Google.com Request Breakdown

Computer Network Assignment

Niranjan Dahal

August 11, 2024

Table of Contents

Introduction

When you type ”google.com” in your browser’s address bar and press enter within 1-2 seconds you get an response with web-page but under-hood , a very complex sequence of events occurs over seven layers of the OSI model and each layer plays an important role ensuring accurate and efficient data flows from your device to Google servers. We will be looking in depth at what really happens between those 1-2 seconds. In those 1-2 seconds

- first it check whether url ip is known or it need to be lookup.
- after ip is known browser prepares an request to google
- after it tcp connection is established with google ensuring data is sent reliably.
- then request is broken into packets,put into frames and transmitted as electrical signals over networks.
- Google receives and processes the request then send back response.
- response travels back through same layers and browser renders web-page by loading html,css,js and image files.

Let’s dive deep and breakdown each events across all osi models layers

1 User Input and URL Parsing

when we type ”google.com” and press enter then browser check whether it is url or search term. in our case it is url not search term.

secondly the browser checks its preloaded HSTS(http strict transport security) list.This is the list of websites that have requested to be contacted via https only.if website is in list then it send https request instead of http.

example of url breakdown of google.com
protocol: ”https” Host: ”www.google.com” Path : ”/”

2 Application Layer

The Application Layer is closest to end users and responsible for network services like file transfers ,email and other network software services.

2.1 DNS Query

- **initial check:** The browser first check it’s local cache to know ip address for ”www.google.com”
- **if cached:** cached ip address is used.
- **if not cached:** browser generate dns query to know ip address of domain
- **DNS Resolver:** send the dns query to dns server(ISP caching dns server)

- **if not found:** dns resolver send query to root dns servers,TLD(top level domain) servers and authoritative nameserver for domain to get the ip address.
- **received ip address:** once dns resolution complete then ip address is obtained eg ”www.google.com” (172.217.12.206)
- **cache ip address:** This resolved ip address is now cached for future use.

2.2 Http Request

The browser prepare an http request to access the ”google.com”

- **constructing request:** in this the browser specifies method(GET,POST,etc) and path (”/”,”/homepage”,etc) and http version (HTTP/1.1)
- **Headers:** it includes metadata like Host:”www.google.com”,”User-Agent”,”Accept” S,”Connection”,etc
- **send request:** http request is sent to the ip address obtained from dns resolution.

3 Transport Layer

after ip address is obtained the browser establish connection using TCP/IP.

3.1 TCP Connection Establishment

3.1.1 three-way handshake

- **SYN:** Client sends a SYN packet to server to start connection which contain initial sequence number.
- **SYN-ACK:** Server responds with SYN-ACK packet which contains acknowledgement of client SYN and server sequence number.
- **ACK:**Client sends ACK packet which acknowledges server SYN-ACK

3.1.2 SSL/TLS Handshake (for HTTPS)

- **Client Hello:** Client sends a SYN packet to server to start connection which contain initial sequence number.
- **Server Hello:** Server responds with SYN-ACK packet which contains acknowledgement of client SYN and server sequence number.
- **Certificate Verification:**Client sends ACK packet which acknowledges server SYN-ACK
- **Session Key Exchange:** Both parties exchange keys to establish a secure session.

3.2 Data Transfer

3.2.1 Segmentation:

- HTTP data is broken into smaller TCP segments, each with its sequence number.

3.2.2 Reliability:

- **Acknowledgments:** Each segment sent must be acknowledged by the receiver.
- **Re transmission:** : Lost segments are re-transmitted.
- **Flow Control:** TCP uses flow control to prevent overwhelming the receiver

3.2.3 Reassembly:

- Segments are reassembled into the original HTTP request on the server.

4 Session Layer (Client)

The Session Layer is responsible for establishing, maintaining, and terminating sessions between applications. It manages the communication session between the client and server, ensuring that data is exchanged reliably.

In many modern implementations, session management is handled by the transport layer or higher-level application protocols, so it may not always be distinctly handled.

in practical implementations, session management often occurs at the application layer or is integrated within transport layer secure connection.for https,the session management is often handled as part of TLS/SSL protocol

5 Network Layer

5.1 Packet Creation

At the Network Layer, the HTTP request (which is segmented into TCP segments from the Transport Layer) is encapsulated in an IP packet. This encapsulation involves adding a header to each TCP segment. The IP packet includes crucial information for routing and delivery.

- **IP Header:**The IP packet header contains fields essential for routing and addressing
  - **Source IP Address:** he IP address of the client device initiating the request.
  - **Destination IP Address:** The IP address obtained from the DNS resolution, which corresponds to "www.google.com."
  - **Version:** Indicates the version of IP being used (IPv4 or IPv6).
  - **Header Length:** Specifies the length of the IP header.
  - **Total Length:** The total length of the IP packet (header plus data).

  - **TTL(Time to live):** A field that limits the lifespan of the packet to prevent it from circulating indefinitely in case of routing loops.
  - **Protocol:** Indicates the protocol used in the data portion of the packet (TCP in this case).
  - **Checksum:** Used for error-checking the header.

5.2 IP Packet Transmission (Client)

Once the IP packet is created, it is transmitted over the network. This involves sending the packet to the next-hop router or directly to the destination server if it is on the same network. .

- **Transmission Medium:**The packet travels through various physical mediums like Ethernet cables, Wi-Fi, or fiber optics.
- **Routing:** Routers along the path use the destination IP address to determine the next hop for the packet. The packet may pass through several routers and networks before reaching the destination server.
- **Forwarding:**Routers forward the packet based on the routing table, which is a list of routes to various network destinations.
- **Fragmentation:** If the packet size exceeds the Maximum Transmission Unit (MTU) of the network, it may be fragmented into smaller packets.

5.3 IP Packet Reception (Server)

The server receives the IP packet from the network. Upon arrival, the IP packet is extracted from the data frame provided by the Data Link Layer and forwarded to the appropriate TCP segment. .

- **Data Frame:**The IP packet is carried within a data frame (e.g., Ethernet frame) which includes additional headers specific to the Data Link Layer.
- **Decapsulation:** At the server's Network Layer, the IP packet is decapsulated from the data frame, and its header information is read.
- **Address Check:** The server checks if the destination IP address in the packet matches its own IP address. If so, the packet is accepted; otherwise, it is discarded.

5.4 IP Packet Processing (Server)

After receiving the IP packet, the server processes it to extract the TCP segment. This segment is then passed to the Transport Layer for further processing .

- **Header Analysis:** The server reads the IP header to extract information necessary for handling the packet, such as source and destination IP addresses and protocol type.
- **TCP Segment Extraction:** The payload of the IP packet contains the TCP segment. The Network Layer extracts this segment and forwards it to the Transport Layer.

- **Checksum Verification:** The server verifies the IP packet's checksum to ensure data integrity and detect any errors that may have occurred during transmission.

6 Data Link Layer

The Data Link Layer is responsible for node-to-node data transfer and error detection and correction on the local network segment. It frames the data packets from the Network Layer and handles their delivery over the physical medium.

6.1 Frame Creation

The IP packet, which contains the encapsulated TCP segment, is further encapsulated into a data frame at the Data Link Layer. This frame is prepared for transmission over the physical network medium.

The process of encapsulation at the Data Link Layer involves adding a header and a trailer around the IP packet. This forms the data frame.

- **Frame Header:**
  - **Source MAC Address:** The Media Access Control (MAC) address of the sender's network interface card (NIC).
  - **Destination MAC Address:** The MAC address of the intended recipient's NIC. If the destination is a remote host, this will be the MAC address of the next-hop router.
  - **Type/Length Field:** Indicates the protocol of the encapsulated data (e.g., IPv4 or IPv6).
  - **Priority Field:** May indicate the priority of the frame in network traffic (e.g., Quality of Service).
- **Frame Trailer:**
  - **Frame Check Sequence (FCS):** A checksum used for error detection. It allows the receiving device to verify if the frame has been transmitted correctly.

6.2 Frame Transmission(Client)

The frame, now containing the IP packet, is transmitted over the network to the next-hop device (e.g., router) or directly to the destination if on the same network.

- **Transmission Medium:** The frame is sent over the physical medium such as an Ethernet cable, Wi-Fi, or fiber optic link.
- **MAC Addressing:** The Data Link Layer uses MAC addresses to deliver the frame to the correct hardware address on the local network.
- **Access Methods:** Various methods (e.g., CSMA/CD for Ethernet, CSMA/CA for Wi-Fi) determine how the network medium is accessed and how collisions are handled. .

6.3 Frame Reception (Server)

The server receives the data frame from the network. The frame is extracted, and the IP packet within it is processed.

- **Decapsulation:** The server's NIC removes the frame header and trailer, revealing the encapsulated IP packet.
- **Error Checking:** The FCS in the frame trailer is checked to verify that no errors occurred during transmission.
- **MAC Address Verification:** The server checks the destination MAC address to ensure the frame was intended for it.

7 Physical Layer

The Physical Layer is responsible for the actual transmission of raw data bits over the physical medium. It deals with the hardware aspects of network communication

7.1 Signal Transmission (Client)

The data frame is converted into electrical, optical, or radio signals for transmission over the physical network medium.

- **Data Encoding:** The data in the frame is encoded into signals that can be transmitted.Manchester encoding and NRZ(Non-Return-to-Zero) encoding are commonly used.
- **Transmission Medium:** Ethernet cables ,fibres optics and wireless medium are used for transmission.

7.2 Signal Reception(Server)

The server's NIC receives the signals from the physical medium. These signals are converted back into a data frame for further processing at the Data Link Layer.

- **Signal Decoding:** The received signals are decoded to retrieve the raw data bits. This involves reversing the encoding process used during transmission.
- **Signal Processing:** Includes amplification and filtering to ensure the signal quality before decoding.
- **Synchronization:** Ensures that the timing of the received signals matches the expected timing to accurately interpret the data bits.

8 Application Layer (Server)

The Application Layer is responsible for handling high-level protocols and services that provide end-user services directly. When a server receives an HTTP request, it processes it to generate an appropriate response. Let's break down the events in this layer in detail.

8.1 HTTP Response Handling

After receiving and processing the HTTP request from the client, the server generates an HTTP response. This response includes a status line, headers, and a body.

8.1.1 Request Processing

The server parses the HTTP request to extract the request line (method, path, and HTTP version), headers, and body (if present).

- **Request Line:** GET /index.html HTTP/1.1
- **Headers:** Host: www.google.com, User-Agent: Mozilla/5.0
- **Body:**May be empty or some form data in post request

After it the server uses the request path (eg "/index.html" to determine which resource or handler should process the request.

8.1.2 Response Construction

It construct the valid http response that will be sent back to the client.

- **Status Line:**
  - **Format:** 'HTTP/1.1 ¡Status Code¿ ¡Reason Phrase¿'
  - **Example:** 'HTTP/1.1 200 OK'
- **Headers:** It provide metadata about the response, such as content type, length, and server information.
  - **Content-Type:** Specifies the media type of the response body (e.g., Content-Type: text/html).
  - **Content-Length:** Indicates the size of the response body in bytes (e.g., Content-Length: 1024).
  - **Cache-Control:** Directives for caching mechanisms (e.g., Cache-Control: no-cache).
- **Body:**It carry the actual content of the response.
  - **HTML**
  - **JSON**
  - **Files**

  In case of google.com request we get html,css and js file and logo.png(Google Logo)

8.1.3 Response Formatting:

It ensure the response adheres to the HTTP standard and is correctly formatted for transmission.line breaks and encoding response body is used for it.

8.2 Http Response Transmission

Server sends the constructed http response back to client over established tcp connection.

8.2.1 TCP Data Transfer:

It transmit the HTTP response over a reliable connection.

- **Segmentation**
- **Transmission**
- **Acknowledgment**

8.2.2 Response Delivery:

It deliver the complete HTTP response to the client.

- **Receiving Packets** The client receives TCP packets containing the response.
- **Reassembly** TCP segments are reassembled into the complete HTTP response
- **Error Handling** Any lost packets are re transmitted, and errors are handled according to TCP protocols.

9 Transport Layer (Client)

The client browser reassembles the HTTP response received from Google's server, which is sent over multiple TCP segments. This ensures that the complete and correct data (the HTML page and other resources) is reconstructed from the segments.

9.1 Receiving TCP Segments

- After sending an HTTP request to Google's server, the client receives TCP segments containing parts of the HTTP response.
- The server responds with these segments in potentially fragmented parts, which are managed by the TCP protocol.
- **Sequence Numbers:**Each segment from Google's server has a sequence number that helps the client browser determine the order of the segments.
- **Out-of-Order Segments:**Sometimes, segments may arrive out of sequence due to network conditions. TCP handles this by reordering the segments based on their sequence numbers.

9.2 Segmentation and Reassembly

- **Segmentation:**Google's server breaks down the large HTTP response (e.g., the HTML content of the Google homepage, images, CSS files) into smaller TCP segments. Each segment is given a header with sequence information. For instance, a large HTML file might be split into several TCP segments for easier transmission.
- **Reassembly** It reconstruct the complete http response from received tcp segments

9.3 Data Assembly and Error Handling

It Combine all TCP segments into the complete HTTP response that the browser can use.

- **Reconstruction:** After all segments are received and verified, the browser reconstructs the complete HTML page and other resources by concatenating the segments based on their sequence numbers.
- **Error Handling:**If segments are lost or corrupted, TCP re- transmits them. For instance, if the segment containing critical parts of the HTML is lost, TCP will request its re-transmission.

for example if the segment with portion of css for displaying google.com page is lost then server will resend it

9.4 Passing Data to Higher Layers

Once the HTTP response is fully reassembled, including the HTML content and any additional resources (like images and scripts), it is handed over to the browser's Application Layer.After it the Application Layer processes the HTTP response to render the web page for the user.

10 Presentation Layer(Client)

After receiving the complete HTTP response, the browser's Presentation Layer decodes the data to convert it into a format that can be interpreted and rendered. This includes decoding any encoded data in the HTTP response such as compressed content or character encoding.

10.1 Http Response Structure

Google's server responds with an HTTP response that includes an HTML document, which is the main content of the page, as well as links to CSS files, JavaScript files, and image resources.

10.2 Decoding Compressed Content:

Content-Encoding Header: Google use gzip compression for the HTML content to reduce the amount of data sent. The HTTP response header includes Content-Encoding: gzip . So the browser decompress the gzip-encoded HTML to load html

10.3 Character Encoding

The response id generally "Content-Type: text/html; charset=UTF-8" and the browser interprets this and decode to display text characters

10.4 HTTP Response Rendering

The browser renders the HTML content, applying CSS styles and executing JavaScript to present the Google homepage to users..This is basically done by

10.4.1 parsing HTML

DOM Construction: The browser parses the HTML and constructs a DOM tree. For Google's homepage, this includes elements like ¡div¿, ¡input¿, and ¡button¿.

10.4.2 applying CSS

The browser fetches and parses styles.css from Google's server.The styles are applied to the DOM

10.4.3 Executing Javascript

The browser fetches and executes scripts.js.

10.4.4 Rendering the Page

- **Layout Calculation:** The browser calculates the layout of all elements based on the DOM structure and CSS styles. For Google, this means placing the search box and button in the center of the page.
- **Painting** The browser paints the content onto the screen. It renders the search box, button, and any text or images, translating the DOM and CSS into visible pixels on the screen.
- **Re-flows and Repaints:** If JavaScript modifies the DOM or CSS, the browser might need to adjust the layout (re-flow) and update the visual display (repaint).