**WHAT HAPPENS WHEN WE TYPE FACEBOOK.COM?**

Nabin Shrestha -PAS078BEI024

Anil Poudel -PAS078BEI004

Mijash Bhugai -PAS078BEI022

Sagar Joshi -PAS078BEI32

Table of Contents

[Introduction 3](#_Toc173449309)

[Purpose of the Report 3](#_Toc173449310)

[Overview of the Process 3](#_Toc173449311)

[URL Parsing 3](#_Toc173449312)

[Components of a URL 3](#_Toc173449313)

[DNS Lookup 4](#_Toc173449314)

[Browser Cache 4](#_Toc173449315)

[OS Cache 4](#_Toc173449316)

[Router Cache 4](#_Toc173449317)

[ISP DNS Cache 4](#_Toc173449318)

[Recursive DNS Lookup 4](#_Toc173449319)

[DNS Recursive Resolver 4](#_Toc173449320)

[DNS Root Nameserver 5](#_Toc173449321)

[TLD Nameserver 5](#_Toc173449322)

[Authoritative Name Server 5](#_Toc173449323)

[Establishing a TCP/IP Connection 5](#_Toc173449324)

[The TCP Handshake Process 5](#_Toc173449325)

[SYN (Synchronize) Packet 5](#_Toc173449326)

[SYN-ACK (Synchronize-Acknowledge) Packet 5](#_Toc173449327)

[ACK (Acknowledge) Packet 6](#_Toc173449328)

[Post-Handshake Connection 6](#_Toc173449329)

[HTTP Request and Response 8](#_Toc173449338)

[Sending the HTTP Request 8](#_Toc173449339)

[Receiving the HTTP Response 10](#_Toc173449340)

[Possible Errors 11](#_Toc173449341)

[Conclusion 12](#_Toc173449342)

[References 12](#_Toc173449343)

# Introduction

## Purpose of the Report

The purpose of this report is to provide a detailed and comprehensive explanation of the technical processes that occur when a user clicks on facebook.com. By dissecting each step from the initial URL entry to the final rendering of the Facebook homepage, this report aims to elucidate the complex interactions and technologies involved in delivering a seamless web browsing experience.

## Overview of the Process

When a user clicks on facebook.com, a series of events is triggered, starting with the parsing of the URL and followed by a DNS lookup to resolve the domain name into an IP address. Once the IP address is obtained, the browser establishes a TCP/IP connection with the Facebook server. This connection involves a three-way handshake to ensure reliable communication. After establishing the connection, the browser sends an HTTP request to the server and receives an HTTP response. Finally, the browser processes the response and renders the Facebook homepage. Each of these steps is crucial in ensuring that the user can access the desired webpage quickly and efficiently.

# 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

1. **Protocol**: The protocol is the set of rules used for data transmission. In the case of Facebook, it is usually https://, indicating a secure communication channel.
2. **Domain**: The domain is the main address of the website. For Facebook, the domain is www.facebook.com.
3. **Port**: This is often implicit and not visible in the URL. For HTTPS, the default port is 443, while for HTTP, it is 80.

# **Understanding HSTS and the Preloaded HSTS List**

# **What is HSTS?**

# HTTP Strict Transport Security (HSTS) is a web security policy mechanism that helps to protect websites against certain types of attacks, such as protocol downgrade attacks and cookie hijacking. HSTS ensures that browsers only communicate with the server over HTTPS, not HTTP.

# **Preloaded HSTS List**

# Modern web browsers include a "preloaded HSTS list," which is a list of websites that have requested to always be contacted via HTTPS. This list helps to enforce the use of HTTPS from the first connection, providing additional security.

# **How It Works**

# 1. **Browser Checks the HSTS List**:

# - When a user enters a URL, the browser first checks its preloaded HSTS list.

# - This list contains websites that have explicitly requested to be contacted via HTTPS only.

# 2. **HTTPS Request**:

# - If the website is in the HSTS list, the browser automatically sends the request via HTTPS.

# - If the website is not in the list, the initial request is sent via HTTP.

# 3. **HSTS Policy Without Preloading**:

# - Websites not in the HSTS list can still use the HSTS policy.

# - The first HTTP request to such a website will receive a response that instructs the browser to use HTTPS for future requests.

# - However, this initial HTTP request can leave the user vulnerable to a downgrade attack, which is why the preloaded HSTS list is beneficial.

# **Example Scenario**

# 1. **Website in HSTS List**:

# - User enters `facebook.com`.

# - Browser checks the HSTS list and finds `facebook.com`.

# - Browser sends the request via HTTPS: `https://facebook.com`.

# 2. **Website Not in HSTS List**:

# - User enters `newsite.com`.

# - Browser does not find `newsite.com` in the HSTS list.

# - Browser sends the initial request via HTTP: `http://newsite.com`.

# - The server responds with an HSTS header, instructing the browser to use HTTPS for future requests.

# **Why HSTS List Matters**

# - **Security**: The preloaded HSTS list helps protect users by ensuring the first connection to a website is secure.

# -**Mitigation of Attacks**: It mitigates the risk of downgrade attacks and other types of attacks that exploit the initial insecure connection.

# DNS Lookup

After the URL has been parsed, the next step is to resolve the domain name to an IP address through the Domain Name System (DNS) lookup. This involves multiple stages of caching and recursive queries to ensure efficient and accurate resolution.

## 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 Nameserver

The root nameservers are the top-level servers in the DNS hierarchy. There are 13 root nameservers worldwide, each with multiple copies to ensure reliability and speed. The root nameserver does not know the exact IP address of www.facebook.com but knows the authoritative servers for the top-level domain (TLD) .com.

### TLD Nameserver

The recursive resolver then queries a TLD nameserver 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 holds the DNS records for all subdomains of facebook.com, including www.facebook.com. It responds to the recursive resolver with the IP address of the Facebook server.

The recursive resolver caches this information and returns the IP address to the browser, completing the DNS lookup process. This allows the browser to initiate a connection to the Facebook server using the resolved IP address.

# 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

### SYN (Synchronize) Packet

* + The client (browser) sends a SYN packet to the Facebook server.
  + The packet includes an initial sequence number (ISN), a randomly chosen number serving as the starting point for the sequence of bytes.
  + The SYN flag in the TCP header is set to indicate that this packet is for initiating a connection.

### SYN-ACK (Synchronize-Acknowledge) Packet

* + The Facebook server responds with a SYN-ACK packet upon receiving the SYN packet.
  + This packet acknowledges the client's SYN packet with an acknowledgment number one more than the client's sequence number.
  + It synchronizes the server's sequence numbers with the client by including the server's own initial sequence number.
  + Both the SYN and ACK flags in the TCP header are set.

### ACK (Acknowledge) Packet

* + The client responds to the server’s SYN-ACK with an ACK packet.
  + This packet acknowledges the server’s SYN-ACK by setting the acknowledgment number to one more than the server's sequence number.
  + The ACK flag is set in the TCP header, and the SYN flag is not set, indicating that the connection is now established.

## 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.

## 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).

# 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.

## Sending the HTTP Request

#### A screenshot of a computer Description automatically generated

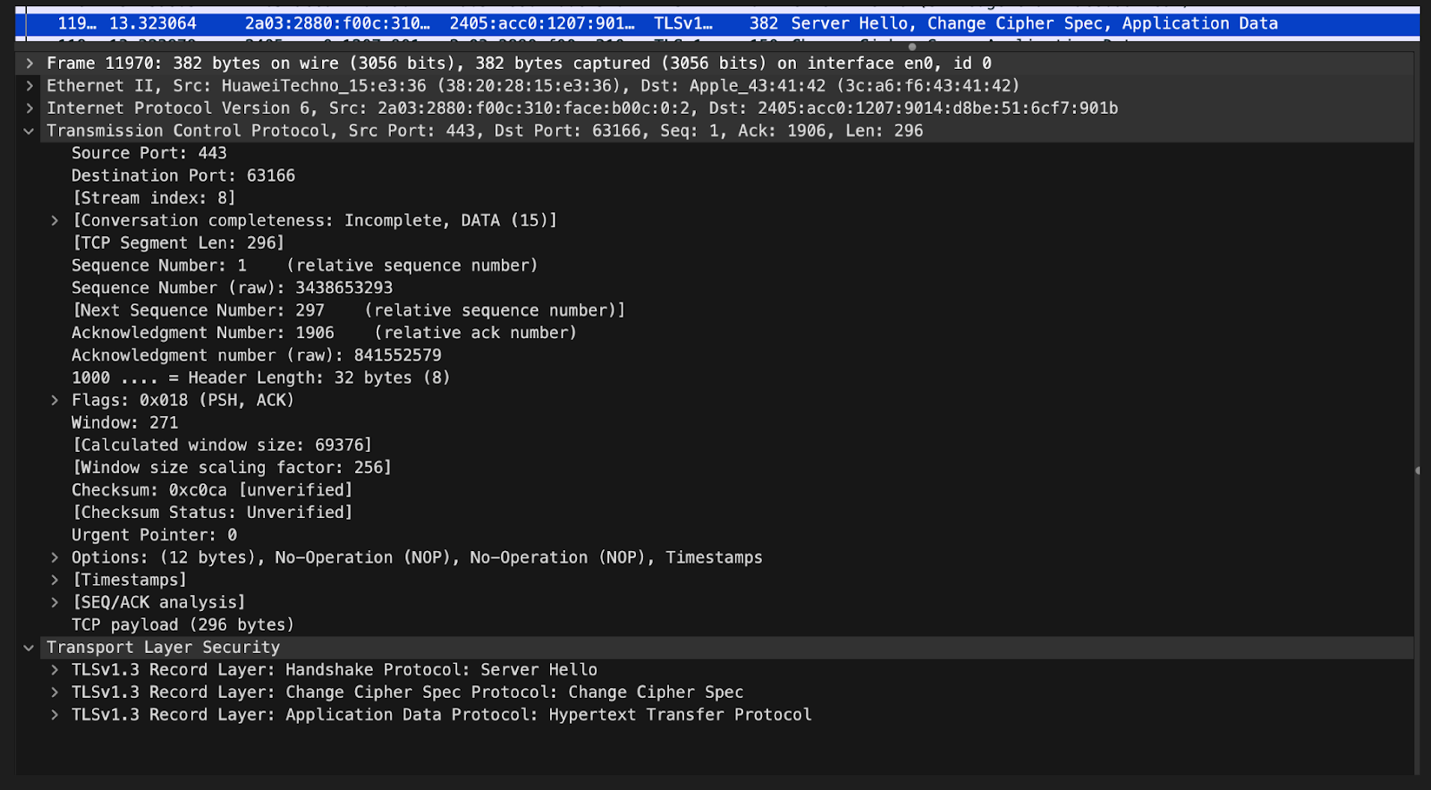
**Analysis of the Request Packet (Frame 13857)**

* **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



**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.

# Possible Errors

During the process of sending the HTTP request and receiving the HTTP response, several errors can occur:

1. **DNS Errors**
   * **DNS Lookup Failed**: The browser cannot resolve the domain name to an IP address.
   * **DNS Timeout**: The DNS server takes too long to respond.
2. **TCP/IP Errors**
   * **Connection Refused**: The server refuses the connection, possibly due to firewall settings.
   * **Connection Timeout**: The server does not respond within a reasonable amount of time.
3. **TLS/SSL Errors**
   * **SSL Certificate Error**: The server's SSL certificate is invalid or expired.
   * **Handshake Failure**: The TLS handshake process fails, preventing secure communication.
4. **HTTP Errors**
   * **404 Not Found**: The requested resource is not found on the server.
   * **500 Internal Server Error**: The server encounters an unexpected condition.
   * **403 Forbidden**: The server refuses to fulfill the request due to insufficient permissions.
   * **502 Bad Gateway**: The server, acting as a gateway, receives an invalid response from the upstream server.

**Server Response**

* The server processes the request and sends back an HTTP response. This response includes:
  + **Status Line**: Indicates the HTTP version, status code (e.g., 200 OK, 404 Not Found), and a reason phrase.
  + **Headers**: Provide metadata such as Content-Type (e.g., text/html), Content-Length, Set-Cookie (for session management), and others.
  + **Body**: Contains the actual content requested, such as HTML, images, or JSON data.

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

# Conclusion

When a user clicks on facebook.com, a series of sophisticated steps occur behind the scenes to deliver the requested webpage efficiently and reliably. The process begins with URL parsing, where the browser dissects the URL into its components. The DNS lookup then resolves the domain name into an IP address, involving multiple caching layers and recursive lookups.

Once the IP address is obtained, the browser establishes a TCP/IP connection with the Facebook server using a three-way handshake. This connection ensures reliable communication. The HTTP request is then sent to the server, which processes it and generates an appropriate HTTP response.

By understanding these processes, you can appreciate the complexity and efficiency of modern web infrastructure, ensuring reliable and quick access to web services like Facebook.

# References

1. **HTTP/1.1: Protocol Overview**
   * Fielding, R., et al. "Hypertext Transfer Protocol -- HTTP/1.1." RFC 2616, 1999.
   * URL: https://tools.ietf.org/html/rfc2616
2. **DNS and BIND**
   * Albitz, P., & Liu, C. "DNS and BIND." O'Reilly Media, 5th Edition, 2006.
   * URL: https://www.oreilly.com/library/view/dns-and-bind/0596100574/
3. **TCP/IP Illustrated, Volume 1: The Protocols**
   * Stevens, W. R. "TCP/IP Illustrated, Volume 1: The Protocols." Addison-Wesley, 1994.
   * URL: <https://www.informit.com/store/tcp-ip-illustrated-volume-1-the-protocols-9780201633467>
4. **TLS and SSL Protocols**
   * Rescorla, E. "The Transport Layer Security (TLS) Protocol Version 1.3." RFC 8446, 2018.
   * URL: https://tools.ietf.org/html/rfc8446