# Application Layer: Web and HTTP

Study-Ready Notes

Compiled by Andrew Photinakis

October 11, 2025

## Contents

**1 Introduction to Application Layer** 4

**2 Web and HTTP Fundamentals** 4
- 2.1 Web Page Composition . . . 4
- 2.2 HTTP Overview . . . 4

**3 HTTP Connections** 5
- 3.1 Connection Types . . . 5
- 3.2 Non-persistent HTTP Example . . . 5
- 3.3 Response Time Analysis . . . 6

**4 Persistent HTTP (HTTP 1.1)** 6
- 4.1 Non-persistent HTTP Issues . . . 6
- 4.2 Persistent HTTP Advantages . . . 6

**5 HTTP Message Format** 7
- 5.1 HTTP Request Message . . . 7
- 5.2 HTTP Request Methods . . . 7
- 5.3 HTTP Response Message . . . 7
- 5.4 HTTP Response Status Codes . . . 8

**6 HTTP in Practice** 8
- 6.1 Trying HTTP with Netcat . . . 8
- 6.2 HTTP State Management: Cookies . . . 8
- 6.3 Cookie Applications and Privacy . . . 9

**7 Web Caching and Performance** 9
- 7.1 Web Caches (Proxy Servers) . . . 9
- 7.2 Cache Control Headers . . . 10
- 7.3 Benefits of Web Caching . . . 10

**8 Caching Performance Analysis** **10**
8.1 Base Scenario . . . . . . . . . . 10
8.2 Option 1: Faster Access Link . . . . . . . . . . 10
8.3 Option 2: Web Cache . . . . . . . . . . 11

**9 Advanced HTTP Features** **11**
9.1 Conditional GET . . . . . . . . . . 11
9.2 HTTP/2 . . . . . . . . . . 11
9.2.1 HOL Blocking Mitigation . . . . . . . . . . 12
9.3 HTTP/3 . . . . . . . . . . 12

# Keywords

**Keywords:**

- Application Layer
- Network Applications
- Client-Server Paradigm
- Peer-to-Peer (P2P) Architecture
- Processes and Sockets
- Port Numbers
- Application-Layer Protocols
- HTTP (Hypertext Transfer Protocol)
- SMTP (Simple Mail Transfer Protocol)
- IMAP (Internet Message Access Protocol)
- DNS (Domain Name System)
- Transport Layer Services
- TCP (Transmission Control Protocol)
- UDP (User Datagram Protocol)
- Data Integrity
- Throughput Requirements
- Timing Constraints
- TLS (Transport Layer Security)
- Socket Programming
- CDNs (Content Delivery Networks)
- Video Streaming
- Network Security
- Protocol Design

# 1 Introduction to Application Layer

- Principles of network applications
- Web and HTTP (part 1 and 2)
- E-mail, SMTP, IMAP
- The Domain Name System: DNS
- P2P applications
- Video streaming, CDNs
- Socket programming with UDP and TCP

[Summary: The application layer is the top layer in network protocols, handling user-facing services like web browsing, email, and file transfer. It defines how applications communicate over networks.]

# 2 Web and HTTP Fundamentals

## 2.1 Web Page Composition

- Web pages consist of multiple **objects**
- Each object can be stored on different web servers
- Objects can be: HTML files, JPEG images, Java applets, audio files, etc.
- Base HTML file includes references to other objects
- Each object is addressable by a **URL**

**URL Structure:**

```
www.someschool.edu/someDept/pic.gif
|                 |    |
host name         |   path name
               domain
```

## 2.2 HTTP Overview

- **HTTP:** Hypertext Transfer Protocol
- Web's application-layer protocol
- Uses **client/server model**:
  - **Client:** Browser that requests, receives, and displays web objects

| | | |
|---|---|---|
| PC running Firefox browser | → HTTP requests/responses | Server running Apache Web server |
| iPhone running Safari browser | → HTTP requests/responses | |

Figure 1: HTTP Client-Server Architecture

- **Server:** Web server that sends objects in response to requests

[Summary: HTTP is the foundation of web communication, using a client-server model where browsers request resources and servers provide them.]

# 3 HTTP Connections

## 3.1 Connection Types

- **Non-persistent HTTP:**
  1. TCP connection opened
  2. At most one object sent over TCP connection
  3. TCP connection closed
  4. Downloading multiple objects requires multiple connections
- **Persistent HTTP:**
  - TCP connection opened to a server
  - Multiple objects can be sent over *single* TCP connection
  - TCP connection closed after all objects transferred

## 3.2 Non-persistent HTTP Example

**Scenario:** User enters URL: www.someSchool.edu/someDepartment/home.index (contains text + 10 JPEG references)

1. HTTP client initiates TCP connection to HTTP server on port 80
2. HTTP client sends HTTP request message with URL
3. HTTP server receives request, forms response with requested object
4. HTTP server closes TCP connection
5. HTTP client receives response, displays HTML, finds 10 referenced JPEGs
6. Steps 1-5 repeated for each of 10 JPEG objects

### 3.3 Response Time Analysis

- **RTT (Round Trip Time):** Time for small packet to travel from client to server and back
- **Non-persistent HTTP Response Time:**

$$\text{Response Time} = 2 \times \text{RTT} + \text{File Transmission Time}$$

  - One RTT to initiate TCP connection
  - One RTT for HTTP request and first bytes of response
  - Object/file transmission time

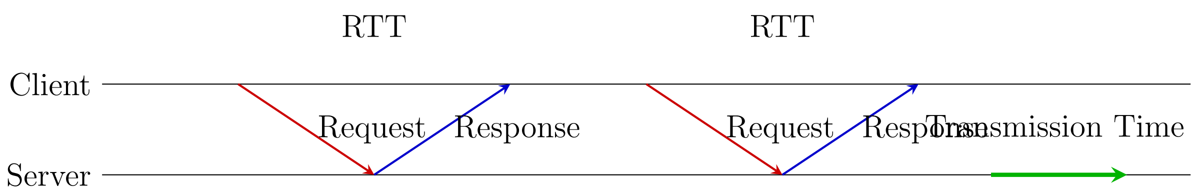


Figure 2: Non-persistent HTTP Response Time Components (colored by message type)

[Mnemonic: Non-persistent = "One and Done" - one object per connection] [Summary: Non-persistent HTTP requires separate connections for each object, leading to 2 RTTs per object, while persistent HTTP reuses connections for multiple objects.]

## 4 Persistent HTTP (HTTP 1.1)

### 4.1 Non-persistent HTTP Issues

- Requires 2 RTTs per object
- OS overhead for each TCP connection
- Browsers open multiple parallel TCP connections to fetch objects

### 4.2 Persistent HTTP Advantages

- Server leaves connection open after sending response
- Subsequent HTTP messages use same connection
- Client sends requests immediately upon encountering referenced objects
- As little as one RTT for all referenced objects
- Cuts response time in half compared to non-persistent

[Concept Map:

- HTTP 1.0 → Non-persistent → Multiple connections → High overhead
- HTTP 1.1 → Persistent → Single connection → Lower overhead
- Performance improvement: 2RTT/object → 1RTT/multiple objects

]

# 5 HTTP Message Format

## 5.1 HTTP Request Message

- ASCII (human-readable format)
- Two types: request and response messages

**General Format:**

```
| method | sp | URL | sp | version | cr | lf |
| header field name | : | value | cr | lf |
| ... |
| cr | lf |
| entity body |
```

## 5.2 HTTP Request Methods

- **GET:** Request object from server
  - Can include user data in URL after '?': www.sombsite.com/animalsearch?monkeys&banana
- **POST:** Send data to server
  - Used for form input
  - User input sent in entity body
- **HEAD:** Request headers only (no body)
- **PUT:** Upload new file to server
  - Completely replaces existing file

## 5.3 HTTP Response Message

- **Status line:** protocol + status code + status phrase
- Example: HTTP/1.1 200 OK

### 5.4 HTTP Response Status Codes

- **200 OK:** Request succeeded, object in message
- **301 Moved Permanently:** Object moved, new location specified
- **400 Bad Request:** Request not understood
- **404 Not Found:** Requested document not found
- **505 HTTP Version Not Supported:** Unsupported protocol version

[Summary: HTTP messages follow specific formats with request methods (GET, POST, HEAD, PUT) and response codes (200, 301, 404, etc.) that indicate request outcomes.]

## 6 HTTP in Practice

### 6.1 Trying HTTP with Netcat

1. Open TCP connection: `% nc -c -v gaia.cs.umass.edu 80`
2. Send GET request:

   ```
   GET /kurose_ross/interactive/index.php HTTP/1.1
   Host: gaia.cs.umass.edu
   ```

3. Examine server response

### 6.2 HTTP State Management: Cookies

- HTTP is **stateless** - no memory between requests
- **Cookies** maintain state between transactions

**Four Cookie Components:**

1. Cookie header line in HTTP *response* message
2. Cookie header line in HTTP *request* message
3. Cookie file on user's host (managed by browser)
4. Back-end database at website

| Client | | Server |
|---|---|---|
| Cookie file | → | Backend database |
| Amazon: 1678 | HTTP with cookie | ID: 1678, user data |

Figure 3: Cookie-based State Management

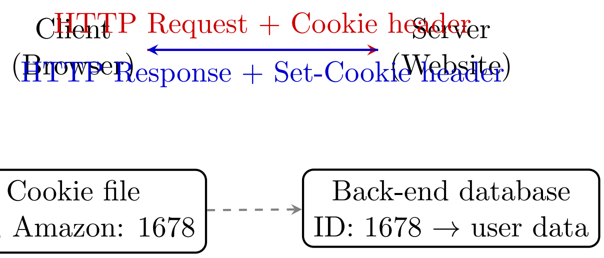


Figure 4: Cookie-based State Management: How client and server maintain user state using cookies.

### 6.3 Cookie Applications and Privacy

- **Uses:** Authorization, shopping carts, recommendations, user sessions
- **Privacy concerns:**
  - Sites learn about user behavior
  - Third-party persistent cookies track across multiple sites
  - Common identity tracking

[Mnemonic: Cookies have 4 C's: Client, Cache, Communication, Control] [Summary: Cookies overcome HTTP's stateless nature by storing user-specific data, enabling personalized experiences but raising privacy concerns.]

## 7 Web Caching and Performance

### 7.1 Web Caches (Proxy Servers)

- **Goal:** Satisfy client requests without origin server involvement
- Browser configured to point to local web cache
- Cache acts as both client and server

**Cache Operation:**

1. Browser sends request to cache
2. If object in cache: return to client
3. Else: request from origin server, cache object, return to client

### 7.2 Cache Control Headers

- **Cache-Control: max-age=¡seconds¿:** Object caching duration
- **Cache-Control: no-cache:** Do not cache object

### 7.3 Benefits of Web Caching

- Reduce client response time (cache closer to client)
- Reduce traffic on institution's access link
- Enable efficient content delivery for providers

## 8 Caching Performance Analysis

### 8.1 Base Scenario

- Access link rate: 1.54 Mbps
- RTT to server: 2 seconds
- Web object size: 100K bits
- Request rate: 15 requests/second
- Data rate to browsers: 1.50 Mbps

**Performance without Cache:**

- Access link utilization: 0.97 (97%)
- LAN utilization: 0.0015
- End-end delay: 2 sec + minutes + microseconds

### 8.2 Option 1: Faster Access Link

- Upgrade to 154 Mbps access link
- Access link utilization: 0.0097 (0.97%)
- Cost: Expensive

### 8.3 Option 2: Web Cache

- Install local web cache
- Cost: Cheap

**Performance with Cache (40% hit rate):**

- 40% requests served by cache (low msec delay)
- 60% requests served by origin
- Access link data rate: $0.6 \times 1.50$ Mbps $= 0.9$ Mbps
- Access link utilization: $0.9/1.54 = 0.58$
- Average end-end delay:

$$\begin{aligned} \text{Delay} &= 0.6 \times (\text{delay from origin}) + 0.4 \times (\text{delay from cache}) \\ &= 0.6 \times 2.01 + 0.4 \times (\text{msecs}) \approx 1.2 \text{ seconds} \end{aligned}$$

[Summary: Web caching dramatically improves performance by serving content locally, reducing both response time and network traffic compared to upgrading infrastructure.]

## 9 Advanced HTTP Features

### 9.1 Conditional GET

- **Goal:** Avoid sending object if cache has up-to-date version
- Client includes: `If-modified-since: <date>`
- Server responses:
  - **304 Not Modified:** Cached copy is current
  - **200 OK:** Object modified, new version sent

### 9.2 HTTP/2

- **Key goal:** Decrease delay in multi-object requests
- Improvements over HTTP 1.1:
  - Object transmission based on client-specified priority (not FCFS)
  - Push unrequested objects to client
  - Divide objects into frames
  - Mitigate Head-of-Line (HOL) blocking

#### 9.2.1 HOL Blocking Mitigation

- **HTTP 1.1:** Objects delivered in request order
  - Small objects wait behind large objects
- **HTTP/2:** Objects divided into frames, transmission interleaved
  - Small objects delivered quickly
  - Large objects slightly delayed

### 9.3 HTTP/3

- **Limitations of HTTP/2:**
  - Single TCP connection means packet loss stalls all objects
  - Browsers open multiple parallel TCP connections
  - No security over vanilla TCP
- **HTTP/3 improvements:**
  - Adds security
  - Per-object error and congestion control over UDP
  - More pipelining capabilities

[Concept Map:

- HTTP 1.0 → Non-persistent → High latency
- HTTP 1.1 → Persistent → Reduced connections
- HTTP/2 → Frame multiplexing → HOL blocking mitigation
- HTTP/3 → UDP-based → Per-object control + security

]

# Exam Questions

## HTTP Fundamentals

1. Compare and contrast persistent vs. non-persistent HTTP connections. What are the performance implications of each?
2. Calculate the response time for downloading a web page with 5 objects using non-persistent HTTP, given RTT = 100ms and transmission time per object = 50ms.
3. Explain how cookies overcome HTTP's stateless nature and discuss the privacy concerns associated with them.

## Caching and Performance

1. A web cache has a hit rate of 60%. The average response time for cache hits is 10ms and for cache misses is 200ms. Calculate the overall average response time.

2. Describe how conditional GET works and why it's beneficial for network performance.

3. Compare the cost-effectiveness of upgrading access link speed vs. installing a web cache for improving institutional network performance.

## HTTP Evolution

1. Explain how HTTP/2 mitigates head-of-line blocking compared to HTTP 1.1.

2. What are the main advantages of HTTP/3 over HTTP/2, and why was UDP chosen as the transport protocol?

3. Describe a scenario where HTTP/2's object prioritization would provide significant performance benefits over HTTP 1.1.

[Exam Questions: These questions cover key concepts including HTTP connection types, performance calculations, caching benefits, and protocol evolution from HTTP 1.1 to HTTP/3.]