Unveiling the Intricate Dance: Accessing a Webpage on Facebook.com

What exactly happens from the moment we type www.facebook.com till we log in?

<u>Submitted By:</u>

Mukesh Amresh Thakur

-PAS078BEI023

Prashant Joshi

-PAS078BEI026

Rohit Kumar Sah

-PAS078BEI027

Uma Nath Thakur

-PAS078BEI044

Table of Contents

Introduction

Purpose of the Report

This report sets out to deliver a thorough and detailed breakdown of the technical processes triggered when a user clicks on facebook.com. We will dissect each step, from the moment the URL is entered to the final display of the Facebook homepage. By doing so, this report aims to shed light on the intricate interplay of technologies that work together to deliver a smooth web browsing experience.

Overview of the Process

Clicking facebook.com kicks off a whole chain of events behind the scenes! First, the browser takes a peek at the web address (URL) to understand where you're trying to go. Then, it uses a special phonebook for websites (DNS) to find the actual address (IP address) of facebook.com. Once it has the right address, the browser high-fives the Facebook server three times (three-way handshake) to make sure they can chat smoothly. With a connection established, the browser sends a message (HTTP request) to Facebook asking for the homepage. Facebook sends back a reply (HTTP response) with all the info to build the page. Finally, the browser takes that info and puts it all together, showing you the familiar Facebook homepage! Every step is important to make sure you get to where you want to go on the web quickly and easily.

URL Parsing

When a user types "facebook.com" into their browser and presses enter, the browser begins by parsing the URL into its fundamental components. This process ensures that each part of the URL is correctly identified and processed for subsequent steps.

Components of a URL

Components of a Uniform Resource Locator (URL):

- **Protocol:** The protocol defines the set of rules governing data transmission between a client (browser) and a server. In the context of Facebook, the commonly used protocol is **HTTPS** (Hypertext Transfer Protocol Secure), indicating a secure communication channel employing encryption to protect data in transit.
- **Domain:** The domain name serves as the primary address for a website, identifying it on the internet. For Facebook, the domain name is **www.facebook.com**. It can be further broken down into subdomains (e.g., www) and the top-level domain (TLD, e.g., .com).
- **Port:** The port number often remains implicit within the URL. However, it plays a crucial role in directing communication to the appropriate service running on the server. For

secure connections using HTTPS, the default port is **443**, while the default for unsecured HTTP connections is **80**.

- **Path:** The path specifies the location of a specific resource within the server's directory structure. For instance, when accessing a user's profile on Facebook, the path would be denoted as **/profile**.
- **Query String:** The query string, denoted by a question mark ("?") followed by key-value pairs separated by ampersands ("&"), is used to transmit additional information to the server. It allows for dynamic content generation based on user input. An example might be a search query on Facebook: **https://www.facebook.com/search?q=friend**, where "q=friend" represents the query string.
- **Fragment:** The fragment, denoted by a hash symbol ("#") followed by an identifier, refers to a specific section within a webpage. It is primarily used for internal navigation without reloading the entire page. For instance, the URL **https://www.facebook.com/home#timeline** directs the browser to the section named "#timeline" on the Facebook home page.

DNS Lookup

Once the URL has been broken down into its components, the system must determine the website's numerical address (IP address). This process is handled by the Domain Name System (DNS). To efficiently find the correct IP address, DNS involves a multi-step lookup process utilizing temporary storage (caching) and a series of inquiries (recursive queries).

Browser Cache

The browser first checks its own cache to see if it has a recent DNS record for `www.facebook.com`. If found, it uses this cached IP address, bypassing further lookup steps.

OS Cache

If the browser cache does not have the required DNS record, the browser queries the operating system's cache. The OS maintains its own cache of recent DNS queries.

Router Cache

If the OS cache also does not have the DNS record, the request is sent to the local router. Many routers have their own DNS caches to speed up the lookup process for frequently accessed domains.

ISP DNS Cache

If the router does not have the DNS information, it forwards the request to the Internet Service Provider's (ISP) DNS server. The ISP maintains a cache of DNS records for its users, which helps in reducing the load on the global DNS infrastructure and improves lookup speed.

Recursive DNS Lookup

If the ISP's DNS server does not have the required DNS record, it initiates a recursive DNS lookup. This involves querying multiple DNS servers in a hierarchical manner to resolve the domain name.

DNS Recursive Resolver

The recursive resolver is the first step in this process. It acts as an intermediary, querying other DNS servers on behalf of the client. It first checks its own cache and, if necessary, sends a request to one of the root DNS servers.

DNS Root Name Server

The Domain Name System (DNS) hierarchy is anchored by **root nameservers**, which occupy the highest level. To ensure reliability and optimal performance, thirteen root name servers exist globally, each having numerous replicas. While these root name servers don't possess the specific IP address for www.facebook.com, they do hold crucial information regarding the authoritative servers responsible for the top-level domain (TLD) ".com".

TLD Nameserver

The recursive resolver then queries a TLD name server responsible for `.com` domains. This server directs the resolver to the authoritative nameserver for `facebook.com`.

Authoritative Name Server

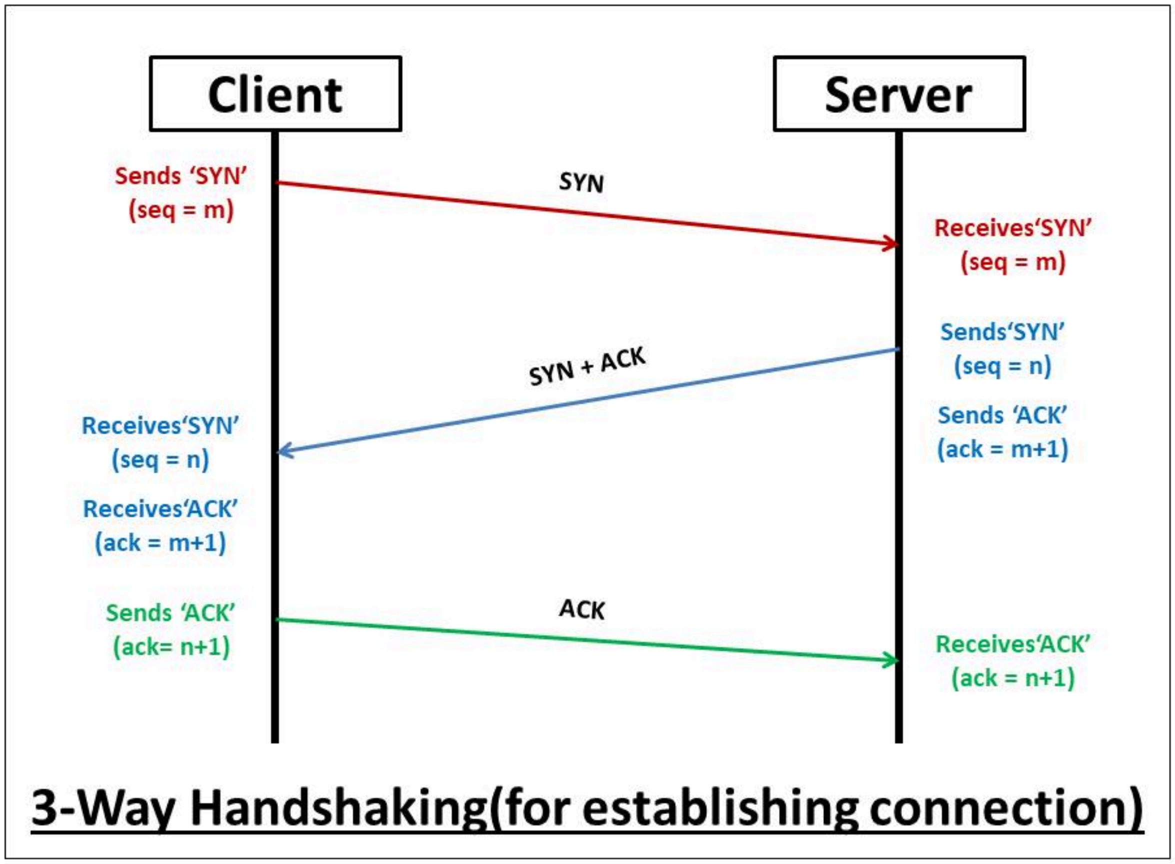
The authoritative nameserver for `facebook.com` harbors the entire set of Domain Name System (DNS) records for all subdomains under that umbrella, including `www.facebook.com`. Upon receiving a recursive query, this authoritative nameserver responds by transmitting the IP address associated with the Facebook server.

This retrieved IP address is then cached by the recursive resolver, further enhancing efficiency. Subsequently, the recursive resolver relays the IP address back to the browser, marking the culmination of the DNS lookup process. Armed with the resolved IP address, the browser can now initiate a connection with the Facebook server, paving the way for content retrieval.

Establishing a TCP/IP Connection

Once the DNS lookup is complete and the IP address of the Facebook server is obtained, the browser needs to establish a reliable connection to the server. This is done using the TCP/IP protocol, which ensures that data is transmitted accurately and in the correct order. The process begins with a three-way handshake.

The TCP Handshake Process



1. SYN Packet:

The client transmits a SYN (synchronize) packet to the server. Embedded within this packet is an initial sequence number (ISN), randomly generated to serve as a baseline for subsequent byte sequencing. The SYN flag in the TCP header is explicitly set to signal the commencement of a connection.

2. SYN-ACK Packet:

Upon receipt of the SYN packet, the server responds with a SYN-ACK (synchronize-acknowledge) packet. This packet acknowledges the client's SYN by incorporating an acknowledgment number incremented by one from the client's sequence number. To synchronize sequence numbers, the server includes its own ISN. Both the SYN and ACK flags are set within the TCP header.

3. ACK Packet:

In response to the server's SYN-ACK, the client sends an ACK (acknowledge) packet. This packet confirms receipt of the server's SYN-ACK by setting the acknowledgment number to one greater than the server's sequence number. The ACK flag is activated, while the SYN flag remains unset, indicating the successful establishment of the TCP connection.

Post-Handshake Connection

- A reliable connection is now established between the client and the server.
- Both the client and server maintain state information about the connection, such as sequence numbers and acknowledgment numbers.
- This ensures that data is transmitted reliably and in order.
- The client can now start sending HTTP requests to the server, and the server can send back the corresponding HTTP responses.

Additional Note: Multiple IP Addresses

- During the DNS lookup process, the DNS server often returns multiple IP addresses for a single domain, such as `www.facebook.com`.
- This is done for load balancing and redundancy purposes.
- The browser typically chooses the first IP address returned by the DNS server but can try other IP addresses if the first connection attempt fails.
- This mechanism enhances the availability and reliability of the service.

Server-Side Load Balancing and Connection Handling

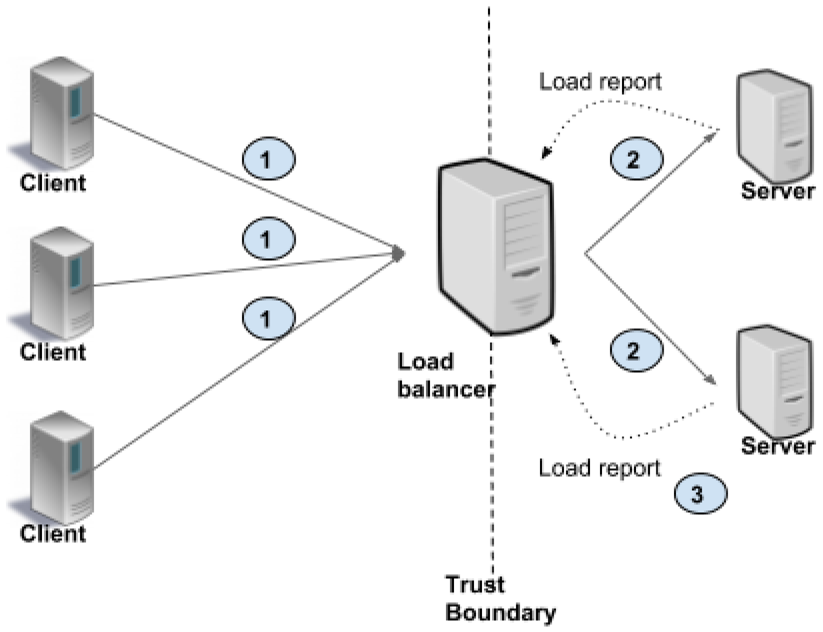
After the TCP connection is established, the HTTP request is received by Facebook's infrastructure. The following steps typically occur to ensure efficient handling and processing of the request:

Load Balancing

- DNS load balancing is a technique employed to distribute incoming traffic across multiple servers by providing distinct IP addresses for a single domain name. This approach serves to optimize performance and enhance system reliability by preventing the overloading of individual servers.

- A variety of algorithms, including round-robin, least connections, and IP hashing, can be implemented to distribute incoming requests among multiple IP addresses, each representing a separate load balancer. Upon reaching one of these designated IP addresses, the traffic is directed to a load balancer.

- The primary function of a load balancer is to evenly distribute incoming requests among a cluster of backend servers. This process ensures that no single server experiences an excessive workload, thereby maintaining optimal system performance and availability.



Listening Socket

- **Initial Connection Handling**:
    - The server has a listening socket on port 80 (for HTTP) or port 443 (for HTTPS). This socket is in an infinite loop, waiting for incoming connection requests.
    - When the server receives a connection request on this port, it accepts the connection and creates a new socket dedicated to communicating with your client.

Processing the Request

- **Parsing the Request**:
    - The server reads the HTTP request from the socket. This involves parsing the request line (e.g., GET /index.html HTTP/1.1), headers, and any body content.
- **Generating the Response**:
    - Based on the request, the server generates an appropriate HTTP response. This might involve querying databases, accessing files, or performing computations.
- **Sending the Response**:
    - The server sends the HTTP response back to your client through the dedicated socket. This response includes the status line (e.g., HTTP/1.1 200 OK), headers (like Content-Type), and the body (e.g., the HTML of the requested webpage).

Connection Management in Operating Systems

Source Ports for Multiple Connections

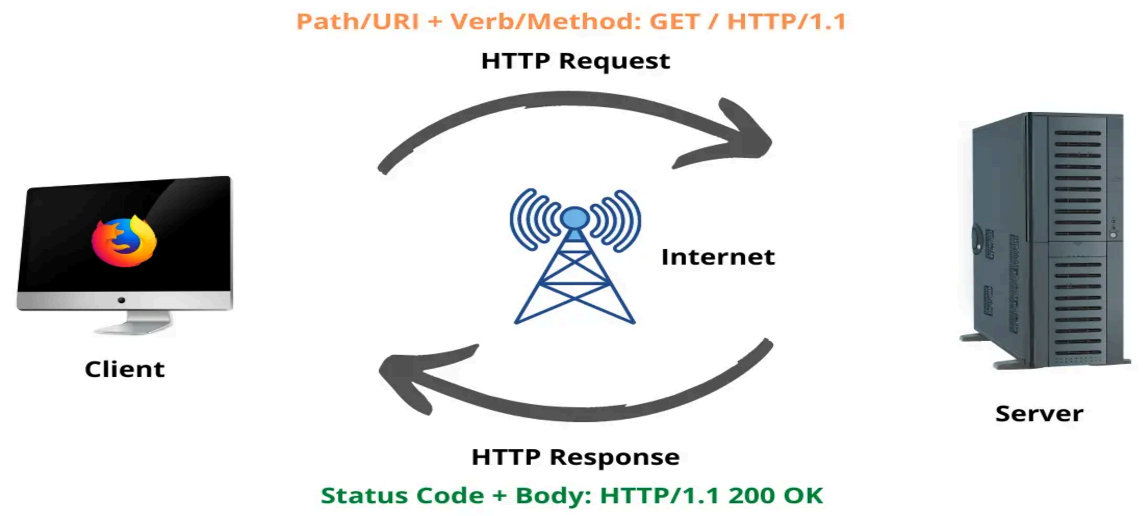
To distinguish between multiple connections originating from a single device, operating systems assign unique source ports to each. This ensures that incoming data is directed to the correct application. For instance, if you're browsing the web with Chrome (localhost:45678) and simultaneously transferring a file with a download manager (localhost:54321), the OS uses these port numbers to differentiate between the two connections. A connection table within the OS meticulously records source and destination IP addresses, along with their corresponding ports, preventing conflicts and ensuring seamless communication.

Handling Multiple Incoming Requests on a Single Port

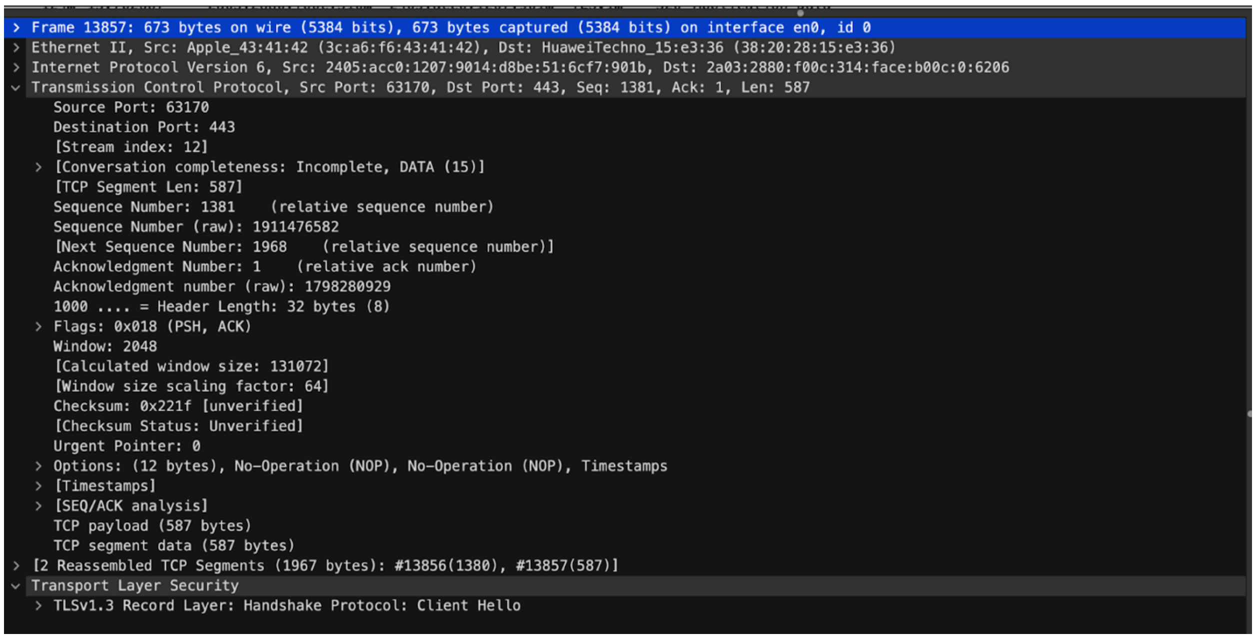
- **Listening and Dedicated Sockets**:
    - A web server handles multiple users' incoming requests on a single port (like port 80) by using a combination of sockets and concurrency mechanisms.
    - When a client connects to the server, it uses a "listening socket" to accept the connection and then creates a "dedicated connection socket" for each user. This dedicated socket manages the communication with that specific client, allowing the server to handle multiple connections simultaneously.
- **Concurrency Mechanisms**:
    - The server can use techniques like multi-threading, multi-processing, or asynchronous I/O to manage these concurrent connections efficiently. This way, even though all requests come in on the same port, the server can process each one separately and respond appropriately to each user.

HTTP Request and Response

After establishing a TCP/IP connection, the browser proceeds to communicate with the server using the Hypertext Transfer Protocol Secure (HTTPS). The HTTPS protocol ensures that data exchanged between the client and server is encrypted and secure. This section describes the analysis of the HTTP request sent by the client and the response received from the server.


Path/URI + Verb/Method: GET / HTTP/1.1
HTTP Request
Client
Internet
Server
HTTP Response
Status Code + Body: HTTP/1.1 200 OK

Sending the HTTP Request



```
> Frame 13857: 673 bytes on wire (5384 bits), 673 bytes captured (5384 bits) on interface en0, id 0
> Ethernet II, Src: Apple_43:41:42 (3c:a6:f6:43:41:42), Dst: HuaweiTechno_15:e3:36 (38:20:28:15:e3:36)
> Internet Protocol Version 6, Src: 2405:acc0:1207:9014:d8be:51:6cf7:901b, Dst: 2a03:2880:f00c:314:face:b00c:0:6206
v Transmission Control Protocol, Src Port: 63170, Dst Port: 443, Seq: 1381, Ack: 1, Len: 587
    Source Port: 63170
    Destination Port: 443
    [Stream index: 12]
  > [Conversation completeness: Incomplete, DATA (15)]
    [TCP Segment Len: 587]
    Sequence Number: 1381    (relative sequence number)
    Sequence Number (raw): 1911476582
    [Next Sequence Number: 1968    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 1798280929
    1000 .... = Header Length: 32 bytes (8)
  > Flags: 0x018 (PSH, ACK)
    Window: 2048
    [Calculated window size: 131072]
    [Window size scaling factor: 64]
    Checksum: 0x221f [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  > Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  > [Timestamps]
  > [SEQ/ACK analysis]
    TCP payload (587 bytes)
    TCP segment data (587 bytes)
> [2 Reassembled TCP Segments (1967 bytes): #13856(1380), #13857(587)]
v Transport Layer Security
  > TLSv1.3 Record Layer: Handshake Protocol: Client Hello
```

Analysis of the Request Packet (Frame 13857)

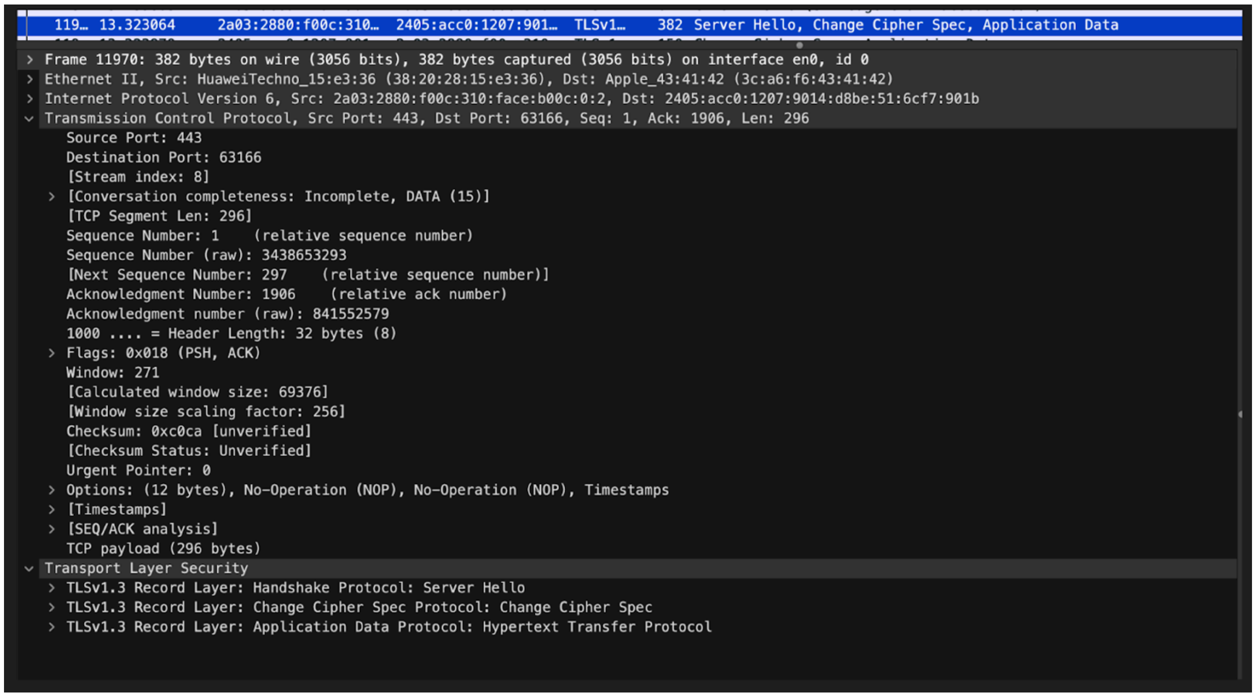
- **Frame Details**
  - Size: 673 bytes on wire (5384 bits), 673 bytes captured.
  - Interface: en0 (likely a wired or wireless network interface).
- **Ethernet II Layer**
  - Source MAC Address: Apple_43:41:42 (3c:a6:f6:43:41:42)
  - Destination MAC Address: HuaweiTechno_15:e3:36 (38:20:28:15:e3:36)
- **Internet Protocol Version 6 (IPv6)**
  - Source IP Address: 2405:acc0:1207:9014:d8be:51:6cf7:901b
  - Destination IP Address: 2a03:2880:f00c:314:face:b00c:0:6206
- **Transmission Control Protocol (TCP)**
  - Source Port: 63170
  - Destination Port: 443 (indicating HTTPS traffic)
  - Stream Index: 12
  - Sequence Number: 1381 (relative sequence number)
  - Next Sequence Number: 1968 (relative sequence number)
  - Acknowledgment Number: 1 (relative ack number)
  - Header Length: 32 bytes (8)
  - Flags: 0x018 (PSH, ACK)
    - PSH (Push): Data should be pushed to the receiving application immediately.
    - ACK (Acknowledgment): Acknowledges receipt of data.
  - Window Size: 2048
  - Checksum: 0x221f [unverified]
  - TCP Payload: 587 bytes
- **Reassembled TCP Segments**
  - 2 Reassembled TCP Segments: #13856 (1380 bytes), #13857 (587 bytes)
- **Transport Layer Security (TLS)**
  - TLSv1.3 Record Layer: Handshake Protocol: Client Hello

Explanation

- **TCP Layer**: The TCP layer indicates this is a packet sent from a high-numbered source port (63170) to the standard HTTPS port (443). The presence of the PSH and ACK flags indicates that this packet is pushing data to the application layer and acknowledging the receipt of data.
- **TLS**: The "Client Hello" message in the TLS layer indicates the beginning of the TLS handshake process, where the client initiates secure communication with the server. This is typical in HTTPS traffic as the client and server negotiate encryption settings.

Receiving the HTTP Response

Upon processing the request, the server constructs an HTTP response comprising a status line, headers, and a body containing the requested data. The browser interprets this response to render the content.



```
  119… 13.323064    2a03:2880:f00c:310…  2405:acc0:1207:901…  TLSv1…    382  Server Hello, Change Cipher Spec, Application Data
> Frame 11970: 382 bytes on wire (3056 bits), 382 bytes captured (3056 bits) on interface en0, id 0
> Ethernet II, Src: HuaweiTechno_15:e3:36 (38:20:28:15:e3:36), Dst: Apple_43:41:42 (3c:a6:f6:43:41:42)
> Internet Protocol Version 6, Src: 2a03:2880:f00c:310:face:b00c:0:2, Dst: 2405:acc0:1207:9014:d8be:51:6cf7:901b
v Transmission Control Protocol, Src Port: 443, Dst Port: 63166, Seq: 1, Ack: 1906, Len: 296
    Source Port: 443
    Destination Port: 63166
    [Stream index: 8]
  > [Conversation completeness: Incomplete, DATA (15)]
    [TCP Segment Len: 296]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 3438653293
    [Next Sequence Number: 297    (relative sequence number)]
    Acknowledgment Number: 1906    (relative ack number)
    Acknowledgment number (raw): 841552579
    1000 .... = Header Length: 32 bytes (8)
  > Flags: 0x018 (PSH, ACK)
    Window: 271
    [Calculated window size: 69376]
    [Window size scaling factor: 256]
    Checksum: 0xc0ca [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  > Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  > [Timestamps]
  > [SEQ/ACK analysis]
    TCP payload (296 bytes)
v Transport Layer Security
  > TLSv1.3 Record Layer: Handshake Protocol: Server Hello
  > TLSv1.3 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
  > TLSv1.3 Record Layer: Application Data Protocol: Hypertext Transfer Protocol
```

Analysis of the Response Packet (Frame 11970)

- **Frame Details**
  - Size: 382 bytes on wire (3056 bits), 382 bytes captured.
  - Interface: en0 (likely a wired or wireless network interface).
- **Ethernet II Layer**
  - Source MAC Address: HuaweiTechno_15:e3:36 (38:20:28:15:e3:36)
  - Destination MAC Address: Apple_43:41:42 (3c:a6:f6:43:41:42)
- **Internet Protocol Version 6 (IPv6)**
  - Source IP Address: 2a03:2880:f00c:310:face:b00c:0:2
  - Destination IP Address: 2405:acc0:1207:9014:d8be:51:6cf7:901b
- **Transmission Control Protocol (TCP)**
  - Source Port: 443 (indicating HTTPS traffic)
  - Destination Port: 63166
  - Stream Index: 8
  - Sequence Number: 1 (relative sequence number)
  - Next Sequence Number: 297 (relative sequence number)
  - Acknowledgment Number: 1906 (relative ack number)
  - Header Length: 32 bytes (8)
  - Flags: 0x018 (PSH, ACK)
    - PSH (Push): Data should be pushed to the receiving application immediately.
    - ACK (Acknowledgment): Acknowledges receipt of data.
  - Window Size: 271
  - Checksum: 0xc0ca [unverified]
  - TCP Payload: 296 bytes

- **Transport Layer Security (TLS)**
  - TLSv1.3 Record Layer: Handshake Protocol: Server Hello
  - TLSv1.3 Record Layer: Change Cipher Spec Protocol
  - TLSv1.3 Record Layer: Application Data Protocol: Hypertext Transfer Protocol

Explanation

- **TCP Layer**: The TCP layer indicates that this is a packet sent from the standard HTTPS port (443) to a high-numbered destination port (63166). The presence of the PSH and ACK flags indicates that this packet is pushing data to the application layer and acknowledging the receipt of data.
- **TLS**: The "Server Hello" message in the TLS layer indicates the server's response in the TLS handshake process, confirming the start of secure communication. The subsequent "Change Cipher Spec" and "Application Data" protocols indicate the transition to encrypted communication and the delivery of the actual HTTP response content.

Potential Errors in HTTP Communication

The process of transmitting and receiving HTTP requests and responses is susceptible to various errors.

DNS Errors:

- DNS Lookup Failure: The domain name cannot be resolved to a corresponding IP address.
- DNS Timeout: The DNS server fails to respond within the allotted timeframe.

TCP/IP Errors:

- Connection Refusal: The server declines the connection establishment, often due to firewall restrictions.
- Connection Timeout: The server fails to respond within a reasonable duration.

TLS/SSL Errors:

- SSL Certificate Validation Failure: The server's SSL certificate is deemed invalid or expired.
- TLS Handshake Failure: The secure communication establishment process terminates prematurely.

HTTP Errors:

- 404 Not Found: The requested resource cannot be located on the server.
- 500 Internal Server Error: The server encounters an unexpected condition, preventing request fulfillment.
- 403 Forbidden: The server denies access to the requested resource due to insufficient permissions.
- 502 Bad Gateway: The server, acting as a gateway, receives an invalid response from the upstream server.

Server Response

Upon processing a client request, the server generates an HTTP response. This response consists of three primary components:

- **Status Line:** This initial segment conveys the request's outcome, including the HTTP version, a status code (e.g., 200 for success, 404 for not found), and a descriptive phrase.
- **Headers:** Metadata about the response is provided through headers. Examples include Content-Type (specifying data format), Content-Length (indicating data size), and Set-Cookie (for session management).
- **Body:** The requested data is contained within the response body. This can range from HTML for web pages to images, JSON, or other formats.

These steps complete the process of sending an HTTP request and receiving an HTTP response, ultimately resulting in the rendering of the requested web page in the browser.

Conclusion

Upon initiating a visit to facebook.com by clicking the URL, a meticulously orchestrated series of events unfolds behind the scenes. This intricate process, designed for efficiency and reliability, culminates in the presentation of the desired webpage.

The journey begins with **URL parsing**. The browser meticulously dissects the URL, meticulously breaking it down into its fundamental components to ensure proper interpretation. This parsed information lays the groundwork for subsequent steps.

Following this, the **Domain Name System (DNS)** steps in. This critical system plays a pivotal role in translating the human-readable domain name (facebook.com) into a numerical IP address that computers can directly understand. To expedite this process, the DNS utilizes a layered caching system and a series of recursive queries.

Once the IP address is acquired, the browser establishes a reliable communication channel with the Facebook server. This vital connection is facilitated through a **TCP/IP three-way handshake**. This handshake serves as a critical protocol, guaranteeing reliable data exchange between the client (browser) and the server.

With a secure connection established, the browser transmits an **HTTP request** to the server. This request conveys the specific resource being sought, such as the Facebook homepage. The server then meticulously processes the request, retrieving and assembling the necessary information.

For optimal performance and scalability, server-side **load balancing mechanisms** are often employed. These mechanisms strategically distribute incoming traffic across numerous servers, ensuring that no single server becomes overloaded, thereby maintaining high performance and reliability.

This intricate dance underpins the smooth and reliable access to web services like Facebook.com that we take for granted in today's digital world.

References

- **Fielding, R., et al. (1999).** Hypertext Transfer Protocol -- HTTP/1.1. RFC 2616. https://www.ietf.org/rfc/rfc2616.txt
- **Albitz, P., & Liu, C. (2006).** DNS and BIND (5th Edition). O'Reilly Media. https://www.oreilly.com/library/view/dns-and-bind/0596100574/
- **Stevens, W. R. (1994).** TCP/IP Illustrated, Volume 1: The Protocols. Addison-Wesley. https://www.informit.com/
- **Incapsula (2021).** What is Load Balancing? Imperva. https://www.imperva.com/products/load-balancer/
- **Rescorla, E. (2018).** The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446. https://datatracker.ietf.org/doc/html/rfc8446
- **GeeksforGeeks. (n.d.). Retrieved from** https://www.geeksforgeeks.org/