Ans to the question no. 01

a) 6, 5, 3, 1, 7, 2, 4

b) Yes, possible. If the 10 peers collectively contain every piece of the torrent. A newcomer can assemble the entire file by downloading different missing ~~file~~ pieces from different peers as long as every piece exists somewhere in the swarm. If even one piece is absent from all 10 peers, then completion is impossible. So success depends on full piece availability among the 10 peers.

c) No, if the cached copy is still fresh, the proxy serves it and does not contact the origin server. The proxy sends a Conditional GET only when the cached ~~resp~~ copy must be validated; otherwise, it serves the catched copy without contacting the origin.

d) It is possible through pre-buffering. The video player downloads and stores content ahead of playback time in a buffer. For example, with 50 Mbps bandwidth which is half the required 100 Mbps, if we wait before playing, enough video data accumulates in the buffer to enable smooth viewing. As viewing continues, the player keeps downloading in the background to maintain the buffer.

Ans to the question no. 02

a) Websites track users without sign-up using cookies, IP address ~~cooking~~ tracking. This tracing is crucial because HTTP is a stateless protocol, each request is independent with no memory of previous interactions. Tracking mechanisms, especially cookies, solve this limitation by maintaining session state across requests, enabling servers to recognize returning users, remember login sessions and preferences. Without tracking, every HTTP request would appear as a new user.

b) Iterative DNS lookup is generally better overall. In Iterative, the local DNS resolver sequentially contacts ~~root~~ multiple servers one by one. Each server simply provides a referral to the next server instead of doing the full lookup itself. This reduces the

workload on DNS servers significantly. Recursive queries are less efficient because each servers must complete the entire lookup before responding. This uses more resources and creates slower response times.

c) i) TTL from Nonters earlier visit (10:30+5h), which is 3:30 PM) had already expired by Phonters visit at 4:00 PM, so the cached record is gone.

∴ Minimum RTT = (2 × 23) ms = 46ms

ii) DNS = 2×22 ms = 46ms

PC to server request time = 2×39 = 78 ms

per-object request/first byte RTT = 78×30
= 2.34s

payload transfer time = 30 × 12 MB = 360 MB
= 2880Mb

$\therefore$ Total time $= 0.046 + 0.078 + 2.34 + (2880/x)$

$= 2.464 + \frac{2880}{x}$ seconds.

iii) Given total time $= 5344\,ms = 5.344\,s.$

$\therefore 5.344 = 2.464 + \frac{2880}{x}$

$\Rightarrow \frac{2880}{x} = 2.88$

$\Rightarrow x = \frac{2880}{2.88} = 1000$ Mbps

(Ans:)

Ans to the question no-03

a) Yes, multiple HTTP requests routinely arrives on the same destination port, the servers distinguishes connections by the 5-tuple (source IP, source port, destination IP, destination port, protocol).

b) UDP only has source port, destination port, length and checksum. It does not have sequence numbers and acknowledgement. Protocol over UDP implements sequence number and acknowledgement by themselves. This way application gets the level of reliability it actually needs.

c) i) Sequence number = 203 + 1 + (3 × 889)

= 2871.

Acknowledgement number = 8924 + 1 + (4 × 234)

= 9861.

ii) Sequence number = 8924 + 1 + (2 × 234).

= 11 733

Acknowledgement number = 203 + 1 + (9 × 889)

= 8205.

iii) Received buffer = 10000 bytes.

Unprocessed in buffer when seg 13 arrives

= 8 (4 × 889) = 3556 [From seg no 6-9].

Out of order segment (11 to 13) = 3 × 889

= 2667.

∴ Total = 3556 + 2667 = 6223.

∴ RWND = 10000 − 6223 = 3777 bytes.