
# TRIBHUVAN UNIVERSITY
# INSTITUTE OF ENGINEERING
# PASCHIMANCHAL CAMPUS

# COMPUTER NETWORK

**Submitted By:**

| | |
|---|---|
| Aastha Paudel | (PAS077BCT002) |
| Anup Bashyal | (PAS077BCT009) |
| Bibek Timilsina | (PAS077BCT016) |
| Tek Singh Ayre | (PAS077BCT046) |

**Date:** July 12, 2024

# What happens when you type google.com in the browser?

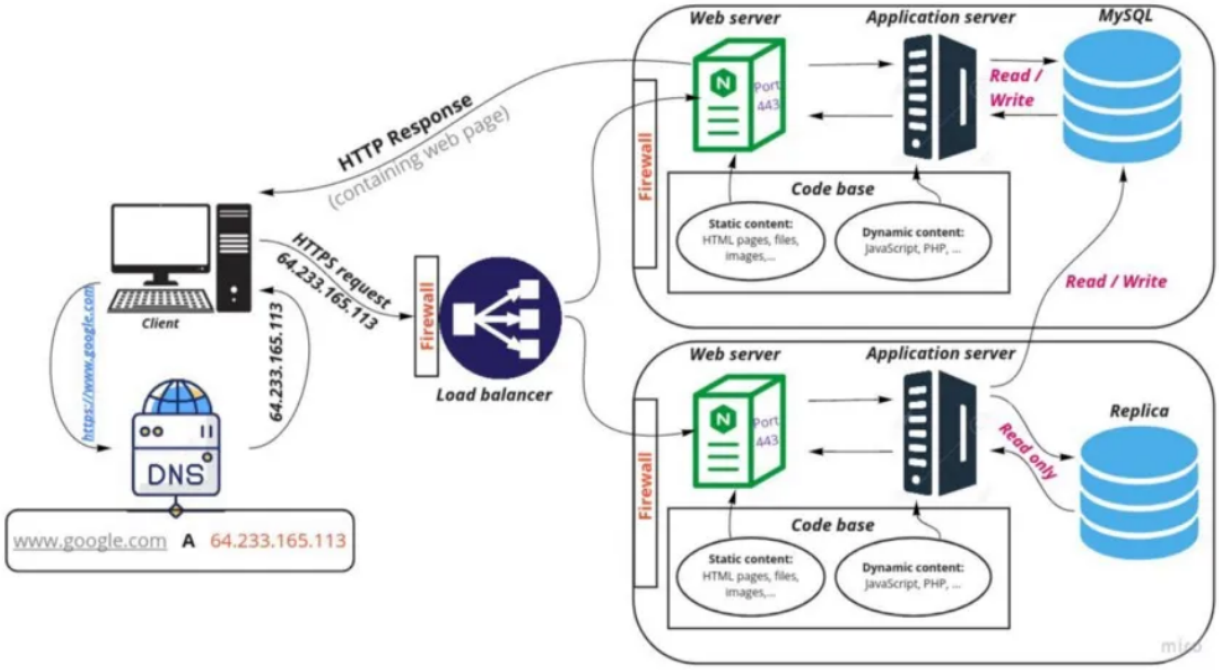


Figure 1: Process

# 1. DNS Lookup to Find IP Address (Application Layer)

When you type ”google.com” in your browser, several steps occur to fetch and display the webpage. Here’s a detailed breakdown of the process, integrating OSI model layers and technical details:

1. **User Action:** User types ”google.com” and presses Enter in the browser.
2. **Browser Request:** Browser sends a DNS lookup request to the local DNS resolver.
3. **Local DNS Resolver:**
    - **Cache Check:** Checks its cache for the DNS record of ”google.com”.
    - **Recursive Query:** If not cached, sends a recursive query to root DNS servers, then to TLD (Top-Level Domain) servers (.com), and finally to the authoritative DNS server for ”google.com”.
4. **Authoritative DNS Server:** Responds with the IP address of google.com.
5. **Response:** Local DNS resolver sends the IP address back to the browser.

Overall ,When a user types ’google.com’ into their browser, the process begins with the browser sending a DNS lookup request to the local DNS resolver. Initially, the resolver checks its cache to see if it already has the IP address for ’google.com’. If not cached, it performs a recursive query starting from root DNS servers, then to the Top-Level Domain (TLD) servers for ’.com’, and finally reaches the authoritative DNS server for ’google.com’. Upon receiving the query, the authoritative DNS server responds with the IP address associated with ’google.com’. This IP address is then returned to the browser by the local DNS resolver. This crucial step in web browsing ensures that users can access websites using human-readable domain names while underlying systems manage the translation to machine-readable IP addresses seamlessly .

The browser sends a request to the local DNS resolver, which is often provided by the internet service provider(ISP). The local DNS resolver checks its cache for the most recent copy of the DNS record for the domain. If it has it, it sends the IP address back to the browser. If the local DNS resolver doesn’t have the most recent copy of the DNS record, it sends a request to a root nameserver. The root nameserver replies with the address of a top level domain (TLD) nameserver, such as .com

- The local DNS resolver sends a request to the TLD nameserver.
- The TLD nameserver responds with the address of the authoritative nameserver for the domain.
- The local DNS resolver sends a request to the authoritative nameserver.
- The authoritative nameserver responds with the IP address for the domain.
- The local DNS resolver sends the IP address back to the browser.
- The browser sends a request to the server at the IP address to retrieve the webpage.

If a DNS record is not found at any of the nameservers or when additional services like DNS load balancing or content delivery networks(CDNs) are involved, the DNS lookup process can become more complex. If the local DNS resolver doesn't have the DNS record cached, it initiates an iterative query process, asking root, top-level domain (TLD), and authoritative nameservers sequentially until it locates the required DNS record. In scenarios where DNS load balancing is employed, the authoritative nameserver might return multiple IP addresses for the same domain, distributing traffic across different servers to optimise load and ensure high load. Similarly, with CDNs, the DNS query might be directed to a specialised DNS service that returns an IP address based on the user's geographic location, ensuring the user is routed to the nearest and most efficient server. This can involve additional steps where the DNS service dynamically selects the best server, enhancing the speed and reliability of content delivery. In cases where the DNS record is not found at any of the nameservers, an NXDOMAIN response is generated, indicating that the domain does not exist.

## 2. Establish TCP/IP Connection (Transport Layer)

Once the DNS lookup has resolved the domain name(e.g., google.com) to an IP address, your browser needs to establish a connection with the server at that IP address to start exchanging data. This is done using the Transmission Control Protocol (TCP) over the Internet Protocol (IP), often referred to together as TCP/IP.

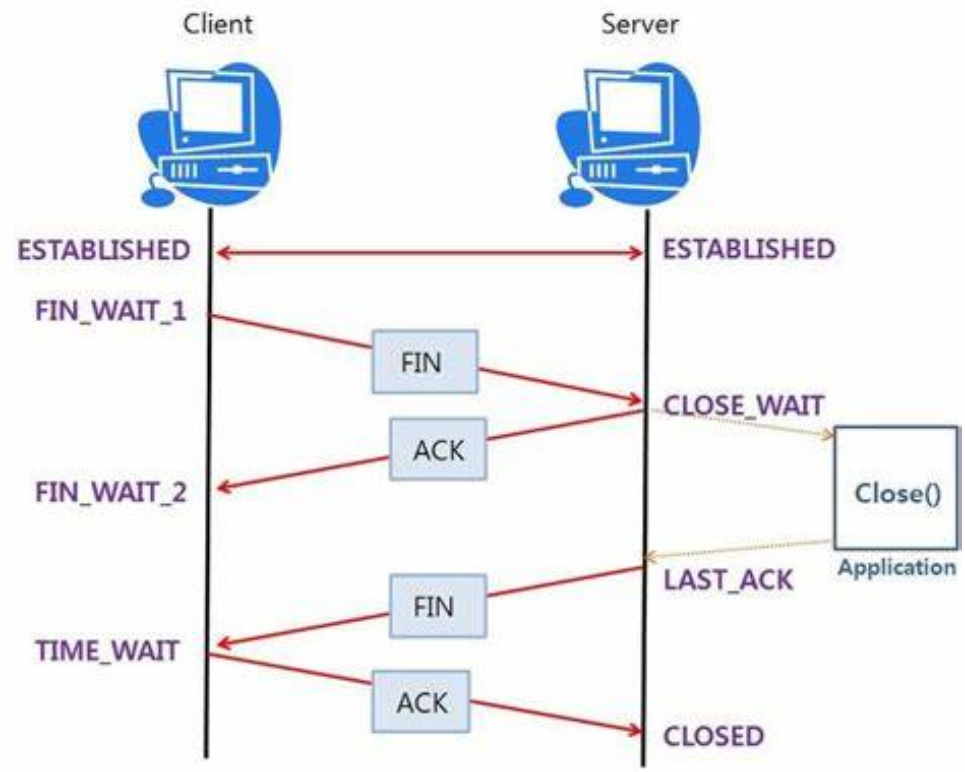


Figure 2: TCP 3-way Handshake

1. **Browser to Google Server:** Initiates a TCP (Transmission Control Protocol) connection to the retrieved IP address of google.com.

2. **Three-Way Handshake:**

    - **SYN (Synchronize):** Browser sends a SYN packet to Google's server.
    - **SYN-ACK (Synchronize-Acknowledgment):** Google's server responds with SYN-ACK packet.
    - **ACK (Acknowledgment):** Browser sends an ACK packet, establishing the TCP connection.

When the browser initiates a TCP (Transmission Control Protocol) connection to the IP address retrieved for 'google.com', it follows a process known as the three-way handshake. First, the browser sends a SYN (Synchronize) packet to Google's server, indicating its intention to establish a connection. Upon receiving the SYN packet, Google's server responds with a SYN-ACK (Synchronize-Acknowledgment) packet, acknowledging the request and signaling its readiness to proceed with the connection. Finally, the browser sends an ACK (Acknowledgment) packet back to the server, confirming the receipt of the SYN-ACK packet. This three-step process ensures that both the browser and the server agree on establishing a reliable and orderly connection for transmitting data. It forms the foundation of TCP/IP communication, enabling robust and error-checked data exchange between client and server in networked environments.

## 0.0.1 The TCP Handshake Process

SYN(Synchronize) Packet:

Initiation: Our computer starts by sending a TCP packet with the SYN (synchronize) flag set. This packet includes a randomly generated sequence number, which will be used to keep the track of the data sent and received. The SYN packet is like your computer calling the server, signalling that it wants to establish a connection and synchronize sequence numbers.

SYN - ACK (Synchronize - Acknowledgement) Packet:

Response: The Google server, upon receiving the SYN packet, responds with a packet that has both the SYN and ACK (acknowledge) flags set. This packet acknowledges receipt of your SYN packet by including your sequence number plus one, and it also contains the server's own sequence number. This SYN-ACK packet is the server answering your call and indicating it's ready to communicate.

ACK (Acknowledgement) Packet:

Finalization: Our Computer responds to the SYN-ACK packet with an ACK packet, acknowledging the server's sequence number by incrementing it by one.This ACK packet completes the handshake, confirming that both sides are ready to start data transmission.

## 0.0.2 Post Handshake: Data Transmission

Once the TCP handshake is complete, a reliable connection is established between the computer and the Google server. This connection is now ready for the exchange of data, such as sending HTTP requests and receiving HTTP responses.

- Data Packets: Data is sent in packets, each with a sequence number so that the recipient can reorder them if they arrive out of order. TCP ensures that all packets are received and can request retransmission if packets are lost.
- Acknowledgements: As data packets are received, the recipient sends back acknowledgements to confirm receipt. This ensures reliable communication.

### 0.0.3 Connection Termination

After the data exchange is complete, the connection is gracefully terminated using a similar handshake process, involving FIN (finish) and ACK packets to ensure that both sides agree to close the connection.

1. FIN Packet: One side (e.g., our computer) sends a FIN packet to indicate it wants to close the connection.
2. ACK Packet: The other side (e.g., the Google server) acknowledges the FIN packet.
3. FIN Packet: The server sends its own FIN packet.
4. ACK Packet: Your computer acknowledges the server's FIN packet.

This ensures that connection is properly closed and all resources are released.

## 3. HTTPS Secure Communication Begins (Transport and Application Layer)

1. **TLS Handshake:** If accessing via HTTPS (secure HTTP), browser and server negotiate encryption parameters:
    - **Encryption Negotiation:** Agree on encryption method and exchange encryption keys.
    - **Secure Channel:** Establish a secure channel for encrypted data transmission.

During the TLS handshake for HTTPS, the browser and server agree on how to encrypt data securely. They exchange encryption methods and keys, and then set up a secure channel to protect all information exchanged between them. This ensures that data sent between your browser and the website remains private and cannot be easily intercepted or understood by unauthorized parties.

## 4. Firewall Inspection (Network Layer)

1. **Firewall Check:** Request passes through any firewalls, which inspect and filter traffic based on security policies.

A firewall is a security system that regulates and monitors incoming and outgoing network traffic based on predetermined security policies. Its primary objective is to safeguard a network from external threats, such as hackers and malware.

When you type a URL like “google.com” into your browser, the request that your browser makes to Google’s server passes through the firewall en route. The firewall examines the incoming request to ensure that it is permitted based on its security policies.

There are two primary types of security policies that a firewall employs to examine incoming requests:

1. Policies that allow or prohibit traffic based on the origin and destination of the request. For example, a firewall may be programmed to block all traffic from specific countries or to allow only specific IP addresses to access the network.

2. Policies that allow or prohibit traffic based on the type of traffic. For example, a firewall may be programmed to block all traffic on certain ports (such as those used by malware) or to allow only certain types of traffic(such as HTTP or HTTPS)

## 5. Load Balancer Distribution (Application and Transport Layer)

1. **Load Balancing:** If Google uses a load balancer, it distributes incoming requests among multiple servers for efficient load management and scalability.

A site as massively popular as Google gets billions of requests from users across the globe every single day. A typical web server would crash under that kind of traffic overload.To handle their enormous scale, Google deploys what’s known as a load balancer. This is a reverse proxy that sits in front of the Google web servers and distributes requests evenly across their server infrastructure.

The load balancer acts like a receptionist, taking incoming requests and forwarding them to available servers for processing. Based on advanced algorithms, it selects an optimal Google web server and routes your connection to that location.

This allows Google to efficiently spread the work across their network of servers. The load balancer also improves redundancy and failover since requests can be shifted from a downed server to a functioning one.

## 6. Web Server Processes Request (Application Layer)

1. **Server Processing:** Web server at the IP address handles the request:

    - **Static Content:** Retrieves static resources (HTML, CSS, JavaScript, images) required to build the webpage.
    - **Dynamic Content:** May generate dynamic content based on user-specific data or interactions.

When the web server receives a request, it processes it by retrieving both static and dynamic content. Static content includes resources like HTML files, CSS stylesheets,

JavaScript files, and images, which are served as-is without modification. These elements form the basic structure and design of the webpage. On the other hand, dynamic content is generated on-the-fly based on user-specific data or interactions. This could involve executing server-side scripts, querying databases, or processing user inputs to customize the webpage content. By combining static and dynamic elements, the server ensures that each user receives a personalized and responsive web experience tailored to their needs.

## 7. Application Server Interaction (Application Layer)

1. **Dynamic Processing:** For interactive or personalized content (e.g., search results):
    - **Application Server:** Executes server-side scripts (e.g., PHP, Python) to process requests and interact with databases.
    - **Database Queries:** Retrieves relevant data from databases (e.g., user preferences, search results).

## 8. Web Server Sends Response (Application Layer)

1. **Response Preparation:** Constructs an HTTP response containing the fully rendered webpage.
2. **Optimization:** Compresses resources (e.g., minifies JavaScript, compresses images) for efficient data transfer.

## 9. Browser Receives and Renders Page (Application Layer)

1. **Page Rendering:** Browser receives the HTTP response:
    - **DOM Construction:** Parses HTML to create the Document Object Model (DOM) structure.
    - **CSS Application:** Applies CSS styles to the DOM for visual presentation.
    - **JavaScript Execution:** Executes JavaScript for interactivity and dynamic content updates.
    - **Rendering:** Displays the rendered webpage on the user's screen.

Once Google gathers all the necessary data and resources through DNS lookups, server requests, and security checks, it sends a complete webpage back to your browser. This includes HTML for structure, CSS for styling, JavaScript for interactivity, and media like images and videos. Your browser then builds the webpage, making it interactive with features like search and animations. It adjusts the layout and style, and finally displays everything on your screen. This process is secured with HTTPS encryption for privacy and optimized for speed using techniques like minifying files and using CDNs.

## 10. Cookies and Local Storage (Application Layer)

1. **Cookie Handling:** Stores cookies locally to track user sessions and preferences across visits to google.com.

2. **Local Storage:** Caches static resources locally for faster load times during repeat visits.

Google utilizes various types of cookies during user interactions to enhance functionality and personalize user experiences. These cookies are small text files that track browsing behavior, store sign-in credentials, and enable features like analytics, personalized ads, and remembering user preferences across Google products. They come in two main types:

1. First-party persistent cookies:These are used by Google directly to maintain basic site functionality and remember user settings over time.

2. Third-party session cookies: These are employed by other entities to track user sessions across different websites, often for advertising purposes.

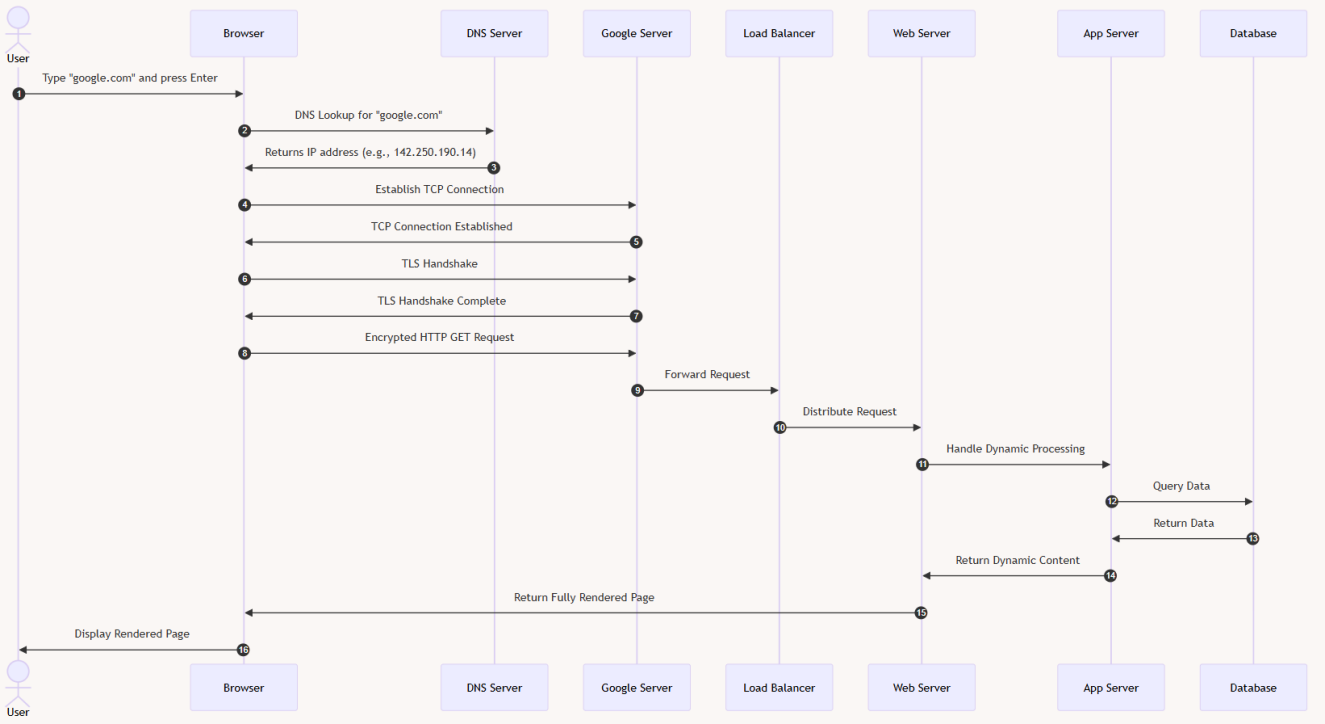


Figure 3: Sequence Diagram

## Conclusion

In conclusion, the process of accessing ”google.com” involves a series of steps across various OSI model layers, ensuring secure, efficient, and personalized web browsing experience. From DNS lookup to rendering the webpage in the browser, each stage plays a crucial role in delivering content seamlessly to users. Understanding this process enhances our knowledge of internet technologies and infrastructure.