# Practice Questions

## Theory Problems

**Que**. Which layers in the Internet protocol stack does a router process? Which layers does a link-layer switch process? Which layers does a host process?

Routers typically process layers 1 through 3 (physical, link and network layers). [Note: Some modern routers sometimes act as firewalls or caching components, and process layer four as well.]
Link layer switches process layers 1 through 2 (physical and link layers).
Hosts process all five layers.

**Que**. How would you describe the overall structure of the Internet? What is the advantage of having such a structure? (Mention at least 2 items)

For the first part, on the **structure of the Internet**:
It is a loose hierarchy from an organizational/provider or routing point of view.
There are several answers:
**Provider hierarchy:** various tiers of providers (tier 1 being the backbone providers and tier 3 closer to local providers) are structured into provider-customer or peering relationships. The hierarchy is loose (not strict) in that tier 2 (or 3) providers can provide peering relationships without going through backbone providers (tier1).
**Routing hierarchy:** there are 2 levels of hierarchy for routing at the autonomous system (AS) (or domain) level; the intra-domain routing and the inter-domain (border gateway [BGP]) routing protocols. BGP establishes peering relationships between border routers at different domains, while the intra-domain routing defines the routing protocol running within a domain.

For the second part **on the advantages**:
The hierarchical structure has multiple advantages:

1- It allows each tier (or domain/autonomous system) to run its own internal protocols and provides uniformity by providing standard protocols and relationships between tiers (or domains)
2- It shields the effects of change in one tier/domain from directly affecting other tiers/domains by keeping the effect local
3- It provides more flexibility in adding or removing domains or ISPs than a non-hierarchical (flat) structure

**Que**. What are the advantages and disadvantages of having a layered protocol architecture for the Internet?

**Advantages:**

1. Allows an explicit structure to identify relationships between various pieces of the complex Internet structure, by providing a reference model for discussion.

2. Provides a modular design that facilitates maintenance, updating/upgrading of protocols and implementations (by various vendors) at the various layers of the stack.
3. Supports a flexible framework for future advances and inventions (such as mobile or sensor networks).

   **Disadvantages:** overhead of headers, redundancy of functions (sometimes not needed) [such as reliability as the transport layer and the link layer, or routing at the network layer and some link layer protocols]

**Que**. What are the five layers in the Internet protocol stack? What are the principal responsibilities of each of these layers?

The five layers in the Internet protocol stack are – from top to bottom – the application layer, the transport layer, the network layer, the link layer, and the physical layer.
The principal responsibilities are:

- **Application layer:** to exchange packets of information (**messages**) with other applications that are distributed over various end systems. Examples include email (SMTP), file transfer (FTP), web (HTTP) and DNS, among others.
- **Transport (or TCP/UDP) layer:** to break down the messages into **segments** at one end (the sender) and reassemble them into messages again (at the other, receiving, end). TCP provides connection-oriented service, flow control (to match the speed of senders and receivers), congestion control (to slow down senders when the network is congested/overloaded) and reliable delivery (through retransmissions). The transport layer operates end-to-end.
- **Network (or IP) layer:** it 'routes' the segments (after breaking them down into IP **datagrams**) through the network [segmentation occurs at the sender, reassembly at the receiver]. The network layer runs the routing protocols that calculate the best (or shortest) paths to get the packets from any point in the network to any other point, based on IP addresses. These protocols also re-calculate routes after failures, and sometime provide load balancing by routing excess traffic over lightly loaded paths. Routing in the Internet occurs hop-by-hop in a store-and-forward fashion. The network layer provides unicast (point-to-point) or multicast (multi-point to multi-point) routing. A routing protocol operates at a domain (multiple-links) level.
- **Link layer:** breaks down the datagrams into **frames** and performs link-specific functions/services, including identification of the MAC layer addresses (sender/receiver) for that hop. Some link layers offer broadcast, multicast or unicast communications. Some offer reliable delivery (on a per-hop or per-link basis) as in WiFi (802.11) via retransmissions or redundancy codes [this is different from the end-to-end reliable TCP service]. For shared media (as in Ethernet, WiFi) the medium access control (MAC) protocol is particularly important at the link layer. Each hop/link the datagram traverses may have a different link layer.
- **Physical layer:** moves **bits** on links. Bit/signal encoding, modulation/demodulation, transmission and reception on the wire (twisted pair, fiber) or on air (radio signals in wireless) are all services provided by the physical layer.

**Que.** How the drawback of circuit switching is overcome in message switching?

Message switching is based on store and forward technique. Instead of establishing a dedicated path, the message is sent to the nearest directly connected node. Each node stores the message, checks for error and forwards it. It allows more devices to share the network bandwidth and one message can be sent to several users. Destination host need not be on at the time of sending message.

**Que.** How packet size affects the transmission time in a packet switching network?

Initially, transmission time decreases as packet size is reduced. But, as packet size is reduced and the payload part of a packet becomes comparable to the control part, transmission time increases.

**Que.** What is the drawback of message switching? How is it overcome in packet switching?

In message switching, large storage space is required at each node to buffer the complete message blocks. On the other hand, in packet switching, messages are divided into subset of equal length, which are generated in the source node and reassembled to get back the initial complete message in destination node. Moreover, to transmit a message of large size, link is kept busy for a long time leading to increase in delay for other messages.

**Que**. Consider sending a packet from a source host to a destination host over a fixed route. List the delay components in the end-to-end delay. Which of these delays are constant and which are variable?

The delay components are processing delays, transmission delays, propagation delays, and queuing delays. All of these delays are fixed, except for the queuing delays, which are variable. [Notes: one can argue that processing delays maybe variable if the load on the machine/router is variable. Also, one can argue that transmission delays are variable in media where the link capacity is variable (as in wireless 802.11)]

**Que.** Distinguish between circuit switching and virtual-circuit packet switching.

In circuit switching, a dedicated path is established. Data transmission is fast and interactive. Nodes need not have storage facility. However, there is a call setup delay. In overload condition, it may block the call setup. It has fixed bandwidth from source to destination and no overhead after the call setup. In virtual-circuit packet switching, there is no dedicated path. It requires storage facility and involves packet transmission delay. It can use different speed of transmission and encoding techniques at different segments of the route.

**Que**. What is the difference between a virus and a worm?

a) Virus: Requires some form of human interaction to spread. Classic example: E-mail viruses.
b) Worms: No user replication needed. Worm in infected host scans IP addresses and port numbers, looking for vulnerable processes to infect.

**Que**. How can a distributed denial of service (DDoS) attack be created?

In DDoS attack, attackers make services (server, bandwidth) unavailable to legitimate users by overwhelming the network or server with bogus traffic and excess requests.
First, the attacker(s) select the target to be attacked. Then, they break into vulnerable hosts in the network via malware to create a botnet. Finally, the compromised hosts send packets or requests to/towards the target (server).

**Que**. Alice and Bob send packets to each other over a network, while Trudy is positioned such that she can capture all the packets sent by Alice and Bob and send whatever she wants to Bob and Alice. List at least three of the malicious things Trudy can do.

i. Trudy can pretend to be Bob to Alice (and vice-versa) and partially or completely modify the message(s) being sent from Bob to Alice. For example, she can easily change the phrase "Alice, I owe you **Rs.**1000" to "Alice, I owe you **Rs.**10,000".
ii. Trudy can pretend to be Bob at a later time via record-and-playback, resending passwords and potentially gaining access to authorized services (that only Bob should access).
iii. Furthermore, Trudy can even drop the packets that are being sent by Bob to Alice (and vise-versa), even if the packets from Bob to Alice are encrypted.

**Que.** Write the functions in sequence which are needed to create a server program using socket API in C. Briefly explain those functions.

**Ans:**

- socket - create an endpoint for communication.
- bind - bind a socket to an address. The address is a pair consisting of an IP-address and a port number.
- listen - specify the maximum number of outstanding connection requests that can be enqueued; that is, the connection request queue length.
- accept - wait to accept an incoming connection request. Use by a server to wait for an incoming request. When a request arrives, a new socket is created and the new socket is used for the connection.
- write, sendto - send data using a connection-oriented (TCP), or connectionless (UDP) protocol, respectively.
- read, recvfrom - read data using a connection-oriented (TCP), or connectionless (UDP) protocol, respectively.
- close - close a connection.

**Que.** What is the principal difference between connectionless communication and connection-oriented communication? Give two example applications for which connection-oriented service is appropriate. Also, give two example applications for connectionless service.

**Que.** Suppose that your department has a local DNS server for all computers in the department. You are an ordinary user (i.e., not a network/system administrator). Can you determine if an external Web site was likely accessed from a computer in your department a couple of seconds ago? Explain.

Yes, we can use dig to query that Web site in the local DNS server. For example, "dig cnn.com" will return the query time for finding cnn.com. If cnn.com is just accessed a couple of seconds ago, an entry for cnn.com is cached in the local DNS cache, so the query time is 0 msec. Otherwise, the query time is large.

**Que.** a. Suppose that Alice wants to send an email message to Bob. This will involve four entities: Alice's mail client (for email composition and sending), Alice's outgoing mail server, Bob's incoming mail server, and Bob's mail client (for email retrieval and viewing). Between which of these four entities does the SMTP protocol operate? What about the IMAP protocol?
b. How does SMTP mark the end of a message body? How about HTTP? Can HTTP use the same method as SMTP to mark the end of a message body? Explain.

**Sol.** a. Answer: SMTP runs between Alices mail client and her server, and also (separately) between her server and Bob's server. IMAP runs between Bob's server and his mail client to retrieve messages from Bob's server.
b. SMTP uses a line containing only a period to mark the end of a message body. HTTP uses "Content-Length header field" to indicate the length of a message body. No, HTTP cannot use the method used by SMTP, because HTTP message could be binary data, whereas in SMTP, the message body must be in 7-bit ASCII format.

**Que.** Explain TCP flow control. Describe Fast retransmit and 3 duplicate ACK techniques.

**Que.** Suppose host A is sending a large file to host B over a TCP connection. If the sequence number for a segment of this connection is m, then the sequence number for the subsequent segment will necessarily be m+1?

**Ans.** False, It can be but not necessarily.

**Que.** Answer true or false to the following questions and briefly justify your answer.

a. With the SR protocol, it is possible for the sender to receive an ACK for a packet that falls outside of its current window.

True. Suppose the sender has a window size of 3 and sends packets 1, 2, 3 at $t0$. At $t1$ ($t1 > t0$), the receiver ACKs 1, 2, 3. At $t2$ ($t2 > t1$) the sender times out and resends 1, 2, 3. At $t3$ the receiver receives the duplicates and re-acknowledges 1, 2, 3. At $t4$ the sender receives the ACKs that the

receiver sent at $t1$ and advances its window to 4, 5, 6. At $t5$ the sender receives the ACKs 1, 2, 3 the receiver sent at $t2$. These ACKs are outside its window.

b. With GBN, it is possible for the sender to receive an ACK for a packet that falls outside of its current window.

True. Same scenario as in (a).

**Que.** In a reliable transfer protocol, can a sender tell the difference between a lost data packet and a lost ACK?

**No**

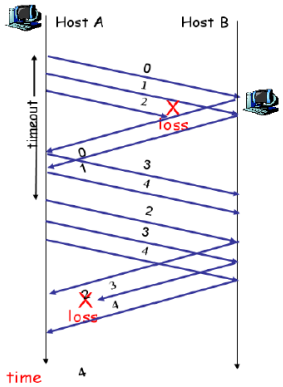
**Que.** How many sequence numbers are needed in a pipelined reliable transfer protocol to avoid ambiguity when the window size is w?

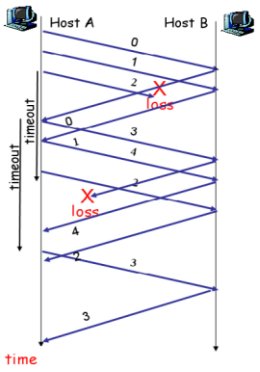
The size of the send window w must be less than $2^m$ (m bits for sequence number). The sequence space wraps to zero after max number is reached. Consider the corner case where ***all*** ACKs are lost - sender does not move its window, but receiver does (since it's unaware the sender is not getting the ACKs). If we don't limit the window size to half the sequence space, we end up with overlapping sender "sent but not acknowledged" and receiver "valid new" sequence spaces. This would result in retransmissions being interpreted as new packets.

**Que.** Explain why for SR, $receive\ window \leq sequence\ num\ range/2$

The receive window in Selective-Repeat is totally different from the one in Go Back-N. The size of the receive window is the same as the size of the send window (maximum $2^{m-1}$) m bits for sequence number. The Selective-Repeat protocol allows as many packets as the size of the receive window to arrive out of order and be kept until there is a set of consecutive packets to be delivered to the application layer. Because the sizes of the send window and receive window are the same, all the packets in the send packet can arrive out of order and be stored until they can be delivered.

**Que.** Draw a diagram with all messages and acknowledgments when sending 5 packets using the GoBackN protocol with a sending window (ie. a pipeline) of size 3. Label each message and acknowledgment with a sequence number between 0 to 4. Assume that the packet with sequence number 2 is lost the first time it is sent, and that the acknowledgment with sequence number 3 is lost the first time it is sent. Repeat for the SR protocol.



# Numerical Problems

**Que**. Suppose users share a 2Mbps link. Also suppose each user requires 1Mbps when transmitting, but each user transmits only 20 percent of the time.

a. When circuit switching is used, how many users can be supported?
b. For the remainder of this problem, suppose packet switching is used. Why will there be essentially no queuing delay before the link if two or fewer users transmit at the same time? Why will there be a queuing delay if three users transmit at the same time?
c. Find the probability that a given user is transmitting.

a) 2 users can be supported because each user requires half of the link bandwidth.
b) Since each user requires 1Mbps when transmitting, if two or fewer users transmit simultaneously, a maximum of 2Mbps will be required. Since the available bandwidth of the shared link is 2Mbps, there will be no queuing delay before the link. Whereas, if three users transmit simultaneously, the bandwidth required will be 3Mbps which is more than the available bandwidth of the shared link. In this case, there will be queuing delay before the link.
c) Probability that a given user is transmitting = 0.2

**Que**. Suppose Host A wants to send a large file to Host B. The path from Host A to Host B has three links, of rates *R1*=500kbps, *R2*=2Mbps, and *R3*=1Mbps.

a. Assuming no other traffic in the network, what is the throughput for the file transfer.
b. Suppose the file is 4 million bytes. Roughly, how long will it take to transfer the file to Host B?
c. Repeat (a) and (b), but now with *R2* reduced to 100kbps.

a) 500 kbps
b) 64 seconds
c) 100kbps; 320 seconds

**Que.** Suppose a 100-Mbps point-to-point link is being set up between Earth and a new lunar colony. The distance from the moon to Earth is approximately 385,000 km, and data travels over the link at the speed of light—$3 * 10^8$ m/s.
(a) Calculate the minimum RTT for the link.
(b) Suppose Mission Control on Earth wishes to download a 25MB image from a camera on the lunar base. What is the minimum amount of time that will elapse between when the request for the data goes out and the transfer is finished?
(c) Using the RTT as the delay, calculate the **delay*bandwidth product** for the link.
(d) Imagine that Mission Control requests one 25MB image and then waits until it starts receiving the file before sending another request (the size of the request is negligible). Use the delay*bandwidth product to determine what percentage of the link is utilized.

**(a)** The time for data to travel from earth to the moon is:

(385,000km*1000m/km) / $3*10^8$m/s = 1.28333s

So the minimum RTT is 1.2833 * 2 = **2.56666s**

**(b)** From (a), the request will take 1.28333s to reach the moon. The time to put the picture on the link is:
(25MB*1024kB/MB*1024byte/kB*8bits/byte) / **(**100Mbps*1000kb/Mb*1000bits/kb)

= 209715200 / 100000000 = 2.097152s

Then it will take another 1.28333s to return to earth. The total time is:

2 * 1.28333s+ 2.097152s = **4.66382 Sec**

**(c)** 2.56666s * 100Mbps * 1000000bits/Mb = **256666000bits**

**(d)** The cycle time for this is 1 RTT. From (c) we see that the link would be fully utilized if 256666000 bits were transferred during this time. The total number of bits transferred is:

25MB * 1024kB/MB * 1024bytes/kB* 8bits/byte = **209715200bits**

The link utilization is 209715200 / 256666000 = **81.707%**

If we assume that the earth doesn't send a request until the entire picture has been received, then the time for a cycle is 4.66382 seconds from (b).

4.66382s *100Mbps * 1000000bits/ Mb = 466382000bits

209715200/466382000 = **44.923%**

**Que.** Calculate the latency (from first bit sent to last bit received) for the following:

(a) A 10-Mbps link with a single store-and-forward switch in the path, and a packet size of 5,000 bits. Assume that each section of the link introduces a propagation delay of 10 microseconds, and that the switch begins retransmitting immediately after it has finished receiving the packet.

(b) Same as (a) but with three switches

(c) Same as (a) but assume the switch implements cut-through switching: it is able to begin retransmitting the packet after the first 200 bits have been received.

(a) To put the packet on the first link it will take:

5000bits/(10 Mbps*1000000bits/Mb) = 0.0005s = 0.5ms

Then it will take another 0.01 ms for the switch to finish receiving the packet for a total of 0.51 ms. It will take another 0.51 ms for the switch to send, and then for the destination to receive all of the packet. The total latency is:

2 * 0.51ms = **1.02ms**

(b) From (a) we know that each link section will take 0.51 ms. Thus, total time will be:

4 * 0.51ms = **2.04ms**

(c) To begin we can calculate the time it will take to send the first 200 bits:

200bits / (10 Mbps*1000000bits/Mb)

= 0.00002s = **0.02ms**

Then .01 ms later the switch can start sending the packet. It will take .5 ms (from (a)) to put the entire packet on the wire (As the switch is sending out the packet, it will continue receiving the rest of the packet). Then it will take .01 ms for the destination to finish receiving the entire packet:

0.02ms + 0.01ms + 0.5ms + 0.01ms = **0.54ms**

**Que**. Two hosts, A and B, connected by a single link of rate *R* bps. Suppose that the two hosts are separated by *m* meters, and that the propagation speed is *s* m/s. Host A is to send a packet of size *L* bits to host B.

a. Express the propagation delay, $d_{prop}$, in terms of *m* and *s*.

b. Determine the transmission time of the packet, $d_{trans}$, in terms of $L$ and $R$.

c. Obtain an expression for the end-to-end delay (ignore queuing and processing delays).

d. Suppose A begins to transmit the packet at time $t$=0. At time t=$d_{trans}$, where is the last bit of the packet?

e. Suppose $d_{prop}$ is greater than $d_{trans}$. At time $t$=$d_{trans}$, where is the first bit of the packet?

f. Suppose $d$prop is less than $d_{trans}$. At time $t$=$d_{trans}$, where is the first bit of the packet?

g. Suppose $s = 2.5 \cdot 10^8$ m/s, $L$=100 bits, and $R$=28kbps. Find the distance $m$ so that $d_{prop}$ equals $d_{trans}$.

**a)** $d_{prop} = m / s$ seconds.
**b)** $d_{trans} = L/ R$ seconds.
**c)** $d_{end\ to\ end} = (m/ s + L / R)$ seconds.
**d)** The bit is just leaving Host A.
**e)** The first bit is in the link and has not reached Host B.
**f)** The first bit has reached Host B.
**g)**

$$m = \frac{L}{R} S = \frac{100}{28 \times 10^3} \left(2.5 \times 10^8\right) = 893\,\text{km}.$$

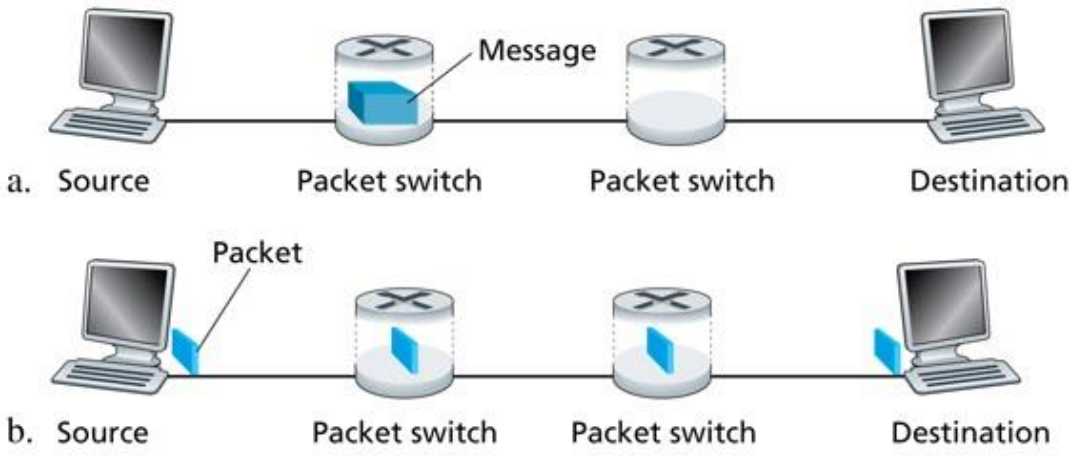
**Que**. In modern packet-switched networks, the source host segments long, application-layer messages (for example, an image or a music file) into smaller packets and sends the packets into the network. The receiver then reassembles the packets back into the original message. We refer to this process as message segmentation. The figure given below illustrates the end-to-end transport of a message with and without message segmentation. Consider a message that is $7.5 \times 10^6$ bits long that is to be sent from source to destination in below Figure. Suppose each link in the figure is 1.5Mbps. Ignore propagation, queuing, and processing delays.

a. Consider sending the message from source to destination without message segmentation. How long does it take to move the message from the source host to the first packet switch? Keeping in mind that each switch uses store-and-forward packet switching, what is the total time to move the message from source host to destination host?

b. Now suppose that the message is segmented into 5,000 packets, with each packet being 1,500 bits long. How long does it take to move the first packet from source host to the first switch? When the first packet is being sent from the first switch to the second switch, the second packet is being sent from the source host to the first switch. At what time will the second packet be fully received at the first switch?

c. How long does it take to move the file from source host to destination host when message segmentation is used/ Compare this result with your answer in part (a) and comment.

d. Discuss the drawbacks of message segmentation.




a) Time to send message from source host to first packet switch =
$\frac{7.5\times10^6}{1.5\times10^6}\text{sec} = 5\,\text{sec}$.
With store-and-forward switching, the total time to move message from source host to destination host = 5sec × 3 *hops* = 15sec

b) Time to send 1st packet from source host to first packet switch =
$\frac{1.5\times10^3}{1.5\times10^6}\text{sec} = 1\,m\,\text{sec}$
Time at which 2nd packet is received at the first switch = time at which 1st packet is received at the second switch = 2 × 1*m*sec = 2 *m*sec

c) Time at which 1st packet is received at the destination host = 1 *m*sec × 3 *hops* = 3 *m*sec . After this, every 1msec one packet will be received; thus time at which last (5000th) packet is received =3 *m*sec+ 4999*1*m*sec = 5.002 sec . It can be seen that delay in using message segmentation is significantly less (almost 1/3rd).

d) Drawbacks:

i. Packets have to be put in sequence at the destination.

ii. Message segmentation results in many smaller packets. Since header size is usually the same for all packets regardless of their size, with message segmentation the total amount of header bytes is more.

**Que.** Calculate the total time required transferring a 1000 KB file in the following cases assuming an RTT of 100 ms, a packet size of 1 KB, and an initial 2 x RTT of "handshaking" before data is sent:

**(a)** The bandwidth is 1.5 Mbps, and data packets can be sent continuously.

We have a 2xRTT initial handshake, followed by the time to transmit 1000 KB, and finally the propagation delay (half RTT) for the last bit to reach the other side. Therefore, we have:

$2*0.1 + 1000*2^{10}*8/(1.5*10^6) + 0.1/2 =$ **5.711 sec**

**(b)** The bandwidth is 1.5 Mbps, but after we finish sending each data packet we must wait for one RTT before sending the next.

After transmitting a packet, we wait for one RTT. Therefore, since RTT > transmission time + propagation delay, by the time we transmit the nextpacket, the first packet has already reached the other side. So, we need the transmission time of a packet + one RTT for each of the first 999 packet. For the last packet, we must wait for the propagation delay for the last bit to reach the other side. Therefore, the total time is as before plus 999 RTTs.

$5.711 + 999 * 0.1 =$ **105.611sec**

**(c)** The bandwidth is "infinite", meaning that we take transmit time to be zero, and up to 20 packets can be sent per RTT.

We have 49 RTTs between batches of 20. The last batch needs half RTT for propagation delay. We also have the 2 initial RTTs, for a total of 51.5 RTTs, i.e. **5.15 sec.**

**(d)** The bandwidth is infinite, and during the first RTT we can send one packet ($2^0$), during the second RTT we can send two packets ($2^1$), during the third RTT we can send four packets ($2^2$), and so on.

Note that $1 + 2 + 4 + \ldots 2^{l} = 2^{l+1} - 1$. Therefore, we want $2^{l+1} - 1$ should be greater than or equal to 1000, which means $l = 9$. We must wait for half an RTT at the end for the propagation delay. With the 2 initial RTTs, this is 11.5 RTTs, i.e. **1.15 sec.**

Questions : Assume that you have base HTML file with 30 embedded images, images & base file are small enough to fit in one TCP segment. How many RTT are required to retrieve base file & images under-following condition :

(i) Non-Persistent connection without parallel connection
(ii) Non-persistent connection with 10 parallel connection
(iii) Persistent connection without pipe-lining
(iv) Persistent connection with pipe-lining
(Assume RTT dominates all other time)

**Ans.** 2RTT is the initial required connection time, one for TCP connection and one for HTML base file.

**Total time = 2RTT + transmit time**

**(i) Non-Persistent connection with no parallel connection:**

Here for each image 2 RTT are required one for TCP connection and one for image to send.

So transmit time for 30 images = 2*(30 RTT) = 60 RTT

Total time = 2 RTT+60 RTT = 62RTT

**(ii) Non-persistent connection with 10 parallel connections:**

Here 10 images can be sent simultaneously.

So for 30 images, it required -> 2*(30/10) = 6RTT

Total time = 2 RTT + 6 RTT = 8RTT

**(iii) Persistent connection without pipelining:**

Here TCP connection is required again and again.

So for 30 images it requires -> 30 RTTs

Total time = 2 RTT + 30 RTT = 32RTT

**(iv) Persistent connection with pipelining:**

Since it is Persistent connection, TCP connection is not required again and again.

In Pipe-lining connection we can send all images in 1RTT.

Total time = 2 RTT + 1 RTT = 3RTT

**Que. a)** Suppose in your web browser you click on a link to obtain a web page. Suppose the associated IP address for the associated URL is not cached in your local host, so that the DNS look-up is necessary to obtain the IP address. Suppose that n DNS servers are visited before your host receives the IP address from DNS; the successive visit incur a RTT of RTT1 , ... , RTTn , Further, suppose that the web page associated with the link contains only one object and a few lines of HTML text. Let RTT0 denote the RTT between the local host and the server containing the object. Assume zero transmission time for the object, how much time elapse from when the client clicks on the link until the object is received by the client.

**b)** Same as in (a), but suppose the page contains three very small objects. Neglecting transmission time, how much time elapse with:

a) Nonpersistent HTTP with no parallel TCP connections,
b) Nonpersistent HTTP with parallel connections,
c) Persistent HTTP with pipelining?

**Ans.: a).** Time to visit n DNS servers and get IP address = RTT1 + RTT2 + ... + RTTn

Time to establish TCP connection = RTT0

Time to send HTTP request and receive reply = RTT0

Total time = 2 * RTT0 + (RTT1 + RTT2 + ... + RTTn)

**Ans. b).** html files containing 3 objects:

**Case 1: where the base html file is considered**

i. Nonpersistent HTTP with no parallel TCP connections: Total time : RTT1 + … +RTTn + 2RTT + 3* (2RTT0).
ii. Nonpersistent HTTP with parallel TCP connections: * Total time : RTT1 + … + RTTn + 2RTT0 + 2RTT0.
iii. Persistent HTTP with pipelining: Total time : RTT1+ … + RTTn + 2RTT0 + RTT0.

**Case 2: where the base html file is not considered**

i. Nonpersistent HTTP with no parallel TCP connections: Total time : RTT1 + … +RTTn + 3* (2RTT0).

ii. Nonpersistent HTTP with parallel TCP connections: Total time : RTT1 + … + RTTn + 2RTT0.

iii. Persistent HTTP with pipelining: Total time : RTT1+ … + RTTn + RTT0.

**Que.** Consider a short, 10-meter link, over which a sender can transmit at a rate of 150 bits/sec in both directions. Suppose that packets containing data are 100,000 bits long, and packets containing only control (e.g., ACK or handshaking) are 200 bits long. Assume that N parallel connections each get 1/N of the link bandwidth. Now consider the HTTP protocol, and suppose that each downloaded object is 100 Kbits long, and that the initial downloaded object contains 10 referenced objects from the same sender. Would parallel downloads via parallel instances of non-persistent HTTP make sense in this case? Now consider persistent HTTP. Do you expect significant gains over the non-persistent case? Justify and explain your answer.

**Ans.** Note that each downloaded object can be completely put into one data packet. Let Tp denote the one-way propagation delay between the client and the server.

First consider parallel downloads via non-persistent connections. Parallel download would allow 10 connections share the 150 bits/sec bandwidth, thus each gets just 15 bits/sec. Thus, the total time needed to receive all objects is given by:
(200/150+Tp + 200/150 +Tp + 200/150+Tp + 100,000/150+ Tp )
+ (200/(150/10)+Tp + 200/(150/10) +Tp + 200/(150/10)+Tp + 100,000/(150/10)+ Tp )
= 7377 + 8*Tp (seconds)

Then consider persistent HTTP connection. The total time needed is give by:
(200/150+Tp + 200/150 +Tp + 200/150+Tp + 100,000/150+ Tp )
+ 10*(200/150+Tp + 100,000/150+ Tp )
=7351 + 24*Tp (seconds)

Assume the speed of light is 300*106 m/sec, then Tp=10/(300*106)=0.03 microsec. Tp is negligible compared with transmission delay.

Thus, we see that the persistent HTTP does not have significant gain (less than 1 percent) over the non-persistent case with parallel download.

**Que.** Suppose you wanted to do a transaction from a remote client to a server as fast as possible. Would you use UDP or TCP? Why?
We would use UDP. With UDP, the transaction can be completed in one roundtrip time (RTT)-the client sends the transaction request into a UDP socket, and the server sends the reply back to the client's UDP socket. With TCP, a minimum of two RTTs are needed -one to set-up the TCP connection, and another for the client to send the request, and for the server to send back the reply.

**Que.** Two HTTP request methods are GET and POST. Are there any other methods in HTTP/1.0? If so, what are they used for? Are there other methods in HTTP/1.1?

**In HTTP 1.0:**

a) HEAD:Asks for the response identical to the one that would correspond to a GET request, but without the response body. This is useful for retrieving meta-information written in response headers, without having to transport the entire content.

**In HTTP 1.1:**

b) PUT: Uploads a representation of the specified resource.
c) DELETE: Deletes the specified resource.
d) TRACE: Echoes back the received request, so that a client can see what (if any) changes or additions have been made by intermediate servers.
e) OPTIONS: Returns the HTTP methods that the server supports for specified URL. This can be used to check the functionality of a web server by requesting instead of a specific resource.
f) CONNECT: Converts the request connection to a transparent TCP/IP tunnel, usually to facilitate SSL-encrypted communication (HTTPS) through an unencrypted HTTP proxy.

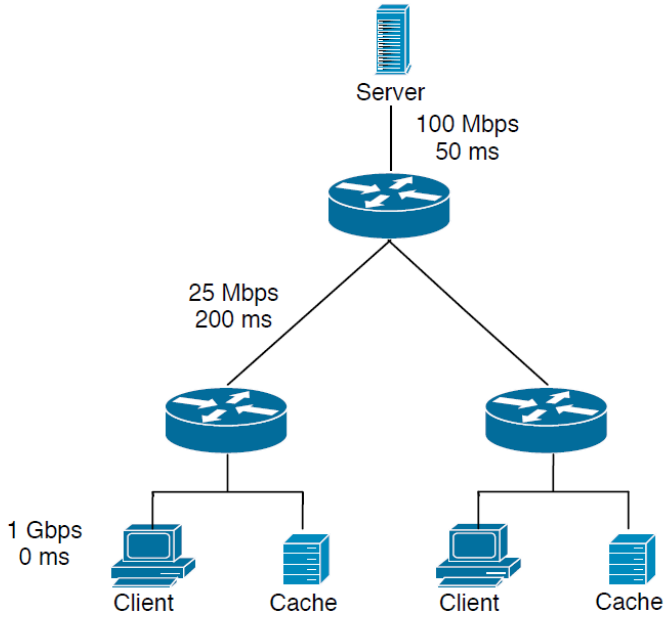
**Que.** A video signal at a resolution of 640 x 480 pixels, 2 bytes/pixel color encoding, and 30 frames/second.

1. Calculate the bandwidth necessary for transmitting in real time.
2. Suppose your cable modem is up to 10 Mbps. Without loss of the resolution and color, how many frames per second can it transfer?

Ans:

1. $640 \times 480 \times 2 \times 8 \times 30 = 147456$ kbps
2. $10 \times 2^{20} / (640 \times 480 \times 2 \times 8) = 2.13$ frames/s

**Que.** Consider the scenario shown in Figure 1 in which a server is connected to a router by a 100Mbps link with a 50ms propagation delay. Initially this router is also connected to two routers, each over a 25Mbps link with a 200ms propagation delay. A 1Gbps link connects a host and a cache (if present) to each of these routers and we assume that this link has 0 propagation delay. All packets in the network are 20,000 bits long




What is the end-to-end delay from when a packet is transmitted by the server to when it is received by the client? In this case, we assume there are no caches, there's no queuing delay at the routers, and the packet processing delays at routers and nodes are all 0.

b. Here we assume that client hosts send requests for files directly to the server (caches are not used or off in this case). What is the maximum rate at which the server can deliver data to a single client if we assume no other clients are making requests?
c. Again we assume only one active client but in this case the caches are on and behave like HTTP caches. A client's HTTP GET is always first directed to its local cache. 60% of the requests can be satisfied by the local cache. What is the average rate at which the client can receive data in this case?

d. What is the average end-to-end delay for the case that 60% of the requests can be satisfied by the local cache?

**Sol.** a. Answer: If all packets are 20,000 bits long it takes 200 usec to send the packet over the 100Mbps link, 800 usec to send over the 25Mbps link, and 20 usec to send over the 1Gbps link. Sum of the three-link transmission is 1020 usec. Thus, the total end-to-end delay is 251.02 msec.
b. Answer: Server can send at the max of the bottleneck link: 25Mbps.
c. Answer: We assume that requests are serially satisfied. 40% of the requests can be delivered at 25Mbps and 60% at 1Gbps. So the average rate is 610Mbps.
d. Answer: 40% of the requests have a delay of 251.02 msec and 60% of the requests have a delay of 20 usec. So the average delay is (150.61 + 12) 162.61 msec.

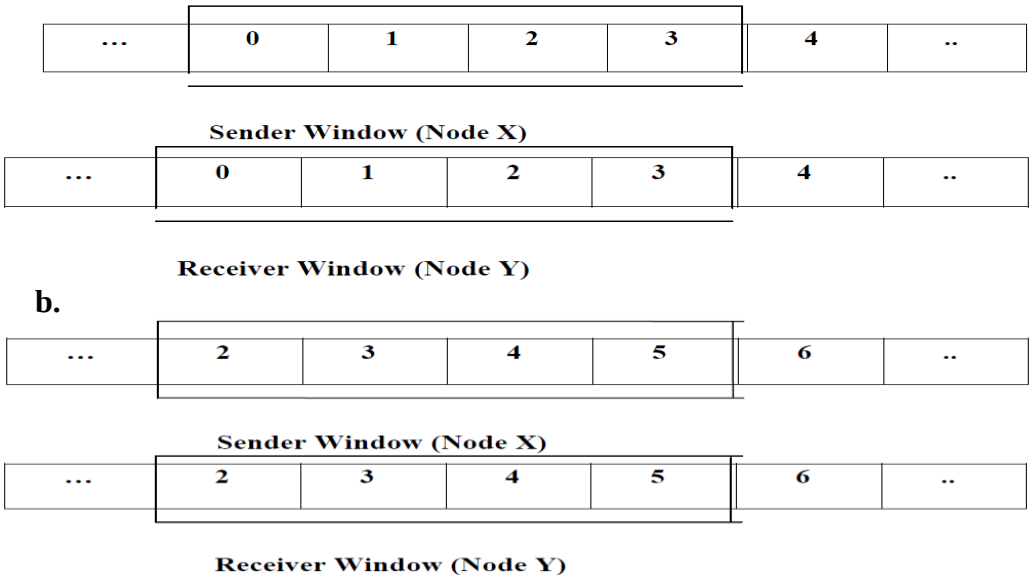
**Que.** What will happen when we double the size of window in TCP congestion control? Consider the case when connection begins and having **CongWin, W** = 1 MSS of 500 bytes & RTT = 200 msec initial rate = 20 kbps. Find out the average throughput for only these two consecutive TCP connections, first having window size W and next one 2W.

**Ans.** The throughput of the link will be doubled.
The average throughput = (W+2W)/2RTT
=1.5 W/RTT
=1.5*(500*8 bits)/ (200 ms) = 1.5*20Kbps = 30Kbps

**Que.** Two neighboring nodes (A and B) use a sliding-window protocol with a 3-bit sequence number. As the ARQ mechanism, Go-back-N is used with a window size of 4. Assuming A is transmitting and B is receiving, show the window positions for the following succession of events at A:
**a**. Before A sends any frames.
**b.** After A sends frames 0, 1,2 and B acknowledges 0, 1 and the ACKs are received by A.
**c.** After A sends frames 3, 4, and 5 and B acknowledges 4 and the ACK is received by A.

**Ans.** For 3- bit sequence number, frame sequence number will be 0, 1, 2…7, 0,..

**a.**

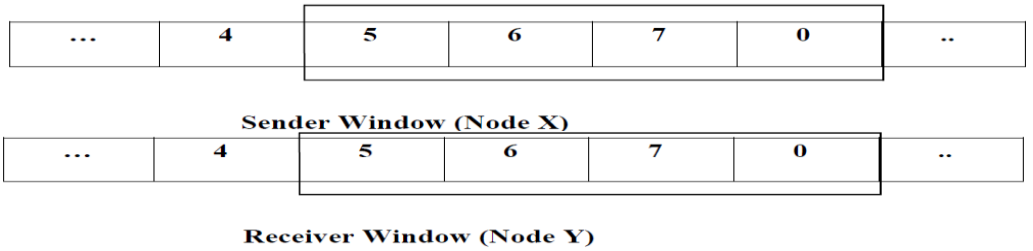



**b.**




**Note: Assuming only ACK for 0 and 1 are sent.**

**c.**




**Note: Assuming only ACK for 3 and 4 are sent.**

**Que.** A client sends a 128-byte request to a server located 100 km away over a 1-gigabit optical fiber. What is the efficiency of the line during the remote procedure call?

Sending 1024 bits over a 1 Gbps line takes 1.024 μsec (1024 bits/($10^9$bits /second)). The speed of light in fiber optics is 200 km/msec (Assume), so it takes 0.5 msec (100 km/(200 km/msec)) for the request to arrive and another 0.5 msec for the reply to get back.

$$U = t_{trans} / t_{trans} + 2 * tprop = 1.024\ \mu sec / 1.024\ \mu sec + 1\ msec$$
$$= 1.024 \times 10^{-3} \text{ or } 0.1\ \%$$

In all, 1024 bits have been transmitted in 1 msec. This is equivalent to 1 megabit/sec, or 0.1% efficiency.

**Que.** Consider the effect of using slow start on a line with a 10-msec round-trip time and no congestion. The receive window is 24 KB and the maximum segment size is 2 KB. How long does it take before the first full window can be sent?

The first bursts contain 2K, 4K, 8K, and 16K bytes, respectively. The next one is 24 KB and occurs after 40 msec because 4 RTT is used.

**Que.** Suppose that the TCP congestion window is set to 18 KB and a timeout occurs. How big will the window be if the next four transmission bursts are all successful? Assume that the maximum segment size is 1 KB.

The next transmission will be 1 maximum segment size. Then the subsequent window size would be: 2, 4, and 8. So after four successes, it will be 8 KB.

**Que.** If the TCP round-trip time, RTT, is currently 30 msec and the following acknowledgements come in after 26, 32, and 24 msec, respectively, what is the new RTT estimate using the Jacobson algorithm? Use α = 0.1.

The successive estimates are 29.6, 29.84, and 29.256.

**Que.** UDP and TCP use 1s complement for their checksums. Suppose you have the following three 8-bit bytes: 01010011, 01100110, 01110100. What is the 1s complement of the sum of these 8-bit bytes? (Note that although UDP and TCP use 16-bit words in computing the checksum, for this problem you are being asked to consider 8-bit sums.) Show all work. Why it that UDP takes the 1s is complement of the sum; that is, why not just use the sum? With the 1s complement scheme, how does the receiver detect errors? Is it possible that a 1-bit error will go undetected? How about a 2-bit error?

```
  0 1 0 1 0 1 0 1
+ 0 1 1 1 0 0 0 0
-----------------
  1 1 0 0 0 1 0 1
```

```
  11000101
+ 01001100
----------
  00010001
```

One's complement = 1 1 1 0 1 1 1 0.

To detect errors, the receiver adds the four words (the three original words and the checksum). If the sum contains a zero, the receiver knows there has been an error. This is an

easy process for detecting the error. All one-bit errors will be detected, but two-bit errors can be undetected (e.g., if the last digit of the first word is converted to a 0 and the last digit of the second word is converted to a 1).

**Que.** Consider a 5-bit sequence number. If the sequence number starts with 0, what is the sequence number of the 100th packet?

A five-bit sequence number can create sequence numbers from 0 to 31. The sequence number in the $N^{th}$ packet is (N mod 32). This means that the 101th packet has the sequence number (101 mod 32) or 5.

**Que.** Using a 5-bit sequence number, what is the maximum size of the send and receive windows for each of the following protocols:

a. Stop-and-Wait **(1)**
b. Go-Back-N **(31)**
c. Selective-Repeat **(16)**

**Que.** Suppose that a sender and a receiver are to perform reliable data delivery.

a. In a Go-Back-N protocol, the window size is 6. Frames with sequence numbers 1, 2, 3, 4 and 5 have been sent. The sender just received an ACK for frame 1. Frames 6, 7, 8, 9 and 10 are waiting to be sent. Draw the time diagram showing this scenario.

b. Which frame(s) can the sender send before it must wait for the next ACK from the receiver? Explain.

Assuming segments 1 through 5 have been sent, the window size of 6 still allows one more segment (segment 6) to be transmitted. When the ACK for segment 1 is received, it opens up the window by an additional 1 segment which means that segment 7 can also be sent.

c. Sometime later, the sender transmitted frames 20, 21, 22, 23, 24, and 26; however, frame 22 got lost. If Go-Back-N is used, what frame(s) would the sender have to retransmit? Explain.

Assuming that the ACKs for 20 and 21 are received, the sender would have to retransmit 22, 23, 24, (25), and 26

d. Suppose the same situation as above but sender and receiver use Selective Repeat. What frame(s) would the sender need to retransmit? Explain. e. Can Selective-Repeat use cumulative ACKs? Explain

Again, assuming that ACKs for 20 and 21 are received, only the lost segment will need to be retransmitted in Selective-Repeat ARQ. (25 and/or 22)

**Que.** Host A is sending a 10,000-byte file to Host B using a sliding window protocol. Packets are limited to 1000 bytes each, packets are numbered by packet number starting at 1, and the window size is 5 packets. Packet 3 is lost.

a. Which packets are retransmitted if Host A and Host B are using the Go-Back N protocol?

(3, 4, 5, 6, 7)

b. Which packets are retransmitted if Host A and Host B are using the Selective Repeat protocol?

(3)

**Que.** Consider a network connecting two systems located 8000 km apart. The bandwidth of the network is 500 x $10^6$ bits per second. The propagation speed of the media is 4 x $10^6$ meters per second. It is needed to design a Go back N sliding window protocol for this network. The average packet size is $10^7$ bits. The network is to be used to its full capacity. Assume that processing delays at nodes are negligible. Then, the minimum size in bits of the sequence number field has to be ?

Given-

- Distance = 8000 km
- Bandwidth = 500 x 106 bps
- Propagation speed = 4 x 106 m/sec
- Packet size = 107 bits

Now,

- For using the network to its full capacity, Efficiency (η) = 1
- Efficiency (η) = 1 when sender window size = 1+2a

**Calculating Transmission Delay-**

Transmission delay (Tt) = Packet size / Bandwidth
= $10^7$ bits / (500 x $10^6$ bits per sec)
= 1 / 50 sec = 0.02 sec

**Calculating Propagation Delay-**

Propagation delay (Tp)
= Distance / Speed
= 8000 km / (4 x $10^6$ m/sec) = 2 sec

**Calculating Value of 'a'-**

a = Tp / Tt
a = 2 sec / 0.02 sec = 100

**Calculating Sender Window Size-**

Sender window size = 1 + 2a
= 1 + 2 x 100 = 201

**Calculating Minimum Size Of Sequence Number Field-**

Minimum number of bits required in the sequence number field = ⌈log2(1+2a)⌉
= ⌈$\log_2$(201)⌉
= ⌈7.65⌉ = 8

Thus, Minimum size of sequence number field = 8 bits.

# Additional Questions (Including TCP congestion Control)

**1.** Consider transferring an enormous file of L bytes from Host A to Host B. Assume an MSS of 536 bytes.

a) What is the maximum value of L such that TCP sequence numbers are not exhausted? Recall that the TCP sequence number field has four bytes.

b) For the L you obtained in (a), find how long it takes to transmit the file. Assume that a total of 66 bytes of transport, network and data-link header are added to each segment before the resulting packet is sent out over a 155 Mbps link. Ignore flow control and congestion control so Host A can pump out the segments back to back and continuously.

Ans.

a) There are 2^32 = 4,294,967,296 possible sequence numbers. The sequence number does not increment by one with each segment. Rather, it increments by the number of bytes of data sent. So the size of the MSS is irrelevant -- the maximum size file that can be sent from A to B is simply the number of bytes representable by 2^32 ≈ 4.19 Gbytes.

b) The number of segments is ceil(2^32 / 536) = 8,012,999

66 bytes of header get added to each segment giving a total of 528,857,934 bytes of header. The total number of bytes transmitted is 2^32 + 528,857,934 = 4.824 × 10^9 bytes. Thus it would take 249 seconds to transmit the file over a 155~Mbps link.

**2.** Hosts A and B are communicating over a TCP connection, and Host B has already received from A all bytes up through byte 126. Suppose Host A then sends two segments to Host B back-to-back. The first and second segments contain 80 and 40 bytes of data respectively. In the first segment, the sequence number is 127, the source port number is 302, and the destination port number is 80. Host B sends an acknowledgment whenever it receives a segment from Host A.

a. In the second segment sent from Host A to B, what are the sequence number, source port number, and destination port number?

b. If the first segment arrives before the second segment, in the acknowledgment of the first arriving segments, what is the ACK number, the source port number and the destination port number?

c. If the second segment arrives before the first segment, in the ACK of the first arriving segment, what is the ACK number?

Ans.

a) In the second segment from Host A to B, the sequence number is 207, source port number is 302 and destination port number is 80.

b) If the first segment arrives before the second, in the acknowledgement of the first arriving segment, the acknowledgement number is 207, the source port number is 80 and the destination port number is 302.

c) If the second segment arrives before the first segment, in the acknowledgement of the first arriving segment, the acknowledgement number is 127, indicating that it is still waiting for bytes 127 and onwards.

3. When establishing a connection in TCP, a ”three-way handshake” is done before exchanging data. Why are three steps needed and not just two? Explain at a self-chosen scenario what could go wrong under which conditions.

Ans: Assume Host A wants to establish a TCP connection with Host B. First, Host A sends a SYN messsage to Host B. Now, Host B knows that Host A wants to establish a connection and replies with a SYNACK message. If there would only be a two-way handshake, Host B would assume the connection to be established and waits for data. But assume the SYNACK message got lost, then Host A would assume that the connection is not established but Host B does so and would wait indefinitely

4. In Stop and wait protocol every 4th packet is lost and we need to send total 10 packets so how many transmission it took to send all the packets ?

**Ans.**

1 2 3 4 5 6 7 8 9 10 (Initially)

^

1 2 3 4 4 5 6 7 8 9 10 (Packet no. 4 retransmitted)

^

1 2 3 4 4 5 6 7 7 8 9 10 (Packet no. 10 retransmitted)

^

1 2 3 4 4 5 6 7 7 8 9 10 10 (Result)

So, we retransmitted packet number 4, 7, 10. Total count = 13

5. In GBN sender Window size = 10 and Tp = 49.5ms & Tt = 1ms. What is the Efficiency of the protocol and throughput given Bandwidth=1000 bps ?

**Ans. –**

Efficiency = N/(1+2a), N = 10 (given), a = Tp/Tt = 49.5
Efficiency = 10/(1+2*49.5) = 10/100 = 0.1 or 10%
Throughput = Efficiency * Bandwidth
= 0.1*1000 = 100

From Last packet is window size to lost pocket we resend the entire window. Total no. of transmissions = 18

6. If there is K bits sequence no., define require sender window size and receiver window size for S&W, GBN & SR?

**Ans –**

Given, K bits, For S&W Ws = 1 and Wr= 1
For GBN, Ws = 2^(K)-1 and Wr = 1
For SR, Ws = 2^(K-1) and Wr = 2^(K-1)

7. In SR Ws= 5 and we are sending 10 packets where every 5th packet is lost find number of retransmissions ?

**Ans.**

1 2 3 4 5 6 7 8 9 10

^

1 2 3 4 5 5 6 7 8 9 10

^

1 2 3 4 5 6 7 8 9 9 10

We see here there is no role of Window size in SR only the lost packet is resent. Total transmissions = 12

8. Consider a selective repeat sliding window protocol that uses a frame size of 1 KB to send data on a 1.5 Mbps link with a one-way latency of 50 msec. To achieve a link utilization of 60%, what will be the the minimum number of bits required to represent the sequence number field ?

Transmission delay = Frame Size/bandwidth

= (1*8*10^3)/(1.5 * 10^6)=5.33ms

Propagation delay = 50ms

**Efficiency = Window Size/(1+2a)** = .6

a = Propagation delay/Transmission delay

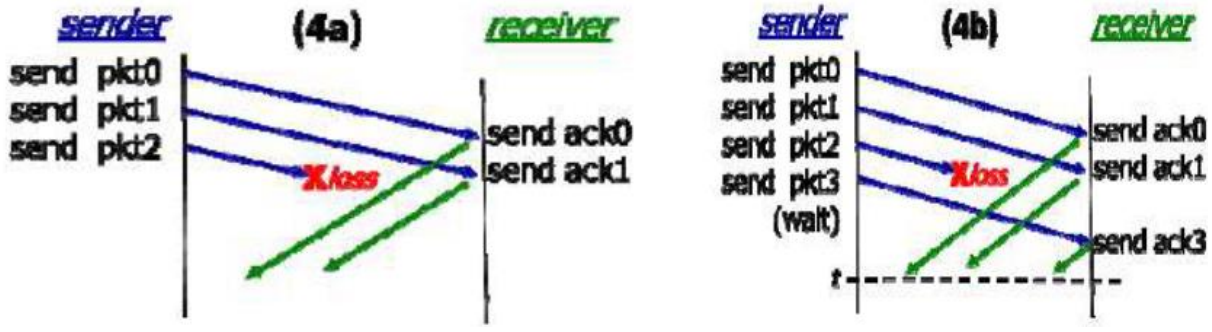
So, window size = 11.856(approx)

**min sequence number = 2*window size** = 23.712

bits required in Min sequence number = $\log_2(23.712)$
Answer is 4.56

Ceil(4.56) = 5

9. Consider the sliding window protocol in Figure (4a) to the right. Does this figure indicate that Go-Back-N is being used, Selective Repeat is being used, or there is not enough information to tell? Explain your answer briefly.

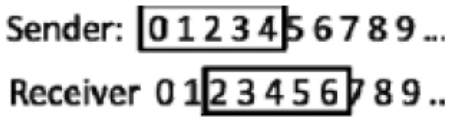


There is not enough information to tell, since both GBN and SR will individually ACK each of the first two messages as they are received correction.

(b.) Consider the sliding window protocol Figure (4b) to the right. Does this figure indicate that Go-Back-N is being used, Selective Repeat is being used, or there is not enough information to tell? Explain your answer briefly.

This must be the SR protocol since pkt 3 is acked even though pkt 2 was lost. GBN uses cumulative ACKs and so would not generate an ACK 3 if pkt 2 was missing

(c.) Consider Figure (4b) again. Suppose the sender and receiver windows are of size N = 4 and suppose the sequence number space goes from 0 to 15. Show the position of the sender and receiver windows over this sequence number space at time t (the horizontal dashed line).




(d.) Give a list of all possible future events at the sender resulting from the ACKs currently propagating from receiver to sender at time t. For each of these events, indicate the action take at the sender (only).

If the next event is ACK0 received, then the sender will advance the window and send pkt4
If the next event is ACK1 received (ACK 0 is lost), then the sender will note that 1 has been ACKed but will not advance the window and will not send anything.
If the next event is ACK2 received (ACK 0 and ACK 1 are lost), then the sender will note that 2 has been ACKed but will not advance the window and will not send anything.
If ACK0, ACK1, and ACK3 are lost the next event will be a timeout, and since no ACKs have been received the sender will resend pkt0, pkt1, pkt 2 and pkt3.

**11.** Station A needs to send a message consisting of 9 packets to Station B using a sliding window (window size 3) and go-back-n error control strategy. All packets are ready and immediately available for transmission. If every 5th packet that A transmits gets lost (but no acks from B ever get lost), then what is the number of packets that A will transmit for sending the message to B? **[Ans. 16 packets]**

**12. Station A uses 32 byte packets to transmit messages to Station B using a sliding window protocol. The round trip delay between A and B is 80 milliseconds and the bottleneck bandwidth on the path between A and B is 128 kbps. What is the optimal window size that A should use?**

**Ans.** Round Trip propagation delay = 80ms
Frame size = 32*8 bits
Bandwidth = 128kbps
Transmission Time = 32*8/(128) ms = 2 ms

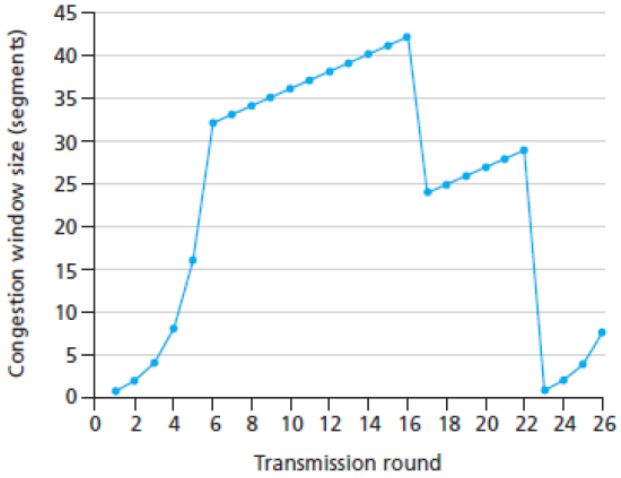
Let n be the window size.

Utilization = n/(1+2a) where a = Propagation time / transmission time
= n/(1+80/2)

For maximum utilization: n = 41.

**13.** Consider Figure below. Assuming TCP Reno is the protocol experiencing the behavior shown above, answer the following questions. In all cases, you should provide a short discussion justifying your answer.

a. Identify the intervals of time when TCP slow start is operating.
b. Identify the intervals of time when TCP congestion avoidance is operating.
c. After the 16th transmission round, is segment loss detected by a triple duplicate ACK or by a timeout?
d. After the 22nd transmission round, is segment loss detected by a triple duplicate ACK or by a timeout?
e. What is the initial value of ssthresh at the first transmission round?
f. What is the value of ssthresh at the 18th transmission round?
g. What is the value of ssthresh at the 24th transmission round?




a) TCP slow start is operating in the intervals [1, 6] and [23, 26]

b) TCP congestion avoidance is operating in the intervals [6, 16] and [17, 22]

c) After the 16th transmission round, packet loss is recognized by a triple duplicate ACK. If there was a timeout, the congestion window size would have dropped to 1.

d) After the 22nd transmission round, segment loss is detected due to timeout, and hence the congestion window size is set to 1.

e) The threshold is initially 32, since it is at this window size that slow start stops and congestion avoidance begins.

f) The threshold is set to half the value of the congestion window when packet loss is detected. When loss is detected during transmission round 16, the congestion windows size is 42. Hence the threshold is 21 during the 18th transmission round.

g) The threshold is set to half the value of the congestion window when packet loss is detected. When loss is detected during transmission round 16, the congestion windows size is 42. Hence the threshold is 21 during the $18^{th}$ transmission round.

**13. Suppose the TCP Congestion window (cwnd) is set to 24 KB and a time out occurs. If you use slow start and AIMD, and the next 6 transmission bursts are all successful then what will be the size of cwnd ? (take MSS=1KB).**

**Ans.**

When Time out occurs the SSthresh=12KBSSthresh=12KB

Before 1 stsuccessfull transmission cwnd = 1KB after cwnd = 2KB

Before 2 stsuccessfull transmission cwnd = 2KB after cwnd = 4KB

Before 3 stsuccessfull transmission cwnd = 4KB after cwnd = 8KB

Before 4 stsuccessfull transmission cwnd = 8KB after cwnd = 12KB

since 16KB>12KB16KB>12KB

Before 5 stsuccessfull transmission cwnd = 12KB after cwnd = 13KB

Before 6 stsuccessfull transmission cwnd=cwnd=13KB after cwnd=14KBcwnd=14KB

**Thus answer is 14KB**

Before 7 stsuccessfull transmission cwnd = 14KB after cwnd = 15KB and so on.

**14.** Consider an instance of TCP's Additive Increase Multiplicative Decrease (AIMD) algorithm where the window size at the start of slow start phase is 2 KB and the threshold at the start of first transmission is 24 KB. Assume that 3 duplicate ACK are received during the 5th transmission what is the congestion window size at the end of 10th transmission?

Ans.

2|4|8|16|24(D.A)|242+3=15|16|17|18|19|2|4|8|16|24(D.A)|242+3=15|16|17|18|19|

At the end of 10th10th transmission, it will be 20K.B

**15.** Host A sends a file consisting of 9 MSS-sized segments to a host B using TCP. Assume that the 4th segment in the transmission is lost. Assume the retransmission timeout is T, the one-way latency is d, and that T > 4*d. Ignore the transmission time of the segments and of the acknowledgements. Also, assume the TCP three-way handshake has completed, but no data has been transmitted.

a) Assume no fast retransmission or fast recovery. Draw the time diagram showing each segment and acknowledgement until the entire file is transferred. Indicate on the diagram all changes in the cwnd and ssthresh. How long does it take to transfer the file?

b) Answer part (a) assuming TCP Reno, i.e., the TCP version that implements both fast retransmission and fast recovery.

c) Answer part (b) assuming that only the 6th segment is dropped.

**Notes**

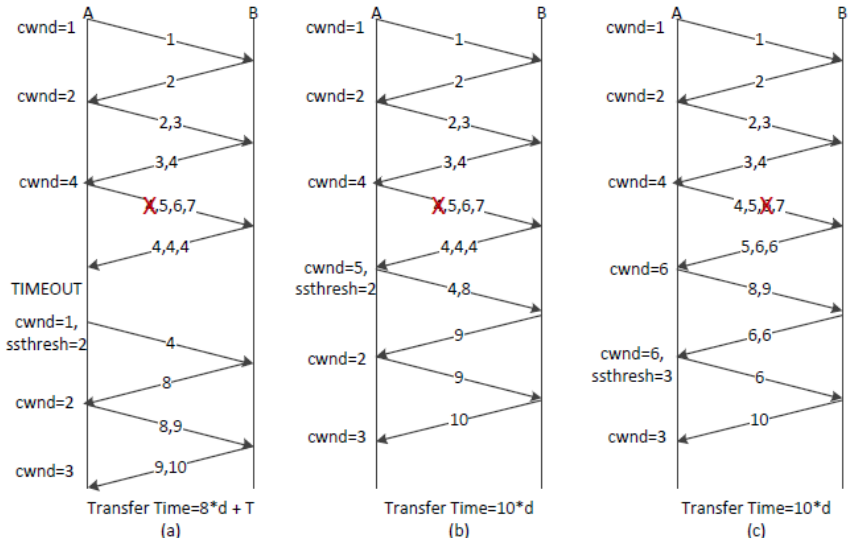
_ For Fast Recovery, assume that each duplicate acknowledgment increases cwnd by 1.

_ For Fast Recovery, assume that, upon receiving a non-duplicate acknowledgment, cwnd drops back to ssthresh.

_ If the value of cwnd is fractional, you should round it to the closest larger integer.

_ The transfer time is the time interval measure at source A from the time the first segment is sent until the acknowledgement of the last segment is received

**Ans.**

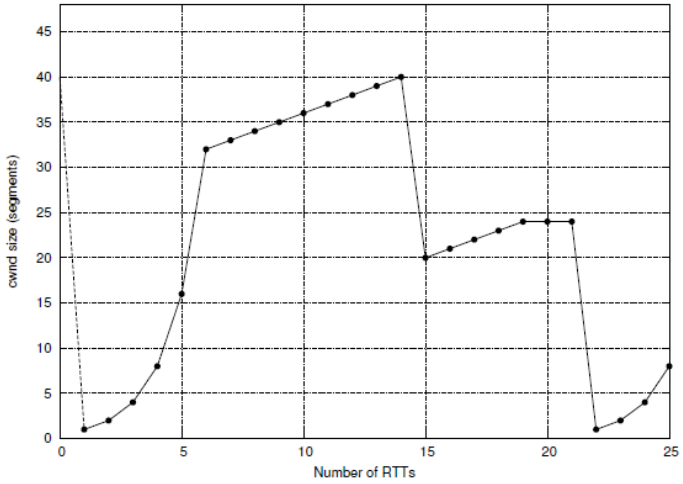



**16.** NASA has built a series of satellites to orbit the Earth, except they lack the requisite knowledge to design a communication protocol. They task you with designing a reliable, sliding window, bytestream protcol similar to TCP for communication over a link with a bandwidth of 1 Gbps, an RTT of 384 ms, and a maximum segment lifetime of 15 sec. The protocol will not include a window scaling option.

**a)** How many bits wide should you make the AdvertisedWindow and SequenceNum fields? Explain why you select your values

**b)** The person in charge of the program scoffs at your suggestions, and instead wishes to use an AdvertisedWindow of 16 bits. Explain why this is problematic.

**c)** Suppose you settle on using TCP as your communication protocol. Two satellites, S1 and S2 want to communicate with a third satellite. Both S1 and S2 are in the Additive Increase, Multiplicative Decrease (AIMD) phases of their transmissions, but S2 has a much higher RTT than S1. As such, you observe that S2 is not receiving its fair share of the bandwidth. Explain how, under AIMD, the TCP congestion window grows with respect to the RTT. Using that intuition, explain why the bandwidth is not being shared fairly between S1 and S2.




**17.** Assume TCP Reno (meaning Fast Retransmit and Fast Recovery) is the protocol experiencing the behavior shown above. Assume that the TCP flow has been operating for some time, meaning that the number of RTTs shown are with respect to when you started observing the flow's behavior.

a) Identify the time periods when TCP slow start is operating.
b) Identify the time periods when TCP congestion avoidance is operating (AIMD).
c) After the 14$^{th}$ RTT, is the segment loss detected by a triple duplicate ACK or by a timeout?
d) What is the initial value of ssthreshold, before the first congestion avoidance interval?
e) What is the value of ssthreshold at the 19$^{th}$ RTT?
f) What is the value of ssthreshold at the 24$^{th}$ RTT?
g) Assuming a packet loss is detected after the 25$^{th}$ RTT by the receipt of a triple duplicate ACK, what will be the values of the congesion-window size and of ssthreshold?

**Ans.**

a) 1-6, 22-25
b) 6-22 (or 19, if you consider 20-22 a"timeout" phase and also accept 19 as an upper range.)
c) Duplicate Acknowledgment. If a timeout happened, cwnd would drop to 1
d) 32. At 32, the flow exits slow start and proceeds to AIMD.
e) 20. At a cwnd size of 40, the cwnd is halved to 20 and the ssthreshold is set to 20.
f) 12, At a cwnd size of 24, a timeout occurs. ssthreshold is set to half of 24 (12), and the cwnd size drops to 1.
g) 4,4. The triple dup ack causes the ssthreshold to be set to half the value of cwnd, which was previously 8 at the conclusion of the 25th RTT. Due to fast recovery,the cwnd is set to 4, or 7 if you considered inflation.

**18.** Consider a residential network that connects to the internet with a DSL link that has a download rate of 4 Mb/s. Assume that there are three UDP flows sharing the link and the remote hosts are sending at rates of 1 Mb/s, 2 Mb/s and 3 Mb/s. Assume that the ISP router has a link buffer that can hold 300 packets (assume all packets have the same length).

**a.** For each flow, what fraction of the packets it sends are discarded?
**b.** For each flow, about how many packets does it have in the queue.

**c.** Now, suppose the queue at the ISP router is replaced by three queues that can each hold 100 packets and that the queues are scheduled using weighted-fair queueing, where the weights are all 0.33. In this case, what fraction of packets are discarded from each flow?

**d.** How many packets does each flow have in the queue?

**e.** Now, suppose the weights are 0.2 for the first flow, 0.6 for the second and 0.2 for the third. In this case, what fraction of packets are discarded from each flow?

**f.** How many packets does each flow have in the queue?

**Ans.**

**a.** They each lose about 33% of the packets they send.

**b.** The first flow has about 50 packets in the queue, the second flow has about 100 and the third has 150.

**c.** In this case, the first flow loses 0, the second loses 25% and the third loses 50%.

**d.** The first flow's queue is empty or close to empty. The other two each have about 100 packets in their queues.

**e.** In this case, the first two flows each lose 0, while the third loses 67%.

**f.** The second flow's queue is empty or close to empty. The third has about 100 packets in its queue. The first flow is a little tricky, as its allocated portion of the link bandwidth exactly matches its data rate. If the packets arrive with uniform spacing, the queue will remain close to empty, but if the times between packets are more random, significant queueing can occur. Indeed, there can even be some packet loss in this case. For a single flow however, it's more likely that the queue will never accumulate a large backlog.

**19.** Consider the effect of using slow start on a line with a 10-msec round-trip time and no congestion. The receive window is 24 KB and the maximum segment size is 2 KB.

a. How long does it take before the first full window can be sent?

b. Suppose the TCP congestion window is set to 18KB and a timeout occurs. How big will the window be if the next four transmission bursts are all successful? Assume that the maximum segment size is 1KB.

**Ans. a.** With slow start, the first RTT sends out 1 segment (or 2KB), the 2nd RTT sends out 2 segments (or 4KB), the 3rd 4 segments (or 8KB), the 4th 8 segments (or 16KB). The 5th RTT would have sent out 16 segments (or 32KB), however, it'll exceed the receiver's window. Therefore, the amont of time it takes BEFORE the 5th RTT (or full window, that is, 24KB) is 4*10=40msec.

**b.** When a timeout occurs, three things happened. First, slow start will be initiated. Second, the congestion window would start at 1. Third, the threshold will be reset to 18KB/2=9KB. If the next four transmission are all successful, then

- 1st transmission: 1 segment, 1KB
- 2nd transmission: 2 segments, 2KB
- 3rd transmission: 4 segments, 4KB
- 4th transmission: 8 segments, 8KB

After these four successful transmissions, the window size is supposed to be 16. However, since the threshold is 9KB, the window size can only be 9KB.

**20.** A TCP machine is sending full windows of 65,535 bytes over a 1-Gbps channel that has a 10-msec one-way delay. What is the maximum throughput achievable? What is the line efficiency?

**Ans.** One window can be sent every 20 msec. This gives 50 windows/sec, for a maximum data rate of about 3.3 million bytes/sec. The line efficiency is then 26.4 Mbps/1000 Mbps or 2.6%.

**21.** What is the total size of the minimum TCP MTU, including TCP and IP overhead but not including data link layer overhead?

**Ans**. The default segment is 536 bytes. TCP adds 20 bytes and so does IP, making the default 576 bytes in total.