Network Theory Study Guide

Units III-VI: A Comprehensive Exam Preparation Guide

Abstract

This study guide provides a comprehensive overview of network theory concepts for undergraduate students preparing for examinations. Covering 28 hours of material across four units (Network Layer, Transport Layer, Application Layer, and Network Security), this guide assumes minimal prior knowledge and builds progressively from foundational concepts to advanced applications. Each unit includes clear explanations, real-world examples, diagrams, and key concept summaries.

Contents

1 Unit III: Network Layer (7 Hours)

1.1 Introduction to the Network Layer

1.1.1 Prerequisites Review

Before diving into the Network Layer, let's review some foundational concepts:

Basic Networking Concepts

- **Network**: A collection of interconnected devices that can communicate with each other
- **Protocol**: A set of rules governing communication between devices
- **Packet**: A unit of data transmitted over a network
- **Binary and Hexadecimal**: Number systems used to represent addresses and data
- **OSI Model**: A 7-layer conceptual framework (Physical, Data Link, Network, Transport, Session, Presentation, Application)

1.1.2 What is the Network Layer?

The Network Layer (Layer 3 of the OSI model) is responsible for **routing packets** from a source host to a destination host across multiple networks. Think of it as the postal service of the internet — it determines the best path for data to travel and ensures packets reach their intended destination.

Key Responsibilities

1. **Logical Addressing**: Assigning unique IP addresses to devices
2. **Routing**: Determining the best path for packets to travel
3. **Packet Forwarding**: Moving packets from input to output ports
4. **Fragmentation and Reassembly**: Breaking large packets into smaller pieces

1.2 IPv4 Addressing

1.2.1 IPv4 Structure and Notation

An IPv4 address is a **32-bit number** divided into 4 octets (8 bits each), typically written in **dotted-decimal notation**.

IPv4 Address Example

Binary: `11000000.10101000.00000001.00000001`
Decimal: `192.168.1.1`

Each octet can range from 0 to 255 (since $2^8 = 256$ possible values).

1.2.2 Classful Addressing

Historically, IPv4 addresses were divided into five classes:

Class	First Octet	Network Bits	Host Bits	Default Mask
A	1-127	8	24	255.0.0.0 (/8)
B	128-191	16	16	255.255.0.0 (/16)
C	192-223	24	8	255.255.255.0 (/24)
D	224-239	Multicast		
E	240-255	Reserved/Experimental		

Special Addresses

- **127.0.0.0/8**: Loopback addresses (127.0.0.1 = localhost)
- **10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16**: Private addresses (RFC 1918)
- **0.0.0.0**: Default route or "this network"
- **255.255.255.255**: Broadcast address

1.2.3 Classless Addressing (CIDR)

Classless Inter-Domain Routing (CIDR) replaced classful addressing to allow more flexible address allocation. CIDR uses a **prefix length** (e.g., /24) to indicate how many bits are used for the network portion.

CIDR Notation

`192.168.10.0/24` means:

- 24 bits for network (192.168.10)
- 8 bits for hosts (0-255)
- Subnet mask: 255.255.255.0
- Usable hosts: $2^8 - 2 = 254$ (excluding network and broadcast addresses)

1.2.4 Subnetting

Subnetting divides a network into smaller sub-networks to improve management and security.

Subnetting Process

1. Determine the number of subnets needed
2. Calculate bits required: $2^n \geq$ number of subnets
3. Borrow bits from the host portion
4. Calculate new subnet mask
5. Identify subnet ranges

Subnetting Example

Given: `192.168.1.0/24`, create 4 subnets

Solution:

- Bits needed: $2^2 = 4$ subnets, so borrow 2 bits
- New mask: /26 (255.255.255.192)
- Hosts per subnet: $2^6 - 2 = 62$
- Subnets:
  - Subnet 1: 192.168.1.0/26 (hosts: .1 to .62)
  - Subnet 2: 192.168.1.64/26 (hosts: .65 to .126)
  - Subnet 3: 192.168.1.128/26 (hosts: .129 to .190)
  - Subnet 4: 192.168.1.192/26 (hosts: .193 to .254)

1.2.5 Variable Length Subnet Masking (VLSM)

VLSM allows subnets of different sizes within the same network, enabling more efficient address utilization.

VLSM Example

Network: `192.168.1.0/24`
Requirements: 100 hosts, 50 hosts, 25 hosts, 10 hosts

Solution:

- 100 hosts: Need $2^7 = 128$ addresses → /25 → 192.168.1.0/25
- 50 hosts: Need $2^6 = 64$ addresses → /26 → 192.168.1.128/26
- 25 hosts: Need $2^5 = 32$ addresses → /27 → 192.168.1.192/27
- 10 hosts: Need $2^4 = 16$ addresses → /28 → 192.168.1.224/28

1.2.6 Network Address Translation (NAT)

NAT allows multiple devices on a private network to share a single public IP address, conserving IPv4 addresses.

Types of NAT

- **Static NAT**: One-to-one mapping (one private IP → one public IP)
- **Dynamic NAT**: Many-to-many mapping (pool of public IPs)
- **PAT (Port Address Translation)**: Many-to-one using different ports

How NAT Works

1. Internal device (192.168.1.10:5000) sends packet to external server
2. NAT router replaces source IP with public IP (203.0.113.5:50000)
3. Server responds to 203.0.113.5:50000
4. NAT router translates back to 192.168.1.10:5000
5. Internal device receives response

1.3 IPv6 Addressing

1.3.1 Why IPv6?

IPv4 provides only $2^{32} \approx 4.3$ billion addresses, which is insufficient for the growing number of internet-connected devices. IPv6 provides 2^{128} addresses — enough for every grain of sand on Earth to have billions of addresses!

1.3.2 IPv6 Address Format

IPv6 addresses are **128 bits** long, written as eight groups of four hexadecimal digits separated by colons.

IPv6 Address Format

Full: `2001:0db8:85a3:0000:0000:8a2e:0370:7334`
Shortened: `2001:db8:85a3::8a2e:370:7334`

Abbreviation Rules:

- Leading zeros in each group can be omitted
- Consecutive groups of zeros can be replaced with `::` (once only)

1.3.3 IPv6 Address Types

Type	**Prefix**	**Description**
Unicast	Various	One-to-one communication
Global Unicast	2000::/3	Internet-routable addresses
Link-Local	fe80::/10	Communication on local link only
Unique Local	fc00::/7	Private addresses (like RFC 1918)
Loopback	::1/128	Localhost (equivalent to 127.0.0.1)
Multicast	ff00::/8	One-to-many communication
Anycast	(no specific)	One-to-nearest communication

1.3.4 IPv6 Header Structure

The IPv6 header is simpler and more efficient than IPv4:

Field	Size	Purpose
Version	4 bits	IP version (6)
Traffic Class	8 bits	Priority and QoS
Flow Label	20 bits	Identify packet flows
Payload Length	16 bits	Size of payload
Next Header	8 bits	Type of next header (like protocol)
Hop Limit	8 bits	TTL equivalent
Source Address	128 bits	Source IPv6 address
Destination Address	128 bits	Destination IPv6 address

IPv6 Improvements

- Fixed 40-byte header (vs variable in IPv4)
- No checksum (faster processing)
- No fragmentation by routers (only by source)
- Built-in security (IPSec)
- Better support for mobile devices

1.3.5 IPv4 to IPv6 Transition Mechanisms

Transition Strategies

- **Dual Stack**: Devices run both IPv4 and IPv6 simultaneously
- **Tunneling**: IPv6 packets encapsulated in IPv4 packets
- **Translation**: Convert between IPv4 and IPv6 (NAT64, DNS64)

1.4 Routing Fundamentals

1.4.1 What is Routing?

Routing is the process of selecting paths in a network along which to send data packets. Routers use **routing tables** to make forwarding decisions.

Routing Table Components

- **Destination Network**: Target network address
- **Next Hop**: IP address of next router
- **Interface**: Outgoing network interface
- **Metric**: Cost/distance to destination

1.4.2 Static vs Dynamic Routing

Aspect	**Static Routing**	**Dynamic Routing**
Configuration	Manual	Automatic
Adaptability	No automatic updates	Adapts to topology changes
Overhead	Low	Higher (routing protocols)
Best For	Small networks	Large networks
Scalability	Poor	Good

1.5 Distance Vector Routing

1.5.1 Bellman-Ford Algorithm

Distance vector protocols use the **Bellman-Ford algorithm**, where each router:

1. Maintains a distance table with costs to all destinations
2. Periodically shares its table with neighbors
3. Updates its table based on neighbors' information

Bellman-Ford Equation

$$D_x(y) = \min_v\{c(x,v) + D_v(y)\}$$

Where:

- $D_x(y)$ = distance from node x to node y
- $c(x,v)$ = cost from x to neighbor v
- $D_v(y)$ = distance from v to y

1.5.2 Routing Information Protocol (RIP)

RIP is a distance vector protocol that uses hop count as its metric.

RIP Characteristics

- Maximum hop count: 15 (16 = unreachable)
- Updates every 30 seconds
- Uses UDP port 520
- Two versions: RIPv1 (classful) and RIPv2 (classless)

1.5.3 Count-to-Infinity Problem

A major issue with distance vector routing where incorrect routing information circulates indefinitely.

Count-to-Infinity Example

1. Network: A—B—C (each link cost = 1)
2. Link B-C fails
3. B hears from A that C is reachable (distance 2)
4. B updates: C reachable via A (distance 3)
5. A hears from B: C reachable (distance 3)
6. A updates: C reachable via B (distance 4)
7. This continues until reaching infinity...

Solutions to Count-to-Infinity

- **Split Horizon**: Don't advertise routes back to the source
- **Route Poisoning**: Set failed routes to infinity (16 in RIP)
- **Hold-Down Timers**: Wait before accepting new routes after failure
- **Triggered Updates**: Send updates immediately on topology change

1.6 Link State Routing

1.6.1 Dijkstra's Algorithm

Link state protocols use **Dijkstra's algorithm** to compute the shortest path tree.

Link State Process

1. Each router discovers its neighbors
2. Measures cost to each neighbor
3. Constructs Link State Advertisement (LSA)
4. Floods LSA to all routers
5. Each router builds complete network topology
6. Runs Dijkstra's algorithm to compute shortest paths

Dijkstra's Algorithm Steps

1. Initialize: Set distance to source = 0, all others = ∞
2. Select unvisited node with smallest distance
3. For each neighbor, calculate: distance = current distance + edge cost
4. If new distance is smaller, update
5. Mark current node as visited
6. Repeat until all nodes visited

Dijkstra's Algorithm Example

Given network: A connected to B(2), C(4); B to C(1), D(7); C to D(3)

Finding shortest paths from A:

- Initial: A=0, B=∞, C=∞, D=∞
- Select A: Update B=2, C=4
- Select B (distance 2): Update C=3 (2+1¡4), D=9 (2+7)
- Select C (distance 3): Update D=6 (3+3¡9)
- Select D (distance 6): Done
- **Result:** A→B=2, A→C=3, A→D=6

1.6.2 Open Shortest Path First (OSPF)

OSPF is a widely-used link state protocol.

OSPF Features

- Uses cost metric (typically based on bandwidth)
- Supports hierarchical design (Areas)
- Fast convergence
- Supports VLSM and CIDR
- Authentication support
- Uses IP protocol 89

OSPF Areas

- **Area 0 (Backbone)**: All areas must connect to Area 0
- **Regular Areas**: Standard OSPF areas
- **Stub Areas**: Don't receive external routes
- **Totally Stubby**: Only default route from ABR

1.6.3 LSA Flooding

Link State Advertisements (LSAs) are flooded throughout the network to ensure all routers have identical topology databases.

LSA Flooding Process

1. Router creates/updates LSA
2. Sends LSA to all neighbors
3. Each neighbor:
   - Checks if LSA is newer (sequence number)
   - If newer, updates database and forwards to all other neighbors
   - If older, discards
4. Process continues until all routers have the LSA

1.7 Network Layer Protocols

1.7.1 Internet Protocol (IP)

IP is the primary network layer protocol responsible for packet delivery.

IPv4 Header (20-60 bytes)

Field	**Size**	**Purpose**
Version	4 bits	IP version (4)
IHL	4 bits	Header length (in 32-bit words)
Type of Service	8 bits	Priority/QoS
Total Length	16 bits	Packet size (header + data)
Identification	16 bits	Identifies fragments
Flags	3 bits	Control fragmentation
Fragment Offset	13 bits	Position in original packet
TTL	8 bits	Time to Live (hop limit)
Protocol	8 bits	Upper layer protocol (TCP=6, UDP=17)
Header Checksum	16 bits	Error detection
Source Address	32 bits	Source IP
Destination Address	32 bits	Destination IP
Options	Variable	Rarely used

1.7.2 Internet Control Message Protocol (ICMP)

ICMP is used for error reporting and diagnostic purposes.

Common ICMP Messages

- **Type 0**: Echo Reply (ping response)
- **Type 3**: Destination Unreachable
- **Type 5**: Redirect
- **Type 8**: Echo Request (ping)
- **Type 11**: Time Exceeded (traceroute)

ICMP in Action: Ping

When you ping a host:

1. Your computer sends ICMP Echo Request (Type 8)
2. Target receives request
3. Target sends ICMP Echo Reply (Type 0)
4. Round-trip time is calculated

1.7.3 Address Resolution Protocol (ARP)

ARP maps IP addresses to MAC addresses on a local network.

ARP Process

1. Host A wants to send to IP address 192.168.1.10
2. A checks ARP cache for MAC address
3. If not found, A broadcasts ARP Request: ”Who has 192.168.1.10?”
4. Host with 192.168.1.10 responds with ARP Reply containing its MAC
5. A caches the mapping and sends the packet

1.7.4 Reverse ARP (RARP)

RARP maps MAC addresses to IP addresses (largely obsolete, replaced by DHCP).

1.7.5 Fragmentation and Reassembly

When packets exceed the **Maximum Transmission Unit (MTU)** of a link, they must be fragmented.

Fragmentation Process

- **MTU**: Maximum packet size for a network (typically 1500 bytes for Ethernet)
- **Fragmentation**: Router breaks large packets into smaller fragments
- **Identification Field**: All fragments have same ID
- **Fragment Offset**: Indicates position in original packet
- **More Fragments Flag**: 1 = more fragments coming, 0 = last fragment
- **Reassembly**: Destination host reassembles fragments

Fragmentation Example

Original packet: 4000 bytes, MTU = 1500 bytes

- Fragment 1: 1500 bytes (offset=0, MF=1)
- Fragment 2: 1500 bytes (offset=1480, MF=1)
- Fragment 3: 1020 bytes (offset=2960, MF=0)

Note: Offset measured in 8-byte units

1.8 Unit III Summary

Key Concepts Checklist

- IPv4: 32-bit addresses, classful/classless, subnetting, VLSM, NAT
- IPv6: 128-bit addresses, simplified header, transition mechanisms
- Routing: Static vs dynamic, routing tables, packet forwarding
- Distance Vector: Bellman-Ford, RIP, count-to-infinity solutions
- Link State: Dijkstra's algorithm, OSPF, LSA flooding
- Protocols: IP header, ICMP messages, ARP process, fragmentation

2 Unit IV: Transport Layer (7 Hours)

2.1 Introduction to the Transport Layer

2.1.1 Prerequisites Review

Before studying the Transport Layer, ensure you understand:

Foundation Concepts

- **IP Addressing**: How devices are identified on networks
- **Packets**: Units of data transmitted across networks
- **End-to-End Communication**: Communication between source and destination hosts
- **Reliability**: Ensuring data arrives correctly and completely
- **Binary Operations**: AND, OR, XOR for checksum calculations

2.1.2 Role of the Transport Layer

The Transport Layer (Layer 4) provides **logical communication between application processes** running on different hosts. Think of it as the layer that ensures your email arrives intact, your video streams smoothly, and your web pages load completely.

Transport Layer Responsibilities

1. **Process-to-Process Delivery**: Unlike the network layer (host-to-host), transport layer delivers to specific applications
2. **Segmentation**: Breaks application data into segments
3. **Multiplexing/Demultiplexing**: Manages multiple connections
4. **Error Detection**: Checksums to detect corrupted data
5. **Flow Control**: Prevents overwhelming the receiver
6. **Congestion Control**: Prevents overwhelming the network

2.2 Transport Layer Services

2.2.1 Process-to-Process Communication

While the network layer delivers packets between hosts, the transport layer delivers data between **processes** (applications) running on those hosts.

Real-World Analogy

Think of houses (hosts) on a street:

- **Network Layer**: Delivers mail to the correct house (IP address)
- **Transport Layer**: Delivers to the correct person in the house (port number)

2.2.2 Port Addressing

Port numbers identify specific processes/applications on a host.

Range	Category	Description
0-1023	Well-Known Ports	Standard services (HTTP=80, HTTPS=443, SSH=22)
1024-49151	Registered Ports	Application-specific (MySQL=3306, PostgreSQL=5432)
49152-65535	Dynamic/Private	Temporary client ports

Common Port Numbers

- HTTP: 80, HTTPS: 443
- FTP: 20 (data), 21 (control)
- SSH: 22, Telnet: 23
- SMTP: 25, DNS: 53
- POP3: 110, IMAP: 143

2.2.3 Sockets

A **socket** is the combination of an IP address and a port number, uniquely identifying a process on a network.

Socket Example

```
192.168.1.10:8080
```

- IP Address: 192.168.1.10
- Port Number: 8080
- Identifies: Web server process on that host

2.2.4 Multiplexing and Demultiplexing

Definitions

- **Multiplexing** (at sender): Gathering data from multiple sockets, encapsulating with headers, passing to network layer
- **Demultiplexing** (at receiver): Delivering received segments to correct sockets based on port numbers

Multiplexing in Action

Your computer runs:

- Web browser (port 50001) → Server (port 443)
- Email client (port 50002) → Server (port 993)
- Chat app (port 50003) → Server (port 5222)

Transport layer multiplexes all three, network layer sees single stream of packets.

2.2.5 Connection-Oriented vs Connectionless

Aspect	Connection-Oriented (TCP)	Connectionless (UDP)
Setup	Requires handshake	No setup
Reliability	Guaranteed delivery	Best effort
Order	Maintains order	No ordering
Speed	Slower (overhead)	Faster
Use Cases	Web, email, file transfer	Streaming, gaming, DNS

2.3 User Datagram Protocol (UDP)

2.3.1 UDP Characteristics

UDP is a **simple, lightweight, connectionless** protocol.

UDP Features

- **No connection establishment**: Immediate data transmission
- **No reliability**: No acknowledgments or retransmissions
- **No flow control**: Sender can overwhelm receiver
- **No congestion control**: Doesn't reduce rate during congestion
- **Low overhead**: Only 8-byte header
- **Fast**: Minimal processing

2.3.2 UDP Header Structure

Field	Size	Description
Source Port	16 bits	Port number of sender (optional)
Destination Port	16 bits	Port number of receiver
Length	16 bits	Length of UDP header + data (minimum 8)
Checksum	16 bits	Error detection (optional in IPv4, mandatory in IPv6)

2.3.3 UDP Checksum Calculation

The checksum detects errors in transmitted data.

Checksum Process

1. Create pseudo-header (source IP, dest IP, protocol, UDP length)
2. Divide data into 16-bit words
3. Add all words together (binary addition)
4. If overflow, wrap around and add to result
5. Take one's complement (flip all bits)
6. Result is checksum

Simple Checksum Example

Data words: 0x1234, 0x5678, 0x9ABC

Calculation:

$$\begin{aligned}&\text{Sum: } 0x1234 + 0x5678 + 0x9ABC = 0x20768\\&\text{Wrap: } 0x0768 + 0x0002 = 0x076A\\&\text{Checksum: } \sim 0x076A = 0xF895\end{aligned}$$

2.3.4 UDP Applications

When to Use UDP

- **Streaming Media**: Some packet loss acceptable, real-time important
- **Online Gaming**: Low latency critical, occasional loss tolerable
- **DNS Queries**: Small, single-packet requests
- **DHCP**: Simple request/response on local network
- **VoIP**: Real-time communication, retransmission not useful
- **TFTP**: Simple file transfer (implements own reliability)

2.4 Transmission Control Protocol (TCP)

2.4.1 TCP Characteristics

TCP provides **reliable, ordered, connection-oriented** communication.

TCP Features

- **Connection-oriented**: Three-way handshake before data transfer
- **Reliable**: Acknowledges received segments, retransmits lost ones
- **Ordered**: Delivers data in correct sequence
- **Flow control**: Sliding window mechanism
- **Congestion control**: Adjusts sending rate based on network conditions
- **Full-duplex**: Simultaneous two-way communication

2.4.2 TCP Header Structure

Field	**Size**	**Description**
Source Port	16 bits	Sender's port number
Destination Port	16 bits	Receiver's port number
Sequence Number	32 bits	Byte number of first byte in segment
Acknowledgment Number	32 bits	Next byte expected from other side
Header Length	4 bits	Length of TCP header (in 32-bit words)
Reserved	6 bits	Reserved for future use
Flags	6 bits	URG, ACK, PSH, RST, SYN, FIN
Window Size	16 bits	Flow control: bytes receiver can accept
Checksum	16 bits	Error detection (mandatory)
Urgent Pointer	16 bits	Points to urgent data (if URG=1)
Options	Variable	MSS, window scaling, timestamps, etc.

TCP Flags

- **SYN**: Synchronize, initiate connection
- **ACK**: Acknowledgment field is valid
- **FIN**: Finish, close connection
- **RST**: Reset connection (error)
- **PSH**: Push data to application immediately
- **URG**: Urgent pointer field is valid

2.4.3 Three-Way Handshake (Connection Establishment)

TCP uses a three-way handshake to establish a connection.

Three-Way Handshake Steps

1. **Client → Server: SYN**
   - Client sends SYN with initial sequence number (ISN)
   - Flags: SYN=1
   - Seq = X (random number)
2. **Server → Client: SYN-ACK**
   - Server responds with SYN and ACK
   - Flags: SYN=1, ACK=1
   - Seq = Y, Ack = X+1
3. **Client → Server: ACK**
   - Client acknowledges
   - Flags: ACK=1
   - Seq = X+1, Ack = Y+1

Connection established, data transfer can begin!

Handshake Example

- Client: SYN, Seq=1000
- Server: SYN-ACK, Seq=5000, Ack=1001
- Client: ACK, Seq=1001, Ack=5001
- Data transfer begins...

2.4.4 Four-Way Termination (Connection Closure)

TCP uses a four-way process to gracefully close connections.

Four-Way Termination Steps

1. **Client → Server: FIN**
   - Client initiates closure
   - Flags: FIN=1, ACK=1
2. **Server → Client: ACK**
   - Server acknowledges FIN
   - Flags: ACK=1
3. **Server → Client: FIN**
   - Server ready to close
   - Flags: FIN=1, ACK=1
4. **Client → Server: ACK**
   - Client acknowledges
   - Flags: ACK=1

Connection closed!

2.4.5 Sequence and Acknowledgment Numbers

TCP uses sequence and acknowledgment numbers to ensure reliable, ordered delivery.

How They Work

- **Sequence Number**: Identifies the first byte of data in the segment
- **Acknowledgment Number**: Specifies the next byte expected from the other side
- **Cumulative ACK**: Acknowledges all bytes up to (but not including) the ACK number

Sequence Number Example

Client sends file to server:

- Segment 1: Seq=1000, Length=500 (bytes 1000-1499)
- Server ACK: Ack=1500 (expecting byte 1500 next)
- Segment 2: Seq=1500, Length=300 (bytes 1500-1799)
- Server ACK: Ack=1800

2.5 Flow Control

2.5.1 Purpose of Flow Control

Flow control prevents a fast sender from overwhelming a slow receiver.

Flow Control Mechanism

- Receiver advertises **receive window (rwnd)** in TCP header
- Window size = buffer space available at receiver
- Sender limits unacknowledged data to rwnd
- Window grows as receiver processes data
- Window shrinks as receiver's buffer fills

2.5.2 Sliding Window Protocol

TCP uses a **sliding window** protocol for efficient flow control.

Sliding Window Concepts

- **Send Window**: Range of sequence numbers sender can transmit
- **Receive Window**: Range receiver can accept
- **Window Slides**: Advances as data is acknowledged
- **Window Size**: Advertised by receiver, adjusts dynamically

Sliding Window Example

Initial: Sender window = [1000-1999], Receiver window size = 1000 bytes

1. Sender transmits bytes 1000-1499 (500 bytes)
2. Receiver ACKs 1500, window = 700 (processed 300 bytes)
3. Sender window = [1500-2199]
4. Sender can send 700 more bytes before waiting

2.5.3 Stop-and-Wait vs Pipelining

Aspect	**Stop-and-Wait**	**Pipelining (Sliding Window)**
Operation	Send one, wait for ACK	Send multiple without waiting
Efficiency	Low (idle time)	High (utilizes bandwidth)
Window Size	1 segment	Multiple segments
Complexity	Simple	More complex
Use	Simple protocols	TCP, modern protocols

2.6 Congestion Control

2.6.1 What is Network Congestion?

Congestion occurs when network demand exceeds capacity, causing:

- Increased queuing delays
- Packet loss (buffer overflow)
- Reduced throughput

- Retransmissions (worsening congestion)

Congestion Analogy

Think of network congestion like traffic on a highway:

- Normal traffic: Cars flow smoothly
- Rush hour: Too many cars, traffic slows
- Accidents: Complete standstill
- TCP's job: Detect congestion and slow down

2.6.2 TCP Congestion Control Algorithms

TCP uses several algorithms to manage congestion:

Congestion Window (cwnd)

- Sender maintains congestion window (cwnd)
- Effective window $= \min(\text{cwnd}, \text{rwnd})$
- cwnd adjusted based on network conditions
- Goal: Maximize throughput without causing congestion

2.6.3 Slow Start

Initially, TCP doesn't know the network capacity, so it starts slowly.

Slow Start Process

1. Initialize cwnd = 1 MSS (Maximum Segment Size)
2. For each ACK received, cwnd += 1 MSS
3. Result: cwnd doubles every RTT (exponential growth)
4. Continue until:
   - Reach slow start threshold (ssthresh)
   - Packet loss detected

Slow Start Example

MSS = 1000 bytes, ssthresh = 16000

- RTT 1: cwnd = 1000 (send 1 segment)
- RTT 2: cwnd = 2000 (send 2 segments)
- RTT 3: cwnd = 4000 (send 4 segments)
- RTT 4: cwnd = 8000 (send 8 segments)
- RTT 5: cwnd = 16000 (reach ssthresh, switch to congestion avoidance)

2.6.4 Congestion Avoidance

After slow start, TCP enters **congestion avoidance** mode.

Congestion Avoidance Process

1. For each RTT, cwnd += 1 MSS
2. Result: Linear growth (additive increase)
3. Continue until packet loss detected
4. On loss:
    - ssthresh = cwnd / 2
    - cwnd = 1 MSS (or ssthresh, depending on loss type)
    - Return to slow start or fast recovery

2.6.5 Fast Retransmit

Fast retransmit quickly detects packet loss without waiting for timeout.

Fast Retransmit Mechanism

1. Receiver sends duplicate ACK when out-of-order segment arrives
2. Sender receives 3 duplicate ACKs (4 identical ACKs total)
3. Interprets as packet loss
4. Immediately retransmits missing segment
5. Faster than waiting for timeout

Fast Retransmit Example

- Sender: Segments 1, 2, 3, 4, 5 (segment 2 lost)
- Receiver: Gets 1, ACKs for 2
- Receiver: Gets 3, sends duplicate ACK for 2
- Receiver: Gets 4, sends duplicate ACK for 2
- Receiver: Gets 5, sends duplicate ACK for 2
- Sender: Receives 3 duplicate ACKs, retransmits segment 2

2.6.6 Fast Recovery

Fast recovery avoids slow start after fast retransmit.

Fast Recovery Process

1. On 3 duplicate ACKs:
   - ssthresh = cwnd / 2
   - cwnd = ssthresh + 3 MSS
   - Retransmit missing segment
2. For each additional duplicate ACK:
   - cwnd += 1 MSS
3. When new ACK arrives:
   - cwnd = ssthresh
   - Enter congestion avoidance

2.6.7 AIMD (Additive Increase Multiplicative Decrease)

TCP's congestion control follows the **AIMD** principle:

AIMD Strategy

- **Additive Increase**: Increase cwnd linearly (add 1 MSS per RTT)
- **Multiplicative Decrease**: Decrease cwnd exponentially (divide by 2 on loss)
- **Result**: "Sawtooth" pattern converging to fair share
- **Fairness**: Multiple flows converge to equal bandwidth

2.7 Reliability Mechanisms

2.7.1 Error Detection

TCP uses checksums to detect errors in transmitted data.

TCP Checksum

- Computed over pseudo-header, TCP header, and data
- Pseudo-header includes source/dest IP, protocol, TCP length
- Same algorithm as UDP, but mandatory
- If checksum fails, segment is discarded

2.7.2 Retransmission Strategies

TCP retransmits segments when:

Retransmission Triggers

- **Timeout**: RTO (Retransmission Timeout) expires
- **Fast Retransmit**: 3 duplicate ACKs received

2.7.3 Timeout Mechanisms

TCP must set appropriate timeout values — too short causes unnecessary retransmissions, too long causes delays.

RTO Calculation

TCP estimates Round-Trip Time (RTT) and sets RTO accordingly:

$$\begin{aligned}
\text{EstimatedRTT} &= (1-\alpha) \times \text{EstimatedRTT} + \alpha \times \text{SampleRTT} \\
\text{DevRTT} &= (1-\beta) \times \text{DevRTT} + \beta \times |\text{SampleRTT} - \text{EstimatedRTT}| \\
\text{RTO} &= \text{EstimatedRTT} + 4 \times \text{DevRTT}
\end{aligned}$$

Typical values: $\alpha = 0.125$, $\beta = 0.25$

RTO Example

- Initial EstimatedRTT = 100ms, DevRTT = 10ms
- SampleRTT = 120ms
- New EstimatedRTT = 0.875 × 100 + 0.125 × 120 = 102.5ms
- New DevRTT = 0.75 × 10 + 0.25 × —120-102.5— = 11.875ms
- RTO = 102.5 + 4 × 11.875 = 150ms

2.8 Unit IV Summary

Key Concepts Checklist

- Transport layer provides process-to-process communication
- Port numbers identify applications, sockets = IP + port
- Multiplexing/demultiplexing manages multiple connections
- UDP: Simple, fast, unreliable, 8-byte header
- TCP: Reliable, ordered, connection-oriented, complex header
- Three-way handshake establishes connections
- Four-way termination closes connections
- Flow control: Sliding window prevents receiver overflow
- Congestion control: Slow start, congestion avoidance, fast retransmit/recovery
- AIMD: Additive increase, multiplicative decrease
- Reliability: Checksums, retransmissions, adaptive timeouts

3 Unit V: Application Layer (7 Hours)

3.1 Introduction to the Application Layer

3.1.1 Prerequisites Review

Before exploring Application Layer protocols, review:

Foundation Concepts

- **Client-Server Model**: Client requests, server responds
- **TCP/UDP**: Transport protocols providing different services
- **Port Numbers**: How applications are addressed
- **DNS Resolution**: Converting domain names to IP addresses
- **Text Encoding**: ASCII, Unicode for representing text

3.1.2 Role of the Application Layer

The Application Layer (Layer 7) is where **network applications and their protocols** reside. It provides services directly to users and applications.

Application Layer Characteristics

- Closest to end users
- Implements specific application protocols
- Uses transport layer services (TCP/UDP)
- Examples: Web browsers, email clients, file transfer

3.2 Domain Name System (DNS)

3.2.1 What is DNS?

DNS is the Internet's phone book — it translates human-readable domain names (like `www.example.com`) into IP addresses (like `93.184.216.34`).

Why DNS Matters

Without DNS, you'd need to remember:

- Google: 142.250.185.46
- Facebook: 157.240.241.35
- Amazon: 205.251.242.103

With DNS, you just remember: google.com, facebook.com, amazon.com

3.2.2 DNS Hierarchy

DNS uses a hierarchical structure:

DNS Hierarchy Levels

1. **Root Level (.)**
   - 13 root server clusters worldwide
   - Top of DNS hierarchy
2. **Top-Level Domain (TLD)**
   - Generic: .com, .org, .net, .edu
   - Country-code: .uk, .de, .jp, .us
3. **Second-Level Domain**
   - Organization's domain: example.com, google.com
4. **Subdomain**
   - Further divisions: www.example.com, mail.example.com

3.2.3 DNS Architecture

DNS Components

- **DNS Resolver (Recursive Resolver)**: Client-side, queries on behalf of applications
- **Root Name Servers**: Direct queries to TLD servers
- **TLD Name Servers**: Direct queries to authoritative servers
- **Authoritative Name Servers**: Provide definitive answers for domains
- **Caching**: Stores results temporarily to reduce queries

3.2.4 DNS Query Types

Type	Description
Recursive	Resolver fully resolves the query, returning final answer to client
Iterative	Server returns best answer it knows or referral to another server

3.2.5 DNS Record Types

Record	Purpose
A	Maps hostname to IPv4 address
AAAA	Maps hostname to IPv6 address
CNAME	Canonical name (alias) for another hostname
MX	Mail exchange server for domain (includes priority)
NS	Name server responsible for domain
PTR	Pointer record for reverse DNS lookup (IP to hostname)
SOA	Start of Authority, domain metadata
TXT	Arbitrary text, often for verification/SPF

DNS Record Examples

```
example.com.        A        93.184.216.34
www.example.com.    CNAME    example.com.
example.com.        MX  10   mail.example.com.
example.com.        NS       ns1.example.com.
example.com.        AAAA     2606:2800:220:1:248:1893:25c8:1946
```

3.2.6 DNS Resolution Process

Complete Resolution Example: www.example.com

1. User enters `www.example.com` in browser
2. Browser checks local cache → not found
3. Query sent to DNS resolver (typically ISP's)
4. Resolver checks cache → not found
5. **Resolver → Root Server:** ”Where is .com?”
6. **Root → Resolver:** ”Ask TLD server at 192.5.6.30”
7. **Resolver → TLD Server:** ”Where is example.com?”
8. **TLD → Resolver:** ”Ask authoritative server at 93.184.216.119”
9. **Resolver → Authoritative:** ”What's IP for www.example.com?”
10. **Authoritative → Resolver:** ”93.184.216.34”
11. Resolver caches result, returns to browser
12. Browser caches result, connects to 93.184.216.34

3.3 Hypertext Transfer Protocol (HTTP/HTTPS)

3.3.1 What is HTTP?

HTTP (Hypertext Transfer Protocol) is the foundation of data communication on the World Wide Web.

HTTP Characteristics

- **Application layer protocol**
- **Client-server model**: Browser (client) requests, web server responds
- **Stateless**: Each request independent (no memory of previous)
- **Uses TCP**: Port 80 (HTTP), port 443 (HTTPS)
- **Text-based**: Human-readable format

3.3.2 HTTP Methods

Method	**Purpose**
GET	Retrieve resource, no side effects (idempotent)
POST	Submit data, may create resource or cause side effects
PUT	Create or replace resource at specific URI (idempotent)
DELETE	Remove specified resource (idempotent)
HEAD	Like GET but only retrieve headers, not body
PATCH	Partially modify resource
OPTIONS	Describe communication options for target

HTTP Method Usage

- **GET /index.html**: Retrieve homepage
- **POST /login**: Submit login credentials
- **PUT /users/123**: Update user 123's information
- **DELETE /posts/456**: Delete post 456

3.3.3 HTTP Request Structure

HTTP Request Format

```
METHOD URI HTTP/VERSION
Header-Name: Header-Value
Header-Name: Header-Value
[blank line]
[optional message body]
```

Sample HTTP Request

```
GET /index.html HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
Accept: text/html,application/xhtml+xml
Accept-Language: en-US,en;q=0.9
Connection: keep-alive
```

3.3.4 HTTP Response Structure

HTTP Response Format

```
HTTP/VERSION STATUS-CODE REASON-PHRASE
Header-Name: Header-Value
Header-Name: Header-Value
[blank line]
[message body]
```

Sample HTTP Response

```
HTTP/1.1 200 OK
Date: Mon, 01 Jan 2024 12:00:00 GMT
Server: Apache/2.4.41
Content-Type: text/html; charset=UTF-8
Content-Length: 1234
Connection: keep-alive

<!DOCTYPE html>
<html>
<head><title>Example</title></head>
<body><h1>Hello World!</h1></body>
</html>
```

3.3.5 HTTP Status Codes

Code	Category	Common Examples
1xx	Informational	100 Continue
2xx	Success	200 OK, 201 Created, 204 No Content
3xx	Redirection	301 Moved Permanently, 302 Found, 304 Not Modified
4xx	Client Error	400 Bad Request, 401 Unauthorized, 403 Forbidden, 404 Not Found
5xx	Server Error	500 Internal Server Error, 502 Bad Gateway, 503 Service Unavailable

Important Status Codes

- **200 OK**: Request succeeded
- **301 Moved Permanently**: Resource permanently moved, update bookmarks
- **304 Not Modified**: Cached version is still valid
- **400 Bad Request**: Malformed request syntax
- **401 Unauthorized**: Authentication required
- **403 Forbidden**: Server refuses to fulfill request
- **404 Not Found**: Resource doesn't exist
- **500 Internal Server Error**: Server encountered unexpected condition
- **503 Service Unavailable**: Server temporarily unable to handle request

3.3.6 HTTP Versions

HTTP/1.0

- One request per TCP connection
- Simple but inefficient (connection overhead)
- Released 1996

HTTP/1.1

- Persistent connections (keep-alive)
- Pipelining (multiple requests without waiting)
- Chunked transfer encoding
- Additional caching mechanisms
- Host header (required)
- Released 1997, still widely used

HTTP/2.0

- Binary protocol (not text)
- Multiplexing (multiple requests/responses simultaneously)
- Server push (proactively send resources)
- Header compression (HPACK)
- Stream prioritization
- Released 2015, increasingly adopted

3.3.7 HTTPS and TLS/SSL

HTTPS = HTTP + TLS/SSL encryption

HTTPS Benefits

- **Confidentiality**: Data encrypted, prevents eavesdropping
- **Integrity**: Detects tampering
- **Authentication**: Verifies server identity (certificates)
- **Port**: 443 (vs 80 for HTTP)

TLS/SSL Handshake (Simplified)

1. Client sends "ClientHello" (supported cipher suites, TLS version)
2. Server sends "ServerHello" (selected cipher, certificate)
3. Client verifies certificate (signed by trusted CA)
4. Key exchange (using public key cryptography)
5. Both derive session keys
6. Secure communication begins

3.4 Email Protocols

3.4.1 Email System Architecture

Email involves multiple components and protocols:

Email Components

- **User Agent**: Email client (Outlook, Gmail, Thunderbird)
- **Mail Server**: Stores and forwards messages
- **Mailbox**: User's received messages
- **Message Queue**: Outgoing messages waiting to be sent

3.4.2 Simple Mail Transfer Protocol (SMTP)

SMTP is used to **send** email from client to server and between servers.

SMTP Characteristics

- Uses TCP, port 25 (or 587 for submission)
- Text-based protocol
- Push protocol (sender initiates)
- Three phases: handshake, message transfer, closure
- Supports only 7-bit ASCII (MIME extends this)

SMTP Session Example

```
S: 220 mail.example.com SMTP Service Ready
C: HELO client.example.com
S: 250 mail.example.com
C: MAIL FROM:<alice@example.com>
S: 250 OK
C: RCPT TO:<bob@example.org>
S: 250 OK
C: DATA
S: 354 Start mail input; end with <CRLF>.<CRLF>
C: Subject: Test Message
C:
C: This is a test email.
C: .
S: 250 OK Message accepted
C: QUIT
S: 221 Closing connection
```

3.4.3 Post Office Protocol 3 (POP3)

POP3 is used to **retrieve** email from a server.

POP3 Characteristics

- Uses TCP, port 110 (or 995 for POP3S)
- Pull protocol (client requests)
- Three phases: authorization, transaction, update
- Downloads messages to client
- Typically deletes from server (can be configured to keep)
- Stateful protocol

POP3 Phases

1. **Authorization**: Login with username/password
2. **Transaction**: Retrieve, mark for deletion
3. **Update**: Actually delete marked messages (on QUIT)

POP3 Session Example

```
S: +OK POP3 server ready
C: USER alice
S: +OK
C: PASS secret123
S: +OK Logged in
C: LIST
S: +OK 2 messages (320 octets)
S: 1 120
S: 2 200
S: .
C: RETR 1
S: +OK 120 octets
S: [message content]
S: .
C: DELE 1
S: +OK Message 1 deleted
C: QUIT
S: +OK Goodbye
```

3.4.4 Internet Message Access Protocol (IMAP)

IMAP provides more advanced email retrieval than POP3.

IMAP Advantages over POP3

- Messages kept on server (access from multiple devices)
- Server-side folder management
- Selective download (headers only, search before download)
- Multiple mailbox support
- Flags (read/unread, starred, etc.)
- Port 143 (or 993 for IMAPS)

IMAP Features

- Create, delete, rename mailboxes
- Search messages on server
- Partial fetch (e.g., just headers)
- Message flags and status
- Server-side filtering

3.4.5 Multipurpose Internet Mail Extensions (MIME)

MIME extends email to support:

MIME Capabilities

- Non-ASCII text (UTF-8, other character sets)
- Binary attachments (images, documents, audio, video)
- Multiple parts (text + HTML + attachments)
- Rich formatting

MIME Headers

- **MIME-Version**: Typically 1.0
- **Content-Type**: text/plain, text/html, image/jpeg, multipart/mixed
- **Content-Transfer-Encoding**: base64, quoted-printable, 7bit, 8bit
- **Content-Disposition**: inline or attachment

3.5 File Transfer Protocol (FTP)

3.5.1 FTP Overview

FTP is a protocol for transferring files between client and server.

FTP Characteristics

- Uses TCP
- Two connections: control (port 21) and data (port 20 or dynamic)
- Stateful (maintains session information)
- Supports authentication
- Text-based commands

3.5.2 FTP Architecture

FTP uses **two separate connections**:

FTP Connections

- **Control Connection (Port 21)**:
  - Persistent throughout session
  - Carries commands and responses
  - Example commands: USER, PASS, LIST, RETR, STOR
- **Data Connection**:
  - Opened when transferring files or directory listings
  - Closed after transfer completes
  - Separate from control connection

3.5.3 Active vs Passive Mode

Aspect	Active Mode	Passive Mode
Initiation	Server initiates data connection to client	Client initiates data connection to server
Firewall Issues	Client firewall may block incoming	Server specifies port, client connects
Data Port	Server uses port 20	Server uses random high port
Use Case	Traditional, less common today	Modern standard, firewall-friendly

Active Mode

1. Client connects to server port 21 (control)
2. Client sends PORT command with its IP:port
3. Server initiates connection to client's specified port (data)
4. Data transfer occurs

Passive Mode

1. Client connects to server port 21 (control)
2. Client sends PASV command
3. Server responds with IP:port to connect to
4. Client initiates connection to server's specified port (data)
5. Data transfer occurs

3.5.4 FTP Commands and Responses

Command	Description
USER	Specify username
PASS	Specify password
LIST	List directory contents
RETR	Retrieve (download) file
STOR	Store (upload) file
CWD	Change working directory
PWD	Print working directory
MKD	Make directory
DELE	Delete file
QUIT	Close connection

FTP Response Codes

- **1xx**: Positive preliminary reply
- **2xx**: Positive completion reply (e.g., 226 Transfer complete)
- **3xx**: Positive intermediate reply (e.g., 331 Password required)
- **4xx**: Transient negative reply
- **5xx**: Permanent negative reply (e.g., 530 Login incorrect)

3.6 Other Application Layer Protocols

3.6.1 Dynamic Host Configuration Protocol (DHCP)

DHCP automatically assigns IP addresses to devices on a network.

DHCP Process (DORA)

1. **Discover**: Client broadcasts ”I need an IP address”
2. **Offer**: DHCP server responds with available IP
3. **Request**: Client requests offered IP
4. **Acknowledge**: Server confirms, provides IP and configuration

DHCP Provides

- IP address
- Subnet mask
- Default gateway
- DNS server addresses
- Lease time (how long IP is valid)

3.6.2 Telnet

Telnet provides remote command-line access to devices.

Telnet Characteristics

- Uses TCP, port 23
- Unencrypted (major security risk)
- Text-based, interactive
- Largely replaced by SSH
- Still used for testing other protocols

3.6.3 Secure Shell (SSH)

SSH provides secure remote access, replacing Telnet.

SSH Advantages

- Encrypted communication
- Strong authentication (password, keys)
- Port forwarding (tunneling)
- Secure file transfer (SCP, SFTP)
- Port 22

3.6.4 Simple Network Management Protocol (SNMP)

SNMP is used for managing and monitoring network devices.

SNMP Components

- **Manager**: Monitoring system that collects data
- **Agent**: Software on managed device
- **MIB (Management Information Base)**: Database of manageable objects
- **Operations**: GET, SET, TRAP (notification)

SNMP Versions

- **SNMPv1**: Original, community-based authentication
- **SNMPv2c**: Improved efficiency, still uses communities
- **SNMPv3**: Adds encryption and authentication (most secure)

3.7 Unit V Summary

Key Concepts Checklist

- DNS: Hierarchical system, record types (A, AAAA, MX, CNAME, NS, PTR), resolution process
- HTTP: Methods (GET, POST, PUT, DELETE), request/response structure, status codes
- HTTP versions: 1.0 (one per connection), 1.1 (persistent), 2.0 (multiplexing)
- HTTPS: HTTP + TLS/SSL, provides confidentiality, integrity, authentication
- SMTP: Send email, port 25/587, push protocol
- POP3: Retrieve email, port 110, downloads and typically deletes
- IMAP: Advanced retrieval, port 143, keeps on server, folder management
- MIME: Extends email for non-ASCII, attachments, multipart
- FTP: File transfer, two connections (control + data), active/passive modes
- Other: DHCP (IP assignment), Telnet (insecure remote access), SSH (secure remote access), SNMP (network management)

4 Unit VI: Network Security (7 Hours)

4.1 Introduction to Network Security

4.1.1 Prerequisites Review

Before studying network security, review:

Foundation Concepts

- **Binary and Hexadecimal**: Number systems for representing data
- **XOR Operation**: Used extensively in encryption
- **Prime Numbers**: Foundation of asymmetric cryptography
- **Network Protocols**: TCP/IP, how data flows through networks
- **Authentication**: Verifying identity

4.1.2 Why Network Security Matters

In our interconnected world, network security protects:

- Personal data (passwords, financial information)
- Business assets (trade secrets, customer data)
- Critical infrastructure (power grids, hospitals)
- National security (government communications)

Security Breach Example

Without proper security:

- Your password could be stolen from a coffee shop WiFi
- Hackers could modify your bank transfer amount
- Attackers could impersonate your bank's website
- Your email could be read by unauthorized parties

4.2 Security Services

4.2.1 Confidentiality

Confidentiality ensures that information is accessible only to authorized parties.

Confidentiality Mechanisms

- Encryption (symmetric and asymmetric)
- Access control
- Physical security
- Authentication

Confidentiality in Action

When you use HTTPS:

- Your credit card number is encrypted
- Eavesdroppers see only scrambled data
- Only your browser and the server can read the actual data

4.2.2 Integrity

Integrity ensures data hasn't been modified in unauthorized ways.

Integrity Mechanisms

- Hash functions (MD5, SHA)
- Message Authentication Codes (MAC)
- Digital signatures
- Checksums

4.2.3 Authentication

Authentication verifies the identity of users, devices, or systems.

Authentication Factors

- **Something you know**: Password, PIN
- **Something you have**: Smart card, phone, token
- **Something you are**: Fingerprint, face recognition, biometrics
- **Multi-Factor Authentication (MFA)**: Combines two or more factors

4.2.4 Non-Repudiation

Non-repudiation prevents denial of actions — proves who did what.

Non-Repudiation Mechanisms

- Digital signatures
- Audit logs
- Timestamps
- Certificates

Non-Repudiation Example

Digital signature on contract:

- You sign document with your private key
- Signature proves you signed it
- You cannot later deny signing
- Like notarizing a document

4.2.5 Availability

Availability ensures authorized users can access resources when needed.

Availability Threats and Defenses

Threats:

- Denial of Service (DoS) attacks
- Hardware failures
- Natural disasters
- Power outages

Defenses:

- Redundancy (backup systems)
- Load balancing
- DDoS protection
- Regular backups

4.3 Threats and Attack Types

4.3.1 Passive vs Active Threats

Aspect	**Passive Threats**	**Active Threats**
Action	Observation only	Modification/disruption
Detection	Very difficult	Easier to detect
Prevention	Easier	More difficult
Examples	Eavesdropping, traffic analysis	Masquerading, modification, DoS
Goal	Information gathering	Cause damage or gain unauthorized access

4.3.2 Eavesdropping

Eavesdropping is secretly listening to private communications.

Eavesdropping Scenarios

- Packet sniffing on public WiFi
- Intercepting unencrypted emails
- Wire tapping phone lines
- Man-in-the-middle attacks

Defense: Encryption (TLS/SSL, VPNs)

4.3.3 Masquerading (Spoofing)

Masquerading is pretending to be someone or something else.

Types of Masquerading

- **IP Spoofing**: Fake source IP address
- **Email Spoofing**: Fake sender address
- **DNS Spoofing**: Redirect to malicious sites
- **Phishing**: Fake website mimicking legitimate one

Phishing Attack

1. You receive email appearing to be from your bank
2. Email says ”verify your account” with link
3. Link goes to fake website (looks like your bank)
4. You enter username/password
5. Attacker now has your credentials

Defense: Authentication, digital signatures, user education

4.3.4 Replay Attack

Replay attack retransmits valid data to repeat or delay the action.

Replay Attack Scenario

1. You send encrypted command: ”Transfer $100 to account X”
2. Attacker intercepts encrypted message
3. Later, attacker resends same encrypted message
4. Bank processes it again: another $100 transferred

Defense: Timestamps, nonces (number used once), sequence numbers

4.3.5 Message Modification

Modification attack alters messages in transit.

Modification Example

- Original: "Transfer $100 to Bob"
- Modified: "Transfer $1000 to Attacker"
- Without integrity checking, modification undetected

Defense: Message Authentication Codes, digital signatures

4.3.6 Denial of Service (DoS)

DoS attacks make systems or networks unavailable.

DoS Attack Types

- **Flooding**: Overwhelm with traffic (SYN flood, UDP flood)
- **Amplification**: Small request → large response (DNS amplification)
- **Resource Exhaustion**: Consume CPU, memory, disk
- **DDoS (Distributed DoS)**: Attack from multiple sources (botnets)

SYN Flood Attack

1. Attacker sends many SYN packets (fake source IPs)
2. Server responds with SYN-ACK, allocates resources
3. Final ACK never arrives (fake source)
4. Server's connection queue fills up
5. Legitimate connections rejected

Defense: SYN cookies, rate limiting, DDoS protection services

4.4 Cryptography Fundamentals

4.4.1 What is Cryptography?

Cryptography is the practice of securing communication through encryption.

Cryptography Terminology

- **Plaintext**: Original, readable message
- **Ciphertext**: Encrypted, unreadable message
- **Encryption**: Converting plaintext to ciphertext
- **Decryption**: Converting ciphertext to plaintext
- **Key**: Secret value used in encryption/decryption
- **Algorithm/Cipher**: Method of encryption

4.5 Symmetric Cryptography

4.5.1 How Symmetric Encryption Works

Symmetric encryption uses the **same key** for both encryption and decryption.

Symmetric Encryption Characteristics

- Single shared key
- Fast and efficient
- Key distribution problem (how to share key securely?)
- Used for bulk data encryption

Symmetric Encryption Process

1. Alice and Bob agree on secret key (K)
2. Alice encrypts message: Ciphertext = Encrypt(Plaintext, K)
3. Alice sends ciphertext to Bob
4. Bob decrypts: Plaintext = Decrypt(Ciphertext, K)

4.5.2 Data Encryption Standard (DES)

DES was a widely-used symmetric cipher, now considered insecure.

DES Characteristics

- 56-bit key (effectively, 64-bit with parity)
- 64-bit block size
- 16 rounds of encryption
- Vulnerable to brute force (key too short)
- Officially retired in 2005

4.5.3 Triple DES (3DES)

3DES applies DES three times to increase security.

3DES Process

- Uses three 56-bit keys (K1, K2, K3)
- Encrypt with K1, Decrypt with K2, Encrypt with K3
- Effective key length: 168 bits (though effective security lower)
- Slower than DES (3x operations)
- Being phased out in favor of AES

4.5.4 Advanced Encryption Standard (AES)

AES is the current standard for symmetric encryption.

AES Characteristics

- Key sizes: 128, 192, or 256 bits
- Block size: 128 bits
- Rounds: 10 (128-bit), 12 (192-bit), 14 (256-bit)
- Fast and secure
- Widely used: WiFi (WPA2), VPNs, SSL/TLS, file encryption

AES Usage

- WhatsApp: End-to-end message encryption (AES-256)
- WiFi: WPA2/WPA3 use AES
- HTTPS: Often uses AES for bulk data encryption
- Full disk encryption: BitLocker, FileVault use AES

4.6 Asymmetric Cryptography (Public Key)

4.6.1 How Asymmetric Encryption Works

Asymmetric encryption uses two different keys: public and private.

Asymmetric Encryption Principles

- **Key Pair**: Public key (shared with everyone) + Private key (kept secret)
- **Encryption**: Encrypt with public key, decrypt with private key
- **Digital Signature**: Sign with private key, verify with public key
- **No key distribution problem**
- **Slower than symmetric** (used for key exchange, not bulk data)

Asymmetric Encryption Process

Bob wants to send secret message to Alice:

1. Alice generates key pair (public + private)
2. Alice publishes public key
3. Bob encrypts message with Alice's public key
4. Bob sends ciphertext to Alice
5. Alice decrypts with her private key
6. Only Alice can decrypt (only she has private key)

4.6.2 RSA Algorithm

RSA is the most widely-used asymmetric algorithm.

RSA Overview

- Based on difficulty of factoring large numbers
- Key sizes: 1024, 2048, 4096 bits (2048+ recommended)
- Used for: Key exchange, digital signatures
- Example: HTTPS uses RSA to exchange AES key

RSA Key Generation (Simplified)

1. Choose two large prime numbers: p and q
2. Calculate n = p × q
3. Calculate (n) = (p-1) × (q-1)
4. Choose e (public exponent), typically 65537
5. Calculate d (private exponent) such that e × d 1 (mod (n))
6. **Public key**: (e, n)
7. **Private key**: (d, n)

RSA Encryption/Decryption

- **Encrypt**: $C = M^e \mod n$
- **Decrypt**: $M = C^d \mod n$
- Where M = plaintext, C = ciphertext, e = public exponent, d = private exponent, n = modulus

4.6.3 Diffie-Hellman Key Exchange

Diffie-Hellman allows two parties to establish a shared secret over an insecure channel.

Diffie-Hellman Process

1. Alice and Bob agree on public values: p (prime) and g (generator)
2. Alice chooses secret a, calculates $A = g^a \mod p$, sends A to Bob
3. Bob chooses secret b, calculates $B = g^b \mod p$, sends B to Alice
4. Alice calculates shared secret: $s = B^a \mod p$
5. Bob calculates shared secret: $s = A^b \mod p$
6. Both have same shared secret: $s = g^{ab} \mod p$
7. Eavesdropper can't determine s from A and B

Diffie-Hellman Example (Small Numbers)

- Public: p = 23, g = 5
- Alice: secret a = 6, sends $A = 5^6 \mod 23 = 8$
- Bob: secret b = 15, sends $B = 5^{15} \mod 23 = 19$
- Alice: shared = $19^6 \mod 23 = 2$
- Bob: shared = $8^{15} \mod 23 = 2$
- Shared secret = 2

4.7 Hash Functions

4.7.1 What is a Hash Function?

A **cryptographic hash function** takes input of any size and produces fixed-size output (hash/digest).

Hash Function Properties

- **Deterministic**: Same input always produces same output
- **Fast**: Quick to compute
- **One-way**: Impossible to reverse (get input from output)
- **Collision-resistant**: Hard to find two inputs with same hash
- **Avalanche effect**: Small input change → completely different hash

Hash Function Analogy

Think of hash as a fingerprint:

- Uniquely identifies a document
- Same document → same fingerprint
- Changed document → completely different fingerprint
- Can't recreate document from fingerprint

4.7.2 MD5 (Message Digest 5)

MD5 Characteristics

- 128-bit (16-byte) hash
- Fast to compute
- **Cryptographically broken** (collisions found)
- Still used for non-security purposes (checksums)
- **DO NOT use for security**

4.7.3 SHA (Secure Hash Algorithm) Family

SHA Variants

- **SHA-1**: 160-bit hash, deprecated (collisions found)
- **SHA-2 Family**:
  - SHA-224, SHA-256, SHA-384, SHA-512
  - Currently secure and widely used
  - SHA-256 most common
- **SHA-3**: Latest standard, different design, additional security

SHA-256 Example

Input: `"Hello, World!"`
SHA-256: `dffd6021bb2bd5b0af676290809ec3a53191dd81`
`c7f70a4b28688a362182986f`
Small change: `"Hello, world!"` (lowercase w)
SHA-256: `315f5bdb76d078c43b8ac0064e4a0164612b1fce`
`77c869345bfc94c75894edd3`
Completely different!

4.7.4 Hash Function Applications

Common Uses

- **Password Storage**: Store hash, not plaintext
- **Data Integrity**: Verify files haven't been modified
- **Digital Signatures**: Sign hash instead of entire document
- **Blockchain**: Link blocks using hashes
- **Certificates**: Fingerprint for SSL/TLS certificates

4.8 Digital Signatures and Certificates

4.8.1 Digital Signatures

Digital signatures provide authentication, integrity, and non-repudiation.

Digital Signature Process

Signing:

1. Alice creates document
2. Alice computes hash of document
3. Alice encrypts hash with her private key (signature)
4. Alice sends document + signature

Verification:

1. Bob receives document + signature
2. Bob computes hash of received document
3. Bob decrypts signature with Alice's public key (gets original hash)
4. Bob compares both hashes
5. If match: Valid signature, document unchanged, Alice is sender

Digital Signature Benefits

- **Authentication**: Proves who sent it
- **Integrity**: Proves it wasn't modified
- **Non-repudiation**: Sender can't deny sending

4.8.2 Digital Certificates

Digital certificates bind public keys to identities.

Certificate Contents

- Subject name (who the certificate is for)
- Subject's public key
- Issuer name (Certificate Authority)
- Validity period (not before/not after dates)
- Digital signature of issuer
- Certificate serial number
- Additional information (domain names, etc.)

Certificate Chain

When you visit `https://www.example.com`:

1. Server presents its certificate
2. Certificate signed by Intermediate CA
3. Intermediate CA certificate signed by Root CA
4. Root CA certificate pre-installed in your browser
5. Browser verifies chain: Root → Intermediate → Server
6. If valid, secure connection established

4.9 Security Protocols

4.9.1 IPSec (IP Security)

IPSec provides security at the IP layer, protecting all traffic.

IPSec Components

- **AH (Authentication Header)**: Authentication and integrity
- **ESP (Encapsulating Security Payload)**: Confidentiality, authentication, integrity
- **IKE (Internet Key Exchange)**: Key management

IPSec Modes

- **Transport Mode**:
  - Encrypts payload only, original IP header intact
  - Used for end-to-end communication
  - More efficient
- **Tunnel Mode**:
  - Encrypts entire IP packet
  - New IP header added
  - Used for VPNs (site-to-site)

IPSec VPN Scenario

Company has offices in New York and London:

- Employees in NY need to access London servers
- IPSec VPN established between offices
- Tunnel mode encrypts all traffic
- Secure communication over public internet
- Appears as single private network

4.9.2 SSL/TLS (Secure Sockets Layer / Transport Layer Security)

TLS (successor to SSL) provides secure communication over networks.

TLS Features

- Operates between application and transport layers
- Provides: confidentiality, integrity, authentication
- Uses: hybrid encryption (asymmetric for key exchange, symmetric for data)
- Port: Varies by application (443 for HTTPS, 993 for IMAPS)

4.9.3 TLS Handshake

TLS Handshake Process (Simplified)

1. **Client Hello**: Client sends supported cipher suites, TLS version
2. **Server Hello**: Server selects cipher suite, sends certificate
3. **Certificate Verification**: Client verifies server certificate
4. **Key Exchange**: Client generates pre-master secret, encrypts with server's public key
5. **Session Keys**: Both derive session keys from pre-master secret
6. **Finished Messages**: Both confirm handshake complete
7. **Secure Communication**: Data encrypted with session keys (AES)

HTTPS in Action

You visit `https://www.bank.com`:

1. Browser initiates TLS handshake
2. Server presents certificate
3. Browser verifies certificate (signed by trusted CA)
4. RSA/ECDHE used to exchange AES key
5. All subsequent data encrypted with AES
6. Lock icon appears in browser

4.9.4 S/MIME (Secure/Multipurpose Internet Mail Extensions)

S/MIME adds security to email.

S/MIME Capabilities

- Encrypted email (confidentiality)
- Digital signatures (authentication, integrity, non-repudiation)
- Uses certificates
- Supported by major email clients

S/MIME Process

Sending Encrypted Email:

1. Alice obtains Bob's certificate (public key)
2. Alice encrypts email with Bob's public key
3. Only Bob can decrypt (with his private key)

Sending Signed Email:

1. Alice signs email with her private key
2. Bob verifies signature with Alice's public key
3. Proves Alice sent it and it's unchanged

4.10 Security Infrastructure

4.10.1 Public Key Infrastructure (PKI)

PKI is the framework for managing digital certificates and public keys.

PKI Components

- **Certificate Authority (CA)**: Issues and signs certificates
- **Registration Authority (RA)**: Verifies certificate requests
- **Certificate Repository**: Stores issued certificates
- **Certificate Revocation List (CRL)**: Lists revoked certificates
- **OCSP (Online Certificate Status Protocol)**: Real-time certificate validation

4.10.2 Certificate Authorities (CAs)

CAs are trusted third parties that issue digital certificates.

CA Trust Model

- **Root CAs**: Highest level, self-signed
- **Intermediate CAs**: Signed by root, issue end-entity certificates
- **Certificate Chain**: End-entity → Intermediate → Root
- **Trust Store**: Pre-installed root certificates in OS/browser

Well-Known CAs

- DigiCert
- Let's Encrypt (free, automated)
- GlobalSign
- Comodo
- VeriSign (now DigiCert)

4.10.3 Firewalls

Firewalls control network traffic based on security rules.

Firewall Types

- **Packet Filtering**: Examines headers (IP, port, protocol)
- **Stateful Inspection**: Tracks connection state
- **Application Layer**: Inspects application data (deep packet inspection)
- **Next-Generation**: Combines multiple techniques + IPS

Firewall Rules Example

- Allow: Outbound HTTP/HTTPS (ports 80, 443)
- Allow: Inbound SSH from admin IP only (port 22)
- Block: All inbound connections to port 23 (Telnet)
- Allow: DNS queries (port 53)
- Default: Deny all other traffic

4.10.4 Intrusion Detection/Prevention Systems

IDS vs IPS

IDS (Intrusion Detection System):

- Monitors network traffic
- Detects suspicious activity
- Alerts administrators
- Passive (doesn't block)

IPS (Intrusion Prevention System):

- Monitors and analyzes traffic
- Detects attacks
- Automatically blocks/prevents
- Active defense

Detection Methods

- **Signature-based**: Matches known attack patterns
- **Anomaly-based**: Detects deviations from normal behavior
- **Hybrid**: Combines both methods

4.10.5 Virtual Private Networks (VPNs)

VPNs create secure connections over public networks.

VPN Benefits

- Encrypts traffic (confidentiality)
- Remote access to private networks
- Hides IP address
- Bypasses geographic restrictions
- Protects on public WiFi

VPN Use Cases

- **Remote Work**: Employee accesses company network from home
- **Site-to-Site**: Connect branch offices securely
- **Privacy**: Encrypt traffic on public WiFi
- **Geo-restrictions**: Access region-locked content

VPN Protocols

- **IPSec**: Secure, widely used for site-to-site
- **OpenVPN**: Open-source, flexible, very secure
- **L2TP/IPSec**: Combines L2TP (tunneling) with IPSec (security)
- **WireGuard**: Modern, fast, simple
- **PPTP**: Obsolete, insecure (avoid)

4.11 Unit VI Summary

Key Concepts Checklist

- Security services: Confidentiality, integrity, authentication, non-repudiation, availability
- Threats: Passive (eavesdropping) vs active (masquerading, modification, DoS)
- Symmetric crypto: DES (obsolete), 3DES (phasing out), AES (current standard)
- Asymmetric crypto: RSA (widely used), Diffie-Hellman (key exchange)
- Hash functions: MD5 (broken), SHA-1 (deprecated), SHA-2/SHA-3 (secure)
- Digital signatures: Authentication + integrity + non-repudiation
- Certificates: Bind public keys to identities, issued by CAs
- IPSec: IP-layer security, AH (auth) vs ESP (encryption), transport vs tunnel modes
- SSL/TLS: Secure communication, handshake process, hybrid encryption
- S/MIME: Secure email
- PKI: Certificate management framework, CAs, trust chains
- Firewalls: Control network traffic, various types
- IDS/IPS: Detect/prevent intrusions
- VPNs: Secure connections over public networks

5 Glossary of Key Terms

ACK (Acknowledgment): TCP flag indicating acknowledgment of received data

AES (Advanced Encryption Standard): Current standard symmetric encryption algorithm

ARP (Address Resolution Protocol): Maps IP addresses to MAC addresses

Asymmetric Encryption: Encryption using public/private key pairs

CIDR (Classless Inter-Domain Routing): IP addressing scheme using prefix notation

Ciphertext: Encrypted, unreadable data

Congestion Control: Mechanism to prevent network overload

DHCP (Dynamic Host Configuration Protocol): Automatically assigns IP addresses

DNS (Domain Name System): Translates domain names to IP addresses

DoS (Denial of Service): Attack making systems unavailable

Flow Control: Mechanism preventing sender from overwhelming receiver

FTP (File Transfer Protocol): Protocol for transferring files

Hash Function: One-way function producing fixed-size output

HTTP (Hypertext Transfer Protocol): Foundation of web communication

HTTPS: HTTP over TLS/SSL (secure)

ICMP (Internet Control Message Protocol): Used for error reporting (ping)

IMAP (Internet Message Access Protocol): Advanced email retrieval protocol

IPSec (IP Security): Security at IP layer

IPv4: 32-bit IP addressing

IPv6: 128-bit IP addressing

MAC (Media Access Control): Physical address of network interface

MTU (Maximum Transmission Unit): Maximum packet size

Multiplexing: Combining multiple signals/connections

NAT (Network Address Translation): Translates private IPs to public IP

OSPF (Open Shortest Path First): Link state routing protocol

Packet: Unit of data transmitted over network

PKI (Public Key Infrastructure): Framework for managing certificates

Plaintext: Original, readable data

POP3 (Post Office Protocol 3): Email retrieval protocol

Port: Logical endpoint for network communication

RIP (Routing Information Protocol): Distance vector routing protocol

RSA: Widely-used asymmetric encryption algorithm

RTT (Round-Trip Time): Time for packet to reach destination and return

SHA (Secure Hash Algorithm): Family of cryptographic hash functions

SMTP (Simple Mail Transfer Protocol): Protocol for sending email

Socket: IP address + port number

SSH (Secure Shell): Secure remote access protocol

SSL/TLS: Protocols providing secure communication

Subnet: Subdivision of IP network

Symmetric Encryption: Encryption using same key for encrypt/decrypt

TCP (Transmission Control Protocol): Reliable, connection-oriented transport

Three-Way Handshake: TCP connection establishment process

TTL (Time to Live): Hop limit for packets

UDP (User Datagram Protocol): Connectionless, unreliable transport

VLSM (Variable Length Subnet Masking): Different-sized subnets in same network

VPN (Virtual Private Network): Secure connection over public network

6 Exam Preparation Tips

6.1 Study Strategies

Effective Study Techniques

1. **Active Recall**: Test yourself regularly without looking at notes
2. **Spaced Repetition**: Review material at increasing intervals
3. **Understand, Don't Memorize**: Focus on concepts, not rote memorization
4. **Practice Problems**: Work through subnetting, routing algorithm examples
5. **Draw Diagrams**: Visualize concepts (TCP state machine, network topology)
6. **Teach Others**: Explaining concepts reinforces understanding

6.2 Key Topics to Master

High-Priority Topics

Unit III - Network Layer:

- Subnetting calculations and VLSM
- Dijkstra's and Bellman-Ford algorithms (know how to execute)
- IPv4 vs IPv6 differences
- Routing protocol comparisons

Unit IV - Transport Layer:

- TCP vs UDP comparison
- Three-way handshake and four-way termination
- Flow control and congestion control mechanisms
- Sequence/acknowledgment number calculations

Unit V - Application Layer:

- DNS resolution process
- HTTP methods and status codes
- Email protocol differences (SMTP vs POP3 vs IMAP)
- FTP modes

Unit VI - Security:

- Symmetric vs asymmetric encryption
- Hash function properties and usage
- Digital signature process
- IPSec modes and TLS handshake

6.3 Common Exam Question Types

Question Categories

- **Calculations**: Subnetting, checksum, timeout values
- **Protocol Comparisons**: TCP vs UDP, RIP vs OSPF, POP3 vs IMAP
- **Process Flows**: DNS resolution, TCP handshake, TLS handshake
- **Algorithm Execution**: Dijkstra's, Bellman-Ford on given topology
- **Security Scenarios**: Choosing appropriate mechanisms
- **Conceptual**: Explain principles, identify correct statements

6.4 Final Checklist

Before the Exam

- □ Review all unit summaries
- □ Practice subnetting problems
- □ Trace through routing algorithms
- □ Understand protocol header structures
- □ Know port numbers for common services
- □ Review security mechanisms and when to use each
- □ Practice with past exams or sample questions
- □ Get adequate rest before exam day

Good luck with your exam!

Remember: Understanding concepts is more valuable than memorizing facts.
Focus on the ”why” and ”how,” not just the ”what.”