Assignment 2

Computer Networking

***Name :Amit Kumar Maurya***

***Class: CS-A(3)***

***Roll No.: 11812095***

***Que1>*** Write TCP congestion control mechanism TCP tahoe, reno and vegas.

***Ans>***

**TCP tahoe**:

Tahoe suggests that whenever a TCP connection starts or re-starts after a packet loss it should go through a procedure called ‘slow-start’. The reason for this procedure is that an initial burst might overwhelm the network and the connection might never get started. Slow starts suggests that the sender set the congestion window to 1 and then for each ACK received it increase the CWD by 1. so in the first round trip time(RTT) we send 1 packet, in the second we send 2 and in the third we send 4. Thus we increase exponentially until we lose a packet which is a sign of congestion. For congestion avoidance Tahoe uses ‘Additive Increase Multiplicative Decrease’. A packet loss is taken as a sign of congestion and Tahoe saves the half of the current window as a threshold value. It then set CWD to one and starts slow start until it reaches the threshold value. After that it increments linearly until it encounters a packet loss.

**TCP reno:**

Reno retains the basic principle of Tahoe. Reno suggest an algorithm called ‘Fast Re-Transmit’. It works as:

**1)**Each time we receive 3 duplicate ACK’s we take that to mean that the segment was lost and we re-transmit the segment immediately and enter ‘Fast Recovery’

**2)**Set SSthresh to half the current window size and also set CWD to the same value.

**3)**For each duplicate ACK receive increase CWD by one. If the increase CWD is greater than the amount of data in the pipe then transmit a new segment else wait. If there are ‘w’ segments in the window and one is lost, the we will receive (w-1) duplicate ACK’s. Since CWD is reduced to W/2, therefore half a window of data is acknowledged before we can send a new segment. Once we retransmit a segment, we would have to wait for atleast one RTT before we would receive a fresh acknowledgement. Whenever we receive a fresh ACK we reduce the CWND to SSthresh. If we had previously received (w-1) duplicate ACK’s then at this point we should have exactly w/2 segments in the pipe which is equal to what we set the CWND to be at the end of fast recovery. Thus we don’t empty the pipe, we just reduce the flow. We continue with congestion avoidance phase of Tahoe after that.

**TCP vegas:**

Vegas is a TCP implementation which is a modification of Reno. It builds on the fact that proactive measure to encounter congestion are much more efficient than reactive ones.

The three major changes induced by Vegas are:

1. *New Re-Transmission Mechanism:* Vegas extends on the re-transmission mechanism of Reno. It keeps track of when each segment was sent and it also calculates an estimate of the RTT by keeping track of how long it takes for the acknowledgment to get back. Whenever a duplicate acknowledgement is received it checks to see if the (current time-segment transmission time)> RTT estimate; if it is then it immediately retransmits the segment without waiting for 3 duplicate acknowledgements or a coarse timeout .
2. *Congestion avoidance:* It does not use the loss of segment to signal that there is congestion. It determines congestion by a decrease in sending rate as compared to the expected rate, as result of large queues building up in the routers. Whenever the calculated rate is too far away from the expected rate it increases transmissions to make use of the available bandwidth, whenever the calculated rate comes too close to the expected value it decreases its transmission to prevent over saturating the bandwidth.
3. *Modified Slow-start:* TCP Vegas differs from the other algorithms during it’s slow-start phase. The reason for this modification is that when a connection first starts it has no idea of the available bandwidth and it is possible that during exponential increase it over shoots the bandwidth by a big amount and thus induces congestion. To this end Vegas increases exponentially only every other RTT, between that it calculates the actual sending through put to the expected and when the difference goes above a certain threshold it exits slow start and enters the congestion avoidance phase.

***Que2>*** Explain how the TCP protocol would be able to detect errors and data loss, and how it would ensure that the lost data is re-transmitted.

***Ans>*** TCP has so-called sequence numbers for every packet. The sequence number addresses bytes, so if the sender says "this is the packet with sequence number 102", he says, that the packet he sent starts with the byte 102 of the stream. The receiver then sends and acknowledgement to the receiver.

For example, if the packet has a length of 10 bytes, the receiver will send and Ack with the sequence number 112, which means "I expect the next packet to be received to start with the sequence number 112". Every packet that has a different sequence number is either a duplicate (too low) or a packet got lost (received sequence number is too high) or it is a phantom (total mismatch of expected sequence number and received one). So in all cases of sequence-number mismatch, the receiver knows that something goes wrong and can react (differs from different TCP versions).

The sender waits for the acknowledgements of the receiver. If he does not receive an expected ack for a certain time, he will retransmit packets, because he assumes, that the packets got lost on their way.

***Que 3>*** Let the size of congestion window of a TCP connection be 32 KB when a timeout occurs. The round trip time of the connection is 100 msec and the maximum segment size used is 2 KB. The time taken (in msec) by the TCP connection to get back to 32 KB congestion window is \_\_\_\_\_\_\_\_\_.

***Ans>*** Current size of congestion window in terms of no. of segments = (size in bytes)/(maximum segment size)

= 32 KB/2 KB

= 16 MSS( maximum segment size)

When the timeout occurs, threshold is reduced to half of the congestion window size. So threshold = 16 MSS/2 = 8 MSS

Then, slow start phase begins i.e. congestion window size becomes 1 MSS and start increasing exponentially until it hits the threshold value:

1 -> 2 -> 4 -> 8 ( 3 RTT required )

After that congestion avoidance phase begins i.e. congestion window size increases linearly ( 1 MSS with each RTT ).

To reach from 8 MSS to 16 MSS, 8 RTT will be required.

Total RTT = 3 + 8 = 11

1 RTT = 100 ms

Total time required = 11 \* 100

= 1100 ms

***Que 4>*** 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>*** For optimal window size, utilization should be maximum possible.

Frame size = 32 byte = 32\*8 bits

RTT = 80 ms

Bandwidth = 128 kbps = 128 \* 10^-3 bps

Time to send 1 frame = (32\*8)/(128\*10^-3)

= 2 ms

Total delay = RTT + time to send 1 frame

= 80 + 2 = 82 ms

Let optimal window size be n frames,

Utilization = (n\*time to send 1 frame) / (total delay)

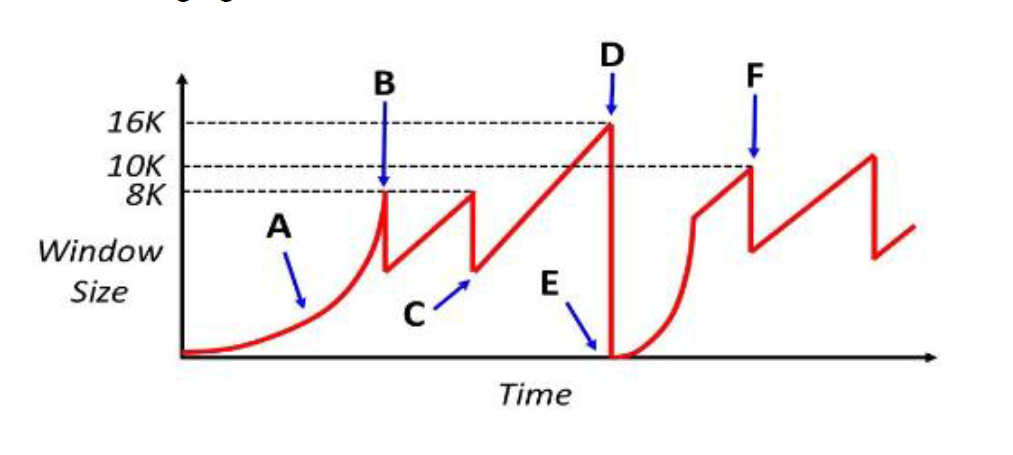
= (n \* 2)/82

= n/41

Utilization will be max when n = 41

So, optimal window size = 41

***Que 5>*** Consider following figure



**a.** In TCP congestion control, name the event at A, B, C, D, and E.

**b.** For a lightly loaded network, is the event at D MORE likely or LESS likely to occur when the sender has multiple TCP segments outstanding. Discuss

**c.** Assume that the network has an MSS of 1000 bytes and the round trip time between sender and receiver of 100 milliseconds. Assume at time t=0 the sender attempts to open the connection. Also assume that the sender can “write” a full window’s worth of data instantaneously, so the only latency you need to worry about is the actual propagation delay of the network. How much time has progressed by point B?

***Ans>*** **a)** A – slow start phase

B – 3 duplicate acks received

C-

D- time out

E- start of slow start phase

F- 3 duplicate acks received

**b)** D is time out. When there is lightly loaded network, time

out is less likely to occur.

**c)** 400 ms

***Que 6>*** Consider an instance of TCP’s AIMD algorithm where the window size at the start of the slow start phase is 2 MSS and the threshold at the start of the first transmission is 8 MSS. Assume that a timeout occurs during the fifth transmission. Find the congestion window size at the end of the tenth transmission.

***Ans>*** Threshold = 8 MSS

Upto 8 MSS window size increases exponentially.

First transmission = 2 MSS

Second transmission = 4 MSS

Third transmission = 8 MSS

Now, window size increases linearly,

Forth transmission = 9 MSS

Fifth transmission = 10 MSS

According to question, time out occurs here.

So, again slow start begins with a window size of 2 MSS and new threshold = half of previous threshold

New threshold = 8/2 = 4 MSS

Upto 4 MSS, increment is exponential,

Sixth transmission = 2 MSS

Seventh transmission = 4 MSS

Now increment is linear.

Eighth transmission = 5 MSS

Ninth transmission = 6 MSS

Tenth transmission = 7 MSS

***Que 7>*** Which of the following statements are TRUE?

(S1) TCP handles both congestion and flow control

(S2) UDP handles congestion but not flow control

(S3) Fast retransmit deals with congestion but not flow control

(S4) Slow start mechanism deals with both congestion and flow control.

***Ans>*** S1 and S3 are TRUE.

Explanation :

**1:** TCP window=>flow control and congestion window=> Congestion control  
**2:** No field are there in UDP header to control flow or congestion  
**3:** It is used by TCP to overcome the problem of out of order segments by re-transmission  
**4:** Slow start, nothing to do with flow control

***Que 8>*** List the fields in the TCP header that are missing from UDP header.

***Ans>*** The sequence number, the acknowledge number, and the Window fields.

***Que 9>*** Why is UDP needed?

***Ans>*** UDP is needed Because there are applications that do not need, or even want, the in-order, guaranteed notification of failure, byte-stream semantics of TCP.

It is also needed for:  
**1.** Minimal amount of header informations for every packet  
**2.** Efficient packet transmission  
**3.** Minimal system call usage for operations