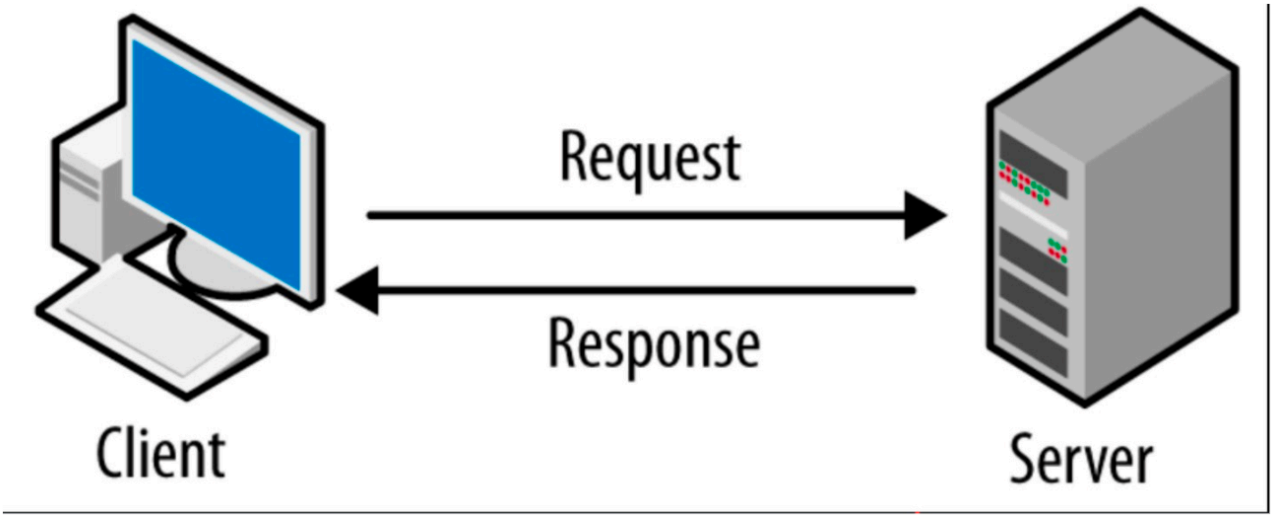
Understanding Web Architecture

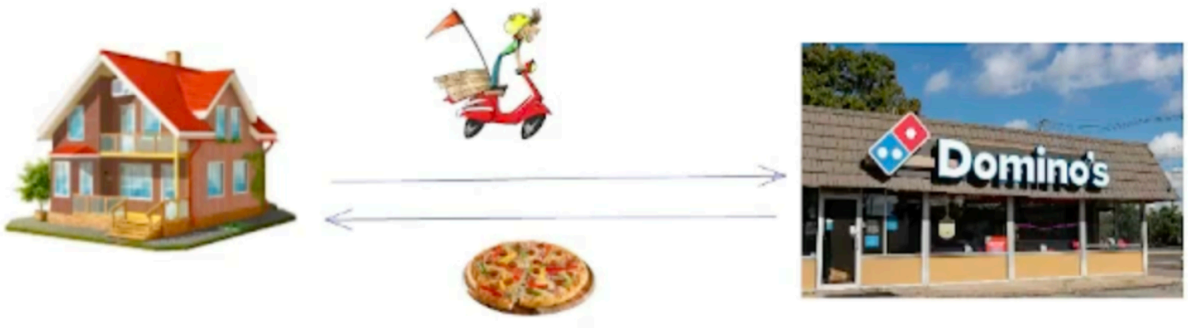
Rushikesh Patil
www.linkedin.com/in/rushikesh-patil-42304b196

When we want to access any website on the internet, we open a browser and enter the URL of that particular web page in the search bar (request). As a result, we receive the web page (response). Therefore, there are mainly two entities involved: the client, which is the place where we access the web page, and the server, which is the place from where we receive the web page.



Let's understand this with an example: Suppose I want to eat a pizza, and I asked my younger brother to get it for me. He hops on his bike, goes to the pizza store, picks up the pizza, and delivers it to me. Similarly, when we make a URL request, the browser needs to know where to get the data from.

Now, the question arises: How does the browser know where to get the data from?



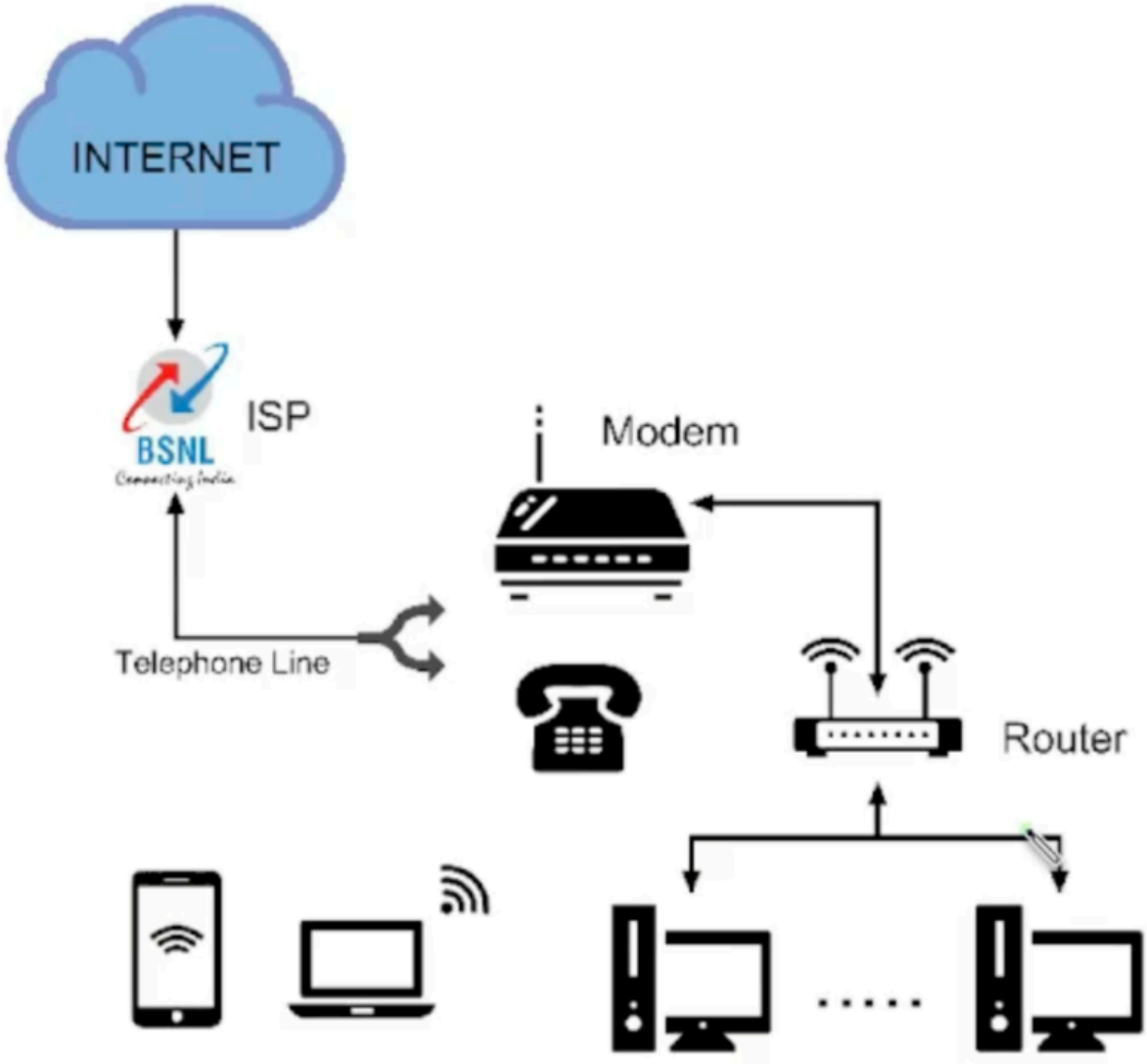
As there are multiple servers across the internet, the browser needs to locate the specific server that contains the data related to the requested URL. To achieve this, we search for the IP address associated with the requested URL. An IP address is a unique numeric value assigned to a device to locate it on the internet.

Therefore, the requested URL goes in search of the IP address mapped to it at the DNS (Domain Name Server). Once the IP address is obtained, it is sent back to the browser. With the help of the IP address, the browser then navigates to the particular server and retrieves the necessary data to load the web page.

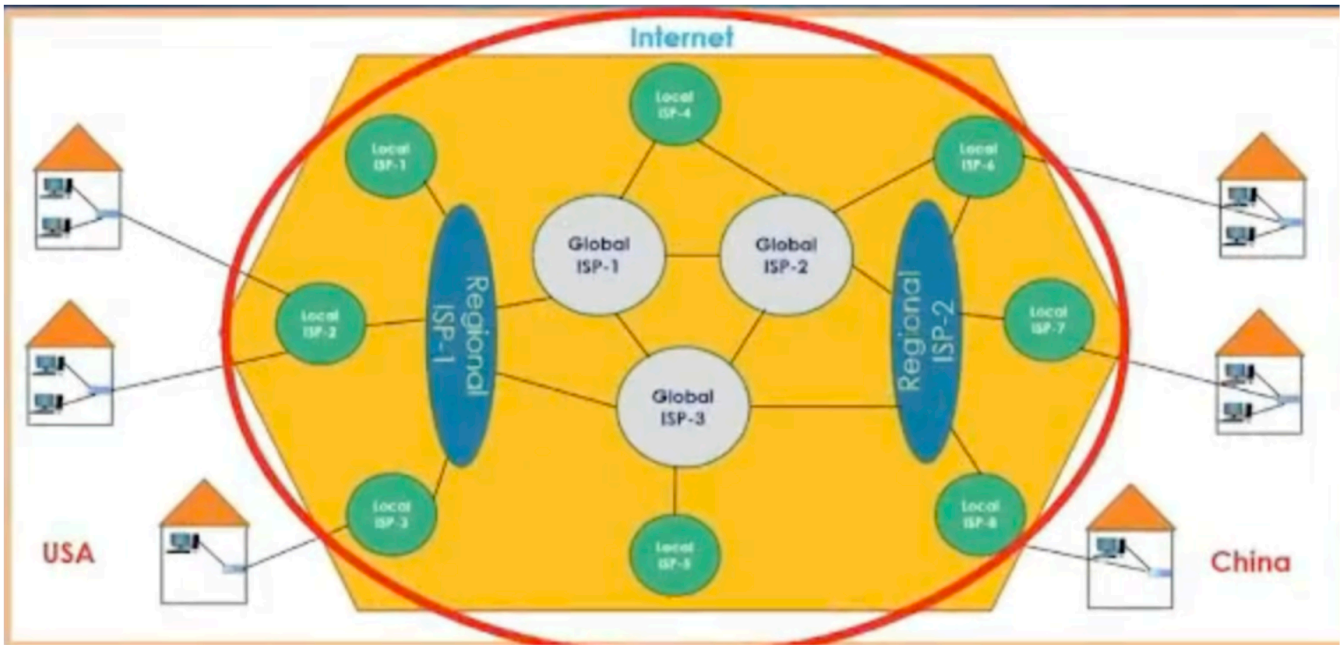
As depicted in above figure, there are two intermediaries between the client and the server that facilitate the transportation of information by establishing a robust network architecture.

1) Cell tower/router: This intermediary plays a crucial role in transmitting data between the client's device and the server. It acts as a connection point and helps relay the information across the network. In wireless communication, the cell tower serves as the primary access point, while routers help in routing data packets efficiently.

2) ISP (Internet Service Provider): ISPs, such as Jio, Airtel, BSNL, and others, serve as essential intermediaries in the network architecture. They provide internet connectivity to clients by establishing the link between their devices and the wider internet infrastructure. ISPs act as a bridge, enabling communication between the client and the server by transmitting data packets across their networks.



In above Figure, we observe that our laptop, PC, or phone is connected to the router in two ways: wired and wireless. The router then establishes an internet connection using an optic fiber cable (via a modem) or a telephone cable (in older setups). From there, the router connects directly to the ISP. These ISPs are in turn connected to the internet.



The URL from our home router reaches the local ISP and further travels to regional ISPs. These regional ISPs help in maintaining rules and regulations for data flow at the country level. Eventually, the data reaches global ISPs, which oversee rules and regulations at a global level. This infrastructure plays a crucial role in maintaining privacy and security.

Referring to above Figure, let's consider a scenario where we are located in the USA and need to retrieve data from a server located in China. In this case, our request first obtains the IP address from the DNS. The request then travels through the local ISP and regional ISP within the USA. Afterward, it passes through global ISPs, followed by the regional and local ISPs in China, until it reaches the server.

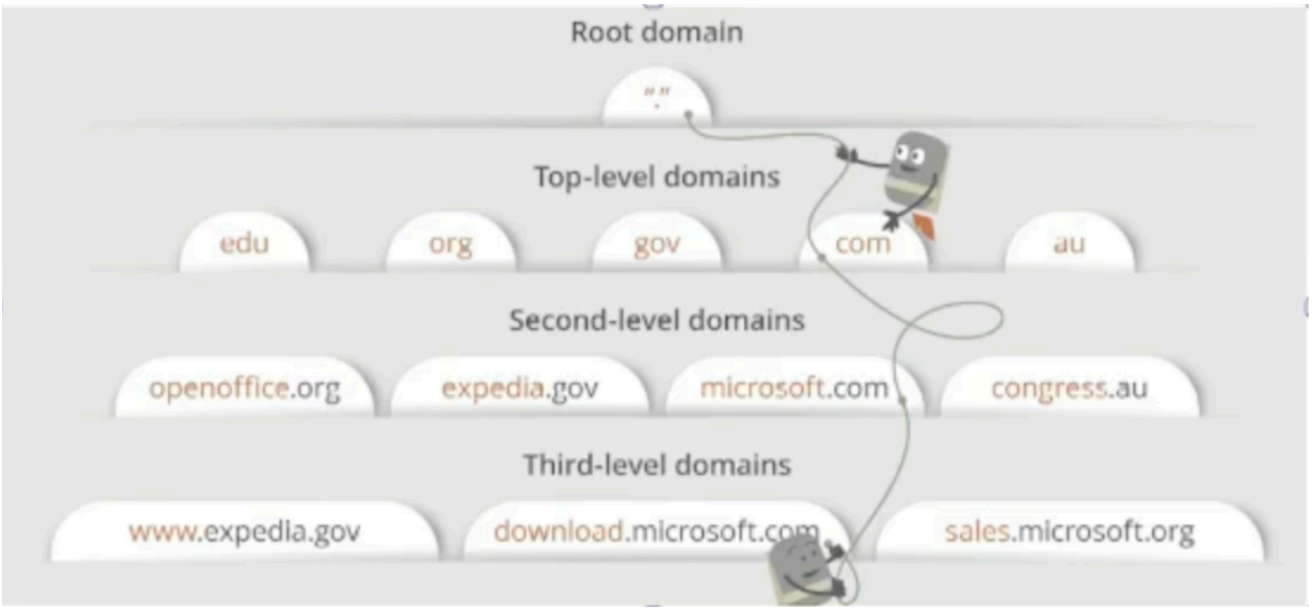
However, in reality, fetching web page data can often take a significant amount of time. To mitigate this, caching techniques are utilized.

Memory cache: A memory cache refers to a small portion of the main memory that is allocated as a temporary storage area for frequently accessed data. Its purpose is to enhance application performance by reducing the time required to access data from the network or main server. By storing frequently accessed data closer to the processing unit, memory caching helps to expedite data retrieval and improve overall system responsiveness.

As it is time-consuming to fetch data from the main server every time, caching is employed at various levels. When a request is sent, the browser first checks its own cache or service worker for data availability. If the data is not found there, it proceeds to check the operating system, followed by the router and ISPs.

Modern routers are now equipped with cache memory that stores frequently requested URLs/IPs, significantly reducing response time. However, if the required IP is still not found, the request needs to be forwarded to the DNS (Domain Name System) for further resolution.

Let's Understand the Domain name Structure in detail:



Let’s suppose the URL "www.google.com" is mapped to the IP address 142.250.190.78, stored in the DNS server. Breaking down the URL:

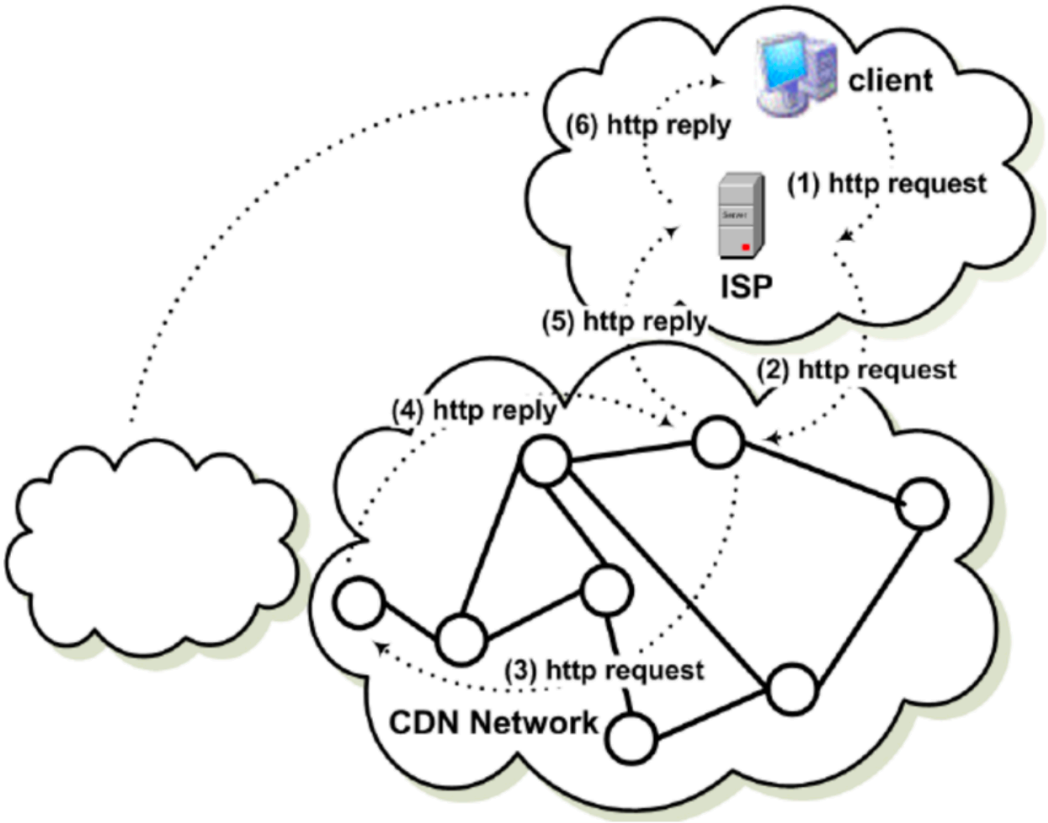
- Top-level domain (TLD): The last part, "com," indicates the type or category of the web page server.
- Second-level domain (SLD): The "google" part denotes the actual site name or entity being searched for.
- Third-level domain: Subdomains like "www," "download," "sales," etc., specify a specific section or type of content within the website.

Combining the TLD, SLD, and subdomain provides the IP address from the DNS. The DNS translates human-readable domain names into IP addresses, allowing browsers to connect to the correct web server.

Optimizing Web Performance: Introducing the Content Delivery Network (CDN)

Here comes the concept called CDN (Content Delivery Network).

CDN stands for Content Delivery Network, which is a geographically distributed network of servers that collaborate to deliver web content, including HTML pages, images, videos, and other multimedia files, to users based on their geographical location. CDNs are specifically designed to enhance the performance, availability, and reliability of web content delivery.



A Generic CDN Architecture (Above Fig): Procedure for Data Object Request

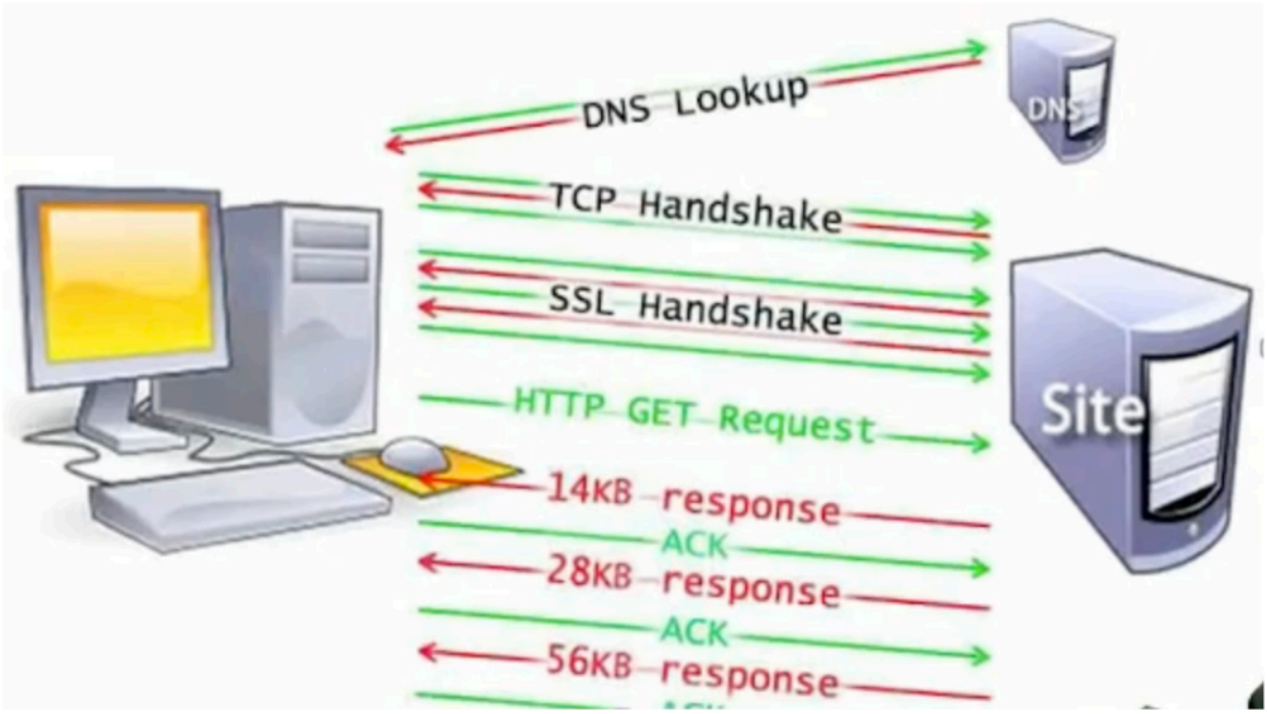
When a client requests a data object, the following procedure is invoked:

Step 1: The client sends an HTTP request to the ISP.
Step 2: The ISP forwards the HTTP request to the nearest CDN server, known as the first-hop CDN server, based on communication times.

Step 3: The first-hop CDN server resolves the DNS entry and identifies the nearest server that holds the replica of the requested data object. The HTTP request is then forwarded to that server.
Step 4: The data object is sent from the identified server to the first-hop CDN server.
Step 5: The first-hop CDN server forwards the data object to the ISP.
Step 6: The ISP sends the data object to the client.

This process allows for efficient retrieval and delivery of data objects through a CDN, minimizing latency and improving content delivery to end-users.

Now let's understand how data is is being tranferred from server in detail



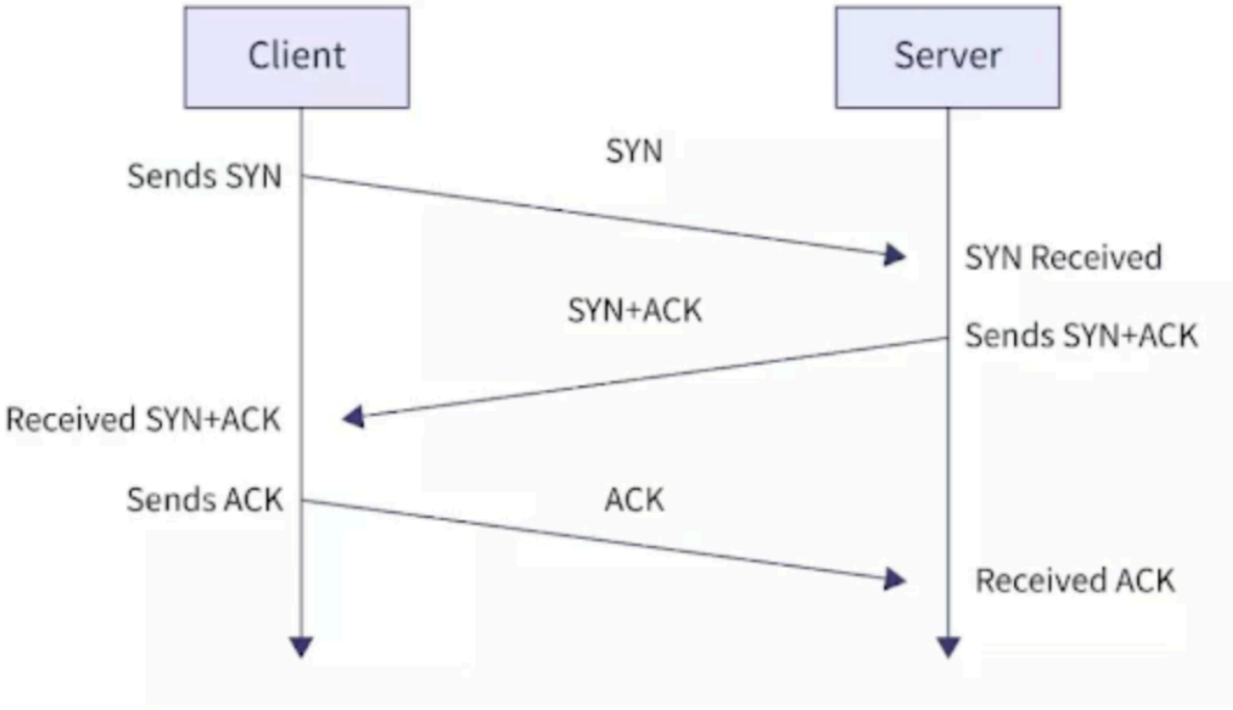
With reference to above fig the DNS Lookup part is explained previously, so let's proceed

TCP Handshake:

The three-way TCP handshake is a process to establish a connection:
1. Client sends SYN packet to server with a sequence number.
2. Server responds with SYN-ACK packet, acknowledging the client's request and providing its own sequence number.

3. Client sends an ACK packet, confirming the server's acknowledgment and completing the connection establishment.

Once the three-way handshake is completed, the devices have agreed upon initial sequence numbers, acknowledged each other's requests, and established a reliable connection.

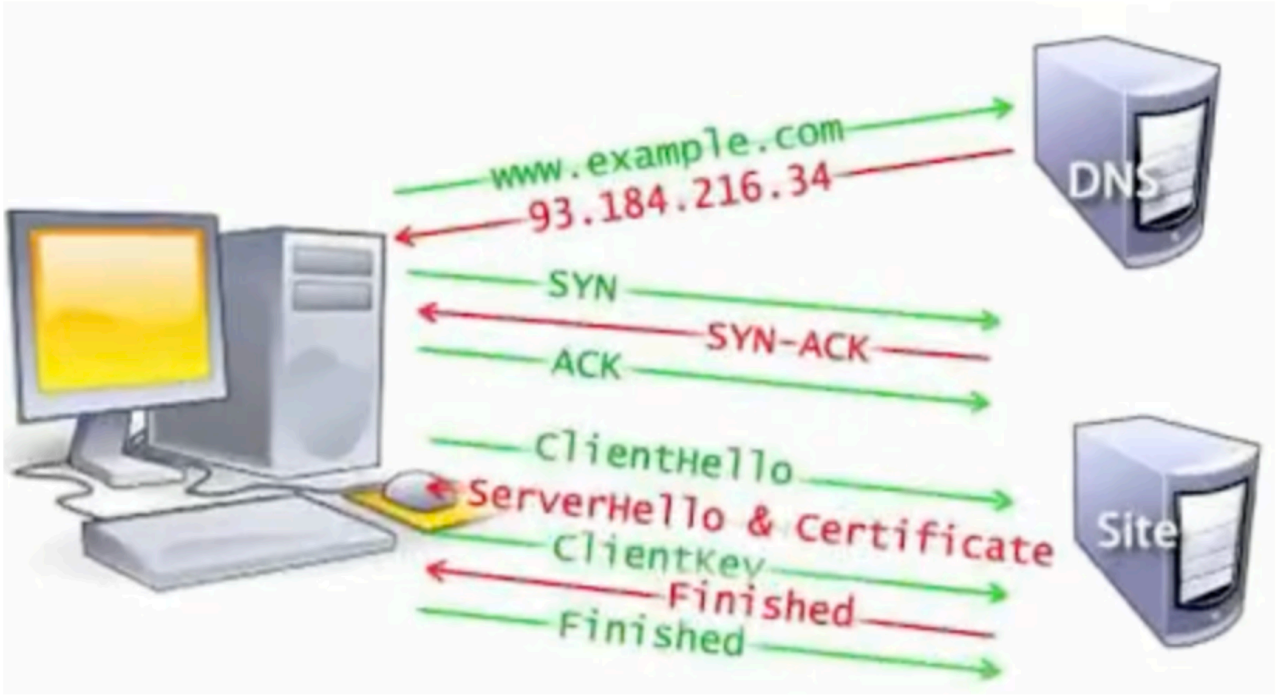


SSL Handshake:

The SSL handshake is a process through which a client and a server establish a secure connection. During this process, the client and server exchange information, verify each other's identity, and agree on encryption methods. The purpose of the SSL handshake is to ensure that data exchanged between the client and server is encrypted and protected.

The handshake involves several steps, including exchanging hello messages, verifying certificates, performing key exchange, and generating session keys. Once the handshake is successfully completed, both the client and server can securely communicate over

an encrypted connection, safeguarding the confidentiality and integrity of the transmitted data.



After SSL Handshake:

The initial data chunk sent is typically small, around 14KB in size. Subsequently, larger data chunks of 28KB and 56KB may be sent.

The reason behind sending smaller data chunks initially is to provide the user with a better browsing experience. By sending a small amount of data first, the browser can quickly render and display the initial content, allowing the user to see and interact with the webpage faster. This gives the impression of a faster loading time and improves the overall user experience.

Once the initial data is received and rendered successfully, the browser establishes trust in the server and its data. Consequently, the browser is more willing to receive and render larger data chunks, leading to the display of the remaining content of the webpage.

By employing this approach of sending smaller chunks first and gradually increasing the data size, web developers can optimize the perceived speed and responsiveness of the webpage, providing a smoother browsing experience for users.

Here's a step-by-step explanation of how files load in a browser after retrieval from the server (Refer the below Figure):

1. Request Page: When you enter a URL or click a link, the browser sends a request to the server for the corresponding webpage or file.

2. Receive HTML: The server responds with the HTML content of the webpage. The browser begins processing the HTML from top to bottom.

3. Build DOM (Document Object Model): The browser parses the HTML and constructs the DOM, which is a tree-like representation of the webpage's structure. Each HTML element becomes a node in the DOM.

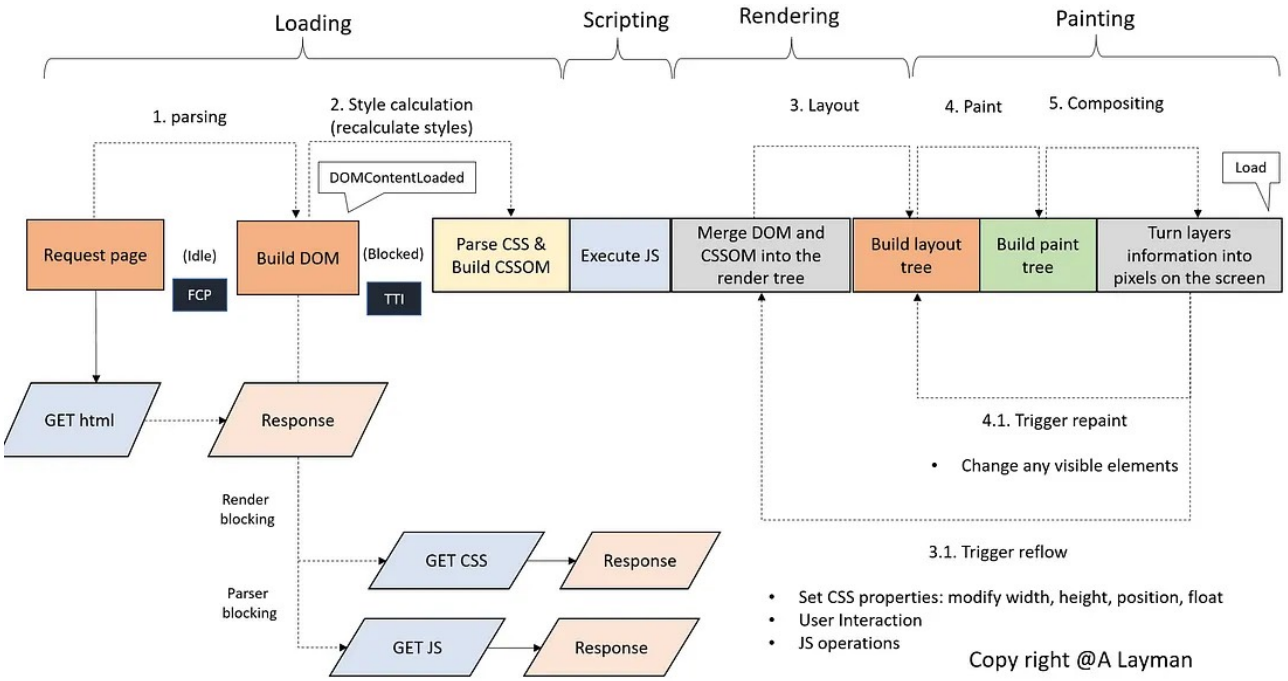
4. Parse CSS & Build CSSOM (CSS Object Model): As the browser encounters CSS stylesheets linked in the HTML, it fetches them from the server. The browser then parses the CSS and builds the CSSOM, which represents the styles applied to the DOM elements.

5. Execute JavaScript: If the HTML contains JavaScript code or references external JavaScript files, the browser executes the JavaScript. This can modify the DOM, apply styles, or interact with the webpage.

6. Merge DOM and CSSOM: The browser combines the DOM and CSSOM to create the render tree. The render tree represents how the DOM nodes should be displayed on the screen, taking into account the applied styles.

7. Build Layout: The browser calculates the layout or geometry of each element in the render tree. It determines their size, position, and relationship with other elements on the webpage.

8. Build Paint: The browser determines which parts of the render tree need to be painted on the screen. It creates a representation of what will be displayed, including colors, fonts, images, and other visual elements.



9. Turn Layer Information into Pixels on Screen: The browser takes the information from the render tree and performs the final step of turning it into pixels on the screen. This process is known as rasterization, where each element is translated into pixels that can be displayed on your device.

As these steps progress, you start to see the webpage's content and visuals rendered on the screen. This process allows the browser to progressively display the webpage while handling different file types, executing scripts, and ensuring a smooth user experience.