CSN – 341 (Computer Networks)

Assignment 2

Group – 16

1. Vineet Kumar – 22114107

Contact No: +91 9263436403

Email ID: vineet_k@cs.iitr.ac.in

2. Ayush Ranjan – 22114018

Contact No: +91 7482958551

Email ID: ayush_r@cs.iitr.ac.in

3. Souvik Karmakar – 22114096

Contact No: +91 8642924659

Email ID: souvik_k@cs.iitr.ac.in

4. Sarvasva Gupta – 22114086

Contact No: +91 7989904811

Email ID: sarvasva_g@cs.iitr.ac.in

5. Anvit Gupta – 22114009

Contact No: +91 9462011044

Email ID: anvit_g@cs.iitr.ac.in

6. Indranil Das – 22114037

Contact No: +91 8902031812

Email ID: indranil_d@cs.iitr.ac.in

Q.1) Explore the mechanisms through which a new peer like Alice in a BitTorrent network acquires her first chunk. Discuss the fairness and efficiency of these mechanisms, and consider how different scenarios (e.g., network congestion, peer availability) might affect Alice's experience.?

Mechanisms:

1. Tracker Request: Alice initially contacts a tracker server, providing her peer ID and the info hash of the file she wants to download. The tracker responds with a list of peers currently downloading or seeding the file.
2. Peer Discovery: Alice connects to peers from the list.
3. Requesting the First Chunk: Alice sends a "have" message to inform the peers about the chunks she already has (initially none) and requests the first chunk.
4. Piece Selection: Peers who have the first chunk and are willing to share it respond with a "have" message and offer to send the chunk.
5. Downloading the Chunk: Alice selects a peer to download the chunk from, considering factors like its upload speed and proximity. She requests the chunk, and the peer sends it.

Fairness and Efficiency:

- Fairness: BitTorrent uses a tit-for-tat mechanism to promote fairness. Peers who contribute by uploading chunks to others are more likely to receive chunks in return.
- Efficiency: BitTorrent is efficient because it allows multiple peers to download different parts of the file simultaneously, maximising bandwidth utilisation. However, network congestion can significantly impact download speeds.

Scenarios:

- Network Congestion: If the network is congested, peers may have difficulty sending or receiving data, slowing down the download process.

- Peer Availability: If there are few peers seeding the file, Alice may have to wait longer to find a peer with the first chunk.

Q.2) Evaluate the challenges and limitations of using POP3 for email access across multiple devices, particularly with the “download and keep” strategy. Propose potential solutions or alternative protocols that could mitigate these challenges and justify your recommendations.

What is POP3 → POP3 (Post Office Protocol version 3) is a standard protocol used for retrieving emails from a mail server to an email client. It allows a user to download emails from a remote server to their local device for offline reading and storage. { Why not SMTP, because it’s a push protocol }

What is the “Download and Keep” Strategy →? There are two types of strategies the POP3 follows “Keep” and “Delete” Modes. As the name suggests, while using Keep mode after the mail is retrieved to the client, it still remains on the server.

Challenges:

- This can lead to data loss if the client crashes before saving the emails and it’s deleted from the server mailbox.
- Multiple Device Access: Synchronising emails across multiple devices can be challenging with POP3. Changes made on one device might not be immediately reflected on others.
- It also doesn’t have other features like mailbox organisation and partial view of mail before complete download.

Solutions and Alternatives:

There are two better ways for Bob (receiver) to retrieve his e-mail from a mail server.

- If Bob is using Web-based e-mail or a smartphone app (such as Gmail), then the user agent will use HTTP to retrieve Bob’s e-mail. This case requires

Bob’s mail server to have an HTTP interface as well as an SMTP interface (to communicate with Alice’s mail server).

- The alternative method is the use of IMAP Internet Mail Access Protocol (IMAP) defined in RFC 3501. Both the HTTP and IMAP approaches allow Bob to manage folders maintained in Bob’s mail server. Bob can move messages into the folders he creates, delete messages, mark messages as important, and so on.

Q.3) Discuss the role of SSL in the TCP/IP stack and its implications for application developers who wish to enhance TCP with SSL. Consider the steps a developer must take and the potential trade-offs involved in implementing SSL at different layers.

Ans.

What is SSL: SSL, or Secure Sockets Layer, is an encryption-based Internet security protocol. It was first developed by Netscape in 1995 for the purpose of ensuring privacy, authentication, and data integrity in Internet communications. SSL is the predecessor to the modern TLS encryption used today

Role of SSL:

- SSL (Secure Sockets Layer) adds a layer of security on top of the TCP/IP stack. It encrypts data transmitted between two communicating parties, ensuring Confidentiality (Keeping Data Private), Integrity (Ensuring Data is Untouched), and Authentication (Verifying Identity).
- SSL operates at the Application Layer (Layer 7), above TCP.

Developer Steps:

1. Choose an SSL/TLS Library: Select a library like OpenSSL or NSS that provides SSL/TLS implementation.
2. Initialise SSL Context: Create an SSL context object that specifies the desired security parameters (e.g., ciphers, certificates).
3. Establish SSL Connection: Use the library's functions to establish an SSL connection over the existing TCP connection. This involves negotiating the security parameters and exchanging certificates.

4. Encrypt and Decrypt Data: Use the library's functions to encrypt data before sending it over the TCP connection and decrypt data received from the other end.

Trade-offs:

- Performance: SSL can introduce some overhead due to encryption and decryption operations. This might impact performance, especially for low-bandwidth connections or computationally intensive tasks.
- Complexity: Implementing SSL can be more complex than using plain TCP, requiring additional code and an understanding of cryptographic concepts.
- Security: SSL provides strong security benefits but requires careful management of certificates, keys, and other security parameters.

Q.4) Examine the process by which non-persistent HTTP connections signal the end of a message to the TCP protocol. Discuss how this mechanism affects the performance and reliability of web communications and compare it with persistent HTTP connections.?

Signalling End of Message:

- In a Non-Persistent Connection Type in HTTP
    - The server CLOSE its end of the TCP Connection after it sends the response or the request made by the client.
    - The Client CLOSE its end of the TCP Connection after it receives the entire response (only one, in case of non-persistent connections)

CLOSE is an operation meaning "I have no more data to send." The notion of closing a full-duplex connection is subject to ambiguous interpretation, of course, since it may need to be clearer how to treat the receiving side of the connection. We have chosen to treat CLOSE in a simplex fashion. The user who CLOSEs may continue to RECEIVE until the TCP receiver is told that the remote peer has CLOSED also. (Src - RFC 9293 Pg-29, 30)

Performance and Reliability

- Performance: Non-persistent connections can be less efficient for multiple requests to the same server, as a new TCP connection needs to be established for each request. This can lead to increased latency and overhead.

- Reliability: Non-persistent connections can be more reliable in some scenarios, as closing the connection after each request can help prevent resource leaks (the server end socket or say the connection is still not terminated OR say resource not deallocated) and connection congestion (multiple requests from a single client simultaneously can overwhelm the server w.r.t to Persistent Connection).

Persistent HTTP Connections:

- Persistent HTTP connections (also known as HTTP keep-alive) allow multiple requests and responses to be sent over the same TCP connection. This can significantly improve performance for multiple requests to the same server.
- However, if not appropriately managed, persistent connections might introduce additional complexity and potential issues.