CS 4220 Computer Networks

HW-1

1 Explain the difference between non-persistent HTTP and persistent HTTP, and why we need persistent HTTP.

Solution: Non-persistent HTTP uses a separate TCP connection for each object. Each object requires its own connection setup and teardown, which adds about two RTTs per object and extra OS overhead. Persistent HTTP reuses a single TCP connection for multiple objects, reducing connection overhead and allowing pipelining, which lowers total response time.

2 What problem do cookies solve for websites? What security issues would there be with cookies enabled?

Solution: Cookies solve HTTP's statelessness. They let a website maintain session state across multiple requests—e.g., keeping a user logged in, remembering items in a shopping cart, storing preferences, and enabling analytics—by associating subsequent requests with the same client.

Enabling cookies also introduces **security and privacy concerns**. If an attacker obtains a user's session cookie, they can impersonate that user, which is known as session hijacking. Cookies can also be exposed through cross-site scripting attacks if malicious scripts can read them. Because browsers automatically attach cookies to outgoing requests, they can be abused in cross-site request forgery attacks to perform actions without the user's intent. In addition, cookies—especially third-party cookies—can be used to track users across websites, raising privacy issues. Improperly configured cookies, such as those sent over unencrypted connections or scoped too broadly, further increase these risks.

3 Explain the differences among HTTP1.1, HTTP 2, and HTTP 3.

Solution: HTTP/1.1 runs over TCP and uses a largely sequential request–response model. Although it supports persistent connections, requests and responses are effectively ordered, so a slow or large response can delay others. This leads browsers to open multiple parallel TCP connections to the same server, increasing overhead and latency.

HTTP/2 also runs over TCP but changes how HTTP data is sent. It introduces multiplexing, allowing multiple requests and responses to be interleaved on a single connection, along with binary framing and header compression to reduce overhead. These features reduce response delay and improve efficiency, but TCP packet loss can still block all active streams on the connection.

HTTP/3 replaces TCP with QUIC, which runs over UDP and integrates reliability, congestion control, and encryption. By moving these functions into QUIC, HTTP/3 avoids transport-level head-of-line blocking, provides faster connection setup, and performs better on lossy or changing networks. **Note: the lectures and the in-class discussion introduce errors.**

In short, HTTP/1.1 is simple but inefficient for modern web workloads, HTTP/2 improves performance through multiplexing and compression but remains constrained by TCP, and HTTP/3 achieves further latency and robustness improvements by using QUIC over UDP.

4 Alice sent an email to Bob asking whether he is available next Monday morning. Explain how the email was sent from Alice and read by Bob from the perspective of the application layer. Hint: keywords: user agent, mail server, SMTP, IMAP.

Solution: From the application-layer perspective, Alice composes the email using her user agent, such as a mail client or webmail interface. When she clicks send, the user agent submits the message to Alice's mail server. That mail server then uses SMTP to transfer the email to Bob's mail server, possibly via one or more intermediate mail servers. SMTP is responsible for reliably pushing the email from the sender's side to the recipient's mail server.

Once the email reaches Bob's mail server, it is stored in Bob's mailbox. When Bob wants to read the message, his user agent connects to his mail server using IMAP. IMAP allows Bob to access, read, and manage the email directly on the server, so the message remains stored there while Bob views it.

5 What is the main purpose of DNS? Why is DNS implemented as a distributed hierarchical system instead of one centralized server? Give two reasons.

Solution:
DNS translates human readable hostnames to IP addresses and also supports aliasing and mail server records. A single centralized DNS server would be a single point of failure, would see enormous query volume, and would create long delays for distant clients. The distributed hierarchical design spreads load, improves performance, and allows each organization to manage its own subtree of the namespace.

6 Explain the difference between recursive and iterative DNS queries from the viewpoint of the local DNS server and upper-level DNS servers.

Solution: In an **iterative** query, each DNS server that is contacted replies with the next server to ask, and the local DNS server iteratively contacts root, TLD, and authoritative servers itself. In a **recursive** query, an upper-level server, such as the root or TLD server, takes on the work of contacting the next servers on behalf of the requester and eventually returns the final answer, which increases its workload.

7 Fill in the protocol names in this table for the common case of fetching a web page over the Internet. Hint: choosing protocols from HTTP, TCP, UDP, IP, and ICMP.

Layer (top to bottom)	Protocol used in this scenario
Application	HTTP
Transport	TCP
Network	IP

8 Consider the usual resolution path when a host in a university network looks up www.amazon.com. Fill each blank with the correct type of DNS server.

Solution:
Client host → (1) Local DNS server → (2) root DNS server → (3) top-level domain DNS server → (4) authoritative DNS server for amazon.com
Possible labels: authoritative, local, root, top-level domain (TLD)

9 For each of the following applications, state whether you would normally use TCP or UDP and briefly justify your choice in terms of reliability, timing, and overhead.

A. File transfer from an enterprise backup server.

B. Real-time voice call.

C. Multiplayer online game.

D. Web browsing.

Solution:

A. File transfer: TCP. Needs reliable, in-order delivery and is not very sensitive to delay, but correctness matters.

B. Real-time voice: usually UDP. Can tolerate some loss, but needs low delay and does not want retransmission delays or head-of-line blocking.

C. Multiplayer game: often UDP. Timeliness is more important than perfect reliability, and the app can do its own lightweight loss handling.

D. Web browsing: TCP. Web pages must arrive reliably and in order, and the overhead of TCP is acceptable.