Assignment No. 1

Name : Shahed Ahmed

Student ID: 22299315

Course : CSE421

Section : 23

Ans: To the ques. no. 1

(a)

Transport layer is responsible for process-to-process delivery. Process-to-process delivery means delivering data to the correct process running on a host. The transport layer uses port numbers to demultiplex to the right application. Host-to-host delivery means moving packets from one host to another host. The network layer is responsible for end-to-end host addressing and routing, but it doesn't know about application processes or ports.

(b)

DNS resolves names to resource records of different types. To differentiate services we use different record types:

i) For web servers we typically use an A record for the host name. Example:

(google.com, 172.10.12.32, A)

ii) For mail delivery we use an MX record to name the value of the mail server. Example:

(google.com, mail.google.com, MX)

(c) (I)

As seen from the given table, the top 4 uploaders are B(100), E(15), F(14) and A(12). C is excluded because it is inactive. Even though the top four uploaders have great upload capacity, the set of pieces they have doesn't include at least one piece that exists only in D. Because all the pieces are available only on D, G cannot assemble the full file from B, E, F, A alone. Therefore, G cannot complete the whole download from that current top four set.

(c) (II)

If D's upload increases to 40 mbps, the top four uploaders are: B(100), D(40), E(15), F(14).

Sum of total upload speed (100+40+15+14)
= 169 mbps

Given, the file size is 890 ~~ma~~megabits.

∴ Computing time = $\frac{\text{File Size}}{\text{Total upload speed}}$

$= \frac{890}{169}$ s

$= 5.27$ s

(approx.)

d

There are several benefits that DASH provide to a user:

i) Adaptive bit-rate: Player can switch among multiple quality levels.

ii) Improved user experience: Smoother playback with fewer interruptions.

iii) HTTP friendly: Uses and gets regular benef-its of HTTP.

iv) Scalable: Supports multiple codecs.

v) Efficient use of network: Chooses optimal optimal chunk sizes and bitrates.

The manifest tells the player what representations are available, how the media is segmented, timing, codecs and adaptation sets. The player parses the manifest to know which URLs to

fetch for which quality and in which order, enabling adaptive switching during playback.

Ans: to the ques. no. 2

(a)

Cookies are stored per browser profile and are not shared between different browsers. Chrome's cookies are ~~Ape~~ separate from Firefox's cookies, so Firefox will not have the cookie entries Chrome stored. The primary reason in the scenario is, cookies are browser-specific. So, Chrome cookies are not available to Firefox.

(b)

A conditional ~~Get~~ GET, which is a GET request with conditional headers. The proxy includes the conditional headers when it contacts the origin. If the origin responds '304 not modified', no object body is sent. The proxy knows its cached copy is ~~stell~~ still fresh and can serve it immediately to the client without transferring the full object. This saves bandwidth and reduces latency, saving time in the process.

(c)(I)

As seen in the given scenario, there were a case of 4 RTTs. One is for the local DNS, the others in the process of iterative DNS lookup.

$$\therefore \text{Total RTT} = 4 \times RTT = (4 \times 19)\ ms$$
$$= 76\ ms.$$

(c)(II)

From the given scenario,

The TCP request starts a TCP handshake which would take 1 RTT.

$$1\ RTT = (2 \times 35)\ ms = 20\ ms$$

After the connection is started, client requests 21 objects and servers processes

and sends them. Each object costs:

$(5 + 125)$ ms

$= 130$ ms

∴ The total object transfer time is

$(70 + 21 \times 130)$ ms

$= 70 + 2730$ ms

$= 2800$ ms

(c) (III)

"From (c)(I)",

The DNS time in RTT = 76 ms

∴ The final total time PC A takes to load the webpage = DNS Time + TCP Handshake + Object retrieve time

$= (76 + 70 + (21 \times 130))$ ms

$= (76 + 70 + 2730)$ ms $= 2876$ ms.

Ans: to the ques. no. 3

(a)

The UDP checksum is used to detect errors in the UDP datagram. It is computed over a pseudo-header, the UDP header and UDP payload. The receiver computes the checksum and compares. If it doesn't match, the datagram is discarded. The checksum provides end-to-end integrity checking across the UDP payload and header fields.

(b)

PCA will set the URG flag in the TCP header to indicate there is urgent ~~data,~~ and data and set the urgent pointer field appropriately. The urgent pointer value indicates the number of bytes from the sequence number to the byte following the urgent data. If PCA's segment begins at sequence number 3001 and the first 700 bytes are urgent, then the urgent pointer should indicate the position after those 700 bytes. The urgent pointer value would be 700, which signals that urgent data spans bytes with sequence numbers

3001 Through (3001 + 200 - 1) or, 3200.

The receiver ~~was~~ sees URG flag set and urgent pointer = 200 and therefore treats bytes in range of (3001 - 3200) as urgent and can deliver them to the application with priority.

~~Ans: to the~~

(c)(I)

Here,

Client ISN = 2045

Data sent = 320 bytes

∴ Next client sequence number = (2045 + 320) bytes

= 2365

Also,

Server ISN = 8935

Data ~~sent~~ received = DS1 + DS2 + DS3

= (952 + 328 + 455) bytes = 1785 bytes

∴ Server acknowledgement number = 8935 + 1785

= 10720

(c)(II)

Here,

Client's receive buffer size = 3020

If upto DS-5 is received, the buffer would hold the data of DS4 and DS5.

Buffered bytes = DS4 + DS5

= (300 + 99) ~~bytes~~ bytes

= 399 bytes

∴ Client new rwnd = (3020 - 399) = 2621

(c)(III)

From 'I',

Upto DS3, client sequence number = 2365

Current sequence number = 2365 + DS4 = 2365 + 399 = 2754

Similarly, The acknowledgement number of server is,

10720 (From 'I') + DS4 = 10720 + 300 = 11020 (Ans.)