Application Layer Overview (Layer 5 - Internet Protocol Stack)

1 Introduction

The Application Layer, the topmost layer of the Internet protocol stack, facilitates communication between applications on different hosts, providing network services for software such as web browsers, email clients, and file transfer programs.

2 Main Functions

- Facilitates process-to-process communication between applications on different hosts.
- Provides network services to user applications (e.g., web browsing, email, file transfers).
- Defines protocols for application communication.

3 Common Application Layer Protocols

Protocol	Purpose
HTTP	Transfers web pages (text, images, etc.)
SMTP	Sends emails between servers
POP3/IMAP	Retrieves emails from mail servers
DNS	Translates domain names to IP addresses
FTP	Transfers files between client and server

4 How It Works

1. A client application (e.g., browser) uses a protocol (e.g., HTTP) to request a service.
2. The server responds using the same protocol.
3. Example: A browser sends an HTTP request to a web server, which replies with the requested web page.

5 Key Concepts

- Client-Server Model: Client initiates communication; server responds (e.g., HTTP, SMTP).
- Peer-to-Peer (P2P) Model: Devices act as both client and server (e.g., BitTorrent).
- Message Format & Syntax: Protocols define message structure and interpretation.
- Port Numbers: Identify specific applications (e.g., 80 for HTTP, 25 for SMTP).

6 Application Layer vs. Transport Layer

- The Application Layer creates messages (e.g., HTTP requests) and relies on the Transport Layer (TCP/UDP) for delivery.
- Example: A browser creates an HTTP request, which TCP wraps and sends to the destination via lower layers (network, link, physical).

7 Application Architectures

1. Client-Server Architecture:
    - Centralized, always-on server handles client requests.
    - Clients do not communicate directly with each other.
    - Examples: Web (HTTP), Email (SMTP/IMAP), Cloud services.
2. Peer-to-Peer (P2P) Architecture:
    - No central server; peers act as both clients and servers.
    - Scalable but complex to manage.
    - Examples: BitTorrent, Skype, Blockchain.

8 Sockets in Process-to-Process Communication

- Definition: A socket is the interface between the application and transport layers, enabling message exchange over the network.
- Identification: Defined by an IP address and port number tuple (e.g., 142.250.190.78:80 for a web server).
- Role: Acts as a communication endpoint for client and server processes.
- Example: A browser (client) sends data to a web server's socket, which responds via its socket.

9 Transport Service Requirements

Application	Data Loss	Throughput	Time Sensitivity
File Transfer	No loss	Elastic	No
Email	No loss	Elastic	No
Web Documents	No loss	Elastic	No
Real-time Audio/Video	Loss-tolerant	5 Kbps–5 Mbps	Yes (10s of ms)
Streaming Audio/Video	Loss-tolerant	10 Kbps–5 Mbps	Yes (few seconds)
Interactive Games	Loss-tolerant	Kbps+	Yes (10s of ms)
Text Messaging	No loss	Elastic	Varies

9.1 TCP vs. UDP

- TCP (Transmission Control Protocol):
  - Features: Reliable data transfer, flow control, congestion control, connection-oriented.
  - Limitations: No timing/throughput guarantees, no built-in security.
  - Applications: Web (HTTP), Email (SMTP/IMAP), File Transfer (FTP).
- UDP (User Datagram Protocol):
  - Features: Connectionless, lightweight, low overhead.
  - Limitations: No reliability, flow/congestion control, or security.
  - Applications: Streaming, VoIP, DNS, online games (prioritize speed).

9.2 Securing Connections with TLS

- Problem: Vanilla TCP/UDP sends data in plaintext, vulnerable to interception.
- Solution: Transport Layer Security (TLS):
  - Provides encryption, data integrity, and authentication.
  - Implemented at the application layer via libraries (e.g., OpenSSL).
  - Ensures secure communication, even if intercepted.

10 HTTP (HyperText Transfer Protocol)

- Purpose: Transfers web resources (HTML, images, etc.) using a client-server model.
- Characteristics:
  - Uses TCP (port 80 or 443 for HTTPS).

  - Stateless: Each request is independent unless cookies are used.
  - Text-based (ASCII commands/responses).

- Connection Types:
  - Non-Persistent HTTP: New TCP connection per object, high latency (2 RTT + transmission time per object).
  - Persistent HTTP (HTTP/1.1 default): Single TCP connection for multiple objects, reducing latency.

- HTTP/2 Improvements:
  - Multiplexing: Splits objects into frames, interleaves them to avoid head-of-line (HOL) blocking.
  - Client-Specified Priorities: Prioritizes important objects.
  - Server Push: Sends objects proactively.
  - Result: Faster page loading, better handling of packet loss.

- HTTP/3: Uses UDP to avoid TCP-level HOL blocking, adds per-stream flow control and encryption.

10.1 HTTP Request Message

- Structure:
  1. Request Line: Method /resource HTTP/version (e.g., GET /index.html HTTP/1.1).
  2. Header Lines: Metadata (e.g., Host, User-Agent, Accept).
  3. Blank Line: Separates headers from body.
  4. Body (optional): Used in POST/PUT requests.

- Methods:
  - GET: Retrieve resource (optionally with query parameters in URL).
  - POST: Send data in body (e.g., form submissions).
  - HEAD: Retrieve headers only (no body).
  - PUT: Upload/update resource.

10.2 HTTP Response Message

- Structure:

1. Status Line: HTTP/version StatusCode StatusPhrase (e.g., HTTP/1.1 200 OK).
2. Header Lines: Metadata (e.g., Date, Content-Type).
3. Blank Line: Separates headers from body.
4. Entity Body: Requested content (e.g., HTML, image).

- Status Codes:
  - 2xx: Success (e.g., 200 OK).
  - 3xx: Redirection (e.g., 301 Moved Permanently).
  - 4xx: Client Error (e.g., 404 Not Found).
  - 5xx: Server Error (e.g., 500 Internal Server Error).

10.3 Cookies for State Management

- Problem: HTTP is stateless, complicating multi-step processes (e.g., logins, shopping carts).
- Solution: Cookies store small data (e.g., userID=12345) on the client.
- Process:
  1. Server sends Set-Cookie header in response.
  2. Client includes Cookie header in subsequent requests.
  3. Server uses cookie to retrieve session data from its database.
- Uses: Authentication, cart tracking, personalization.
- Privacy Concerns: Tracking behavior, third-party cookies.

10.4 Conditional GET

- Purpose: Avoids resending unchanged cached objects.
- Process:
  1. Client sends If-Modified-Since header with cached object's date.
  2. Server responds with 304 Not Modified if unchanged or sends new object if modified.
- Benefits: Reduces bandwidth, speeds up responses.

11 Web Caching (Proxy Servers)

- Definition: Stores copies of web content to reduce latency and bandwidth usage.
- Operation:
  1. Browser sends HTTP requests to the cache.
  2. Cache returns cached objects or fetches from the origin server, storing a copy.
- Benefits:
  - Faster response times.
  - Reduced network traffic.
  - Supports small content providers via CDNs.
- Control: Servers use Cache-Control headers (e.g., max-age, no-cache).

11.1 Caching Performance Example

- Scenario: LAN (1 Gbps), Access Link (1.54 Mbps), RTT (2s), 100 Kbit objects, 15 requests/sec.
- Without Cache:
  - Access Link Utilization: 97% (high delays).
  - Total Delay: 2s + minutes.
- With Cache (40% hit rate):
  - Utilization: 58% (lower delays).
  - Average Delay: $\sim$1.2s.
- Conclusion: Caching is cost-effective compared to upgrading bandwidth.

12 FTP (File Transfer Protocol)

- Purpose: Transfers files between client and server (RFC 959).
- Operation:
  - Control Connection (TCP port 21): Persistent, sends commands/responses (out-of-band).
  - Data Connection (TCP port 20): Temporary, transfers files/listings.
- Stateful: Tracks session state (e.g., directory, authentication).

- Why Two Connections?: Allows simultaneous command and data transfer for flexibility.

13 Email System

13.1 Components

1. User Agents: Software for composing/reading emails (e.g., Outlook, Gmail).
2. Mail Servers: Store incoming emails (user mailbox) and queue outgoing emails.
3. SMTP: Transfers emails between servers using TCP (port 25).

13.2 SMTP Operation (RFC 5321)

- Phases:
    1. Handshaking: Client and server exchange greetings (e.g., HELO, 220 Service ready).
    2. Message Transfer: Commands (MAIL FROM, RCPT TO, DATA) and responses (e.g., 250 OK).
    3. Closure: Client sends QUIT, server closes connection.
- Message End: Indicated by a single period (.) in SMTP, unlike HTTP's Content-Length.

13.3 Email Retrieval

- POP3: Downloads emails (optionally deletes from server), stateless.
- IMAP: Manages emails on server, supports folders, stateful, multi-device sync.
- HTTP: Used by webmail (e.g., Gmail) for access.

13.4 Email Message Format (RFC 2822)

- Header: Includes To, From, Subject.
- Body: Message content, separated by a blank line.

14 DNS (Domain Name System)

- Purpose: Translates domain names (e.g., www.google.com) to IP addresses (e.g., 142.250.190.78).

- Why Distributed?: Avoids single point of failure, handles high query volumes (e.g., 2.2T/day at Akamai).
- Hierarchy:
  - Root Servers: Direct to TLD servers (13 logical servers, managed by ICANN).
  - TLD Servers: Handle domains like .com, .edu.
  - Authoritative Servers: Store hostname-to-IP mappings.
  - Local DNS Servers: Cache responses, forward queries if needed.

14.1 DNS Query Types

- Recursive: Local DNS server resolves fully, client waits.
- Iterative: Client follows referrals from servers to resolve the query.

14.2 DNS Caching

- Process: Servers cache name-to-IP mappings with a TTL (e.g., 600s).
- Issue: Cached entries may be outdated until TTL expires.

14.3 DNS Records

Type	Purpose
A	Maps hostname to IPv4 (e.g., www.example.com → 192.0.2.1)
NS	Maps domain to name server (e.g., example.com → ns1.example.com)
CNAME	Maps alias to canonical name (e.g., www.ibm.com → servereast.backup2.ibm.com)
MX	Maps domain to mail server (e.g., example.com → mail.example.com)

14.4 DNS Registration

1. Register domain with a registrar (e.g., GoDaddy).
2. Provide authoritative server names/IPs.
3. Registrar adds NS/A records to TLD server.
4. Set up authoritative server with custom records (A, MX).

14.5 DNS Attacks

- DDoS: Overwhelm root/TLD servers (mitigated by caching, anycast).
- Redirection:

  - Man-in-the-Middle: Intercepts/modifies responses.
  - Cache Poisoning: Injects false IP mappings.
- Amplification: Spoofs victim's IP for large DNS responses.

15 Video Streaming and CDNs

15.1 Video Basics

- Video: Sequence of images (frames, e.g., 24 fps).
- Digital Image: Pixels with color values (bits).
- Compression:
  - Spatial Coding: Encodes repeated colors (e.g., (Purple, 5)).
  - Temporal Coding: Sends differences between frames.

15.2 Streaming Challenges

- Variable Bandwidth: Congestion affects delivery.
- Packet Loss/Delay: Causes buffering or quality reduction.
- Interactivity: Users need pause/rewind functionality.

15.3 DASH (Dynamic Adaptive Streaming over HTTP)

- Process:
  1. Video split into chunks, encoded at multiple bitrates.
  2. Manifest file lists chunk URLs/qualities.
  3. Client selects chunk quality based on bandwidth, requests from optimal CDN server.
- Benefits: Adapts quality dynamically, reduces buffering, scales via CDNs.
- Client Decisions:
  - When: Requests chunks to avoid buffer starvation/overflow.
  - What: Chooses quality based on bandwidth.
  - Where: Selects fastest CDN server.

15.4 CDNs

- Purpose: Store content on geographically distributed servers.
- Benefits: Faster delivery, reduced latency, load balancing.

15.5 Video Encoding Standards

Standard	Bit Rate
MPEG-1	1.5 Mbps
MPEG-2	3–6 Mbps
MPEG-4	64 Kbps–12 Mbps

16 Network Infrastructure

- End Systems: Use all five layers (application, transport, network, link, physical).
- Core Devices:
  - Routers: Network, link, physical layers (route packets).
  - Switches: Link, physical layers (forward frames).
  - Hubs: Physical layer (repeat signals).
- Role: Move data, not run application-layer services.

17 Program vs. Process

Feature	Program
State	Passive (code)
Storage	Disk
Lifespan	Permanent
Example	chrome.exe

- Communication:
  - Same Host: Uses IPC (e.g., shared memory).
  - Different Hosts: Uses sockets for message exchange.

18 IP Addresses

- Private IP: Local network use (e.g., 192.168.0.1), not Internet-accessible.
- Public IP: Globally unique, Internet-accessible (e.g., 8.8.8.8).