Wenxuan Liu 805152602

Problem 1.

1. ACE
2. AB
3. BCD
4. AD
5. BDCE
6. DE
7. AD
8. BE
9. AD
10. ABDE
11. D
12. CE
13. ACE
14. A
15. BE

Wenxuan Liu 805 152 602

Problem 2.

1. For iterated query mode. the local DNS Server is the resolver. User first sends a query to the local DNS server, then local DNS Server sends the query to the root DNS Server. and ~~DNS~~ get the answers of which TLD DNS Server to ask. Then local DNS Server ~~ask~~ sends query to TLD DNS server, and then Local DNS Server sends query ~~to~~ ~~the~~ ~~asks~~ authoritative DNS Server, and hence get the IP of the hostname in the query.

In The recursive query, the local DNS Server first sends query to the root DNS Server, and the root DNS Server is the resolver, sending DNS query to authoritive DNS server and recursively get IP, and the root DNS sends back to the Local DNS Server.

2. First, users use DNS to get the IP of v-store server, and v-store server and DNS uses the request from client to direct user to the appropriate CDN Server provided by Lime light, instead of v-store's subdomain. ~~which~~ It reduces the burden of v-store server, and ~~users to~~ ~~users~~ uses DNS to direct users to the nearest ~~CDN~~ CDN. ~~and diverse users across the world.~~ requests

Problem 2. Wenxuan Liu 805152602

3. Because Only Email needs Username and Password to verify the user in the certain mail server (For POP and IMAP). And SMTP is using push while HTML uses pull, so SMTP needs verify the authorization ~~po~~ before pushing data ~~to the se~~ from server to server. HTTP also is stateless and all user information is in the backend database, so no ~~nee~~ need ~~to~~ for ~~of~~ authorization

4. For 1st packet: ~~0~~ 0 for delay

2nd packet : $\frac{L}{R}$

3rd packet : $\frac{2L}{R}$

⋮

nth packet : $\frac{(n-1)L}{R}$ So the last packet needs $\frac{(n-1)L}{R}$ for the queueing delay.

5. Peer-to-peer is better. First, because P2P is scalable, so when the user number is becoming large, the ~~to~~ rate for data communication is always equal or faster than client-to-server, since the ~~upload~~ server doesn't have to upload N copies of file to all users. Second, Client-server must have server be always on, so they cannot run their app on the on and off computer for Client-Server.

Wenxuan Liu 805152602

Problem 4.

500 bytes = 500×8 bits = 4000 bits.

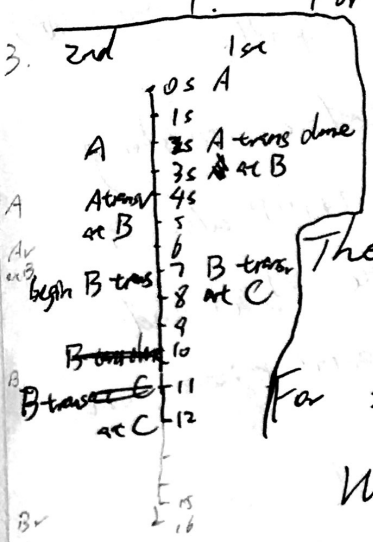
1. For first packet:

Delay for all packets Transmission From A to B : $\frac{4000 \text{ bits}}{2\times10^6 \text{ bits/s}} = 0.002\text{ s}$

From B to C : $\frac{4000 \text{ bits}}{1\times10^6 \text{ bits/s}} = 0.004\text{ s}$

The done time for first is $0.0025 + 0.0045 + 2\times0.0015 = 0.008\text{s}$.

3.



For second packet: ~~begin~~ queue delay is 0.002 s. at A

When arrives at B at $0.002\text{ s} + 0.0025 + 0.0015\text{ s} = 0.005\text{ s}$.

first packet is still transmitting (done at 0.007s) so ~~delay~~ queue at B is 0.002s. So for 2 packet, the done time is $0.0025 + 0.0045 + 2\times0.0015\text{ s} + \underset{\text{(queue)}}{2\times0.002\text{s}} = 0.012\text{s}$.

So the difference between 1 and 2 is

$0.012\text{s} - 0.008\text{s} = \underline{0.004\text{s}}$

2. For third packet queuse delay at A is 0.004s.

arrives at B at $0.004\text{s} + 0.0015 + 0.002\text{s} = 0.007\text{s}$,

when 2 packet just began transmitting. so queue at B is 0.004s. In all, the time for 3rd packet is

$$\underset{\text{to B}}{0.007\text{s}} + \underset{\text{queue}}{0.004\text{s}} + \underset{\text{prop}}{0.0015} + \underset{\text{trans}}{0.004\text{s}}$$

$= 0.016\text{ s}$.

So the time for C receive all 3 packet is $\underline{0.016\text{ s}}$.

Wenxuan Liu 835152602

Problem 5

1. He'll Use DNS, HTTP in application-layer

Use TCP (for HTTP) and UDP (for DNS) in transport layer

2. Client first sends query to local DNS server, and local DNS iteratively get the ~~that~~ IP of ~~good~~ google and CNN server ~~to~~.

Then Client use HTTP to send request (GET) to server base on TCP. ~~Server~~ Client and server first build connection, then server ~~sends~~ transmit data in HTTP Response to client, and client parse the HTML file and objects on to the browser.

3. Persistent without pipeline: $2RTT + RTT \times 20 = 22RTT = 22sec$

Non-persistent: $2RTT + 2RTT \times 20 = 42RTT = 42$ sec

Non-per with parallel: $2RTT + 2RTT \times (20/5) = 10RTT = 10sec$

Non-persistent with parallel is faster.

4. DNS, SMAP, POP3/IMAP UDP, TCP

Wenxuan Lin 805132602

Survey

1. ~~[illegible]~~ match

2. Average

3. Average

4. Yes

Problem 3 (8 points): Joe is writing programs with a TCP client and a TCP server using stream sockets. The following is the SERVER code that Joe wrote. Can you help Joe to find four errors (there can be more) in his code? You need to answer which line of code is wrong and the specific error. You can use the Appendix for references.

```
#include <server.h>  /* assume all headers are included correctly */
#define PORT 8080
int main(int argc, char const *argv[])
{
    int server_fd, new_fd; /* listen on server_fd, new connection on new_fd */
    struct sockaddr_in address;
    int addrlen = sizeof(address);
    char buffer[1024] = {0};
    char *hello = "Hello from server";

    if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) > 0) {
        perror("socket failed");
        exit(EXIT_FAILURE);
    }
    address.sin_family = PF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = htohs(PORT);

    if (bind(server_fd, (struct sockaddr *)&address, addrlen) < 0) {
        perror("bind failed");
        exit(EXIT_FAILURE);
    }
    if (accept(server_fd, 3) < 0) {
        exit(EXIT_FAILURE);
    }
    if ((new_fd = listen(server_fd, (struct sockaddr *)&address, (socklen_t*)&addrlen)) < 0) {
        exit(EXIT_FAILURE);
    }
    int valread = read(server_fd, buffer, 1024);
    printf("%s\n", buffer);
    sendto(new_fd, hello, strlen(hello), 0);
    printf("Hello message sent\n");
}
```

1 → should be "==". (>0 is not an error)

2 Type error, should be "htons".

3 accept and listen should be exchanged.

4 → should be "new_fd".