Workshop – 2 Submission

Answer – 1:

The basic difference between the UDP and TCP is that TCP is connection oriented whereas UDP is not. UDP in the case is actually a thin layer that is present over IP. For the transmission of UDP datagrams the fields such as 16-bit client side packet gets originated, 16 destination port, 16 UDP number of bytes, 16 bit UDP checksum needs to be filled by the programmer or the computer which in turn forwards the data together with the header information. The packets are also needed to be send with the IP address of destination and port because it is not a connection oriented and there might be a possibility that the data might be lost during the transmission. UDP is also unreliable transmission protocol. The receiver side notifies that a particular computer with certain address and port has to send a packet, along with size specification which can also be read by the user. Since UDP is connection less every detail has to be coded, the data packets have to be manually calculated and broken down in order to send that information (Kuros and Ross, pp. 258) [1].

Answer – 2:

HTTP 1.1 has the host header while the HTTP 1.0 does not have a host header, but if we want to, we can add it.

The main benefit of the host header is that it allows the client to route a message throughout the proxy server, and the major difference between 1.0 and 1.1 versions HTTP are:

- HTTP 1.1 has persistent connections so that we can have more than one request/response on the same connection.
- In HTTP 1.0 for each request and response, you need to open a new connection.
- HTTP 1.0 has a pragma while in HTTP 1.1 has Cache-Control, which is similar pragma.

The well-known transport protocol of the IP suite is the TCP. It is widely used for connection-oriented transmissions, whereas the connectionless UDP is used for simpler messaging transmissions. TCP is the complex protocol, due to its stateful design incorporating steadfast transmission and data stream services. Together, TCP and UDP consist of basically all traffic on the internet and are the only protocols executed in every major OS. More, transport layer protocols that have been specified and executed include the DCCP and the SCTP.

The recently released HTTP 1.1 was intended to focus on this problem by promoting multiple transfers of objects over one single connection. Spontaneously, changes in Web content are expected to decline the number of embedded objects, which will enhance network performance (Kuros and Ross, pp. 104 - 105) [1].

.Answer – 3:

Basically a TCP sender is supposed to monitor the transmission rate and based on the rate needs to take the decision. It will try to work together with other routers to obtain the data. For this also, they will share a common resource also. Under TCP Reno, the congestion window is set to cwnd = 6 • MSS and then grows linearly (Kuros and Ross, pp. 277)[1].

For the exploitation, it is actually done by using more number of TCP streams so that there is a gain and getting availability and exploiting internet usage. Also, there is one such algorithm named as Additive increase and multiple decrease which has been used in situation where the sending computer increases the input indefinitely until it increases its maximum window size (Kuros and Ross, pp. 278-279)[1].

Answer – 4:

The use of an easy buffer to keep the sequence number of the packet and the time-stamp it was sent at, will fix the problem in execution. Keeping the time-stamp of all the packets that are yet ACKed, eventually help in getting the timeout for the individual packet. The buffer will contain all the packets yet to be ACKed and their corresponding SENT time stamp, if the packet is ACKed then it will be deleted from this buffer and if a new packet is SENT then that packet is added to this buffer (Kuros and Ross, pp. 248)[1].

Now If we re-start the timer every time a packet is sent, the only way to know, how much time the packet has waited for ACKed, is to find the time-stamp the latest SENT packet, and with that along with the desired packet sent time and timer's time, we can find the waiting time for the desired packet(Kuros and Ross, pp. 271, 272)[1].

If we re-start the timer every time a packet is ACKed, then we need to store the time of the last packed ACKed, with that, the desired packet's sent time and timer's time, we can get the waiting time for the desired packet. We need to store the extra variable here that is the time of the last packet ACKed (Kuros and Ross, pp. 271)[1].

Answer – 5:

The following topics, I think are most valuable and really helpful if we could review it in the workshop session.

- TCP Reno and TCP Tahoe Algorithms
- Major benefits of HTTP 1.1 over HTTP 1.0
- The concept of selective repeat for the timers, it was eventually a lot harder to find because very less information I was able to get from the book.
- A quick recap to Wireshark DNS Quiz

References:

[1] Kurose, James F., and Keith W. Ross. *Computer networking: a top-down approach*. Addison Wesley.
[2] https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol
[3] https://en.wikipedia.org/wiki/TCP_congestion_control