Application Layer - Computer Networks

Study-Ready Notes

Compiled by Andrew Photinakis

November 15, 2025

Contents

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 Application Layer Overview

1.1 Overview

Goal of this is to introduce the application layer of the Internet protocol stack: its goals, common application-layer protocols such as HTTP, SMTP/IMAP, DNS, P2P, streaming/CDNs, and practical programming considerations (socket API, UDP/TCP). Here we'll contrast application-level requirements with transport services (TCP vs UDP) and touches on security (TLS).

1.2 Learning Objectives

- Understand conceptual and implementation aspects of application-layer protocols
- Study transport-layer service models
- Learn client-server and peer-to-peer paradigms
- Examine popular application-layer protocols:
  - HTTP (Web)
  - SMTP, IMAP (Email)
  - DNS (Domain Name System)
- Study video streaming systems and CDNs
- Learn socket programming with UDP and TCP

[Summary: The application layer focuses on network applications, their protocols, and how they use underlying transport services. Key paradigms include client-server and P2P architectures.]

2 Network Applications and Paradigms

2.1 Common Network Applications

- Social networking
- Web browsing
- Text messaging
- Email
- Multi-user network games
- Streaming stored video (YouTube, Hulu, Netflix)

- P2P file sharing
- Voice over IP (Skype)
- Real-time video conferencing (Zoom)
- Internet search
- Remote login

[Mnemonic: WESTS - Web, Email, Streaming, Texting, Social - covers major application categories]

2.2 Creating Network Applications

- Programs run on different end systems
- Communication occurs over network
- Example: web server software communicates with browser software
- **Key Insight**: No need to write software for network-core devices
  - Network-core devices don't run user applications
  - Applications reside only on end systems
  - Enables rapid application development and propagation

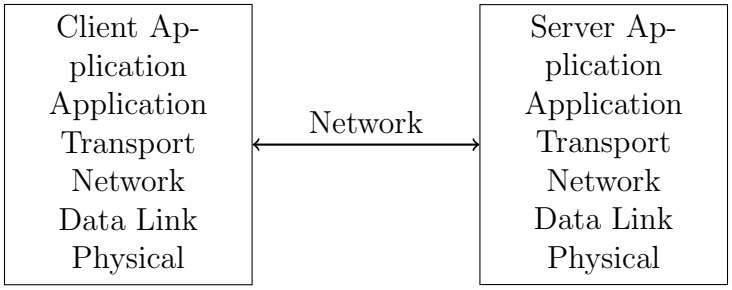

Figure 1: Network Application Architecture: Applications run on end systems using the protocol stack

2.3 Client-Server Paradigm

- **Server**:
  - Always-on host
  - Permanent IP address
  - Waits for and serves client requests

- **Clients**:
  - Contact and communicate with server
  - May be intermittently connected
  - May have dynamic IP addresses
  - Do not communicate directly with each other
- **Examples**: HTTP, IMAP, FTP

[Summary: Client-server model features dedicated servers that are always available and multiple clients that initiate connections. This is the foundation of most traditional web services.]

2.4 Peer-to-Peer (P2P) Architecture

- No always-on server
- Arbitrary end systems directly communicate
- Peers both request and provide services
- **Key Advantages**:
  - Self-scalability: New peers bring new service capacity
  - Distributed nature reduces single points of failure
- **Challenges**:
  - Peers are intermittently connected
  - Peers change IP addresses
  - Complex management and coordination
- **Example**: P2P file sharing (BitTorrent)

[Concept Map: Application Architectures → Client-Server (centralized, reliable) vs P2P (decentralized, scalable) → Hybrid approaches combine both]

3 Process Communication and Sockets

3.1 Processes Communicating

- **Process**: Program running within a host
- **Client process**: Process that initiates communication
- **Server process**: Process that waits to be contacted

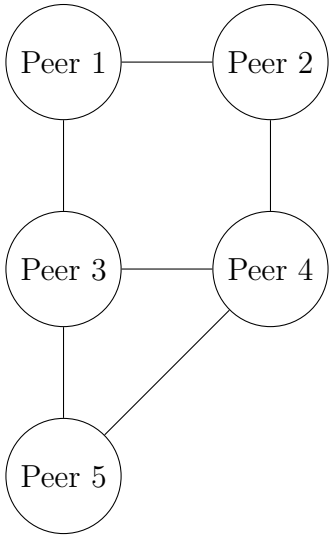

Figure 2: P2P Architecture: Peers connect directly to each other in a mesh network

- Processes on same host use inter-process communication (IPC)
- Processes on different hosts communicate by exchanging messages
- Note: P2P applications have both client and server processes

3.2 Sockets

- Process sends/receives messages to/from its socket
- **Analogy**: Socket is like a door
  - Sending process shoves message out the door
  - Transport infrastructure delivers message to receiving process's socket
- Two sockets involved: one on each communicating process
- **Developer Control**: Application developer controls application layer
- **OS Control**: Operating system controls transport layer and below

3.3 Addressing Processes

- To receive messages, process must have an identifier
- Host has unique 32-bit IP address
- IP address alone is insufficient - many processes can run on same host
- Complete identifier includes: IP address + port numbers

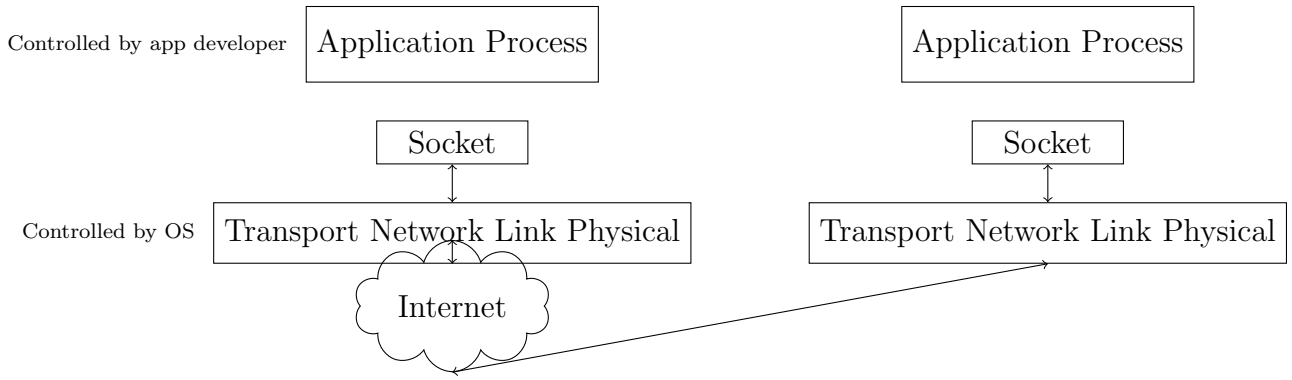

Figure 3: Socket Communication: Applications use sockets as interface to network services

- **Common Port Numbers**:
  - HTTP server: port 80
  - Mail server: port 25
  - Example: Sending HTTP to cs.rit.edu
    - IP address: 128.119.245.12
    - Port number: 80

[Summary: Processes communicate through sockets, which act as endpoints. Addressing requires both IP address and port number to uniquely identify applications on hosts.]

4 Application-Layer Protocols

4.1 Protocol Definition

An application-layer protocol defines:

- **Types of messages exchanged**: Request, response messages
- **Message syntax**: Fields and how they are delineated
- **Message semantics**: Meaning of information in fields
- **Rules**: When and how processes send and respond to messages

4.2 Protocol Types

- **Open Protocols**:
  - Defined in RFCs (Request for Comments)
  - Everyone has access to protocol definition
  - Enables interoperability

  - Examples: HTTP, SMTP

- **Proprietary Protocols**:
  - Privately owned and controlled
  - May provide competitive advantages
  - Examples: Skype, Zoom

[Mnemonic: SSTR - Syntax, Semantics, Timing, Rules - the four components of protocols]

5 Transport Service Requirements

5.1 Application Needs

Different applications have different transport service requirements:

- **Data Integrity**:
  - Some apps require 100% reliable data transfer (file transfer, web transactions)
  - Other apps can tolerate some loss (audio)
- **Throughput**:
  - Some apps require minimum throughput (multimedia)
  - Other apps are elastic (use whatever throughput available)
- **Timing**:
  - Some apps require low delay (Internet telephony, interactive games)
- **Security**: Encryption, data integrity, authentication

5.2 Common Application Requirements

[Summary: Applications have varying requirements for data integrity, throughput, and timing. Real-time applications tolerate some loss but need low delay, while data transfer applications require reliability but can tolerate delay.]

6 Internet Transport Protocols

6.1 TCP Service

- **Reliable transport** between sending and receiving process
- **Flow control**: Prevents sender from overwhelming receiver
- **Congestion control**: Throttles sender when network overloaded

Application	Data Loss	Throughput	Time Sensitive?
File transfer/download	No loss	Elastic	No
E-mail	No loss	Elastic	No
Web documents	No loss	Elastic	No
Real-time audio/video	Loss-tolerant	Audio: 5Kbps-1Mbps	
Video: 10Kbps-5Mbps	Yes, 10's msec		
Streaming audio/video	Loss-tolerant	Same as above	Yes, few secs
Interactive games	Loss-tolerant	Kbps+	Yes, 10's msec
Text messaging	No loss	Elastic	Yes and no

Table 1: Transport Service Requirements for Common Applications

- **Connection-oriented**: Setup required between client and server
- **Does not provide**: Timing, minimum throughput guarantee, security

6.2 UDP Service

- **Unreliable data transfer** between processes
- **Does not provide**: Reliability, flow control, congestion control, timing, throughput guarantee, security, or connection setup

Q: Why bother with UDP? Why is there a UDP?

- Lower overhead than TCP
- No connection establishment delay
- Simpler header and no congestion control overhead
- Suitable for applications that can tolerate some loss but need low latency
- Applications can implement their own reliability if needed

6.3 Applications and Transport Protocols

[Concept Map: Transport Protocols → TCP (reliable, connection-oriented) vs UDP (unreliable, connectionless) → Application choice depends on reliability vs latency requirements]

Application	Application Layer Protocol	Transport Protocol
File transfer/download	FTP [RFC 959]	TCP
E-mail	SMTP [RFC 5321]	TCP
Web documents	HTTP 1.1 [RFC 7320]	TCP
Internet telephony	SIP [RFC 3261], RTP [RFC 3550], or proprietary	TCP or UDP
Streaming audio/video	HTTP [RFC 7320], DASH	TCP
Interactive games	WOW, FPS (proprietary)	UDP or TCP

Table 2: Internet Applications and Their Protocols

7 Transport Layer Security (TLS)

7.1 Security in TCP/UDP

- **Vanilla TCP & UDP sockets**: No encryption
- Cleartext passwords sent into socket traverse Internet in cleartext
- Major security vulnerability

7.2 Transport Layer Security (TLS)

- Provides encrypted TCP connections
- Ensures data integrity
- Provides end-point authentication
- TLS is implemented in application layer
- Applications use TLS libraries, which use TCP in turn
- Cleartext sent into "socket" traverses Internet encrypted

[Summary: TLS provides security for TCP connections by adding encryption, data integrity, and authentication. It's implemented at the application layer but provides transport-layer security services.]

8 Course Topics Overview

8.1 Application Layer Coverage

The course will cover:

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

[Mnemonic: WED P2P VS - Web, Email, DNS, P2P, Video Streaming - major application layer topics]