What Happens When We Type GOOGLE.com and Hit Enter

A Comprehensive Analysis Based on the TCP/IP Model

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

July 25, 2024

Introduction

In today's digital world, a simple action we do many times a day involves a complex series of events behind the scenes. Typing "GOOGLE.com" into our web browser and pressing Enter kicks off an amazing process through the world of modern networking technology. This quick connection to the world's top search engine shows off years of advancements in computer networking and the sturdy structure of the Internet.

At the core of this process is the TCP/IP model. This model is the main framework for how data is packaged, addressed, sent, routed, and received over the Internet. It has four layers that manage a set of protocols and processes working together to make sure our request gets to Google's servers and brings back the search results.

We'll dive into each layer of the TCP/IP model to understand the journey of our Google search request. Starting from the application layer where our browser creates the request, moving through the transport layer that ensures data is sent reliably, then to the Internet layer that routes our packets worldwide, and finally to the link layer that deals with the actual data transmission. We'll see how these protocols and technologies work together to make our Internet experience smooth and efficient.

This journey will not only highlight the technical wonders behind our daily Internet use but also give us an understanding of the strength, scalability, and efficiency of the systems that keep us connected. Whether you're a budding network engineer, a curious tech enthusiast, or just someone who's ever wondered how that Google search bar works, this exploration of the TCP/IP model will be an enlightening adventure into the foundations of our digital world.

1.1 Overview of the TCP/IP Model

The TCP/IP model is divided into four distinct layers, each responsible for different aspects of data communication:

1.1.1 Application Layer

The Application Layer is where the user interacts with the network. This layer includes protocols like HTTP, FTP, SMTP, and DNS, which are responsible for managing communication between network applications.

1.1.2 Transport Layer

The Transport Layer ensures reliable data transmission. Protocols like TCP and UDP operate at this layer, providing services such as error checking, data flow control, and re-transmission of lost packets.

1.1.3 Network Layer

The Network Layer is responsible for routing packets across the network. This layer uses protocols like IP, ICMP, and ARP to manage logical addressing and routing of data packets from the source to the destination.

1.1.4 Network Access Layer

The Network Access Layer handles the physical transmission of data. It includes technologies like Ethernet, Wi-Fi, and PPP that manage the actual hardware and transmission protocols required for data to travel across network mediums.

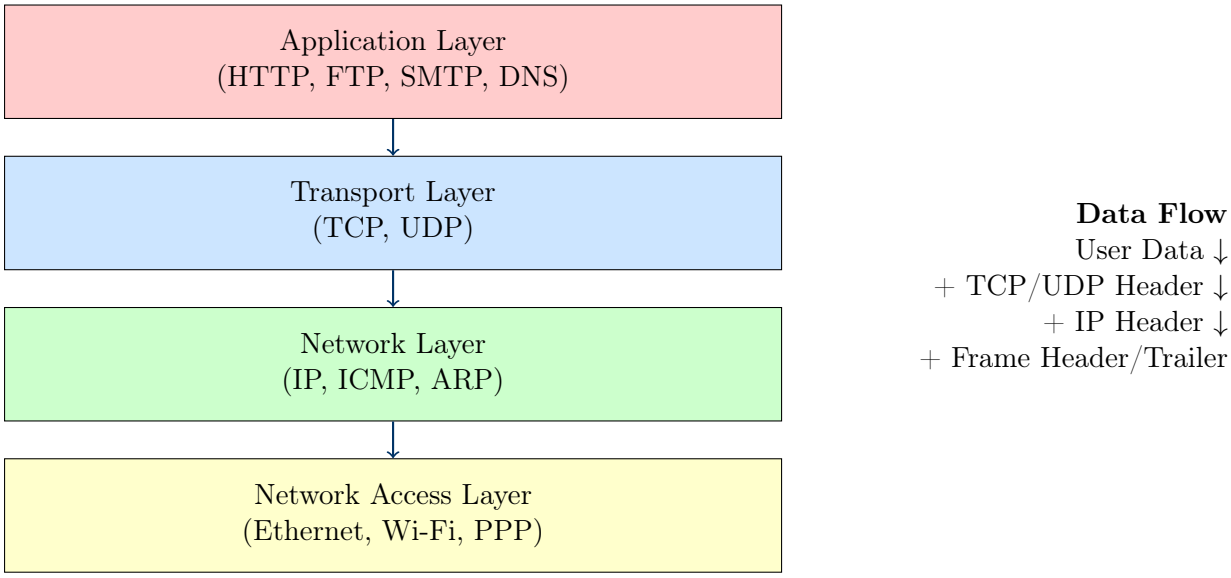


Figure 1.1: TCP/IP Model Layers and Their Functions

Application Layer

2.1 Application Layer Overview

Application Layer Functions

The application layer, being the topmost layer , serves as the direct point of contact for users and software applications. It's responsible for:

- Identifying communication partners
- Determining resource availability
- Synchronizing communication
- Providing application services to software applications

Unlike lower layers, which focus on moving data, the application layer is concerned with the semantics of the data being communicated.

2.2 Detailed Application Layer Processes

2.2.1 URL Parsing and Validation

URL Parsing Steps

1. **Scheme Identification:**
   - Browser identifies the scheme (protocol) to be used
   - If no scheme is specified in "google.com", defaults to "http://"
   - Modern browsers might attempt "https://" first
2. **Domain Extraction:**
   - Extracts "google.com" as the domain
   - Separates into second-level domain "google" and top-level domain "com"
3. **Path and Query String Analysis:**
   - No specific path or query string in this case
   - Defaults to requesting the root path "/"
4. **Fragment Identifier Check:**
   - No fragment identifier (e.g., section) in this URL

2.2.2 HSTS (HTTP Strict Transport Security) Evaluation

HSTS Evaluation Process

1. **Preloaded HSTS Check:**
   - Browser checks its preloaded HSTS list
   - Google.com is typically in this list for major browsers
2. **Previously Stored HSTS Policy Check:**
   - If not preloaded, checks for a previously stored HSTS policy
3. **HSTS Policy Application:**
   - If found, automatically upgrades the request to HTTPS
   - Occurs before any network traffic is sent

2.2.3 DNS (Domain Name System) Resolution

DNS Resolution Process

1. **Local DNS Cache Check:**
   - Browser first checks its local DNS cache
   - Cache typically stores DNS records for a short period
2. **Operating System DNS Cache Check:**
   - If not found in browser cache, OS's DNS cache is checked
3. **Hosts File Check:**
   - System checks the local hosts file for manual IP mapping
4. **Resolver Cache Check:**
   - Configured DNS resolver checks its cache
5. **Recursive DNS Query:**
   - If IP not found in any cache, initiates recursive DNS query
   - Queries root DNS server, then TLD server, then authoritative server
6. **DNS Record Types:**
   - Typically looks for A record (IPv4) or AAAA record (IPv6)
   - Other record types like CNAME might be involved
7. **TTL (Time To Live) Processing:**
   - Each DNS record comes with a TTL value
   - Browser and intermediate DNS servers cache the result for TTL duration

2.2.4 Application Protocol Selection

Protocol Selection

- **Protocol Determination:**
  - Based on HSTS check and scheme identified in URL parsing
  - Selects HTTPS for Google.com
- **Port Selection:**
  - For HTTPS, port 443 is selected by default
  - If HTTP was used (unlikely for Google), it would be port 80

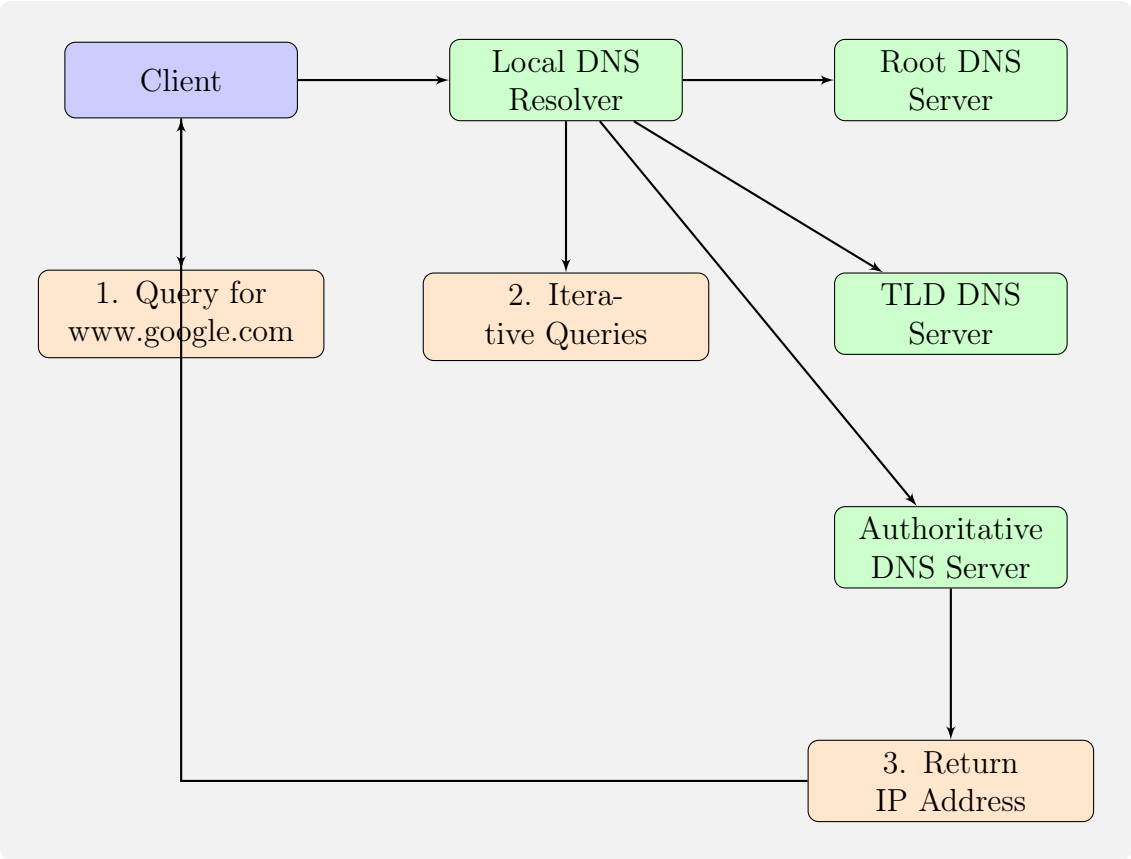


Figure 2.1: DNS Resolution Process

2.2.5 TCP Socket Initialization

TCP Socket Setup

- **Socket Creation:**
  - Browser creates a TCP socket
  - Specifies destination IP and port 443
- **TCP Handshake:**
  - SYN packet sent to server
  - Server responds with SYN-ACK
  - Client sends ACK

2.2.6 TLS (Transport Layer Security) Handshake

TLS Handshake Process

1. **Client Hello:**
   - Browser sends Client Hello message
   - Includes supported TLS versions, cipher suites, compression methods
   - Sends ClientRandom for key generation
2. **Server Hello:**
   - Server responds with chosen TLS version, cipher suite, compression method
   - Sends ServerRandom and digital certificate
3. **Certificate Validation:**
   - Browser validates server's certificate
   - Checks issuer, expiration, and domain name
4. **Key Exchange:**
   - For RSA: Browser generates and encrypts pre-master secret
   - For Diffie-Hellman: Exchange parameters for shared secret
5. **Finished Messages:**
   - Both client and server send encrypted "Finished" messages

2.2.7 HTTP Request Preparation

HTTP Request Components

- **Request Line Construction:**
  - Constructs `GET / HTTP/2`
- **Header Compilation:**
  - Adds various headers (Host, User-Agent, Accept, etc.)
- **Cookie Handling:**
  - Checks for stored cookies for "google.com"
  - Adds Cookie header if found
- **Request Body:**
  - Typically no request body for GET request to homepage

2.2.8 Request Transmission

Request Transmission Process

- **HTTP/2 Framing:**
  - Request divided into frames if using HTTP/2
- **TLS Encryption:**
  - Entire HTTP request encrypted using TLS session keys
- **Packet Fragmentation:**
  - Encrypted data fragmented into TCP packets
  - Each packet includes sequence numbers for reassembly

2.2.9 Response Processing

Response Handling

1. **Initial Response Parsing:**
   - Browser receives and decrypts data from server
2. **Status Line Interpretation:**
   - Reads status line (e.g., `HTTP/2 200 OK`)
3. **Header Processing:**
   - Processes various response headers
4. **Body Decompression:**
   - Decompresses content if compressed
5. **Content Parsing Initiation:**
   - Begins parsing HTML content as it's received

2.2.10 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

2.2.11 JavaScript Execution

JavaScript Processing

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

Transport Layer

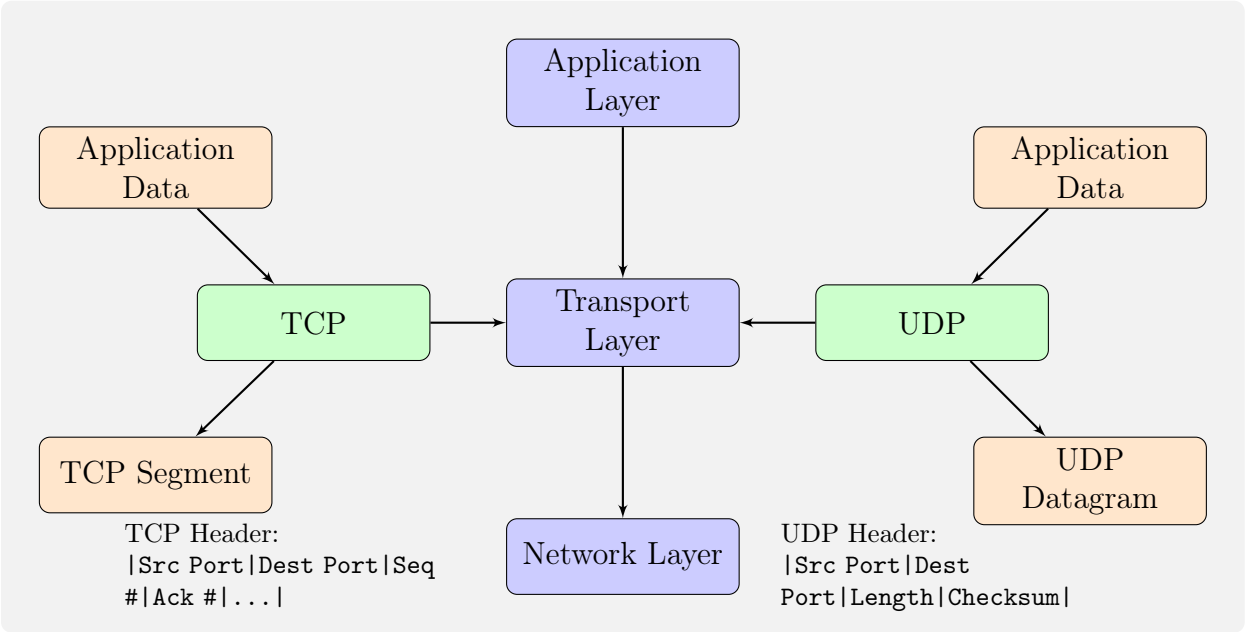


Figure 3.1: Transport Layer in the TCP/IP Model

Transport Layer Functions

The transport layer, the second layer , is responsible for end-to-end communication between applications. Its primary functions include:

- Segmentation and reassembly of data
- Connection-oriented and connectionless communication
- Reliability through error detection and recovery
- Flow control
- Congestion control
- Multiplexing and demultiplexing of application data

The two main protocols in the transport layer are TCP (Transmission Control Protocol) and UDP (User Datagram Protocol).

3.1 Detailed Transport Layer Processes

3.1.1 Protocol Selection

Protocol Selection Process

1. **Application Requirements Analysis:**
   - Web browsing typically requires reliable, ordered delivery
2. **Protocol Choice:**
   - TCP is chosen for HTTP/HTTPS connections to google.com
   - UDP might be used for some background services or DNS lookups
3. **Port Number Assignment:**
   - Client OS assigns an ephemeral source port (typically > 1024)
   - Destination port is set to 80 (HTTP) or 443 (HTTPS) for google.com

3.1.2 TCP Connection Establishment (Three-Way Handshake)

TCP Handshake Process

1. **SYN Packet:**
   - Client sends SYN packet with initial sequence number
   - SYN flag set, ACK flag clear
2. **SYN-ACK Packet:**
   - Server responds with SYN-ACK packet
   - Acknowledges client's sequence number
   - Sends its own initial sequence number
3. **ACK Packet:**
   - Client sends ACK packet
   - Acknowledges server's sequence number
4. **Connection Establishment:**
   - After ACK, connection is established
   - Both sides can begin sending data

3.1.3 Data Segmentation

Segmentation Process

1. **Receiving Application Data:**
   - Transport layer receives data stream from application layer
2. **Maximum Segment Size (MSS) Determination:**
   - Typically negotiated during connection setup
   - Often based on MTU (Maximum Transmission Unit) of the network
3. **Segmentation:**
   - Divides data into segments of appropriate size
   - Adds TCP header to each segment
4. **Sequence Number Assignment:**
   - Assigns sequence numbers to ensure proper ordering

3.1.4 Flow Control

Flow Control Mechanisms

- **Window Size Advertisement:**
  - Receiver advertises available buffer space in window field
  - Sender limits data in flight to this window size
- **Dynamic Window Sizing:**
  - Window size can change during transmission
  - Allows receiver to slow down sender if necessary
- **Zero Window Handling:**
  - If receiver advertises zero window, sender stops transmission
  - Sender periodically sends window probe packets

3.1.5 Error Detection and Correction

Error Handling Mechanisms

1. **Checksum Calculation:**
   - Calculates checksum over segment header and data
   - Includes pseudo-header with IP addresses for additional protection
2. **Acknowledgment System:**
   - Receiver acknowledges successfully received data
   - Uses cumulative ACKs: ACK number indicates next expected byte
3. **Retransmission:**
   - Sender retransmits unacknowledged segments after timeout
   - Fast Retransmit: retransmit after receiving duplicate ACKs

3.1.6 Congestion Control

Congestion Control Mechanisms

- **Slow Start:**
  - Begins with small congestion window
  - Exponentially increases window size until threshold
- **Congestion Avoidance:**
  - Linear increase of congestion window after threshold
  - Aims to find equilibrium point
- **Fast Recovery:**
  - After packet loss, reduces window but not to initial size
  - Allows quicker recovery from congestion
- **Explicit Congestion Notification (ECN):**
  - If supported, allows routers to signal congestion
  - Helps prevent packet drops due to congestion

3.1.7 Data Transmission

Data Transmission Process

1. **Segment Preparation:**
   - Adds necessary headers (sequence number, checksum, etc.)
2. **Transmission:**
   - Passes segments to IP layer for transmission
3. **Acknowledgment Handling:**
   - Processes incoming ACKs
   - Updates sliding window and congestion window
4. **Retransmission Timer Management:**
   - Sets and adjusts retransmission timers
   - Uses adaptive algorithms to estimate appropriate timeout values

3.1.8 Data Reception and Reassembly

Data Reception Process

1. **Segment Reception:**
   - Receives segments from IP layer
2. **Checksum Verification:**
   - Verifies segment integrity using checksum
3. **Sequence Number Check:**
   - Ensures segments are in correct order
   - Handles out-of-order segments
4. **Acknowledgment Generation:**
   - Sends ACKs for received segments
   - May use delayed ACKs to reduce overhead
5. **Data Reassembly:**
   - Reassembles segments into complete data stream
   - Passes reassembled data to application layer

3.1.9 Connection Termination

TCP Connection Termination Process

1. **FIN Packet (Initiator):**
   - Either client or server sends FIN packet
2. **ACK Packet (Receiver):**
   - Other side acknowledges FIN
3. **FIN Packet (Receiver):**
   - Other side sends its own FIN when ready to close
4. **ACK Packet (Initiator):**
   - Original initiator acknowledges final FIN
5. **Time-Wait State:**
   - Initiator enters TIME-WAIT state
   - Ensures all packets have died out in network

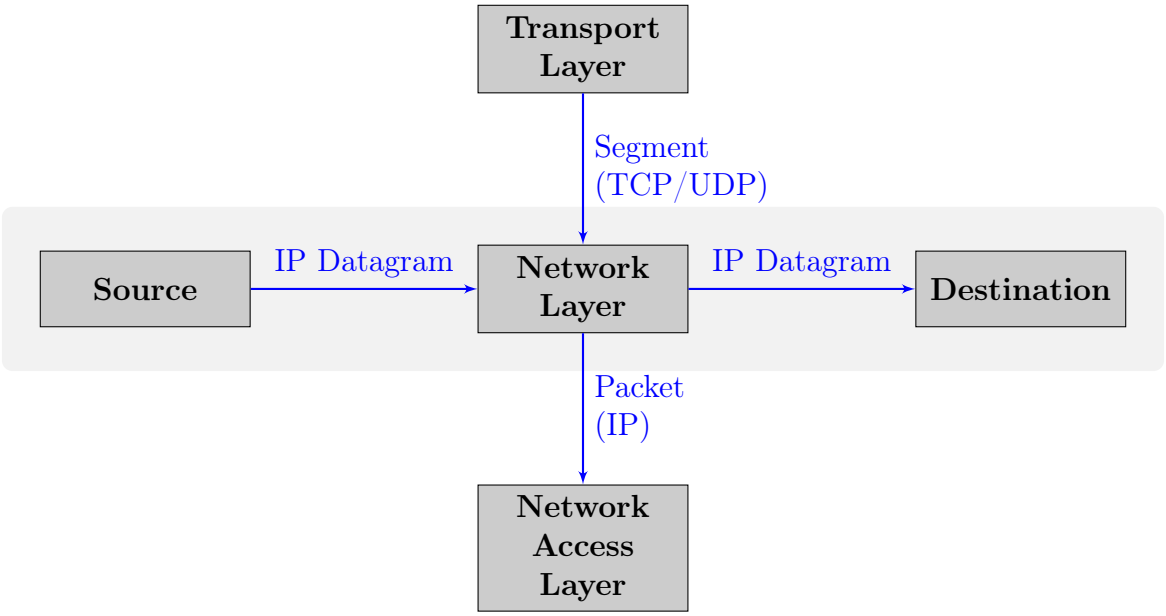
3.2 UDP Considerations

UDP Operations

While TCP is the primary protocol for web browsing, UDP may be used for some aspects:

- **DNS Lookups:**
  - UDP is typically used for DNS queries (port 53)
  - Provides faster, lightweight communication for name resolution
- **QUIC Protocol:**
  - Google's QUIC protocol uses UDP as its base
  - Provides TCP-like reliability with reduced latency
- **Stateless Nature:**
  - No connection establishment or termination
  - Each datagram handled independently
- **Simple Header:**
  - Contains only source and destination ports, length, and checksum

Network Layer



IP Header Structure:
`|Ver|IHL|ToS|TLen|ID|Flags|Frag|TTL|Prot|Checksum|Src IP|Dest IP|`

Figure 4.1: Network Layer in the TCP/IP Model

Network Layer Functions

The Network Layer is responsible for packet forwarding including routing through intermediate routers. Its primary functions include:

- Logical addressing
- Routing
- Path determination
- Packet forwarding
- Fragmentation and reassembly

4.1 Detailed Network Layer Processes

4.1.1 IP Address Resolution

IP Address Resolution Process

1. **DNS Resolution Completion:**
   - Receives resolved IP address for google.com from upper layers
   - Typically resolves to multiple IP addresses for load balancing
2. **ARP (Address Resolution Protocol) Process:**
   - Translates IP addresses to MAC addresses for local network communication
   - Broadcasts ARP request if destination MAC is unknown
   - Caches results for future use

4.1.2 IP Packet Formation

Packet Formation Process

1. **Packet Header Creation:**
   - Constructs IPv4 or IPv6 header
   - Sets source IP (your device) and destination IP (Google server)
2. **Header Fields Population:**
   - Version: Set to 4 (IPv4) or 6 (IPv6)
   - IHL (Internet Header Length): Typically 5 for IPv4 (20 bytes)
   - DSCP (Differentiated Services Code Point): Set for QoS if applicable
   - ECN (Explicit Congestion Notification): If supported and enabled
   - Total Length: Sum of header and payload lengths
   - Identification: Unique identifier for packet fragments
   - Flags: Sets DF (Don't Fragment) for IPv4 if path MTU discovery is active
   - Fragment Offset: 0 for unfragmented packet
   - TTL (Time To Live): Typically starts at 64 or 128
   - Protocol: Set to 6 for TCP
   - Header Checksum: Calculated over the header
3. **Options and Padding:**
   - Rarely used in modern networks
   - Ensures header is multiple of 32 bits

4.1.3 Routing Decision

Routing Decision Process

1. **Destination IP Analysis:**
   - Compares destination IP with local subnet mask
2. **Routing Table Lookup:**
   - Consults local routing table for next hop
   - Determines if packet should be sent to default gateway
3. **Policy-Based Routing:**
   - Applies any configured routing policies
   - May route based on source IP, protocol, or other criteria

4.1.4 Path Determination

Path Determination Process

1. **Metric Calculation:**
   - Considers factors like hop count, bandwidth, delay
   - Uses routing protocol's metric system (e.g., OSPF cost, BGP attributes)
2. **Best Path Selection:**
   - Chooses optimal path based on calculated metrics
   - Considers redundant paths for fault tolerance
3. **Equal Cost Multi-Path (ECMP):**
   - Distributes traffic across multiple equal-cost paths if available

4.1.5 Packet Forwarding

Packet Forwarding Process

1. **Next Hop Determination:**
   - Identifies next router in the path to Google's servers
2. **TTL Decrement:**
   - Decreases TTL value by 1
   - Drops packet and sends ICMP Time Exceeded if TTL reaches 0
3. **Checksum Recalculation:**
   - Recalculates header checksum after TTL modification

4.1.6 Fragmentation and Reassembly

Fragmentation and Reassembly Process

1. **MTU Consideration:**
   - Compares packet size with outgoing interface's MTU
2. **Fragmentation Process (if necessary):**
   - Splits packet into smaller fragments if larger than MTU
   - Sets fragmentation flags and offsets in IP header
3. **Path MTU Discovery:**
   - Uses ICMP to determine smallest MTU along the path
   - Adjusts packet size to avoid fragmentation

4.2 Additional Considerations

4.2.1 Quality of Service (QoS)

QoS Implementation

- **Traffic Classification:**
  - Identifies packet type (e.g., HTTP traffic to google.com)
- **Policy Application:**
  - Applies QoS policies based on classification
  - May prioritize or rate-limit based on configured rules
- **DSCP Marking:**
  - Sets appropriate DSCP value in IP header for end-to-end QoS

4.2.2 Network Address Translation (NAT)

NAT Process

- **Source NAT:**
  - Replaces private source IP with public IP if behind NAT
  - Updates IP header and recalculates checksum
- **Port Address Translation:**
  - Modifies source port if using PAT (Port Address Translation)
  - Maintains NAT translation table

Network Access Layer

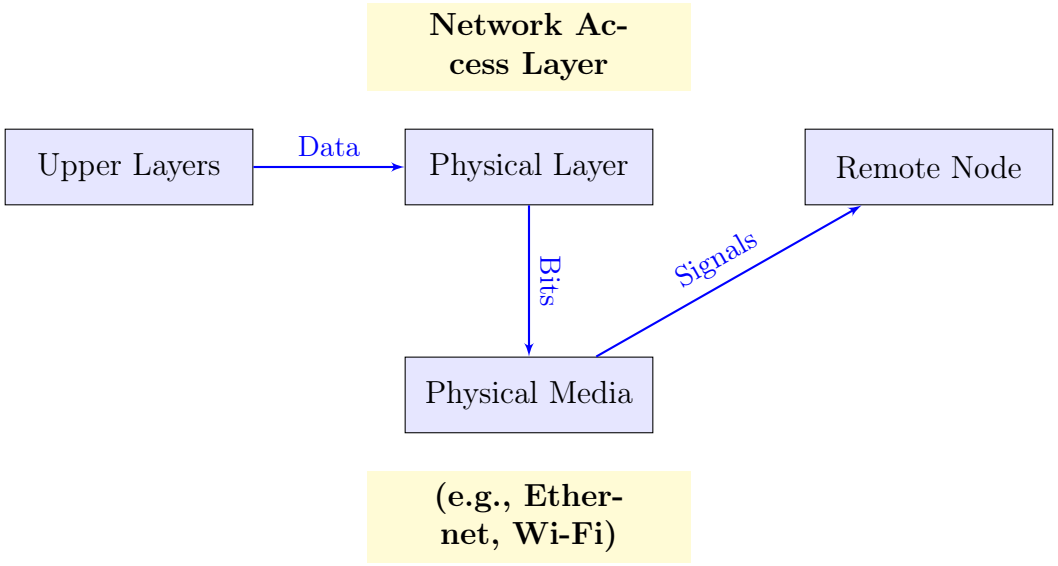


Figure 5.1: Network Interface Layer Diagram

Network Interface Layer Functions

The Network Interface Layer is responsible for the actual transmission of data over the physical network. Its primary functions include:

- Physical addressing (MAC addressing)
- Frame formation and transmission
- Media access control
- Error detection and handling

5.1 Detailed Network Interface Layer Processes

5.1.1 Physical Addressing

Physical Addressing Process

1. **MAC Address Resolution:**
   - Uses ARP (Address Resolution Protocol) to resolve IP addresses to MAC addresses
   - Broadcasts ARP request if the MAC address is not in the ARP cache
2. **MAC Address Assignment:**
   - Assigns source MAC address (of the network interface card)
   - Sets destination MAC address (usually the gateway router for internet traffic)

5.1.2 Frame Formation

Frame Formation Process

1. **Encapsulation:**
   - Encapsulates the IP packet received from the Internet Layer
   - Adds frame header and trailer
2. **Frame Header Creation:**
   - Includes source and destination MAC addresses
   - Adds frame type/length field
3. **Frame Trailer Creation:**
   - Appends Frame Check Sequence (FCS) for error detection

5.1.3 Media Access Control

Media Access Control Process

1. **Channel Access:**
   - For Ethernet: Implements CSMA/CD (Carrier Sense Multiple Access with Collision Detection)
   - For Wi-Fi: Implements CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance)
2. **Transmission Timing:**
   - Determines when it's safe to transmit data on the shared medium
   - Handles collisions and retransmissions if necessary

5.1.4 Physical Transmission

Physical Transmission Process

1. **Signal Encoding:**
   - Converts digital data into appropriate signals for the physical medium
   - For Ethernet: Electrical signals
   - For Wi-Fi: Radio waves
2. **Transmission:**
   - Sends the encoded signals over the physical medium
   - For Ethernet: Through cables
   - For Wi-Fi: Through the air

5.1.5 Error Detection and Handling

Error Detection and Handling Process

1. **Error Checking:**
   - Uses Frame Check Sequence (FCS) to detect transmission errors
2. **Error Handling:**
   - Discards frames with detected errors
   - Relies on higher layers (e.g., TCP) for retransmission of lost data

5.2 Specific Technologies

Specific Network Interface Technologies

- **Ethernet:**
  - Uses CSMA/CD for media access control
  - Defines frame format with preamble, SFD, addresses, type, data, and FCS
- **Wi-Fi (IEEE 802.11):**
  - Uses CSMA/CA for media access control
  - Implements additional features like authentication and encryption
- **PPP (Point-to-Point Protocol):**
  - Used for direct connections (e.g., dial-up internet)
  - Provides features like authentication and compression

5.3 Interaction with Upper Layer

Interaction with Internet Layer

- **Receiving from Internet Layer:**
  - Accepts IP packets from the Internet Layer
  - Prepares these packets for transmission over the physical network
- **Sending to Internet Layer:**
  - Receives frames from the physical network
  - Decapsulates the IP packets and passes them to the Internet Layer

5.4 Security Considerations

Security Measures

- **MAC Filtering:**
  - Can implement MAC address filtering for basic network access control
- **Encryption:**
  - For Wi-Fi: Implements WPA/WPA2/WPA3 encryption
- **Physical Security:**
  - Ensures physical security of network cables and devices

5.5 Performance Optimization

Optimization Techniques

- **Frame Aggregation:**
  - In Wi-Fi: Combines multiple frames for more efficient transmission
- **Quality of Service (QoS):**
  - Implements QoS mechanisms to prioritize certain types of traffic
- **Link Speed Negotiation:**
  - Negotiates optimal link speed based on network capabilities

Conclusion

As we wrap up our exploration of what happens when you type "GOOGLE.com" and press Enter, it's amazing to see how the TCP/IP model's four layers work together so smoothly. What seems like a simple action starts a complex process involving many protocols and steps, each crucial for getting information from the web to us.

We started at the **Application Layer**, where your request begins. Here, HTTP creates your query, and DNS converts "GOOGLE.com" into an IP address that computers can understand. This layer prepares your request for its journey across the Internet.

Next, we looked at the **Transport Layer**. TCP acts like a careful postal service, making sure your data packets are correctly addressed and ordered. It sets up a reliable connection with Google's servers, managing data flow and dealing with any issues like network congestion or lost packets. This layer helps keep everything running smoothly even when networks are busy.

Then, we moved to the **Network Layer**, where IP handles routing your data. This layer breaks your request into packets and sends them through the Internet. IP makes quick decisions to find the best route through a network of routers and networks, showing how robust and scalable the Internet is.

Finally, we reached the **Network Interface Layer**. This layer turns your digital request into physical signals that travel through networks, whether by Ethernet cables or Wi-Fi. It manages the details of sending data and checking for errors.

Looking back, it's impressive how efficiently and elegantly this process works. From the high-level tasks of the Application Layer to the physical data transmission of the Network Interface Layer, every part plays a role in getting your request to Google's servers and back in just seconds.

In a nutshell, this simple web search shows the essence of the Internet age—connecting people, ideas, and information quickly and reliably, a feat that once seemed like science fiction. As technology keeps advancing, the principles of the TCP/IP model will keep shaping the future of digital communication.