**Q1.) Email Messages**

SMTP is a push protocol, which relies on sender’s mail server to push the email to the recipient’s mail server. Transferring from the recipient’s mail server to their personal email is a pull operation.

**Q2.) Institutional Network**

Average object size = 900,000 bits.

Average request rate from institution’s browsers 🡪 origin server = 1.5 requests per second.

Internet Delay: Time from when router forwards HTTP request to receiving a response = 2 seconds.

Total response time = sum of average access delay + average internet delay

**How to do this qn wtf?**

**Q3.) DNS and UDP**

Why does DNS use UDP, instead of TCP for its query and response messages?

- TCP involves requires a 3-way handshake (SYN, SYN-ACK, ACK), while UDP does not.

- Therefore with TCP, it will be a lot more expensive / have a higher overhead in delay.

- This is acceptable with larger transfers, but DNS queries and responses are very small.  
(DNS requests are very small and fit well within UDP segments)

- UDP is better for a small packet size and imposes a smaller load on the name servers due to its simplicity.

**Q4.) Hotmail Account**

IMAP: is an email protocol that stores email messages on a mail server, but allows the end user to view / manipulate messages as though they were stored locally. This allows users to organise messages into folders, have multiple client applications know which messages have been read, flag messages for urgency or save draft messages on the server.

- Email goes from your computer to your mail server over HTTP.

- Your mail server sends your email/file to your friend’s mail server over SMTP.

- Your friend downloads the email/file from their mail server over IMAP.  
“https://en.wikipedia.org/wiki/Internet\_Message\_Access\_Protocol”

**Q5.) Instant Messaging**

In what way is instant messaging a hybrid of client-server and P2P architecture?

Client-Server: Web server is always up: send a message, connection is made, gives files that are requested.

P2P: Every host plays both the client AND server: Server will not always be on.

- The initiator contacts a centralised server to locate the IP address of the receiver (client-server model)

- After the connection is initiated, the two users may send messages to each other directly (p2p)

**Q6.)** **Multiple Choice Questions (see Open Learning)**

Out of Band Signalling: Out of band signalling is passing control data on a separate connection / channel to from main data.

**Q7.) Iterated DNS query**

How can iterated DNS queries improve overall performance?

Iterated request can improve overall performance by offloading the processing of requests from root and TLD servers to local servers. In recursive queries, root servers can be tied up ensuring the completion of numerous requests, which can results in a substantial decrease in performance. Iterated requests move that burden to local servers, and distributed the load more evenly throughout the Internet. With less work at the root servers, they can perform much faster.

**Q8.) Iterated DNS query**

Non-Persistent HTTP: At most one object sent over HTTP connection, then connection is closed.

Initiate connection 🡪 Send request 🡪 File transmission 🡪 Receive response.

With multiple objects on a webpage, you will need multiple TCP connections for each object.

Persistent HTTP: Server maintains TCP connection after sending a response. Subsequent files/objects sent between the same client / server are sent over the same TCP connection

Persistent without Pipelining: Client issues a new request only when previous response has been received.

Persistent with Pipelining: Default in HTTP/1.1. Client sends request as soon as it encounters a referenced object.

MAIN DIFF: One RTT for each referenced object vs. One RTT for ALL referenced objects.

**Q9.) Web Browser**

n DNS servers are visited before retrieves the IP address from the DNS. Each visit to DNS = RTT.

- Total time visiting DNS iteratively for n DNS servers = RTT1 + RTT2 + … + RTTn

After finding the IP address for the URL:

- One RTT for setting up the TCP connection

- One RTT for requesting and receiving the object

Total time from clicking on the link + receiving the object **= (2 \* RTT0) + (RTT1 + RTT2 + … + RTTn)**

**Q10.) Caching**

Total delay in downloading objects assuming that web-cache hit ratio = 40%

Assume cache hit ratio = 50% (50% requests are satisfied locally from the web cache).

If cache-hit ratio = 50% 🡪 Access link utilisation = 50%.

Total Delay = Internet Delay + Access Delay + LAN Delay

= Internet Delay + (0.5 \* delay from origin servers) + (0.5 \* delay from web cache)

= Internet Delay + (0.5 \* 2.01) + (0.5 \* ~milliseconds)

= 1 + 0.1 **= 1 to 1.1 seconds**

**Q11.) DHT**

Yes, search performance can be impacted if the assignment of keys to peers does not take into account network topology (note: Network Topology is the arrangement of various elements such as links, nodes of a communication network).

Search performance can be impacted because peers may be separated by a long distance so there could be large delays between successive nodes. Also, throughput may be low due to the underlying network topology having a low bandwidth, which means higher delay in search performance.

**Q12.) Circular DHT**

P15 = Designated contact peer for DHT

P6 requests to join.

- P6 will as P15 to join.

- P15’s successor is P1. Forward’s request to P1.  
- P1’s successor is P2. Forward’s request to P3.

- P3’s successor is P4. Forward’ request to P4.

- P4’s successor is P5. Forward’s request to P5.

- P5’s successor is P8, therefore makes P6 its successor.

- P6 makes P8 its successor.

**Q13.) BitTorrent (P2P choke / unchoke qn)**

Optimistic unchoke: A peer periodically selects one of its neighbours at random as a peer for uploading, whether or not this neighbour is uploading data or not. Alice may be chosen at random as a result of this, hence will receive her first chunk.

**Q14.) Minimum Distribution Time (P2P calculation qn)**

Distributing a file F = 10Gbits to N peers.

Server Upload rate i\_s = 20mbps

Per Peer download rate d\_i = 1mbps / upload rate u\_i = u

N = 10, 100, 1000 and u = 200kbps, 600kbps, 1mbps, give the minimum distribution time for each of the combinations of N and u for both client-server and P2P distribution.

Client Server

**Dc-s = max { N \* F / us, F / dmin }**

|  |  |  |  |
| --- | --- | --- | --- |
|  | n = 10 | n = 100 | n = 1000 |
| US = 20mbps | (10 \* 10Gb) / 20mbps | 10Gb\1mbps  { U = 100Gb / 20mbps |10 \* 1000mb)  { U = 5,000 | D = 10,000} **MAX = 10,000** | (100\*10Gb)/20mbps|10Gb/1mbps  { U = 1,000Gb/20mbps|10\*1000mb)}  { U = 50000 | 10000} **MAX = 50,000** | (1000\*10Gb/20mbps,10gb/1)  { U = 10,000/20mbps, 10000}  { U = 500,000/ 10,000 }  **MAX = 500,000** |
| US = 20mbps | **10,000** as server upload is the same | **50,000** as server upload is the same | **MAX = 500,000** |
| US = 20mbps | **10,000** as server upload | **50,000** as server upload is the same | **MAX = 500,000** |

P2P Distribution

**Dp2p = max { F/us , F/dmin , N\*F/(us + sum(ui) }**

|  |  |  |  |
| --- | --- | --- | --- |
|  | n = 10 | n = 100 | n = 1000 |
| U = 200 | 10,000 / 20 | 10,000 | 100Gb / (20mbps+(200kbps\*10))  = 500 | 10,000 | 100,000 / (22) = 500 | 10,000 | 4,545.45 **MAX = 10,000** | Calculating only the P2P part:  100 \* 10Gb / 20mbps + (200kbps\*100)  = 1,000,000mb / 20mbps + 20mbps  = 1,000,000 / 40mbps  = 25,000  **MAX = 25,000** | Calculating only P2P:  = 1000\*10Gb / 20mbps + (200kbps \* 1000)  = 10,000,000mb / 20mbps +  200mbps  = 10,000,000mb / 220mbps  = 45,454.54  **MAX = 45,454.54** |
| U = 600 | 500 | 10,000 | 100Gb / (20mbps + (600kbps\*10)  = 500 | 10,000 | 100,000 / (20 + 6)  = 500 | 10,000 | 3,846.15  **MAX = 10,000 { us + sum(ui) } is increasing** | Calculating only the P2P part:  1,000,000mb / 20mbps + (600kbps\*100)  = 1,000,000mb / 20mbps + 60mbps  = 1,000,000mb / 80mbps  = 12,500  **MAX = 12,500** | As above:  = 10,000,000mb / 20mbps + 600,000kbps = 10,000,000mb / 620mbps = 16,129  **MAX = 16,129** |
| U = 1mbps | 500 | 10,000 | 100Gb / (20mbps + (1mbps\*10))  = 500 | 10,000 | 100,000 / (20 + 10)  = 500 | 10,000 | 3,333,33  **MAX = 10,000 { us + sum(ui) } is increasing** | Calculating only the P2P part:  1,000,000mb / 20mbps + 1mbps \* 100  = 1,000,000mb / 120mbps  = 8,333.33  **MAX = 10,000** | As above:  = 10,000,000mb / 20mbps +  1,000mbps  = 10,000,000mb / 1020mbps  = 9,803.92  **MAX = 10,000** |