CSN-341 (Computer Networks)

Assignment-3 by Group 16
August 27, 2024

Anvit Gupta
22114009
anvit_g@cs.iitr.ac.in

Ayush Ranjan
22114018
ayush_r@cs.iitr.ac.in

Indranil Das
22114037
indranil_d@cs.iitr.ac.in

Sarvasva Gupta
22114086
sarvasva_g@cs.iitr.ac.in

Souvik Karmakar
22114096
souvik_k@cs.iitr.ac.in

Vineet Kumar
22114107
vineet_k@cs.iitr.ac.in

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.

DNS Caching Overview:

- **What is DNS Caching?** DNS caching is a mechanism that stores DNS query results (mapping domain names to IP addresses) locally on a DNS resolver or client machine for a specified period (TTL, Time-to-Live). This eliminates the need for repeated DNS lookups for the same domain, improving response time and reducing network traffic.

Benefits of DNS Caching:

- **Reduced Latency:** Caching reduces the time it takes to resolve domain names by retrieving stored results from the local cache, speeding up user access to frequently visited websites.
- **Decreased DNS Traffic:** By minimizing the need to query DNS servers repeatedly, caching reduces the load on authoritative DNS servers and overall DNS traffic on the network.
- **Scalability in Large-Scale Systems:** In large networks or distributed systems, DNS caching can significantly reduce the strain on centralized DNS services, improving scalability and reliability.

Drawbacks of DNS Caching:

- **Stale Data:** If the cached DNS information becomes outdated (e.g., due to domain IP changes), users may be directed to incorrect or non-functional addresses until the cache is refreshed.
- **Security Risks:** DNS cache poisoning attacks exploit caching mechanisms to inject malicious IP addresses into the cache, redirecting users to fraudulent websites.
- **Load Balancing and Failover Issues:** Some systems rely on DNS to implement load balancing or failover mechanisms. Cached results can interfere with these processes by preventing dynamic IP resolution.

Real-World Examples/Case Studies:

- **Akamai CDN:** Akamai's Content Delivery Network (CDN) leverages DNS caching to distribute content globally, reducing latency and balancing load. However, it also must handle cache expiry to ensure content delivery remains dynamic and responsive to network conditions.

- **Google Public DNS:** Google's Public DNS is designed with large-scale DNS caching in mind, improving global internet performance by reducing the time taken to resolve domain names. It addresses potential drawbacks by using optimised cache refresh policies to avoid stale data.

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.

HTTP (Hypertext Transfer Protocol):

- **Primary Use:** HTTP is mainly used for transferring web content (HTML, CSS, JavaScript) and files over the web.
- **Security:** HTTP can be secured using HTTPS (HTTP over SSL/TLS), which encrypts the data in transit. This is vital for sensitive data transfers.
- **Ease of Use:** HTTP is user-friendly, with files accessed through web browsers, requiring no special software.
- **Performance:** HTTP/2 and HTTP/3 protocols improve speed by allowing multiple simultaneous requests over a single connection, reducing latency.
- **Use Case Example:** Websites like Google Drive or Dropbox use HTTP/HTTPS to allow users to easily upload and download files through their browser interfaces.

FTP (File Transfer Protocol):

- **Primary Use:** FTP is traditionally used for transferring large files between systems, often between a client and a server.
- **Security:** FTP transfers data in plaintext, which can expose sensitive information. Secure FTP (SFTP) uses SSH to encrypt file transfers, enhancing security.
- **Ease of Use:** FTP generally requires a dedicated client (e.g., FileZilla) and server setup, making it less accessible for casual users compared to HTTP.
- **Performance:** FTP is often faster for bulk file transfers due to its ability to handle larger file sizes more efficiently in certain network configurations.
- **Use Case Example:** Web developers commonly use FTP to upload website files to a remote server, especially when dealing with bulk uploads or system backups.

Comparison:

- **Security:** HTTPS (HTTP over TLS) **offers built-in encryption**, making it preferable for secure file transfers over the web. SFTP offers similar security for FTP but requires additional setup.
- **Speed:** FTP can outperform HTTP in environments **focused on large file transfers**, but modern HTTP protocols (like HTTP/2) have closed the gap for general file retrieval scenarios.
- **Ease of Use: HTTP is more user-friendly and doesn't require specialized software**, making it ideal for widespread file distribution (e.g., via websites). FTP, while less user-friendly, is better suited for professional environments where large-scale file transfers and server management are common.

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.

Role of Cookies in Web Applications:

- **Benefits**:
  - **Session Management:** Cookies help maintain user sessions across multiple requests, enabling features like login persistence and shopping carts.
  - **Personalization:** Websites use cookies to store user preferences and deliver personalized experiences (e.g., language settings, tailored content).
  - **Analytics and Tracking:** Cookies are essential for tracking user behavior, gathering analytics, and enabling targeted advertising.

Privacy Concerns:

- **Tracking and Profiling:** Cookies can be used to track users across different websites, often without their explicit consent, leading to concerns about user profiling and behavioral advertising.
- **Data Breaches:** If cookies contain sensitive information and are not properly secured, they can be intercepted, leading to data breaches or session hijacking.

Impact of GDPR:

- **Consent Requirements:** Under GDPR, websites must obtain explicit consent from users before setting non-essential cookies (e.g., tracking cookies). Users must be informed about what data is being collected and how it will be used.

- **Right to Access and Erasure:** Users have the right to access the data collected via cookies and request its deletion, impacting how companies store and manage cookie data.

Best Practices for Developers:

- **Implement Consent Management:** Use a **cookie consent banner** that allows users to opt-in or opt-out of non-essential cookies, ensuring compliance with regulations like GDPR.
- **Use Secure Cookies:** Set cookies with the **Secure**(Travel only on HTTPS channels) and **HttpOnly**(accessible only by HTTP requests and not by scripting) flags to protect them from being accessed via insecure channels or by JavaScript
- **Minimize Data Storage:** Only store essential data in cookies and consider **using alternative storage mechanisms** (e.g., localStorage) where appropriate. Avoid storing sensitive data in cookies

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.

SMTP Overview and Inherent Vulnerabilities

- **Lack of Encryption:** SMTP by itself does not provide encryption, meaning emails sent over SMTP are transmitted in plaintext and can be intercepted.
- **Spoofing:** SMTP lacks built-in mechanisms to authenticate senders, making it vulnerable to email spoofing, where attackers send emails appearing to come from trusted domains.
- **Phishing and Man-in-the-Middle Attacks:** Without encryption and authentication, SMTP is susceptible to phishing attempts and man-in-the middle (MITM) attacks.

Additional Protocols/Practices to Secure Email:

- **TLS (Transport Layer Security):** STARTTLS is used to encrypt SMTP traffic, preventing unauthorized access to email content during transmission.
- **DKIM (DomainKeys Identified Mail):** DKIM adds a digital signature to emails, allowing the receiving server to verify that the email hasn't been altered and is from a legitimate source

- **SPF (Sender Policy Framework):** SPF helps prevent spoofing by allowing domain owners to specify which IP addresses are authorized to send emails on their behalf.
- **DMARC (Domain-based Message Authentication, Reporting, and Conformance):** DMARC builds on SPF and DKIM, providing policies for handling unauthorized emails and enabling reports on email authentication.

Looking at SPF, DKIM and DMARC together :

- DKIM → Verifies if the e-mail send mail-box server is who it claims to be
- SPF → Check if that server is authorised to send mail of the specific domain
- DMARC → Policy published by domain owner on how to handle mails that fail either of these checks

Balancing Security and Ease of Use:

- **Trade-offs:** While encryption and authentication protocols enhance email security, they can complicate setup and management for end-users and administrators. For example, misconfigured SPF or DKIM records can lead to legitimate emails being marked as spam.
- **Ease of Use vs. Security:** Email providers must balance robust security measures (e.g., enforcing TLS encryption) with user-friendly interfaces that don't overwhelm non-technical users. Some services offer security features as optional configurations, but this can lead to inconsistent security practices across users.

Example:

- **Gmail:** Gmail enforces encryption between Gmail servers and uses features like DKIM and SPF by default. It balances security and ease of use by managing complex configurations in the background, offering a secure service that is still accessible to non-technical users.