What Happens When We Type GOOGLE.com and Hit Enter

IN APPLICATION LAYER

A Comprehensive Analysis Based on the TCP/IP Model

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

July 22, 2024

1 Introduction

When a user types "google.com" into a web browser's address bar and presses Enter, a complex series of events unfolds at the application layer.

2 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.

3 Detailed Application Layer Processes

3.1 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

3.2 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

3.3 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

3.4 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

3.5 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

3.6 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

3.7 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

3.8 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

3.9 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

3.10 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

3.11 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

4 Conclusion

The application layer processes involved in entering "google.com" into a browser are complex and multifaceted. From URL parsing to JavaScript execution, each step plays a crucial role in delivering the final web page to the user. Understanding these processes is key to optimizing web applications and troubleshooting issues in web communication.