Problem 1

Consider a short link, over which a sender can transmit at a rate of 200 bits/sec in both directions (meanwhile the propagation delay is so small that we are omitting it in this question). Suppose that packets containing data are 100,000 bits long, and packets containing only control (e.g., ACK or hand-shaking) are 200 bits long. Assume that N parallel connections each get 1/N of the link bandwidth. Now consider the HTTP protocol, and suppose that each downloaded object is 100 Kbits long, and that the initial downloaded object contains 10 referenced objects from the same sender. Would parallel downloads via parallel instances of non-persistent HTTP make sense in this case? Now consider persistent HTTP. Do you expect significant gains over the non-persistent case? Justify and explain your answer.

Clarification: For simplicity, we work on a simplified model instead of actual HTTP and TCP. You can assume the connection establishment contain a pair of packets of 200 bits each. After the connection establishment, we only consider the request and response, with the size of 200 and 100000 bits respectively. Other traffic (e.g. connection teardown) is omitted in this question.

Write your solution to Problem 1 in this box

Non-persistant

(1 initial object + 10 referenced objects) · 2 packet/connection · 200 bits/packet / 200 bits/sec =

22 · 1 = 22 sec for connection

22/2 = 11 sec for request

11 objects · 100,000 bits/object / 200 bits/sec = 5500 sec for data transfer

total time = 5500 + 22 + 11 = 5533 sec

Persistant

2 packets/connection · 200/200 = 2 sec for connection

11 packet/connection · 200/200 = 11 sec for request

same data transfer time, only connection time changes

total time = 5500 + 11 + 2 = 5513 sec

Parallel downloads wouldn't make much sense since the bottleneck for this is relevant on the connection time, not the data transfer which we've shown to be the same.

Persistant does create a significant gain of about 20 seconds.

Problem 2

(a) Why do most HTTP versions (except recent HTTP/3), SMTP (used to deliver emails), and SSH run on top of TCP rather than on UDP?

(b) Suppose you wanted to do a transaction from a remote client to a server as fast as possible. Would you use UDP or TCP? Why?

(c) Suppose that the UDP receiver computes the Internet checksum for the received UDP segment and finds that it matches the value carried in the checksum field. Can the receiver be absolutely certain that no bit errors have occurred? Explain.

Write your solution to Problem 2 in this box

a) It is more reliable — really important in the communication on HTTP, SMTP and SSH.

b) UDP — it allows far faster, although less reliable, transactions because it contains less packet header information and its a connectionless protocol. Therefore, it is lightweight in computation and size.

c) NO, it is possible for errors to occur where the checksum is valid but unnoticable by only checking the checksum. This is a collision and thus can't guarantee no angle bit errors

Problem 3

Consider the DNS query procedure.

(a) What are the root DNS servers used for?

(b) If a local DNS server needs to query root DNS servers, how does it know their IP addresses?

(c) If you go to `https://www.iana.org/domains/root/servers`, you will find that there are only 13 root server addresses. Will this be a problem? Why or why not?

Write your solution to Problem 3 in this box

a) Root DNS resolve TLD nameservers — it forms the basis for DNS and are the first step in translating DNS domain names into IP addresses when a local DNS server cannot resolve a domain name in the cache.

b) When a local DNS is setup, it receives a list of root server IP address of which it can parse through and add to the cache

c) No, it is just a representation of the root DNS infrastructure. The existence of only 13 root server addresses just simplifies configuration and management.

Problem 4

Suppose your computer is connected to a WiFi network, which gives you the IP address of the local DNS server. It was just rebooted and its cache is completely empty. RTT between your computer and the local DNS server is 5ms while the RTT between the local DNS server and *any* other DNS server is 60ms. Assume the iterated query is used and all responses have TTL of 5 hours. Also suppose that you are the only user of the local DNS server in the duration of this problem and you made no web requests other than what's mentioned in the problem.

(a) If you try to visit www.cs.ucla.edu, what would be the amount of time that you need to wait before the web browser is able to initiate connection to the web server of UCLA CS?

(b) If you try to visit bruinlearn.ucla.edu one minute later, what would be the waiting time?

(c) If you try to visit gradescope.com one minute later, what would be the waiting time?

(d) If you try to visit google.com one minute later, what would be the waiting time?

Write your solution to Problem 4 in this box

a) 5ms + 60ms + 60ms + 60ms + 60ms = 245ms

connection; ucla domain req/res; cs.ucla domain req/res

b) 5ms + 60ms = 65ms

connection; route to bruinlearn

c) 5ms + 60ms + 60ms = 125ms

connection + connection w/ gradescope

d) 5 + 60 = 65ms

Problem 5

(a) One process in Host A uses a UDP socket with port number 8888. Two other hosts X and Y each send a UDP segment to Host A. Both segments specify destination port number 8888. At host A, will both segments be forwarded to the same socket? If so, can the process at Host A know that these two segments are from two different hosts, and how? If not, would that cause any problem for the process? Discuss and explain.

(b) Consider an application that runs over UDP. Can it still achieve reliable data transfer? If so, how?

(c) Present two application scenarios where UDP is preferred over TCP. Explain why.

Write your solution to Problem 5 in this box

a) They both have the same destination port, so they will be forwarded to the same socket. Host A can inspect the source IP of the packets to determine where they came from to identify that they came from 2 separate hosts

b) Yes, you can implement acknowledgements, timeouts, error and flow control, and other TCP-like processes to achieve reliable data transfer.

c)

Streaming — low latency is preferable for user experience. if a packet is lost, the host can simply retrieve the packet later and create a small jitter over better delivery.

DNS — UDP allows for faster responses. Even if packet is lost, UDP can re-retrieve and retransmit the request w/o significant issues on the end-host.

Problem 6

Answer True or False to the following questions and briefly justify your answer:

(a) With the Selective Repeat protocol, it is possible for the sender to receive an ACK for a packet that falls outside of its current window.

(b) With Go-Back-N, it is possible for the sender to receive an ACK for a packet that falls outside of its current window.

(c) The Stop&Wait protocol is the same as the SR protocol with a sender and receiver window size of 1.

(d) Selective Repeat can buffer out-of-order-delivered packets, while GBN cannot. Therefore, SR saves network communication cost (by transmitting less) at the cost of additional memory.

Write your solution to Problem 6 in this box

a) False — if ACK falls outside of it's current window, the sender has already moved its window forward, meaning the ACK is outdated or duplicated.

b) False — the sender expects cumulative ACK's when the window is forwarded, so if an ACK is sent outside the window the sender won't receive it because it doesn't receive a cumulative ACK that confirms the sending of the ACK.

c) True — sender can send a packet and receive an ACK one packet at a time.

d) True — SR can buffer out-of order delivered packets ⇒ allows selective retransmission of corrupted/missing packets. GBN requires for all unacknowledged packets from earliest detection. Thus SR saves communication cost at expense of additional memory