CMSC 417 Computer Networks **Fall 2020**

Second Third-Term Exam

Open book and notes; Take home *Tuesday, Nov. 17th*

⊕ *Do not forget to write your name on the first page. Initial each subsequent page.*
⊕ *Be* **neat** *and* **precise. I will not grade answers I cannot read.**
⊕ *You should draw simple figures if you think it will make your answers clearer.*
⊕ *Good luck and remember, brevity is the soul of wit*

- All problems are mandatory
- I cannot stress this point enough: **Be precise**. If you have written something incorrect along with the correct answer, you should **not** expect to get all the points. I will grade based upon what you **wrote**, not what you **meant**.
- Maximum possible points: 50 + bonus.

Name: Dazhi Peng

Problem	Points
1	
2	
3	
4	
5	
Total	

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	2	4	8	16	32	64	128	256	512	1024	2048	4096	8192	16384	32768	65536

D.P

1. Nomenclature

(a) Describe the following terms: (2 points each)

- Sender Policy Framework (SPF)

An email authentication method designed to detect forging sender addresses during the delivery of the email.

~~For example, if an email is claimed to come from "cs.umd.edu", SPF performs a reverse DNS lookup on the IP address that the mail is coming from; check whether that IP address is authenticated by "cs.umd.edu."~~

Achieved by using DNS; in DNS, there's a TXT record for the domain which specifies which IP addresses are allowed to send messages on behalf of that domain. We go through that TXT and verifies whether the sender IP address is in that TXT.

- Negative Ack

Stands for negative acknowledgement. Designed for the receiver to tell the sender which data is missing.

For example, if LFR=5 & RWS=f_0 & LAF=9, if frame 7 arrived, the receiver could send a NAK for frame 6 to tell the sender frame 6 is missing.

- Distributed Hash Table

Unlike a normal hash table which stores all data on a local data structure; the distributed hash table distributed data across different nodes in a network. Each node can efficiently retrieve the value associated with a given key. Benefits include decentralization, fault tolerance, and scalibitiy. A example would be Chord.

- Resolver

The DNS resolver acts like a "client interface" in the DNS process. i.e. providing a "getHostByName" function for clients. When a client contacts a resolver, the resolver contacts the nameservers, which actually hold the mapping information, in order to get the necessary IP address/domain names. (for reverse)

- TCP Fast Recovery

Fast Recovery governs the transmission of data until a non-duplicate ACK arrives (after fast retransmit sends what appears to be the missing segment.) It works as follows:

ss_thresh = cwind/2

cwind = ss_thresh +3.

If dup ACK arrives, cwind +=1 (Retransmit a segment if we can)

If new ACK arrives, cwind = ss_thresh, begun congestion avoidance.

Reason: since we know ACKs are flowing, fast recovery help us avoid go back to slow start thus increase efficiency.

D.P

2. Reliable Transfer/UDP

(a) What are the send and receive window sizes in the "Alternating Bit Protocol"? (2 points)

Both = 1.

(b) What service(s), beyond checksumming, does UDP provide over IP? (2 points)

Demultiplexing. Because UDP packet has a source and destination port, for a given IP address, it can deliver the datagram to whichever process is sitting on the destination port.

(c) What is the maximum end-to-end throughput you could achieve on a 100 Mbps, 250ms RTT link, with send window-size $\leq$ 40 maximum-sized segments, segment size $\leq$ 125 bytes. Show your work. (3 points)

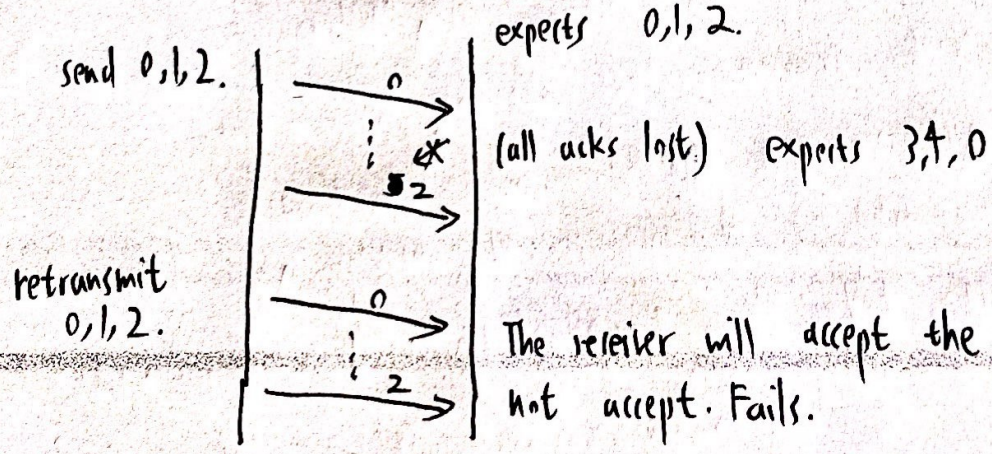
$1000 \div 250 = 4$ RTT

100 Mbps $= 10^8$ bps

$40 \times 125 \times 8 \times 4 = 160000$ bits

Since $160000 < 10^8$, the maximum end-to-end throughput is 160000 bits per second.

(d) Give an example, using a single waterfall diagram, where a sliding window transfer protocol that uses 5 sequence numbers fails when RWS = 3, SWS= 3. Explain your assumptions. (3 points)

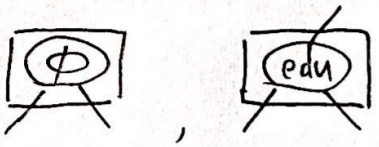


D.P

3. DNS/Application-Layer

(a) What is a DNS "zone"? (2 points)

It's a subtree in the DNS the structure that is autonomously administered. And each zone must have two NSs with separate power supplies.

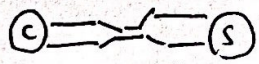
ex. 

(b) You administer the `umd.edu.` domain, and want to delegate the `a.umd.edu.` subdomain. Explain what records you would add to your zone to enable this. (2 points)

We need two pieces of information: a.umd.edu.'s nameserver and the nameserver's IP address. To achieve that we use glue record to prevent circular dependency.

ex.
```
a.umd.edu.  NS  ns.a.umd.edu.
ns.a.umd.edu.  A  128.8.1.4
```

(c) Describe the deficiencies with opening concurrent TCP connections to download HTTP resources when the network is congested. (3 points)

In a congested network, there's a bottleneck in the network: 

These concurrent connections will simultaneously hit the bottleneck of network. TCP is not able to adapt at all because all these connections are independently trying to figure out their window size. At the same time, these connections are also doing slow start and exponentially hit the network at the same time.
thus
all

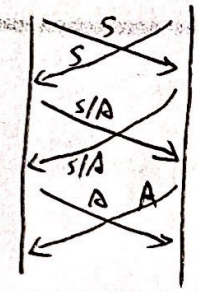
(d) Describe two uses of the "*domain*" argument of the HELO message in SMTP. (3 points)

(1) The receiving server can perform a reverse DNS lookup on the IP-address that the mail is coming from and see if it maps to the "domain".

(2) Perform SPF; in DNS, there's a TXT record for the "domain", which specifies which IP addresses are allowed to send email on behalf of "domain". We go through that TXT and checks whether the sender IP is in that TXT.

4. TCP

(a) What sequence of segments lead to a TCP "simultaneous open" (You may use a waterfall diagram to show the sequence.)? When can either end send data? (2 + 1 points)



Both can start sending data after receiving the ACK of (SYN & ACK).

(b) (Why) would a TCP Window Scale value of 16 be incorrect? (2 points)

Since the window is 16 bits, the maximum allowable window scale value will be 15 to make sure the sender window size (31) cannot be larger than half of the sequence space (32).

(c) (Why) is the the TCP `TIME WAIT` required? Are there cases when both TCP end-points enter `TIME-WAIT`? (2+1 points)

Because even if the old connection has been closed; if we immediately establish new connection using the 5-tuple, some messages from the old connection may leak in. (same)

Yes, the rare will be simultaneous (FIN crossed) close, in which both end-points goes from:
ESTABLISHED -> FIN_WAIT1 -> CLOSING -> TIME_WAIT.

(d) How/why is TCP able to guarantee reliable byte stream delivery even if segments are/may be reordered by IP? (2 points)

~~Reliable: Uses sliding window.~~

By not repeating a sequence number within a MSL; therefore the old packet either show up or doesn't, it can't show up infinite late. other out of order (cases) will be taken care by the sliding window.

D.P

5. Applications/Design

(a) What is a PTR record in DNS? Explain with an example. (2 points)

A record that maps an IP address to a domain name.

Ex. Suppose you want to find the domain of 128.8.126.63

You do a query 63.126.8.128.in-addr.arpa with type PTR.

(b) How is stream multiplexing in QUIC different from HTTP/1.1? (3 points)

Unlike TCP which may has Head of Line Problem (because TCP is providing a reliable byte stream); QUIC still multiplex inside one connection but sending them independently by using the crypto parameters from the connection setup. Thus, the multiplexing is independent of the transport byte stream.

(c) The function `handle_clients` takes `fds`, an array of non-blocking client file descriptors, and `nfds`, the number of descriptors passed in. Without blocking on a single client, it then reads `EXPECTED` bytes from all passed in clients and dispatches some callback when all expected bytes are read. Identify as many errors as you can with the implementation on the following page. You may consult the man pages. (5 points)

Line 2: should be calloc(sizeof(struct pollfd), nfds)

Line 13: C does not have a bool type by default.

~~Line 12: This for loop will end early and not advance to the Line 20. Since recv is for TCP, the server s~~

Line 33: Should free pfds after the while loop.

Line 12: This for-loop implementation will not get the right file descriptor with POLLIN; instead, it should check all file descriptors and see whether each one has POLLIN.

```
1  void handle_clients(int *fds, nfds_t nfds) {
2    struct pollfd *pfds = calloc(sizeof(pfds), nfds);
3    for(nfds i = 0; i < nfds; ++i) {
4      pfds[i].events = POLLIN;
5      pfds[i].fd = fds[i];
6    }
7
8    int finished = 0;
9
10   do {
11     int ready = poll(pfds, nfds, 10);
12     for(int i = 0; i < ready; ++i) {
13       bool cleanup_fd = false;
14       cleanup_fd = pfds[i].revents == POLLIN ||
15                    pfds[i].revents == POLLHUP;
16       if (pfds[i].revents == POLLIN) {
17         char buffer[EXPECTED];
18         size_t n = 0;
19         do {
20           ssize_t rv = recv(pfds[i].fd, buffer + n, EXPECTED - n, 0);
21           n += rv;
22         } while(n <= EXPECTED);
23         callback(buffer);
24       }
25
26       if (cleanup_fd) {
27         ++finished;
28         close(pfds[i].fd);
29         pfds[i].events = 0;
30         pfds[i].fd = -1; /* ignore this entry */
31       }
32     }
33   } while(finished < nfds);
34 }
```

(revents only return POLLHUP)
(ignore event & revent returns 0)