1. What are the 3 timers used in the TCP layer. Explain their purpose.

Retransmission timer: Started when a packet is sent for which an ACK is expected. If the ACK is not received before the timer fires, the packet will be retransmitted.

Persistence timer: Started when a non zero window size is advertised following a zero window size advertisement. This is to prevent deadlocks. If the non-zero Window size advertisement is lost both the sender and the receiver will be waiting for a packet from the other side. If no response is received until the time the persistence timer expires, the non-zero window size should be re-advertised.

Keep alive timer: This is to keep a connection open even when the sender and receiver may have no application data to exchange between them. This timer is started every time is a packet is sent or received. If no packet has been sent or received when till the time the keep alive timer expires, a dummy TCP packet with no data is sent to keep the connection alive.

2. Silly Window Syndrome:

Caused when the application at the receiver reads bytes in small chunks and waits for long durations between chunks. In such cases the TCP buffer fills up mandating a window size 0 advertisement. The application then reads a few bytes resulting in a non zero window size advertisement. The buffer is again filled up forcing the TCP layer to send another zero window size advertisement, and so on.

3. Explain the need for dynamic window sizes in TCP.

The window size advertised corresponds to the size of the storage buffer available for the TCP connection. The initial window size represent the size of the number of bytes allocated. When TCP packets are received from the other end, the application data (TCP payload) will need to be stored in the buffer until the application is ready to read the data (after which the portions that have been read by the application can be cleared). Every TCP packet indicates the current buffer space available.

The congestion window is used to prevent swamping of a congested network with additional packets. The TCP end points set a congestion window size which is initially equal to the MTU size (about 1400 bytes of application payload as the MTU size is restricted by the payload size of Ethernet – which is 1500 bytes.) Every time an ACK is received in time, the congestion window size is doubled. The effective window size is the minimum of the congestion window size and the current storage window size advertised by the other end.

The congestion window size is doubled every time (1 MTU, 2 MTU, 4 MTU, 8 MTU etc, if ACKs are received in time every time) only till a threshold window size is reached. After the threshold is reached (say 40 MTU), the window size is increased linearly (41 MTU, 42 MTU etc). At any point if the ACK is not received in time the congestion window size is reset to 1 MTU ands the threshold is chosen as half of the last used effective Window size.

4. A process A initiates a TCP connection to a process B. Over the connection, A sends 10,000 bytes of application data to B. Process B sends 5,000 bytes to A. Make the following assumptions:

- the starting sequence numbers of A and B are 100,000 and 200,000 respectively;
- A and B allocate buffers of size 8000 bytes to process the connection.
- The maximum number of application data bytes is restricted to 1400 bytes in each packet.

Make other reasonable assumptions if required. However, you should explicitly specify the assumptions.

Indicate the contents of the TCP headers (SEQ numbers, ACK numbers, flags set, Window size) of all packets exchanged between A and B.
Packets from A to B are indicated in green. Packets from B to A are indicated in black

10000 bytes will be sent in 7 packets with payload of 1400 bytes and one packet with payload of 200 bytes (7*1400+200 = 10000)

Packet 1: A to B, Seq 100000, ACK NONE; Flags:SYN; W 8000
Packet 1: B to A, Seq 200000, ACK 100001; Flags:SYN, ACK; W 8000;
Packet 2: A to B, Seq 100000, ACK 200001; Flags:ACK ; W 8000;
Packet 3: A to B, Seq 100001, ACK NONE; Flags:NONE ; W 8000; payload 1400;
Packet 2: B to A, Seq 200000, ACK 101400; Flags:ACK W 6600; (1400 bytes in buffer)
Packet 4: A to B, Seq 101401, ACK NONE; Flags:NONE; W 8000; payload 1400;
(At this point B's buffer has 2800 bytes, assume application layer of B reads the application data and clears the buffer)
Packet 5: A to B, Seq 102801, ACK NONE; Flags:None; W 8000; payload 1400;
Packet 6: A to B, Seq 104201, ACK 101401; Flags:None; W 8000; payload 1400;
Packet 7: A to B, Seq 105601, ACK NONE; Flags:None; W 8000; payload 1400;
(At this point B's buffer has 4200 bytes, assume application layer of B reads the application data and clears the buffer)
Packet 8: A to B, Seq 107001, ACK NONE; Flags:None; W 8000; payload 1400;
Packet 9: A to B, Seq 108401, ACK 101401; Flags:None; W 8000; payload 1400;
Packet 4: B to A, Seq 200000, ACK 109801; Flags:ACK; W 5200;
Packet 10:A to B,Seq 109801, ACK 200001,Flags:FIN,ACK W 8000; payload 200;
(At this point B's buffer has 3000 bytes, assume application layer of B reads the application data and clears the buffer)
Packet 5: B to A, Seq 200000, ACK 110000, Flags:FIN,ACK W 8000;
Packet 11:A to B, Seq 110000, ACK 200001,Flags ACK W 8000;

Indicate the difference (if any) in the TCP headers of packets exchanged for the following three cases:
   a) A and B are processes running on two computers that are not in the same LAN;
   b) A and B are processes running on different computers in the same LAN;
   c) A and B are processes running on the same machine.

No difference! TCP layer does not care.