CS 341 Quiz 2 Solutions

FULL MARKS 20

1. **Consider the following: A user program sends 6000 bytes of data through a TCP socket with the call “write (sid, buf, 6000)”. The following parameters are defined: MTU is 1500 bytes; Maximum IP datagram size (data part) is 4KB. The source and destinations are on the same Ethernet segment. Assume that the receive window size is large enough to receive all these bytes of information.**
Making reasonable assumptions about unspecified fields, show
i) the TCP segments sent
ii) the IP datagrams sent
iii) the MAC frames sent (including any address resolution required).

Marks: 8 (2+4+2)

Ans: (**Note**: *Before going through the solution, first read the following IP header fields which are important for fragmentation: Identification, Flags (DF and MF), and Fragment offset. Wikipedia link:* https://en.wikipedia.org/wiki/IPv4)

As the given maximum IP payload (excluding IP header) size is 4000 bytes, TCP cannot send one segment (IP payload) to IP layer of 6000 bytes user data. It needs to create two IP packets with of 4000 bytes IP payload (20 bytes TCP header + 3980 bytes user data), and 2040 bytes IP payloads (20 bytes TCP header + 2020 bytes user data). The sequence numbers of TCP headers for the two segments will be 0 and 3980. (**Note**: *No other way of dividing. TCP is greedy which always tries to send as much as data as early as possible, and transport layer does not know the MTU of data link layer*.)

As MTU is of 1500 bytes, the both the IP packet needs to be fragmented in the network layer

For the 1st IP packet of 4000 bytes IP payload:
3 fragments to be created all with packet identification number X. 1st fragment will have 20 bytes IP header + 1480 bytes IP payload. Header will contain MF flag as 1, DF flag as 0, and fragment offset as 0. 2nd fragment will have 20 bytes IP header + 1480 bytes IP payload. Header will contain MF flag as 1, DF flag as 0, and fragment offset as 1480/8 = 185. The 3rd fragment will have 20 bytes IP header + 1040 bytes IP payload. Header will contain MF flag as 0, DF flag as 0, and fragment offset as (2*1480)/8 = 370.

For the 2nd IP packet of 2040 bytes IP payload:
2 fragments to be created all with packet identification number Y!=X (usually X+1). 1st fragment will have 20 bytes IP header + 1480 bytes IP payload. Header will contain MF flag as 1, DF flag as 0, and fragment offset as 0. 2nd fragment will have 20 bytes IP header + 560 bytes IP payload. Header will contain MF flag as 0, DF flag as 0, and fragment offset as 1480/8 = 185.

MAC frames will have same header values for all the 5 IP fragment packets

2. **Assume a TCP extension that allows window sizes much larger than 64 kB. If this extended TCP is used over a 1-Gbps link with a latency of 100 ms to transfer a 10-MB file, and the TCP receive window is 1 MB. TCP sends 1-kB packets (assume no congestion and packet loss).**

(a) How many RTTs required for slow start to reach to send window size to 1 MB?

(b) How many RTTs required to send the file?

(c) If the time to send the file is given by the number of required RTTs multiplied by the link latency, what is the effective throughput for the transfer? What percentage of the link band-width is utilized?

Marks: 5 (1+2+2)

Ans:

(a) In slow start, the size of the window doubles every RTT. At the end of the ith RTT, the window size is 2^i KB. It will take 10 RTTs before the send window has reached 2^{10} KB = 1 MB. (the 11th send uses that 1 MB window to send 1 MB data)

(b) After 9 RTTs, $2^0 + 2^1 + \ldots + 2^9 = 1023$ KB = 1 MB − 1 KB has been transferred, and the window size is now 1 MB. As the receiver window size is 1MB, from 10th RTT onward TCP can send only 1 MB data in each RTT. Therefore, after 9+10 = 19 RTTs, TCP sends 10 MB – 1 KB data. The last 1 KB is sent in the 20th RTT. So total 20 RTTs are required to send the file.

(c) Given link latency is 100 ms or 0.1 s. Therefore, 1 RTT takes 2 * 0.1 = 0.2 s.
Accordingly, it takes 4 s = 20 RTTs * 0.2 s to send the file. The effective throughput is (10 / 4) = 2.5 Mbps = 20 Mbps.
This is only 2% utilization of the available link bandwidth.

3.
i) A computer on a 20Mbps network is regulated by a token bucket. The token bucket is filled at a rate of 2Mbps. It is initially filled to capacity with 12 Megabits. How long the can the computer transmit at the full 20 Mbps?
ii) What can be done to restrict the maximum transmit rate to 10 Mbps?

Marks: 2 (1.5+0.5)

Ans:
(i) Let the duration for which 20 Mbps rate is possible is t.
20t = 2t + 12
18t =12
t= 12/18 = 0.66 s

ii) Reduce the token bucket capacity to 8 Mb. If data is constantly being transmitted, bucket will have some space for incoming tokens. As token inflow rate is 2 Mb, in any situation, the token bucket will not allow more than 10 Mb to be transmitted in one second, resulting the maximum transfer rate as 10 Mbps.

4

(a) If an Autonomous system already has OSPF as the intra-AS routing protocol, then what is the purpose of iBGP?

(b) What is count-to-infinity problem in distance vector protocols and what are the available solutions for the problem? Do they solve the problem completely?

Marks: 5 (2+3)

Ans:

(a) OSPF maintains the reachability information of the intra AS routers but it does not know the next-hop information for the destinations outside of AS. iBGP informs internal routers about the border routers(within AS) for the reachabilities of external destinations. To route packets from one internal router to any of the border router (within AS) OSPF is used.

(b) **Note**: *Read from online resources about count to infinity problem of Distance vector routing. One link is given here: https://www.geeksforgeeks.org/route-poisoning-and-count-to-infinity-problem-in-routing/*
The answer provided here is just a hint and incomplete.

The main issue with **D**istance Vector **R**outing (DVR) protocols is Routing Loops since Bellman-Ford Algorithm cannot prevent loops. This routing loop in the DVR network causes the Count to Infinity Problem. Routing loops usually occur when an interface goes down or two routers send updates at the same time.

Reverse poisoning:
if z routes through y to get to destination x, then z will advertise to y that its distance to x is infinity, that is, z will advertise to y that $D_z(x) = \infty$ (even though z knows $D_z(x) = 5$ in truth). z will continue telling this little white lie to y as long as it routes to x via y. Since y believes that z has no path to x, y will never attempt to route to x via z, as long as z continues to route to x via y (and lies about doing so).

Split Horizon:
But according to the Split horizon Rule, Node A does not advertise its route for C (namely A to B to C) back to B. On the surface, this seems redundant since B will never route via node A because the route costs more than the direct route from B to C.

No, these solutions cannot solve the Count to Infinity Problem if the routing loop is with more than two nodes.