Problem 1

A timer is a useful component in various protocol designs: because a communicating end cannot see what is going on either inside the network or at the other end, when needed it sets up an "alarm", and takes some action when the alarm goes off.

(a) Does HTTP use any timers? If so, please briefly describe how each is used. If not, please explain why it does not need one.

(b) Does DNS use any timers? If so, please briefly describe how each is used. If not, please explain why it does not need one.

(c) Does TCP use any timer? If so, please briefly describe how each is used. If not, please explain why it does not need one.

(d) Does UDP use any timer? If so, please briefly describe how each is used. If not, please explain why it does not need one.

Write your solution to Problem 1 in this box

A) No, HTTP is part of application layer. The reason transport layer needs a timer is because transport layer only occurs at each end of the network and needs to know when to do something on one side regardless if the other side has transmitted an instruction or not. In application layer, the host at one end only worries about the information it is given and nothing else.

B) DNS also does not have any timers. The only protocol I know with timers is TCP, but DNS uses UDP. DNS must be fast and instantaneous, so creating a connection with a handshake (TCP) is too slow and can't be done many times in succession. DNS does include a time to live timer that deals with the cache and expiration, so I guess that does count as a timer, but the translation and propagation part of DNS has no timer.

C) TCP has timers to provide RDT. Things such as the retransmission timer, timeout clock, and others all exist to help packets be delivered reliably and to redeliver packets when errors and drops happen. We even had calculations for EstRTT, DevRTT, and timeout when dealing with TCP.

D) UDP itself has no timers because it is merely a best-fit transfer mechanism that has 0 guarantees and is literally just a pipe for data.

Problem 2

TCP is a very symmetric protocol, but the client/server model is not. Consider an asymmetric TCP-like protocol in which only the server side is assigned a port number visible to the application layers. Client-side sockets would simply be abstractions that can be connected to server ports. Can you propose header data and connection semantics to support this. What will you use to replace the client port number?

Write your solution to Problem 2 in this box

If only the server side has port numbers visible to the application layer, then the TCP header would have to include lots of extra information. For example, the client side has to be able to demultiplex incoming traffic. The only way this can happen is if the TCP header includes a value to replace source port # to differentiate it between other machines that are talking to the server. Since we are replacing a number, the overall structure of the TCP header could stay the same with src port still being 16 bits, but algorithms for demultiplexing and mux would have to change. Making clients not have source port number means we can't demux normally. This means we have to replace it with a specific number.

Problem 3

Suppose that instead of a multiplicative decrease, TCP decreased the window size (upon timeout or duplicate ACK) by a constant amount. Would the resulting AIAD algorithm converge to an equal share (fair) algorithm?

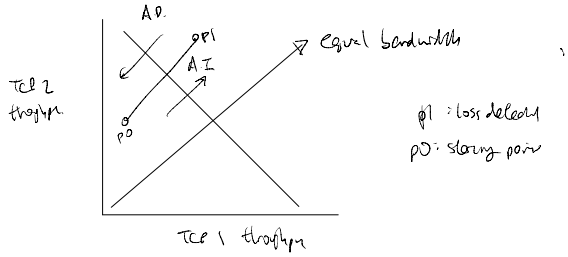
Write your solution to Problem 3 in this box

No. no convergence to fair algorithm

UNLESS

initial start is equal bandwidth sharing btwn both connections

→ increase linearly + decrease linearly means oscillation around same line



Problem 4

Consider a router that interconnects three subnets: Subnet 1, Subnet 2, and Subnet 3. Suppose all of the interfaces in each of these three subnets are required to have the prefix 223.1.17/24. Also suppose that Subnet 1 is required to support at least 60 interfaces, Subnet 2 is to support at least 90 interfaces, and Subnet 3 is to support at least 12 interfaces. Provide three subnet addresses (of the form a.b.c.d/x) that satisfy the constraints. You may use the following link to help verify your result: `http://jodies.de/ipcalc`.

Write your solution to Problem 4 in this box

Subnet 1: 60 interface: $2^6 = 64 \rightarrow 32 - 6 = 26 = x$

223.1.17.128/26 → (129 – 190) = 62.

Subnet 2: 90 interface: $2^7 = 128 \rightarrow 32 - 7 = 25 = x$

223.1.17.0/25 → (1 – 126) = 126.

subnet 3: 12 interface: $2^4 = 16 \rightarrow 32 - 4 = 28 = x$

223.1.17.192/28 → (193 – 172 = 19.

Problem 5

Suppose two packets arrive to two different input ports of a router at exactly the same time. Also suppose there are no other packets anywhere in the router.

(a) Suppose the two packets are to be forwarded to two different output ports. Is it possible to forward the two packets through the switch fabric at the same time when the fabric uses a shared bus?

(b) Suppose the two packets are to be forwarded to two different output ports. Is it possible to forward the two packets through the switch fabric at the same time when the fabric uses switching via memory?

(c) Suppose the two packets are to be forwarded to two different output ports. Is it possible to forward the two packets through the switch fabric at the same time when the fabric uses a crossbar?

Write your solution to Problem 5 in this box

a) no. cant forward two packets through switch fabric simultaneously for a shared bus.

Bus is shared → only 1 packet can go through @ same time.

one packet will wait in queue.

b) not possible again. each packet must be copied to memory and then copied to output port buffer. However Read/write can be only 1 pkt @ a time.

c) yes. possible b/c 2 diff output ports and packets can be sent in parallel through switch fabric.