What Happens When We Type GOOGLE.com and Hit Enter

TCP/IP MODEL

A Comprehensive and In-depth Sequential Analysis of All Layers

Submitted By:
Aayush Adhikari
Roshan Tiwari
Shishir Sharma Rijal
Sudip Acharya

August 8, 2024

Contents

List of Figures

1 Introduction

When a user types "google.com" into a web browser's address bar and presses Enter, a complex series of events unfolds across all layers of the TCP/IP model. This document provides a comprehensive and in-depth sequential analysis of these processes, exploring each layer in detail and in the order they occur.

2 TCP/IP Model Overview

TCP/IP Model Layers

The TCP/IP model consists of four layers, each with specific responsibilities:

1. Application Layer: Handles high-level protocols, APIs, and data formatting.
2. Transport Layer: Manages end-to-end communication and data flow.
3. Internet Layer: Deals with packet forwarding and routing between different networks.
4. Network Access (Link) Layer: Concerned with physical addressing and data transmission over the network medium.

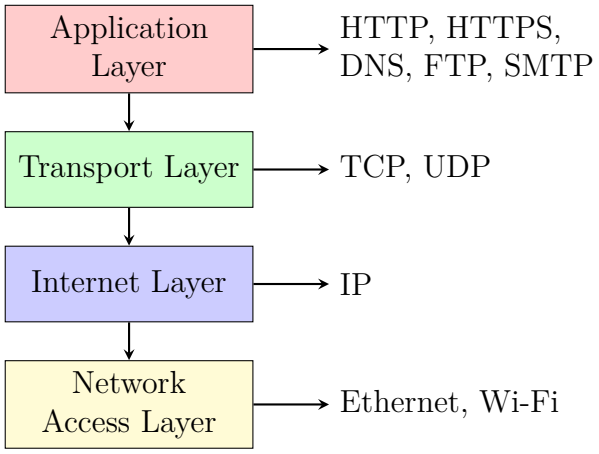


Figure 1: TCP/IP Model Layers and Protocols

3 Step-by-Step Process

3.1 User Input and Application Layer Processing

3.1.1 URL Parsing and Validation

When the user types "google.com" and hits Enter, the following steps occur:

URL Parsing Steps

1. **User Input Reception:**
   - Browser receives the input "google.com".
2. **Scheme Identification:**
   - Browser identifies "https://" as the default scheme if not specified.
   - Note: Modern browsers typically use HTTPS by default due to HSTS (HTTP Strict Transport Security) policies.
3. **Domain Extraction:**
   - Extracts "google.com" as the domain.
   - Validates characters (a-z, 0-9, hyphen) and formatting.
4. **Path and Query String Analysis:**
   - Identifies "/" as the default path if not specified.
   - No query parameters in this case.
5. **URL Normalization:**
   - Converts the domain to lowercase.
   - Removes default port numbers if present (e.g., :443 for HTTPS).

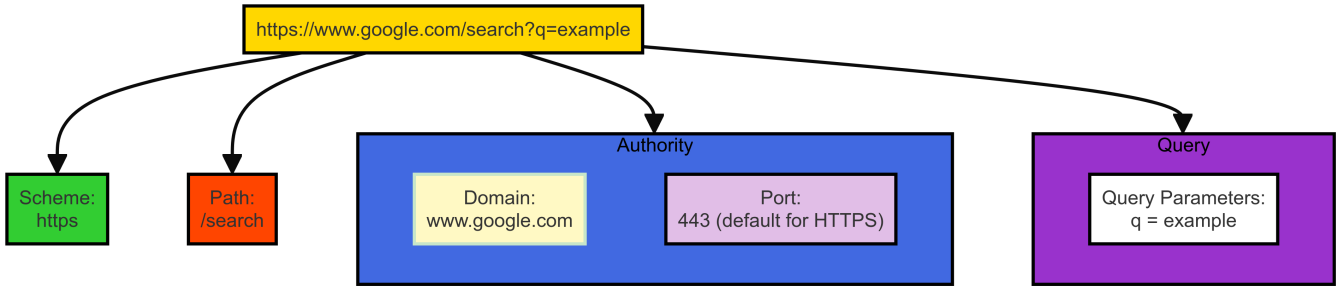
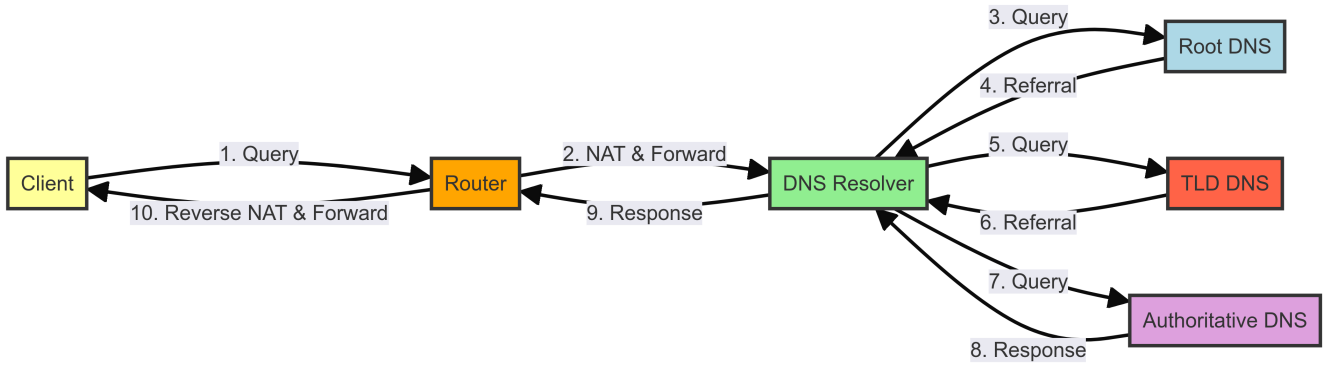


Figure 2: URL Parsing Process

3.1.2 DNS Resolution

After parsing the URL, the next step is to resolve the domain name "google.com" to an IP address. This process involves several steps, including local cache checks, DNS queries, and network address translation (NAT):



Detailed DNS Resolution Process

1. **Local DNS Checks:**
   (a) Browser checks its local DNS cache (typically short TTL, e.g., 60 seconds).
   (b) Operating system checks its DNS cache (usually longer TTL than browser cache).
   (c) System checks the local hosts file (e.g., /etc/hosts on Unix systems).
2. **DNS Request Creation:**
   (a) Our device creates a DNS query to resolve "google.com" into an IP address.
   (b) DNS Query Packet is formed:
   - Encapsulated in a UDP packet.
   - Source Port: A temporary port on our device.
   - Destination Port: Port 53 (DNS standard port).
   - Payload: Contains the domain name and query details.
3. **Sending DNS Request to Router:**
   (a) Device determines router's IP address (default gateway).
   (b) If needed, ARP resolution is performed to get router's MAC address.
   (c) DNS query packet is encapsulated in an Ethernet or Wi-Fi frame with the router's MAC address.
   (d) Packet is sent to the router.
4. **Router Processing:**

(a) Router receives the DNS packet from our device.

(b) NAT is performed (if necessary):

- Source IP address changed from private IP to router's public IP.
- Source port updated to a port on the router.
- NAT mapping is maintained.

(c) Router identifies DNS server to forward the request to (using pre-configured settings, DHCP-provided information, or default settings).

(d) Router checks its ARP cache for the MAC address of the next hop.

(e) If the MAC address isn't in the cache, router performs an ARP request to get the MAC address.

(f) Router encapsulates the DNS query in an Ethernet frame with the obtained MAC address.

(g) Router sends the DNS query to the DNS server's IP address on port 53.

5. **DNS Server Query:**

(a) DNS resolver performs recursive query through DNS hierarchy:

i. Query Root DNS servers for TLD server information.
ii. Query TLD (.com) DNS servers for authoritative DNS servers.
iii. Query Authoritative DNS servers for the IP address of google.com.

(b) DNS server resolves "google.com" to its IP address.

(c) DNS server sends the response back to the router.

6. **Router Receives and Forwards DNS Response:**

(a) Router receives the response from the DNS server.

(b) Reverse NAT is performed (if necessary):

- Destination IP and port translated back to device's private IP and port.

(c) Router encapsulates the DNS response in a packet addressed to our device's MAC address.

(d) Router sends the response over the local network.

7. **Device Receives and Processes DNS Response:**

(a) Our device receives the DNS response containing the IP address for "google.com".

(b) The IP address may be cached for future use, respecting the TTL.

(c) The IP address is used to establish a connection with "google.com".

```
└─ dig google.com

; <<>> DiG 9.10.6 <<>> google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 47026
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;google.com.                    IN      A

;; ANSWER SECTION:
google.com.             278     IN      A       142.250.195.14

;; Query time: 46 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Wed Aug 07 00:48:56 +0545 2024
;; MSG SIZE  rcvd: 55
```

Figure 3: Dig Google.com

3.2 Transport Layer Processing

3.2.1 TCP Connection Establishment

Once the IP address is obtained, the client initiates a TCP connection:

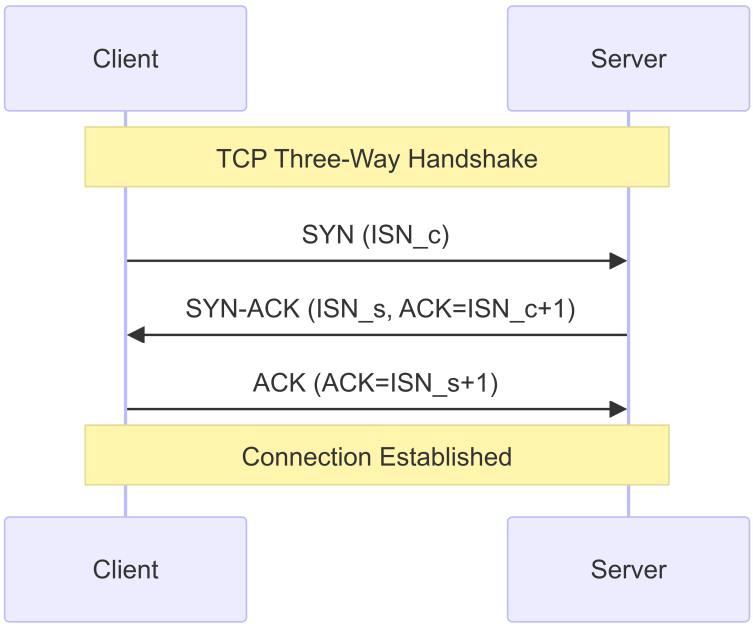


Figure 4: TCP Three-Way Handshake

TCP Three-Way Handshake

1. **SYN:**
   - Client sends SYN packet with initial sequence number (ISN).
   - Includes TCP options (e.g., Maximum Segment Size, Window Scale).
2. **SYN-ACK:**
   - Server responds with SYN-ACK packet.
   - Acknowledges client's ISN and sends its own ISN.
3. **ACK:**
   - Client sends ACK packet, acknowledging server's ISN.
   - Connection is now established and ready for data transfer.
4. **Connection Parameters:**
   - Both sides agree on initial sequence numbers.
   - Window sizes for flow control are established.
   - Any additional TCP options are negotiated.

3.2.2 TLS Handshake (for HTTPS)

After establishing the TCP connection, the TLS handshake begins for secure communication:

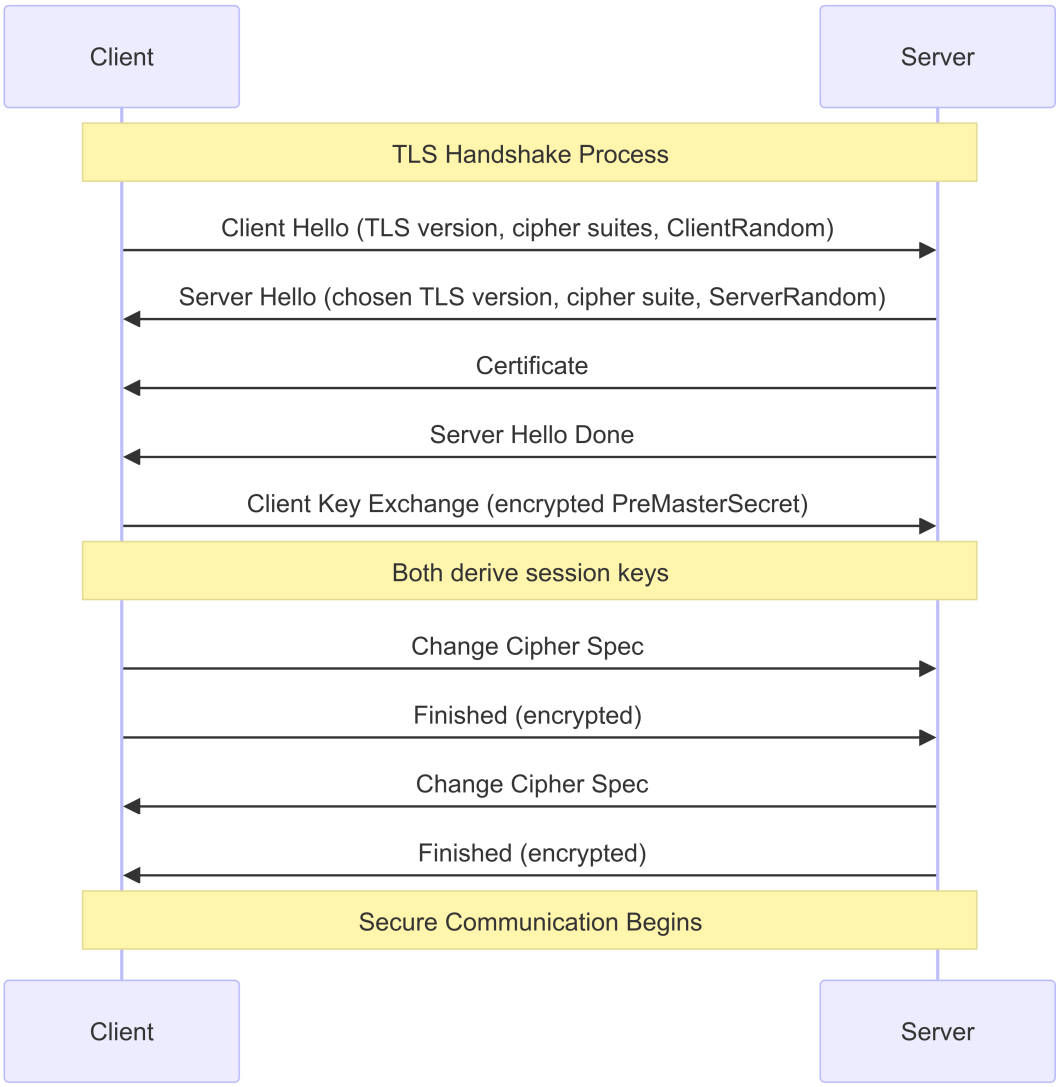


Figure 5: TLS Handshake

TLS Handshake Process

1. **Client Hello:**
    - The client initiates the handshake by sending supported TLS versions, cipher suites, and a random number (ClientRandom).
2. **Server Hello:**
    - The server responds by selecting the TLS version and cipher suite, and sends its own random number (ServerRandom).
3. **Certificate:**
    - The server sends its digital certificate to the client.
4. **Server Hello Done:**
    - The server signals that it's done with its initial messages.
      Both client and server derive the session keys from the shared information
5. **Client Key Exchange:**
    - The client sends a PreMasterSecret, encrypted with the server's public key.
6. **Change Cipher Spec:**
    - Both parties inform each other that they're switching to the agreed-upon cipher suite for encrypted communication.
7. **Finished:**
    - Both parties send an encrypted hash of all handshake messages to verify the process's integrity.
8. **Secure Communication:**
    - After the handshake, secure communication begins using the derived session keys.

3.3 Application Layer (HTTP) Processing

After the secure connection is established, the HTTP request is prepared:

HTTP Request Formation

1. **Request Line:**
   - Method: GET (for retrieving the homepage)
   - Path: / (root path)
   - HTTP Version: HTTP/1.1, HTTP/2, or HTTP/3
2. **Headers:**
   - Host: www.google.com
   - User-Agent: (browser information)
   - Accept: text/html,application/xhtml+xml,application/xml ;q=0.9,image/webp,*/*;q=0.8
   - Accept-Language: en-US,en;q=0.5
   - Connection: keep-alive
   - Cookie Handling: Checks for stored cookies for "google.com" and adds Cookie header if found
3. **Body:**
   - Empty for GET requests

For HTTP/2 and HTTP/3, the request format is different, using binary framing instead of plain text. These protocols also support multiplexing, allowing multiple requests to be sent over a single connection simultaneously.

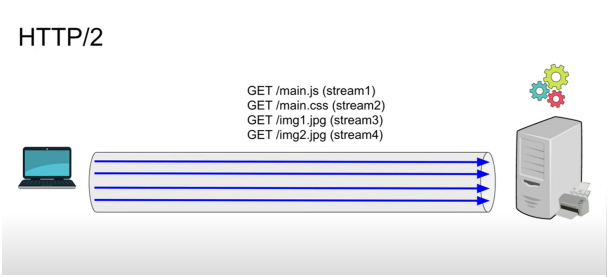


Figure 6: HTTP/2

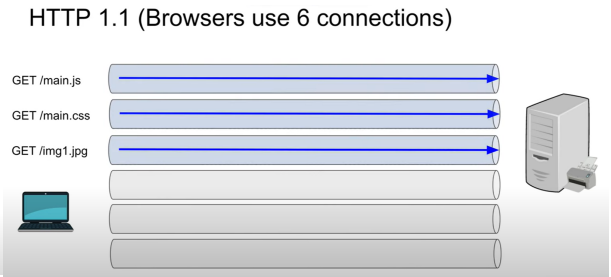


Figure 7: HTTP/1.1

3.4 Transport Layer (TCP) Segmentation

The HTTP request is then passed to the Transport Layer for segmentation:

TCP Segmentation Process

1. **Segmentation:**
   - TCP breaks the HTTP request into segments.
   - Each segment size is determined by the Maximum Segment Size (MSS).
2. **Sequence Numbering:**
   - Each byte in the data stream is assigned a sequence number.
   - The sequence number of the first byte in a segment is used in the TCP header.
3. **TCP Header Creation:**
   - Source Port (ephemeral port assigned by OS)
   - Destination Port (443 for HTTPS)
   - Sequence Number
   - Acknowledgment Number (if carrying an ACK)
   - Data Offset (header length)
   - Flags (e.g., ACK, PSH)
   - Window Size
   - Checksum
   - Urgent Pointer (if URG flag is set)

3.5 Internet Layer Processing

3.5.1 IP Packet Formation

The TCP segments are then encapsulated into IP packets:

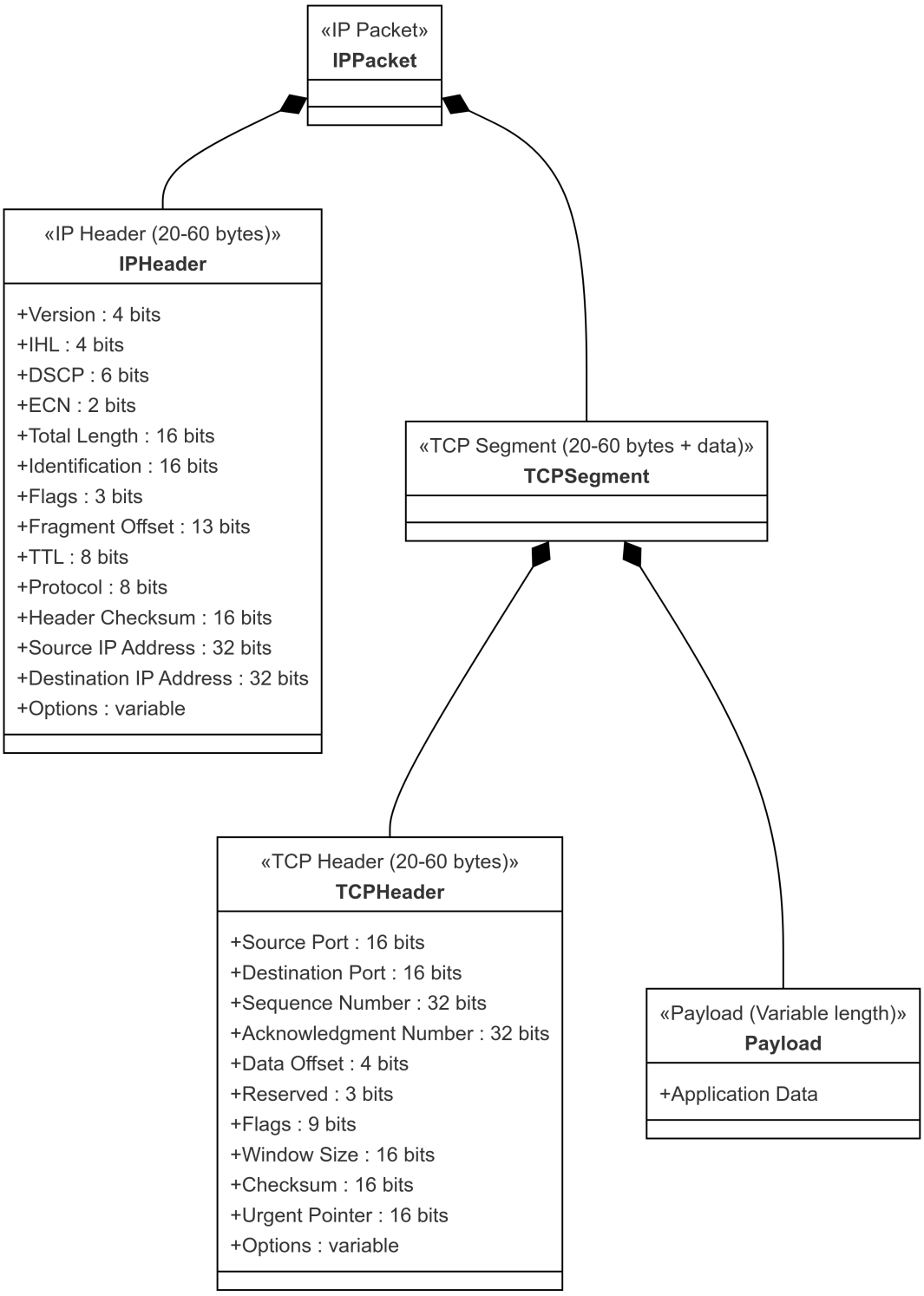


Figure 8: IP Packet Structure

IP Packet Creation Process

1. **IP Header Creation:**
    - Version (IPv4 or IPv6)
    - Header Length (for IPv4)
    - Type of Service (ToS) / Traffic Class (for QoS)
    - Total Length of the packet
    - Identification (for fragmentation)
    - Flags and Fragment Offset
    - Time to Live (TTL)
    - Protocol (6 for TCP)
    - Header Checksum (IPv4 only)
    - Source IP Address
    - Destination IP Address
2. **Encapsulation:**
    - TCP segment becomes the payload of the IP packet
    - IP header prepended to the TCP segment
3. **Fragmentation (if necessary):**
    - Occurs if packet size exceeds MTU of outgoing interface
    - Sets fragment flags and offset in IP header
    - Each fragment gets its own IP header

Note: IPv6 handles fragmentation differently, relying on Path MTU Discovery to avoid fragmentation in transit.

3.5.2 IP Routing

The IP packet is then routed to its destination:

IP Routing Process

1. **Local Routing Table Check:**
   - System checks if destination IP is on local network
   - If local, packet sent directly to destination
2. **Default Gateway:**
   - If not local, packet sent to default gateway
   - Gateway's MAC address obtained via ARP (for IPv4) or NDP (for IPv6)
3. **Intermediate Routers:**
   - Each router performs longest prefix match on destination IP
   - Decrements TTL and updates header checksum
   - Forwards packet to next hop
4. **Routing Protocols:**
   - Interior Gateway Protocols (e.g., OSPF, RIP) for intra-AS routing
   - Exterior Gateway Protocols (e.g., BGP) for inter-AS routing

The exact routing process can vary depending on network configurations and whether IPv4 or IPv6 is used.

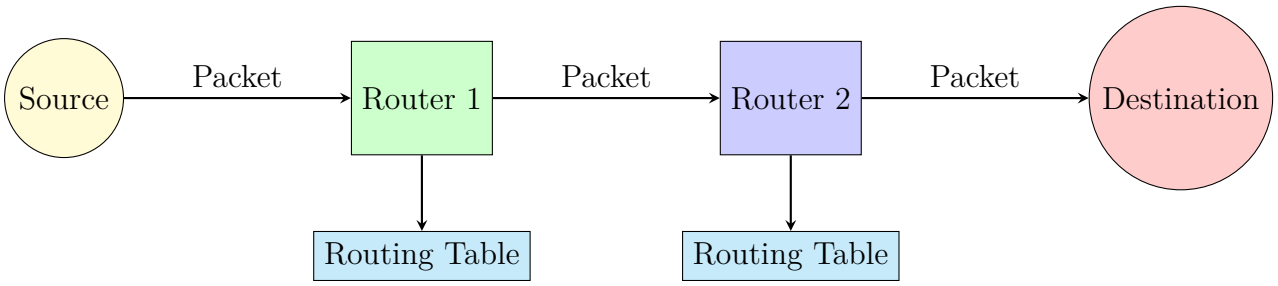


Figure 9: IP Routing Process

3.6 Network Access (Link) Layer Processing

3.6.1 ARP (Address Resolution Protocol)

For each hop in the routing process, ARP is used to resolve IP addresses to MAC addresses:

ARP Process

1. **ARP Cache Check:**
   - System first checks local ARP cache for IP-to-MAC mapping
   - If found, uses cached MAC address
2. **ARP Request:**
   - If not in cache, broadcasts ARP request on local network
   - Contains sender's IP and MAC, target IP
   - Destination MAC is broadcast address (FF:FF:FF:FF:FF:FF)
3. **ARP Reply:**
   - Device with matching IP sends unicast ARP reply
   - Contains its MAC address
4. **ARP Cache Update:**
   - Sender updates ARP cache with received mapping
   - Cached entries typically expire after a timeout (e.g., 20 minutes)

For IPv6, the Neighbor Discovery Protocol (NDP) is used instead of ARP, performing similar functions but with some differences in implementation.

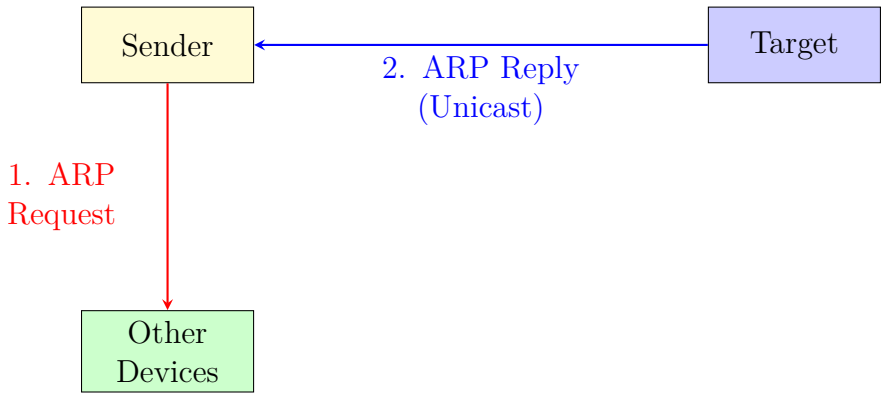


Figure 10: ARP Process

3.6.2 Frame Formation

Finally, the IP packet is encapsulated in a link-layer frame:

Ethernet Frame Creation

1. **Frame Structure:**
   - Preamble and Start Frame Delimiter (SFD)
   - Destination MAC Address (6 bytes)
   - Source MAC Address (6 bytes)
   - EtherType (2 bytes) - indicates payload protocol (e.g., 0x0800 for IPv4)
   - Payload (IP packet)
   - Frame Check Sequence (FCS) - 32-bit CRC
2. **MAC Address Assignment:**
   - Source MAC: Network interface's MAC address
   - Destination MAC: Obtained via ARP for next hop IP
3. **Frame Size Considerations:**
   - Minimum frame size: 64 bytes (including header and FCS)
   - Maximum frame size: 1518 bytes (standard Ethernet)
   - Jumbo frames: Up to 9000 bytes (if supported by network)

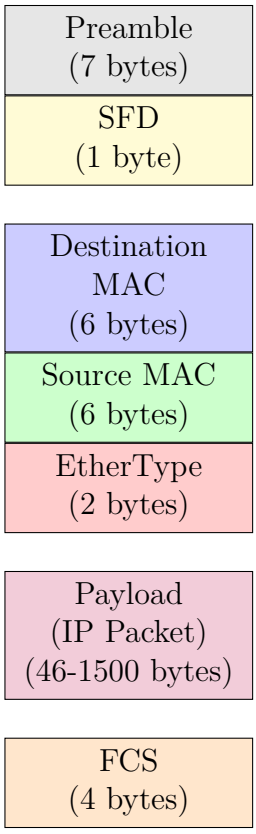


Figure 11: Ethernet Frame Structure

3.7 Physical Layer Transmission

The final step involves transmitting the Ethernet frame over the physical medium:

Physical Layer Transmission

1. **Signal Encoding:**
   - Convert digital data to appropriate signals (e.g., electrical, optical)
   - Use encoding scheme (e.g., Manchester encoding, 8b/10b)
2. **Media Access Control:**
   - CSMA/CD for half-duplex Ethernet
   - Full-duplex operation for most modern Ethernet
3. **Physical Transmission:**
   - Copper: Electrical signals over twisted pair cables
   - Fiber: Optical signals over fiber optic cables
   - Wi-Fi
4. **Error Detection and Correction:**
   - Use of Frame Check Sequence (FCS) for error detection
   - Automatic retransmission of corrupted frames

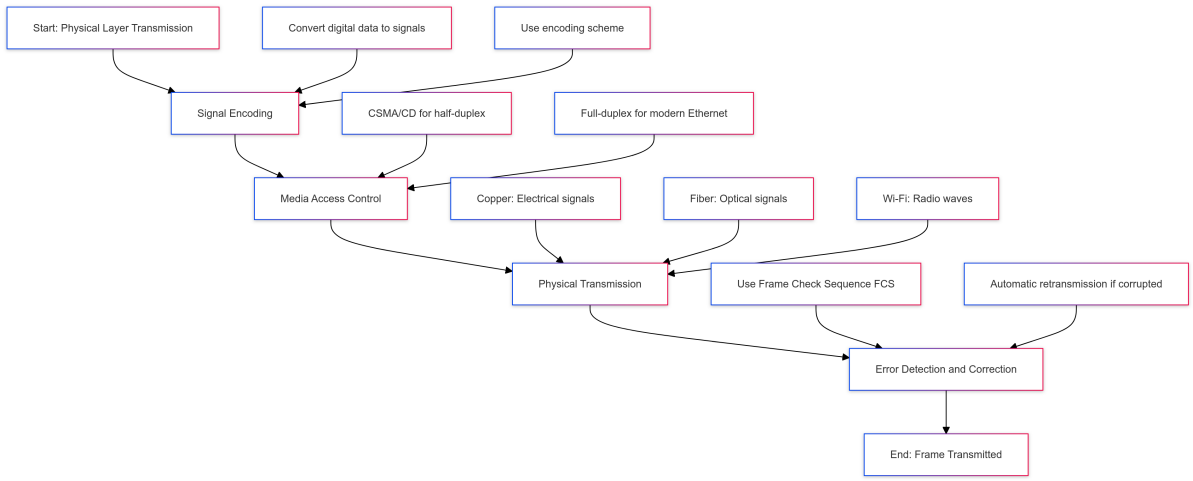


Figure 12: Physical Layer Transmission

3.8 Response Processing

Response Handling

1. **The server decapsulates the frames, processes the IP packet and TCP segment, and finally reads the HTTP GET request and send response.**
2. **Initial Response Parsing:**
   - Browser receives and decrypts data from server
3. **Status Line Interpretation:**
   - Reads status line (e.g., `HTTP/2 200 OK`)
4. **Header Processing:**
   - Processes various response headers
5. **Body Decompression:**
   - Decompresses content if compressed
6. **Content Parsing Initiation:**
   - Begins parsing HTML content as it's received

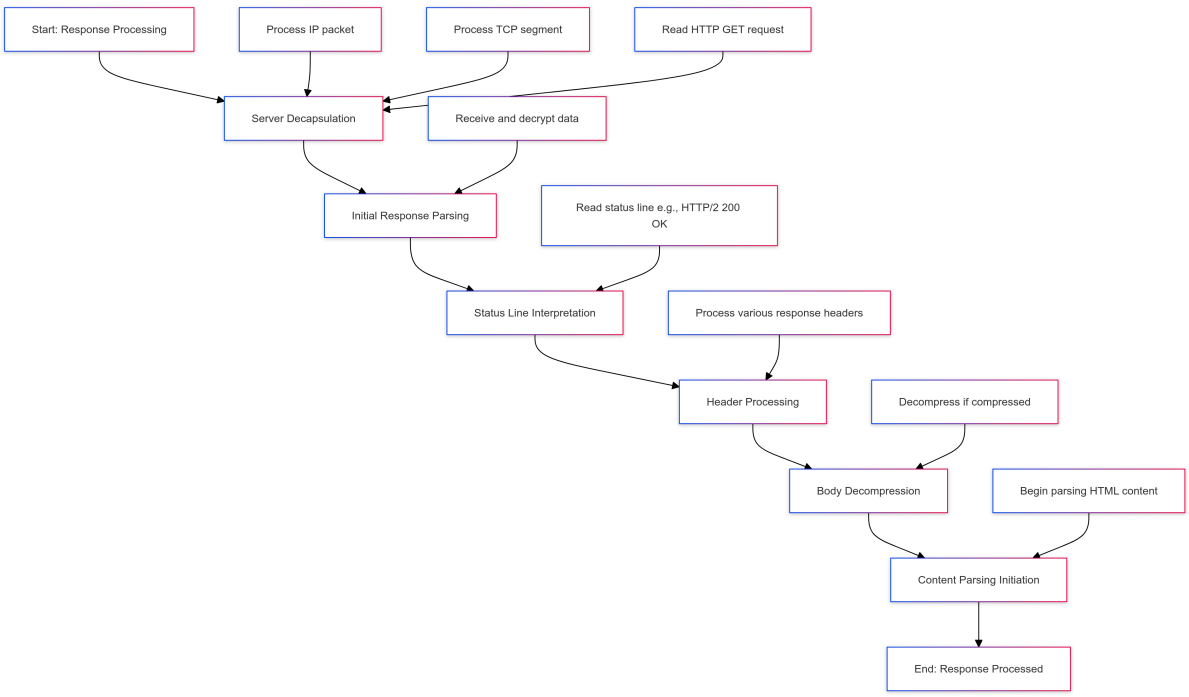


Figure 13: Response Processing

3.9 Content Rendering and Additional Requests

Rendering Process

- **DOM Construction:**
  - Constructs Document Object Model from HTML
- **Resource Identification:**
  - Identifies additional resources needed (CSS, JS, images, fonts)
- **Resource Fetching:**
  - Initiates new requests for each identified resource
- **Rendering Pipeline:**
  - Executes style calculation, layout, painting, and compositing

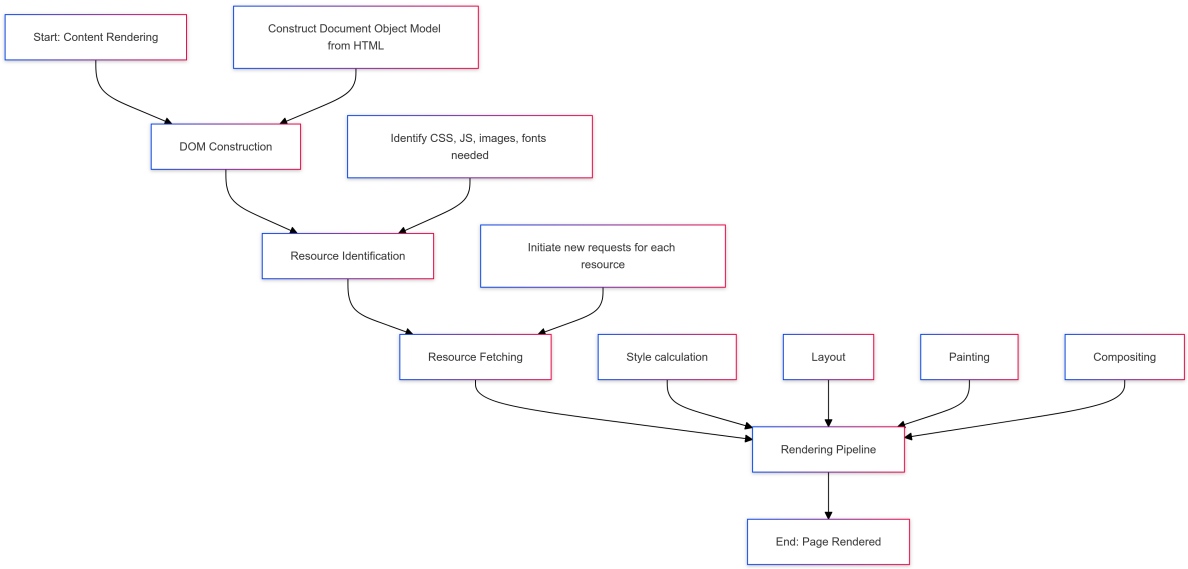


Figure 14: Rendering Process

3.10 JavaScript Execution

JavaScript Processing

- **Parsing:**
  - Parses JavaScript files as they're received
- **Execution:**
  - Executes parsed scripts, potentially modifying DOM
- **Event Handling:**
  - Sets up event listeners as specified in JavaScript

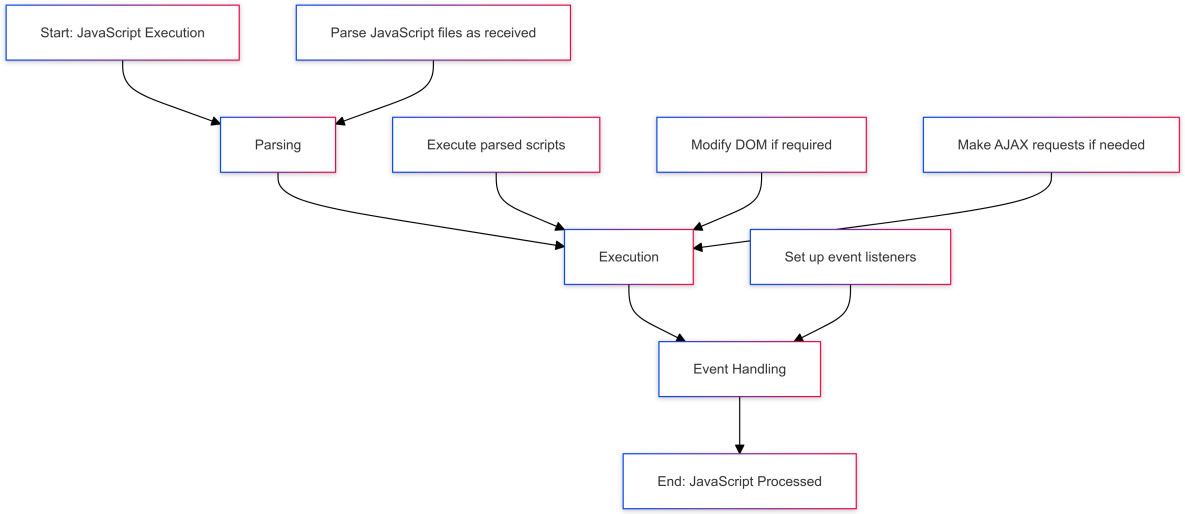


Figure 15: JavaScript Processing

4 Conclusion

This document has provided a comprehensive and sequential analysis of the TCP/IP model layers involved when a user types "google.com" into a web browser and hits Enter. We've explored each step in detail, from the initial user input to the final physical transmission of data.

Key points covered include:

- URL parsing and DNS resolution in the Application Layer
- TCP connection establishment and TLS handshake in the Transport Layer

- IP packet formation and routing in the Internet Layer
- ARP process and Ethernet frame creation in the Network Access Layer
- Physical signal transmission in the Physical Layer

Additional important aspects discussed:

- The role of HSTS in enforcing HTTPS connections
- Differences between HTTP/1.1, HTTP/2, and HTTP/3
- Variations in processes for IPv4 and IPv6

Understanding this process provides valuable insights into network operations, troubleshooting, and optimization. It highlights the complex interactions between different protocols and technologies that make modern internet communication possible.