CSN-341
Computer Networks
Assignment - 3

Group-6

Name	Enrollment No
Meet Sindhav	22114053
Mohammed Haaziq	22114055
Aditya Mundada	22114058
Nayan Kakade	22114060
Sarvesh Baheti	22114087
Roopam Taneja	22125030

Question 1: Analyze how DNS caching works and discuss its potential benefits and drawbacks on network performance, particularly in large-scale systems. Include real-world examples or case studies to support your explanation.

Since the user always remembers the name of the website by its name and not by its IP address, DNS (Domain Name System) comes into play. Whenever a user requests access to a website, the DNS server translates the domain name into an IP address. Now, every time a request is sent by the user, the request goes to the root server. To decrease this load, we use caching. It is used to store previously queried domain name resolutions temporarily.

Each time a server receives a query for a name that is not in its domain, it needs to search its database for a server IP address. Reduction of this search time would increase efficiency. DNS handles this with a mechanism called caching. When a server asks for a mapping from another server and receives the response, it stores this information in its cache memory before sending it to the client. If the same or another client asks for the same mapping, it can check its cache memory and resolve the problem. However, to inform the client that the response is coming from the cache memory and not from an authoritative source, the server marks the response as unauthoritative.

Benefits
The benefits of DNS cache are generic and are similar to cache in memory:

- **Reduced Latency and Lower Bandwidth Usage** are the primary advantages of cached DNS. By reducing responses to remote servers, it reduces the time it takes to resolve domain names, thus speeding up website access. This also decreases bandwidth consumption, which particularly helps in large-scale systems with heavy traffic.
- In large-scale systems, such as those in large organizations or smart cities, caching reduces the load on DNS infrastructure, improving overall system scalability.
  Real-World Example: Google Public DNS has a variety of extensive caching mechanisms to handle billions of queries daily. They thus help provide fast and reliable DNS resolution throughout the world.
- In case of root server outages, cached entries allow continued access to previously visited domains for at least some amount of time before the TTL is up, thus mitigating the impact of server failures.

Drawbacks of DNS Caching

Caching speeds up resolution, but it can also be problematic:

- **Outdated Entries:** If a server caches a mapping for a long time, it may send an outdated mapping to the client. A recent example would be the 2016 DDoS attack on Dyn's DNS infrastructure, which caused widespread internet outages. Cached DNS entries pointed to unavailable services, thus making matters worse.
- **Security:** The security of the cache is easier to breach compared to the authoritative server, thus malicious actors can inject false DNS records into a cache, redirecting users to fraudulent websites or intercepting data. In 2008, a significant vulnerability was discovered in DNS caching mechanisms (Kaminsky vulnerability). This allowed attackers to poison caches, redirecting vast amounts of internet traffic.
- Cache memory must be searched periodically and those mappings with an expired TTL must be purged, thus selecting an appropriate TTL is important. If TTL values are set too high, stale data might persist longer, increasing the chances of directing users to outdated IP addresses. Conversely, if TTL values are too low, caching benefits diminish, leading to frequent DNS queries and increased load on DNS servers.

Question 2: Compare and contrast HTTP and FTP in the context of file retrieval. Discuss scenarios where each protocol is preferable, taking into account factors like security, speed, ease of use, and typical use cases. Support your discussion with specific examples and justify your choices.

HyperText Transfer Protocol (HTTP) and File Transfer Protocol (FTP) are the protocols used for file transfer between client and server.

HTTP:

- It stands for HyperText Transfer Protocol. It is the backbone of WWW. It is an internet standard that allows the process of transfer of web pages over the internet. It also defines how the web browser will respond to any web request.
- The web address of all the web pages contains a protocol, domain name, and path to the web page. Most of the web address contains `http://` in their URL to show the HTTP protocol.
- HTTP operates over TCP (Transmission Control Protocol) and is widely used for accessing resources like HTML documents, images, and videos on the internet.

FTP:

- It stands for File Transfer Protocol. It is an internet standard that allows the process of file downloading and uploading on different computers from the internet.
- An FTP site consists of different types of files (text, graphics, videos, images, etc).
- FTP supports two separate Transmission Control Protocols: the first one is a control connection or command port to authenticate the user, and the second one is a data connection or data port to transfer the files.
- It requires a specific username and password for access.

In the following section, we compare these protocols on several factors:

1. **Security:**
   - **HTTP:** With HTTPS, HTTP offers strong encryption and is widely supported, making it suitable for secure file transfers over the internet.
   - **FTP:** Basic FTP lacks security, but FTPS and SFTP provide encryption. All data, including the username and password, is transferred in plain text, which makes it easy for an attacker to intercept the transmission.
2. **Speed:**

- **HTTP:** While optimized for web traffic, HTTP's performance may suffer with very large files due to its stateless nature and the overhead of HTTP headers and cookies. It is efficient in transferring small files.
- **FTP:** Typically faster for large file transfers, especially in batch operations, due to its dedicated design for file transfer.

3. **Ease of Use:**
   - **HTTP:** Very easy to use, especially for casual users who only need a web browser. No additional software is required.
   - **FTP:** Requires a dedicated client, making it less accessible to non-technical users. However, for large-scale file management, FTP clients offer powerful features.
4. **Use Cases:**
   - **HTTP:** Best for retrieving files that are publicly accessible or when integrating with web services and APIs. Ideal for environments where security (HTTPS) and ease of access are primary concerns.
   - **FTP:** Ideal for environments where large files need to be transferred quickly and efficiently, such as web development or data migration tasks.

Justification of Choices:

- **HTTP/HTTPS** is preferable when transferring files that are part of web content or when ease of access and security via HTTPS are required. For example, downloading a software installer from a trusted website would typically use HTTPS due to its security and simplicity.
- **FTP** is preferable in scenarios requiring the transfer of large files, such as backing up a website or transferring large datasets between systems. FTP's design allows for faster and more efficient bulk file transfers compared to HTTP, making it ideal for these use cases.

In summary, HTTP is better for general web-related file retrieval with good security and ease of access, while FTP is more suited for efficient, large-scale file transfers.

Question 3: Explore the role of cookies in web applications, focusing on both their benefits and privacy concerns. How have recent regulations, such as GDPR, influenced the way cookies are used? Suggest best practices for developers to balance functionality and user privacy.

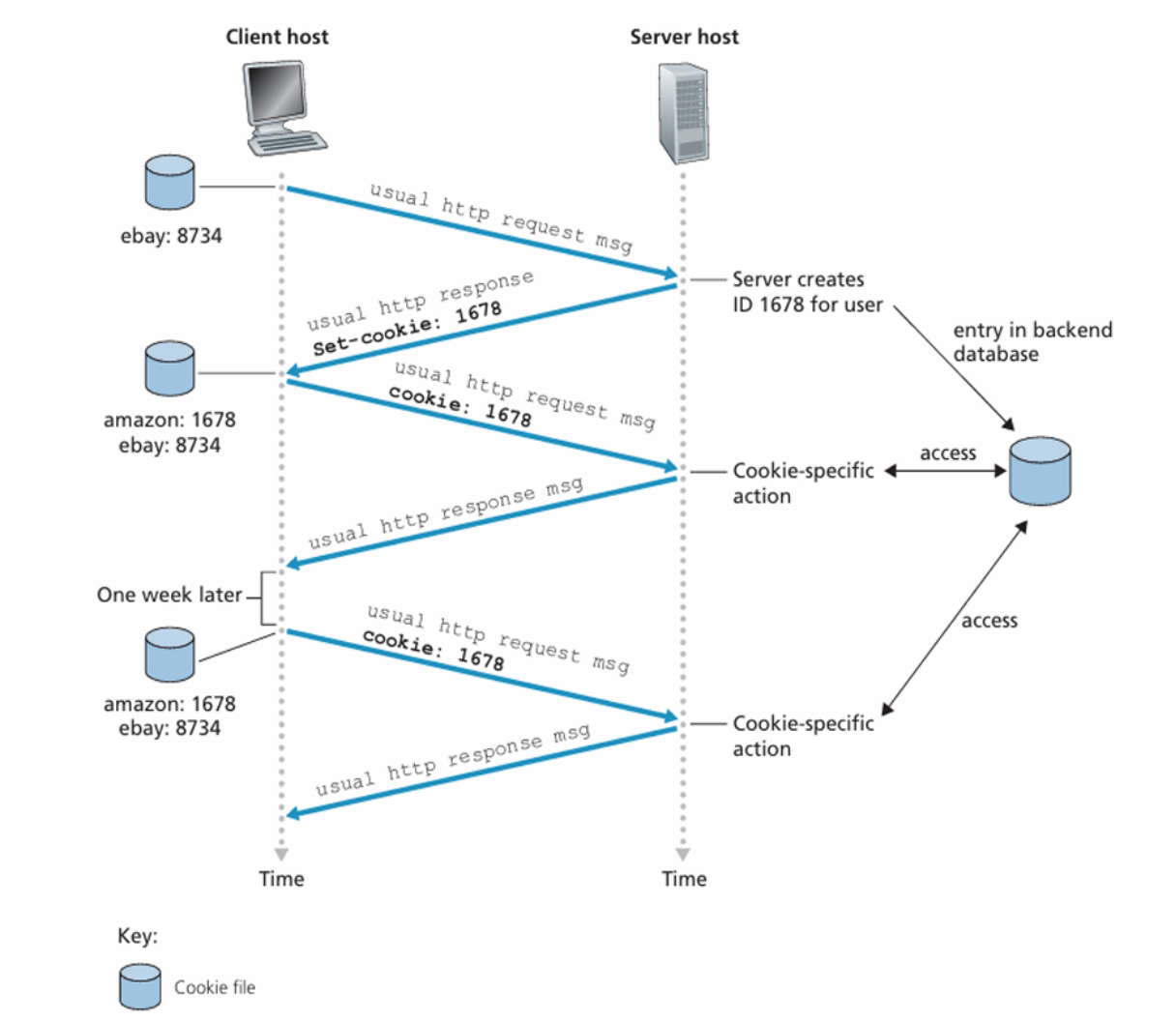
The HTTP protocol that is used by clients to access web pages from the server was originally designed to be **stateless**. This means that the server would not track and store any information of the client. Therefore, if a client asks for the same object twice within a few seconds, the server would not respond saying that it already provided the object since it did not store information regarding the previous communication.

However, it often becomes necessary by the server to identify the users since it may want to restrict access based on the type of user or it may want to serve the objects based on the user identity. Eg: For an e-commerce platform users must be able to select the required items and put them in a cart and pay for all the items together at the end of their shopping. Thus, the server must be able to store the information regarding the selections made by this client. For these purposes cookies are used.

Process of Cookie Creation and Usage

1. When the user client requests a web page from the e-commerce platform, let's say Amazon for the first time, the Amazon web server creates a unique identification number and generates a new entry in its back-end database indexed by this number.
2. The cookie number is returned by the server to the client in its response under the header: `Set-cookie`.

How Do Cookies work?



3. Every user browser maintains a special file that stores its cookie details. This file contains a list of hostnames of various servers and the cookie number associated with that server for the particular client. Now, since our user is visiting the Amazon server for the first time, a new cookie entry is inserted in this file corresponding to the server and the cookie number.
4. Now, in the future whenever the user requests something from the Amazon server, it searches the cookie file for the server name. As the server name is already present, each request that now goes to the Amazon server from our client contains additional information about the Cookie number in the `Cookie` header.
5. In this way, the server can now track the activity of each user based on its cookie number and store any more information like address and credit card details in its database corresponding to the given cookie number.
6. So, basically we first register with the server and then provide our ID to do our work.

The various benefits of cookies include:

1. They restrict access to registered clients, preventing unknown access.
2. They help a lot in personalization and optimization of services since the recommendations to each user are based

on their consumption pattern.

3. They help collect analytics and track data which helps organizations understand its consumers better.

Problems with cookies:

1. **Third party cookies:** Advertising agencies often place their URL on main websites that are popular. Whenever a user visits that website, a request for the advertisement banner object is sent to that URL. Whenever this request is sent, the advertisement website registers the user by setting cookies. Now, the same website can track a user by placing such ads on multiple sites and get a lot of search data about the user which can be sold further and threatens user privacy.
2. A hacker can also run some Cross-Site Script (XSS) on vulnerable websites that may extract cookie data and sensitive information about the user.
3. Excessive cookies when stored also cause performance overhead and it slows down browser performance as excess overhead cookie data must be sent in HTTP requests.

Regulations related to cookie handling:
Certain regulations like General Data Protection Regulations (GDPR) in the EU and the California Consumer Privacy Act (CCPA) were recently enacted to protect user privacy and regulate the usage of user data. Their provisions include:

1. Cookie banners must now be displayed by websites to users that inform them regarding the type of data collected, its purpose, and third parties that may get access to this data. This increased transparency.
2. Consent must be asked from users when collecting non-essential cookies related to advertising and data tracking.
3. Users must be able to manage and change their cookie preferences.
4. Cookies must have a justified lifespan after which they must be deleted.
5. Strict regulations are imposed on third-party advertising cookies regarding what kinds of data they may track and where they can be placed.
6. Organizations must properly document their cookie practices and data collection methodologies.

Certain practices that can help developers balance user privacy and good design are:

1. **Data minimization:** Collection of only the most necessary data and discarding data not essential for improving user experience.
2. Cookies must be secured using encryption techniques to prevent unauthorized data leaks.
3. **Data Anonymization:** Data must be anonymized wherever possible so that it can't be traced back to the user.
4. Minimization of third-party interactions is necessary to comply with user privacy.
5. Using new techniques like:
   (a) **Contextual advertising:** Where ads are shown according to website content and not user behavior.
   (b) **Server-Side tracking:** Tracking server-side data instead of client-side data.
6. Apart from these, organizations must regularly audit and update their data protection and monitoring practices keeping user privacy as their primary responsibility.

Question 4: Analyze the Simple Mail Transfer Protocol (SMTP) in the context of email security. What vulnerabilities are inherent in SMTP, and what additional protocols or practices are used to secure email communications? Discuss the balance between security and ease of use in email systems

Simple Mail Transfer Protocol (SMTP)

- SMTP is the principal application layer protocol. It uses the data transfer service of TCP to transfer mail from sender to recipient.
- SMTP has two sides, client and server side, which operates on the sender and receiver's server.

Following are the vulnerabilities in SMTP:

- SMTP restricts the body of all the messages to be 7-bit ASCII. So sending messages which are not in this format require encoding techniques like Base64. Improper encoding can raise privacy issues and decoding issues.
- The basic SMTP version lacks encryption. Mails sent through SMTP are susceptible to man-in-the-middle attacks and eavesdropping from bad actors.
  - Attacks can happen while the message is in transit. The attacker can access the message, modify the content of the message (e.g., insert a link which, when clicked, can download malware on a personal computer), or reroute the message to another attacker.
- Attackers inject their own malicious SMTP commands in the original message. This is known as SMTP command injection.

Consequences of SMTP command injection:

- **Sending Spam:** Attackers can send large volumes of spam mails. They are less likely to be flagged as spam as they appear to come from legitimate senders.
- **Phishing Attacks:** Attackers can inject phishing mails that appear to come from legitimate senders. This can trap receivers into sharing their sensitive information like passwords, OTP, or other personal data.

Steps to secure SMTP:

- SMTPS (Simple Mail Transfer Protocol Secure) is a method for securing SMTP through enabling Transport Layer Security (TLS) at the transport layer.
  - The client and server connect normally through SMTP at the application layer, but the connection is secured through TLS or SSL at the time when the TCP connection is established and before the message transfer has taken place.
  - Port 587 and 465 are both frequently used for SMTPS traffic. Port 587 is often used to encrypt SMTP messages using STARTTLS, which allows the email client to establish secure connections by requesting that the mail server upgrade the connection through TLS. Port 465 is used for implicit TLS and can be used to facilitate secure communications for mail services.
- But SMTPS does not provide security against phishing attacks, malicious attachments, spoof emails. For that, the following email standards can help:
  - **Sender Policy Framework (SPF):** Helps recipients of your messages verify that messages from your domain are in fact coming from you. SPF tells the world which servers you send email from. If a message is sent from your domain that does not originate from these servers, the SPF check fails.
  - **Domain Keys Identified Mail (DKIM):** Provides an extra layer of email authentication by giving the messages a digital signature.
  - **Domain-based Message Authentication Reporting and Conformance (DMARC):** Authenticates messages by aligning SPF and DKIM capabilities.

Some additional security measures:

- **Pretty Good Privacy (PGP):** It provides cryptographic privacy. When a user sends email, the mail content is first encrypted with a session key. Then the session key is encrypted with the recipient's public key. The recipient can decrypt the encrypted session key using their private key and then decrypt the message.

- **Secure/Multipurpose Internet Mail Extensions (S/MIME):** It is a standard for public key encryption and signing of MIME (Multipurpose Internet Mail Extensions) data. It also extends the original SMTP protocol to support non-ASCII text and binary data.

Tradeoff between Security and Ease of Use:

- If we implement strong authentication methods, they increase security but at the same time the login process can become frustrating for the user due to multi-factor authentication.
- Single sign-on (SSO) allows users to access multiple services with a single password. But this can create a single point of failure if the password becomes compromised.

Ensuring the right balance between security and ease of use is necessary by implementing adaptive security measures that adjust based on user behavior. For example, multi-factor authentication can be asked during suspicious activity.