Mcn (4351602) - Problem Solver

Antigravity AI

June 4, 2026

Mcn

First Edition: 2026

Author: Antigravity AI
Website: milav.in
YouTube: @milav.dabgar

To my students,
who constantly inspire me to find new analogies,
break down complex theories, and make learning
an engineered science.

Preface

About this Book

Welcome to **Computer Networks**! This textbook is meticulously designed to serve as a comprehensive problem solver and study companion. It provides a robust and intuitive foundation in the core principles of this subject, bridging the gap between theoretical models and practical applications. A unique feature of this book is its focus on decoding complex concepts using clear, step-by-step methodologies and real-world analogies.

Target Audience

This book is intended for students and professionals looking to master the concepts of Computer Networks. Whether you are revising for exams, seeking to solidify your fundamental understanding, or looking for practical examples to clarify abstract concepts, this book has been structured to support your learning journey. It serves as an accessible guide designed to remove the fear of complex topics, replacing it with an appreciation for the underlying mechanics.

Scope and Coverage

The coverage is divided logically to follow standard academic curriculums. Each chapter focuses on key topics, breaking them down into digestible sections:

- **Theoretical Insights** – Core concepts explained intuitively.
- **Method Blocks** – Standardized, step-by-step procedures for solving specific types of problems.
- **Worked Examples** – Fully solved problems that apply the methods in practice.

We hope this resource proves invaluable in your studies and helps you achieve excellence in **Computer Networks**. Happy learning!

Acknowledgments

Writing a comprehensive book that connects the fundamental forces of Chemistry with practical engineering applications requires more than just academic knowledge—it requires a community of support. I would like to express my sincere gratitude to everyone who contributed to the realization of this project.

Specially, I extend my heartfelt gratitude to the **Department of Technical Education (DTE), Government of Gujarat**, and **Government Polytechnic, Palanpur**, for providing an environment that fosters innovation, academic rigor, and continuous professional development.

I am deeply thankful to my family for their unwavering patience and support during the countless hours spent researching, formatting, and refining these chapters.

Finally, my deepest appreciation goes to my students. Your curiosity, challenging questions, and continuous feedback are the true driving force behind the unique analogies and streamlined structure of this book. This textbook was engineered for you.

Antigravity AI

About the Author

Milav Jayeshkumar Dabgar is an engineering educator and R&D professional with over 9 years of experience in electronics hardware development, embedded systems, AI/ML, and full-stack software engineering. He currently serves as a **Lecturer (Class - II) & Innovation Leader** at Government Polytechnic, Palanpur, under the Education Department of the Government of Gujarat.

Milav holds a Master of Engineering (ME) in Communication Systems from L.D. College of Engineering and a Bachelor of Engineering (BE) in Electronics & Communication. He is also currently pursuing a **BS in Data Science and Applications from IIT Madras**, where he has completed both the **Diploma in Programming** and the **Diploma in Data Science** with distinction.

Most recently, he completed the **AICTE QIP PG Certificate Program in Deep Learning from SVNIT Surat** with a **perfect 10/10 CGPA**, topping his batch.

A dedicated mentor, Milav has guided numerous student teams in securing over **INR 45 lakhs** in innovation funding, including INR 25 lakhs from Shark Tank India and INR 20 lakhs through iHub Gujarat, resulting in two student patents. He is recognized as an **NPTEL Evangelist** and **NPTEL Discipline Star** in Computer Science, reflecting his commitment to continuous learning and academic excellence.

His technical expertise spans from embedded systems (8051, STM32, Arduino) to modern web technologies (Next.js, Python, PostgreSQL) and Data Science (TensorFlow, PyTorch). Through this textbook, he aims to simplify complex Engineering Chemistry concepts by integrating relatable, real-world analogies drawn from modern tech and engineering landscapes.

Milav is also an active content creator, running a popular **YouTube channel** (@milav.dabgar) where he shares technical tutorials and academic resources. He maintains a personal blog and resource hub at milav.in and can be reached for professional collaborations on LinkedIn.

Contents

List of Figures

List of Tables

CHAPTER 1

Networking Essential

1.1 Differentiate Different Models of Network Computing

Network computing involves distributing tasks and data among different computers. The two primary models are the **Client-Server** model and the **Peer-to-Peer (P2P)** model. To differentiate their performance, we calculate the time required to distribute a file of size F to N peers.

Methodology:

File Distribution Time in Client Server and P2P This method calculates the minimum time required to distribute a file from a single source to multiple destinations under different network models.

1. **Identify Parameters:**
   - F: File size (in bits).
   - N: Number of downloading peers.
   - u_s: Upload capacity of the server (in bps).
   - $d_{\min}$: Minimum download capacity among all peers (in bps).
   - $\sum_{i=1}^{N} u_i$: Total upload capacity of all N peers (in bps).
2. **Client-Server Distribution Time (D_{CS}):** The server must transmit the file N times sequentially. The bottleneck is either the server's upload capacity or the slowest peer's download capacity.

$$D_{CS} = \max\left(\frac{N \cdot F}{u_s}, \frac{F}{d_{\min}}\right)$$

3. **Peer-to-Peer Distribution Time (D_{P2P}):** The server must upload the file at least once, the slowest peer must download it, and the total system must upload $N \cdot F$ bits using all available upload bandwidth.

$$D_{P2P} = \max\left(\frac{F}{u_s}, \frac{F}{d_{\min}}, \frac{N \cdot F}{u_s + \sum_{i=1}^{N} u_i}\right)$$

Worked Example:

Calculating File Distribution Time Consider a network where a server needs to distribute a file of size $F = 2$ Gigabits (2×10^9 bits) to $N = 10$ peers. The server's upload capacity is $u_s = 200$ Mbps. Each peer has a download capacity of $d_{\min} = 50$ Mbps and an upload capacity of $u_i = 10$ Mbps. Calculate the minimum distribution time for both the Client-Server and P2P architectures.

Step 1: Convert units to standard bits and bits per second (bps).

- $F = 2,000$ Megabits $= 2,000$ Mb
- $N = 10$
- $u_s = 200$ Mbps

- $d_{\min} = 50$ Mbps
- $u_i = 10$ Mbps (Total peer upload capacity $\sum u_i = 10 \times 10 = 100$ Mbps).

Step 2: Calculate Client-Server Distribution Time (D_{CS}).

$$\begin{aligned} D_{CS} &= \max\left(\frac{10 \times 2000}{200}, \frac{2000}{50}\right) \\ &= \max(100 \text{ seconds}, 40 \text{ seconds}) \\ &= 100 \text{ seconds} \end{aligned}$$

Step 3: Calculate Peer-to-Peer Distribution Time (D_{P2P}).

$$\begin{aligned} D_{P2P} &= \max\left(\frac{2000}{200}, \frac{2000}{50}, \frac{10 \times 2000}{200 + 100}\right) \\ &= \max\left(10, 40, \frac{20000}{300}\right) \\ &= \max(10, 40, 66.67) \\ &= 66.67 \text{ seconds} \end{aligned}$$

Conclusion: The P2P model completes the distribution in 66.67 seconds compared to 100 seconds for the Client-Server model. This computationally demonstrates the performance advantage of utilizing peer upload bandwidth.

1.2 Explain OSI Model and TCP/IP

The OSI and TCP/IP models define how data is encapsulated with protocol headers at each layer before transmission. To analyze network efficiency and performance computationally, we calculate encapsulation overhead and transmission delay.

Methodology:

Encapsulation Overhead and Transmission Delay This method calculates the effective data rate and the time required to transmit a packet across a link while accounting for the headers added by various protocol layers.

1. **Identify Payload and Header Sizes:** Determine the application payload size (P) and the header sizes added by Transport (H_t), Network (H_n), and Data Link (H_d) layers.
2. **Calculate Total Packet Size (S):**

$$S = P + H_t + H_n + H_d$$

3. **Calculate Overhead Percentage (O):**

$$O = \left(\frac{H_t + H_n + H_d}{S}\right) \times 100\%$$

4. **Calculate Transmission Delay (T_d):** Given the link bandwidth B (in bps), convert total packet size S to bits and compute delay:

$$T_d = \frac{S \text{ (in bits)}}{B}$$

Worked Example:

TCP-IP Encapsulation Analysis An application generates a message payload of 1200 Bytes. The message is sent using TCP/IP over an Ethernet link. The header sizes are: TCP Header = 20 Bytes, IP Header = 20 Bytes, Ethernet Header and Trailer = 18 Bytes. The network link has a bandwidth of 100 Mbps. Calculate the total packet size, overhead percentage, and transmission delay.

Step 1: Identify Payload and Header Sizes.

- $P = 1200$ Bytes
- $H_t = 20$ Bytes
- $H_n = 20$ Bytes
- $H_d = 18$ Bytes

Step 2: Calculate Total Packet Size (S).

$$\begin{aligned} S &= 1200 + 20 + 20 + 18 \\ &= 1258 \text{ Bytes} \end{aligned}$$

Step 3: Calculate Overhead Percentage (O).

$$\text{Total Header Size} = 20 + 20 + 18 = 58 \text{ Bytes}$$

$$\begin{aligned} O &= \left(\frac{58}{1258}\right) \times 100\% \\ &\approx 4.61\% \end{aligned}$$

Step 4: Calculate Transmission Delay (T_d). Convert total packet size to bits: $S_{\text{bits}} = 1258 \times 8 = 10,064$ bits.
Bandwidth $B = 100 \times 10^6$ bps.

$$\begin{aligned} T_d &= \frac{10064}{100,000,000} \\ &= 0.00010064 \text{ seconds} \\ &= 100.64 \text{ microseconds } (\mu\text{s}) \end{aligned}$$

1.3 Describe Data Traffic and Congestion Management

Congestion management ensures network stability by shaping data traffic. Two fundamental computational algorithms used for traffic shaping and policing are the Leaky Bucket and Token Bucket algorithms.

Methodology:

Leaky Bucket Algorithm The Leaky Bucket algorithm regulates the burstiness of incoming traffic, producing a steady outgoing stream. If the bucket overflows, incoming packets are discarded.

1. **Define System Parameters:**
   - C: Capacity of the bucket (in Bytes).
   - R: Outflow or leak rate (in Bytes/sec).
   - A: Inflow or arrival rate of bursty traffic (in Bytes/sec).
2. **Check Overflow Condition:** If $A \leq R$, the bucket never overflows.
   If $A > R$, data accumulates in the bucket at a rate of $(A - R)$ Bytes/sec.

3. **Calculate Time to Overflow (t):**
$$t = \frac{C}{A - R}$$
4. **Calculate Data Discarded:** If the burst lasts for a duration T (where $T > t$), the amount of data discarded is:
$$\text{Discarded Data} = (T - t) \times (A - R)$$

Worked Example:

Leaky Bucket Overflow Calculation A network node uses a leaky bucket algorithm for traffic shaping. The bucket has a capacity of $C = 2500$ Bytes and a steady leak rate of $R = 250$ Bytes/sec. A traffic burst arrives at a rate of $A = 1000$ Bytes/sec and lasts for $T = 5$ seconds. Calculate the time until the bucket overflows and the total amount of data discarded.

Step 1: Check Overflow Condition. The arrival rate $A(1000)$ is greater than the leak rate $R(250)$. The bucket will accumulate data.

Accumulation rate $= 1000 - 250 = 750$ Bytes/sec.

Step 2: Calculate Time to Overflow (t).

$$\begin{aligned} t &= \frac{C}{A - R} \\ &= \frac{2500}{750} \\ &= 3.33 \text{ seconds} \end{aligned}$$

Step 3: Calculate Data Discarded. The burst lasts for $T = 5$ seconds, which is strictly longer than $t = 3.333$ seconds.

$$\begin{aligned} \text{Time spent overflowing} &= T - t = 5 - 3.333 = 1.667 \text{ seconds} \\ \text{Discarded Data} &= 1.667 \times 750 \\ &= 1250 \text{ Bytes} \end{aligned}$$

Verification:

- Total Data Arrived $= 5 \times 1000 = 5000$ Bytes.
- Total Data Leaked during burst $= 5 \times 250 = 1250$ Bytes.
- Data stored safely in bucket $= 2500$ Bytes.
- Discarded Data $= 5000 - 1250 - 2500 = 1250$ Bytes.

Methodology:

Token Bucket Burst Calculation The Token Bucket algorithm allows brief bursts of data to exceed the average transmission rate as long as unused tokens are available in the bucket.

1. **Define System Parameters:**
   - C: Token bucket capacity (in tokens/Bytes).
   - R: Token generation rate (in Bytes/sec).
   - M: Maximum output transmission rate of the link (in Bytes/sec).
2. **Calculate Maximum Burst Duration (S):** The maximum time data can be sent at the peak rate M before the bucket completely empties is:
$$S = \frac{C}{M - R} \quad (\text{for } M > R)$$

3. **Calculate Maximum Data Transmitted in Burst (D):**

$$D = M \times S$$

Worked Example:

Token Bucket Max Burst Calculation A token bucket has a capacity of $C = 4000$ Bytes. Tokens are consistently added at a rate of $R = 500$ Bytes/sec. The maximum physical transmission rate of the output link is $M = 2500$ Bytes/sec. Calculate the maximum duration of a burst at the peak transmission rate and the total data sent during this exact burst.

Step 1: Determine Parameters. $C = 4000$ Bytes, $R = 500$ Bytes/sec, $M = 2500$ Bytes/sec.

Step 2: Calculate Maximum Burst Duration (S).

$$\begin{aligned} S &= \frac{C}{M - R} \\ &= \frac{4000}{2500 - 500} \\ &= \frac{4000}{2000} \\ &= 2 \text{ seconds} \end{aligned}$$

Step 3: Calculate Maximum Data Transmitted (D).

$$\begin{aligned} D &= M \times S \\ &= 2500 \times 2 \\ &= 5000 \text{ Bytes} \end{aligned}$$

Verification: In 2 seconds, 4000 tokens initially available $+(500 \times 2)$ tokens generated dynamically $= 5000$ tokens, matching the 5000 Bytes successfully sent.

CHAPTER 2

Protocol and Addressing Scheme

2.1 Protocols of the OSI Layer

The OSI (Open Systems Interconnection) model uses specific protocols at each layer, wrapping data with headers as it travels down the stack. Computational problems here involve calculating total packet sizes and Protocol Data Unit (PDU) sizes at various layers due to encapsulation overhead.

Methodology:

Calculation of Protocol Data Unit Sizes To determine the size of the Protocol Data Unit (PDU) at any given layer during encapsulation, use the following steps:

1. **Identify Application Data:** Start with the original data size generated by the Application, Presentation, and Session layers.
2. **Transport Layer PDU (Segment):** Add the Transport layer header size (e.g. TCP Header is typically 20 bytes, UDP is 8 bytes).

$$\text{Segment Size} = \text{Data Size} + \text{Transport Header Size}$$

3. **Network Layer PDU (Packet):** Add the Network layer header size (e.g. IPv4 Header is typically 20 bytes).

$$\text{Packet Size} = \text{Segment Size} + \text{Network Header Size}$$

4. **Data Link Layer PDU (Frame):** Add the Data Link layer header and trailer size (e.g. Ethernet Header is 14 bytes and FCS trailer is 4 bytes).

$$\text{Frame Size} = \text{Packet Size} + \text{Frame Header Size} + \text{Frame Trailer Size}$$

Worked Example:

Calculating Encapsulation Overhead An application generates 400 bytes of data. It uses UDP at the Transport layer (8 bytes header), IPv4 at the Network layer (20 bytes header), and Ethernet at the Data Link layer (14 bytes header and 4 bytes trailer). Calculate the total size of the Ethernet frame and the percentage of overhead.

Step 1: Calculate Segment Size at Transport Layer

$$\text{Segment Size} = 400 \text{ bytes (Data)} + 8 \text{ bytes (UDP)} = 408 \text{ bytes}$$

Step 2: Calculate Packet Size at Network Layer

$$\text{Packet Size} = 408 \text{ bytes (Segment)} + 20 \text{ bytes (IPv4)} = 428 \text{ bytes}$$

Step 3: Calculate Frame Size at Data Link Layer

$$\text{Frame Size} = 428 \text{ bytes (Packet)} + 14 \text{ bytes (Ethernet Header)} + 4 \text{ bytes (Trailer)} = 446 \text{ bytes}$$

Step 4: Calculate Overhead Percentage

$$\text{Total Overhead} = 8 + 20 + 14 + 4 = 46 \text{ bytes}$$

$$\text{Overhead Percentage} = \left(\frac{46}{446}\right) \times 100 \approx 10.31\%$$

2.2 IPv4 and IPv6 Addressing Scheme

IP addresses uniquely identify devices on a network. IPv4 addresses are 32-bit numbers, while IPv6 addresses are 128-bit numbers.

Methodology:

Identifying IPv4 Address Classes IPv4 addresses are divided into classes based on the value of the first octet.

1. **Class A:** First octet is 1 to 126. Default mask is 255.0.0.0 (/8).
2. **Class B:** First octet is 128 to 191. Default mask is 255.255.0.0 (/16).
3. **Class C:** First octet is 192 to 223. Default mask is 255.255.255.0 (/24).
4. **Class D:** First octet is 224 to 239 (Used for Multicasting).
5. **Class E:** First octet is 240 to 255 (Experimental).

Worked Example:

Classifying IPv4 Addresses Determine the class and default subnet mask for the IP address 172.16.45.10.
Step 1: Examine the first octet The first octet is 172.
Step 2: Match with class ranges The value 172 falls in the range 128 to 191, which corresponds to Class B.
Step 3: State the default subnet mask For Class B, the default subnet mask is 255.255.0.0.

Methodology:

Compressing IPv6 Addresses IPv6 addresses are written as eight groups of four hexadecimal digits. To compress them:

1. **Omit Leading Zeros:** In any 16-bit block, leading zeros can be removed. For example $004B$ becomes $4B$.
2. **Zero Compression:** Consecutive blocks of zeros can be replaced by a double colon (::). This rule can only be applied **once** per address to avoid ambiguity.

Worked Example:

IPv6 Address Compression Compress the following IPv6 address:
$2001 : 0db8 : 0000 : 0000 : 0000 : ff00 : 0042 : 8329$
Step 1: Omit leading zeros Remove leading zeros in each block:

$$2001 : db8 : 0 : 0 : 0 : ff00 : 42 : 8329$$

Step 2: Apply zero compression Replace the longest sequence of zero blocks $(0 : 0 : 0)$ with a double colon (::):

$$2001 : db8 :: ff00 : 42 : 8329$$

2.3 Subnetting and Supernetting of IPv4

Subnetting divides a large network into smaller networks, while supernetting (route aggregation) combines multiple smaller networks into a larger one.

Methodology:

IPv4 Subnetting Calculations To perform subnetting, borrow n bits from the host portion of the address. Let h be the remaining host bits.

1. **Number of Subnets:** 2^n
2. **Number of Usable Hosts per Subnet:** $2^h - 2$ (Subtract 2 for Network ID and Broadcast Address).
3. **New Subnet Mask:** Add the n borrowed bits (set to 1) to the default subnet mask.
4. **Block Size (Increment):** 256 − (interesting octet value in subnet mask).
5. **Subnet Ranges:** Start at 0, and keep adding the block size to find consecutive Network IDs.

Worked Example:

Subnetting a Class C Network Given the network address 192.168.10.0/24, you need to create at least 5 subnets. Calculate the new subnet mask, number of usable hosts per subnet, and the network range of the first two subnets.
Step 1: Determine borrowed bits (n) We need ≥ 5 subnets. $2^n \geq 5 \implies 2^3 = 8$. So $n = 3$ bits are borrowed.
Step 2: Calculate new subnet mask Original mask is /24. New mask is $/24 + 3 = /27$. In binary, the last octet is 11100000, which is $128 + 64 + 32 = 224$. New subnet mask: 255.255.255.224.
Step 3: Calculate usable hosts per subnet Total bits in IPv4 = 32. Remaining host bits $h = 32 - 27 = 5$. Usable hosts $= 2^5 - 2 = 32 - 2 = 30$ hosts.
Step 4: Calculate block size Block size $= 256 - 224 = 32$.
Step 5: Determine the ranges for the first two subnets

Subnet No.	Network ID	First Valid IP	Last Valid IP	Broadcast IP
0	192.168.10.0	192.168.10.1	192.168.10.30	192.168.10.31
1	192.168.10.32	192.168.10.33	192.168.10.62	192.168.10.63

Methodology:

Determining Subnet ID from an IP Address Given an IP address and its subnet mask (or CIDR notation), find the Network ID (Subnet ID) to which it belongs.

1. Convert the IP address and the Subnet Mask into binary format.
2. Perform a bitwise AND operation between the binary IP address and binary Subnet Mask.
3. Convert the resulting binary value back to decimal to get the Network ID.
4. Alternatively using decimal: The block size is 256 − mask value. Find the multiple of the block size that is less than or equal to the IP's value in the interesting octet.

Worked Example:

Finding the Network Address Find the Network ID and Broadcast Address for the host IP 10.1.5.130/26.
Step 1: Identify the interesting octet and block size The mask /26 means 255.255.255.192. The interesting octet is the 4th octet (192). Block size $= 256 - 192 = 64$.

Step 2: Find the Network ID The 4th octet of the IP is 130. Multiples of the block size (64): $0, 64, 128, 192$. The largest multiple less than or equal to 130 is 128. Therefore the Network ID is 10.1.5.128.
Step 3: Find the Broadcast Address The next subnet starts at 10.1.5.192 (since $128 + 64 = 192$). The broadcast address of the current subnet is one less than the next Network ID. Broadcast Address = 10.1.5.191.

Methodology:

IPv4 Supernetting (Route Aggregation) Supernetting combines multiple contiguous networks into a single summary route to reduce routing table size.

1. **List Networks in Binary:** Write out the network addresses in binary format.
2. **Identify Matching Bits:** Find all contiguous bits from left to right that are identical across all network addresses.
3. **Create Supernet Mask:** Count the number of matching bits. This becomes the new CIDR prefix.
4. **Determine Supernet Address:** Use the matching bits followed by all zeros to form the supernet network address.

Worked Example:

Aggregating Four Class C Networks Aggregate the following four network addresses into a single supernet route:

- 192.168.16.0/24
- 192.168.17.0/24
- 192.168.18.0/24
- 192.168.19.0/24

Step 1: Convert the varying octet (3rd octet) to binary The first two octets (192.168) are identical. Let us expand the 3rd octet:

$$\begin{aligned} 16 &\rightarrow \quad 00010000 \\ 17 &\rightarrow \quad 00010001 \\ 18 &\rightarrow \quad 00010010 \\ 19 &\rightarrow \quad 00010011 \end{aligned}$$

Step 2: Identify common bits Comparing the binary representations:

$$\begin{aligned} &000100|00 \\ &000100|01 \\ &000100|10 \\ &000100|11 \end{aligned}$$

The first 6 bits of the 3rd octet are common (000100).
Step 3: Calculate the new subnet mask The first two octets provided $8 + 8 = 16$ common bits. The 3rd octet provides 6 common bits. Total common bits $= 16 + 6 = 22$. The supernet mask is /22, which is 255.255.252.0.
Step 4: Determine the supernet address Take the common bits and append zeros: 00010000 in decimal is 16. The aggregated supernet route is 192.168.16.0/22.

CHAPTER 3

Introduction to Mobile Computing

Mobile computing involves the transmission of data, voice, and video via a computer or any other wireless-enabled device without having to be connected to a fixed physical link. In this chapter, we explore the core architectural models, network formations like ad hoc networks, the role of middleware and gateways, and the diverse applications of mobile computing.

3.1 Architecture for Mobile Computing

The architecture of mobile computing systems defines how devices, networks, and backend systems interact to deliver services. The most standard approach is the 3-Tier Architecture.

Methodology:

Three Tier Mobile Architecture The 3-tier architecture divides the system into three distinct logical layers:

1. **Presentation Tier (Client/Device Layer):** The user interface and mobile application running on the mobile device (e.g. smartphone or tablet). It handles user interaction and displays data.
2. **Application Tier (Middleware/Processing Layer):** This tier acts as a bridge between the client and the data. It contains application servers, middleware, and gateways that process business logic, handle requests, and format data for the mobile device.
3. **Data Tier (Database/Backend Layer):** The backend databases and legacy systems that store and manage the actual data.

Worked Example:

Analyzing a Mobile Banking System **Problem:** Map the components of a mobile banking application to the 3-tier mobile architecture.
Solution:

1. **Step 1: Identify Presentation Tier Components** The mobile app installed on the user's smartphone where they log in, view balances, and initiate transfers forms the Presentation Tier.
2. **Step 2: Identify Application Tier Components** The bank's application servers that authenticate the user, validate the transfer limits, and ensure secure communication (via APIs and gateways) represent the Application Tier.
3. **Step 3: Identify Data Tier Components** The central database servers in the bank's data center that actually hold the account balance records and transaction history form the Data Tier.

3.2 Ad Hoc Networks

An ad hoc network is a decentralized type of wireless network. The network is ad hoc because it does not rely on a pre-existing infrastructure, such as routers in wired networks or access points in managed (infrastructure) wireless networks.

Methodology:

Ad Hoc Network Routing Protocol In Mobile Ad Hoc Networks (MANETs), nodes must dynamically discover routes. A common algorithm is the Ad hoc On-Demand Distance Vector (AODV) routing protocol.

1. **Route Request (RREQ):** When a source node needs to send data to a destination but does not have a route, it broadcasts an RREQ packet to its neighbors.
2. **Forwarding RREQ:** Neighbors forward the RREQ to their neighbors, establishing a reverse path back to the source.
3. **Route Reply (RREP):** When the RREQ reaches the destination (or an intermediate node with a valid route), an RREP packet is sent back along the reverse path.
4. **Data Transmission:** Once the RREP reaches the source, a forward path is established, and data packets are transmitted.

Worked Example:

AODV Route Discovery Simulation **Problem:** Consider a simple ad hoc network with 4 nodes: A, B, C, and D arranged in a line where communication range allows A to reach B, B to reach A and C, C to reach B and D, and D to reach C. Trace the AODV route discovery process for Node A to send data to Node D. **Solution:**

1. **Step 1: Broadcast Route Request** Node A needs to send data to Node D. It has no route. Node A broadcasts an RREQ message.
2. **Step 2: First Hop Forwarding** Node B receives the RREQ from A. It records a reverse route to A. B does not have a route to D, so it broadcasts the RREQ.
3. **Step 3: Second Hop Forwarding** Node C receives the RREQ from B. It records a reverse route to B (and ultimately A). C does not have a route to D, so it broadcasts the RREQ.
4. **Step 4: Destination Reached and Route Reply** Node D receives the RREQ from C. D is the destination. D generates an RREP packet and sends it back to C using the reverse route.
5. **Step 5: RREP Propagation** Node C receives the RREP, updates its forward routing table for D, and forwards the RREP to B. Node B receives the RREP, updates its table, and forwards it to A.
6. **Step 6: Path Established** Node A receives the RREP. The path A $\rightarrow$ B $\rightarrow$ C $\rightarrow$ D is now established, and Node A begins transmitting data.

3.3 Middleware and Gateway

Middleware and gateways are essential components in the Application Tier of mobile computing, handling the mismatch between wireless environments and fixed wired networks.

Methodology:

Gateway Protocol Translation A Mobile Gateway acts as a proxy that translates protocols from the mobile network to standard internet protocols.

1. **Request Reception:** The gateway receives a request from a mobile device using a mobile-optimized protocol (e.g. WAP).
2. **Translation:** The gateway translates the mobile protocol request into a standard web protocol request (e.g. HTTP).
3. **Forwarding:** The translated HTTP request is sent to the target web server.

4. **Response Processing:** The gateway receives the HTTP response (e.g. HTML) from the server.
5. **Reverse Translation:** The gateway translates and compresses the response into a mobile-friendly format (e.g. WML or optimized JSON) and sends it back to the mobile device.

Worked Example:

WAP Gateway Operation **Problem:** Trace the steps taken by a WAP Gateway when a mobile user accesses a standard web page.
Solution:

1. **Step 1:** The mobile client sends a Wireless Session Protocol (WSP) Get request over the cellular network.
2. **Step 2:** The WAP Gateway intercepts the WSP request and converts it into a standard HTTP Get request.
3. **Step 3:** The HTTP Get request is routed over the Internet to the target Web Server.
4. **Step 4:** The Web Server processes the request and returns an HTTP response containing HTML content.
5. **Step 5:** The WAP Gateway receives the HTML. It parses and translates the HTML into Wireless Markup Language (WML), which is optimized for mobile screens.
6. **Step 6:** The Gateway sends the WML response back to the mobile client using WSP.

3.4 Applications of Mobile Computing

Mobile computing transforms how businesses and individuals operate by providing ubiquitous access to information.

Methodology:

Categorizing Mobile Applications Mobile computing applications can be grouped by their core computational function:

1. **Location-Based Services (LBS):** Uses GPS coordinates to provide contextual information (e.g. finding nearby restaurants or navigation).
2. **Mobile Commerce (M-Commerce):** Financial transactions using mobile devices (e.g. mobile banking, digital wallets).
3. **Enterprise Mobility:** Applications that connect mobile workers to corporate databases (e.g. inventory management, sales force automation).
4. **Emergency and Health Services:** Real-time data transmission for critical situations (e.g. remote patient monitoring, disaster response).

Worked Example:

Classifying Real World Systems **Problem:** Categorize the following systems into the appropriate mobile application category: 1. Google Maps Navigation. 2. Apple Pay. 3. A delivery driver's package tracking app. 4. A smartwatch measuring heart rate and sending it to a doctor.
Solution:

1. **System 1 (Google Maps):** This heavily relies on GPS and mapping algorithms. It is classified as **Location-Based Services (LBS)**.

2. **System 2 (Apple Pay):** This handles secure financial transactions via NFC. It is classified as **Mobile Commerce (M-Commerce)**.

3. **System 3 (Package Tracking App):** This connects a field worker (driver) to the company's central inventory/logistics system. It is classified as **Enterprise Mobility**.

4. **System 4 (Heart Rate Monitor):** This transmits critical health data over a wireless link for medical tracking. It is classified as **Emergency and Health Services**.

CHAPTER 4

Mobile Network and Transport Layer

The Mobile Network Layer and Transport Layer ensure that data reaches a mobile device correctly and efficiently as it moves across networks. In this chapter, we focus on the algorithms and procedural steps for routing, handover, and optimizing TCP for wireless conditions.

4.1 Packet Delivery Handover and Location Management

When a Mobile Node (MN) moves from its home network to a foreign network, standard IP routing fails because the IP address implies a physical location. Mobile IP solves this using a two-address mechanism and tunneling.

Methodology:

Mobile IP Packet Delivery Procedure The process of delivering a packet to a roaming mobile node involves the following steps:

1. **Addressing:** The Correspondent Node (CN) sends a packet to the Mobile Node (MN) using its standard Home Address.
2. **Interception:** The Home Agent (HA) intercepts the packet on the home network, knowing that the MN is currently registered at a Foreign Network.
3. **Encapsulation (Tunneling):** The HA wraps the original packet inside a new IP packet. The new destination address is the Care-of Address (CoA) provided by the Foreign Agent (FA).
4. **Decapsulation:** The FA receives the tunneled packet, removes the outer IP header, and delivers the original packet to the MN via the local wireless link.
5. **Reverse Path:** When the MN replies to the CN, it sends packets directly to the CN using standard IP routing, bypassing the HA. This is known as Triangle Routing.

Worked Example:

Tracing Mobile IP Packet Routing **Problem:** A server (CN) with IP 198.51.100.10 wants to send data to a Mobile Node whose Home Address is 203.0.113.50. The MN is currently in a foreign network with a Home Agent at 203.0.113.1 and a Foreign Agent providing a Care-of Address (CoA) 192.0.2.200. Trace the source and destination IP addresses at each stage of the forward and reverse packet delivery.
Solution:

1. **CN to HA (Original Packet):**
   - Source IP: 198.51.100.10 (CN)
   - Destination IP: 203.0.113.50 (MN Home Address)
2. **HA to FA (Tunneled Packet):**
   - Outer Source IP: 203.0.113.1 (HA)

- Outer Destination IP: 192.0.2.200 (FA CoA)
- Inner Source IP: 198.51.100.10 (CN)
- Inner Destination IP: 203.0.113.50 (MN)

3. **FA to MN (Decapsulated Packet):**
   - Source IP: 198.51.100.10 (CN)
   - Destination IP: 203.0.113.50 (MN)
4. **MN to CN (Reverse Path):**
   - Source IP: 203.0.113.50 (MN)
   - Destination IP: 198.51.100.10 (CN)

Methodology:

Handover and Location Management Steps Handover occurs when an MN changes its point of attachment (e.g. moving from one FA to another). Location management ensures the network knows where the MN is.

1. **Agent Discovery:** The MN listens for Agent Advertisement messages from routers or explicitly sends Agent Solicitation messages to find a new FA.
2. **Registration Request:** Upon moving to a new FA, the MN sends a Registration Request to the new FA.
3. **Forwarding Registration:** The FA forwards this request to the MN's HA.
4. **Binding Update:** The HA updates its mobility binding table, linking the MN's Home Address to the new CoA.
5. **Registration Reply:** The HA sends a Registration Reply back to the FA, which then forwards it to the MN, completing the handover.

Worked Example:

Calculating Handover Latency **Problem:** A Mobile Node moves to a new Foreign Agent. The one-way propagation delay between the MN and FA is 2 ms. The one-way delay between the FA and HA is 30 ms. Processing time at the FA and HA is 5 ms each. Calculate the total handover registration delay from the moment the MN sends the Registration Request until it receives the Registration Reply.
Solution:

1. Time for MN to send Request to FA: 2 ms
2. FA processing time: 5 ms
3. FA to HA transmission: 30 ms
4. HA processing time (updating binding): 5 ms
5. HA to FA Reply transmission: 30 ms
6. FA processing time for Reply: 5 ms
7. FA to MN transmission: 2 ms

Total Delay $= 2 + 5 + 30 + 5 + 30 + 5 + 2 = 79$ ms.

4.2 Differentiate Indirect TCP Snooping TCP and Mobile TCP

Standard TCP assumes all packet losses are due to network congestion, which is false in wireless networks where signal fading causes errors. Mobile transport protocols split or modify the connection to prevent unnecessary congestion control mechanisms.

Methodology:

Indirect TCP Algorithm Indirect TCP (I-TCP) splits the TCP connection into two distinct segments at the access point (Foreign Agent).

1. Establish a standard TCP connection between the Fixed Host (CN) and the Access Point (AP).
2. Establish a separate, optimized TCP connection over the wireless link between the AP and the Mobile Node (MN).
3. The AP acts as a proxy. It acknowledges packets from the CN immediately upon receipt, before they reach the MN.
4. If a packet is lost on the wireless link, the AP retransmits it locally without informing the CN.

Methodology:

Snooping TCP Algorithm Snooping TCP leaves the end-to-end TCP connection intact but adds a transparent proxy agent at the AP.

1. The connection remains end-to-end between CN and MN.
2. The AP "snoops" or monitors passing TCP packets and caches them.
3. The AP also snoops on returning ACKs from the MN.
4. If the AP detects duplicate ACKs (indicating a lost packet on the wireless link), it locally retransmits the cached packet to the MN.
5. The AP suppresses the duplicate ACKs to prevent the CN from triggering congestion control.

Methodology:

Mobile TCP Algorithm Mobile TCP (M-TCP) focuses on frequent disconnections and maintains end-to-end semantics.

1. The AP splits the connection but does not act as an independent proxy (unlike I-TCP).
2. When the AP detects a wireless disconnection (MN loses signal), it sends a TCP window size of 0 (Zero Window Advertisement) to the CN.
3. The CN enters a persistent standby state, freezing its timers and preventing congestion window reduction.
4. Once the MN reconnects, the AP reopens the window, and transmission resumes at full speed.

Worked Example:

Selecting the Appropriate TCP Mechanism **Problem:** You are designing a mobile network. Identify whether I-TCP, Snooping TCP, or M-TCP is best for the following scenarios:

1. A system requiring strict end-to-end semantics (the sender must know the receiver actually got the data) but suffering from minor wireless bit errors.

2. A system where the mobile device frequently passes through dead zones (tunnels) causing total loss of signal for several seconds.

3. A legacy fixed-network sender communicating with a highly constrained mobile device over a very slow, error-prone wireless link, requiring complete isolation from wireless delays.

Solution:

1. **Snooping TCP:** It maintains strict end-to-end semantics because the AP does not generate artificial ACKs. It handles minor bit errors well by locally retransmitting lost packets.

2. **M-TCP:** It is designed specifically to handle frequent, long disconnections by freezing the sender's state with a zero window advertisement, preventing timeout drops.

3. **Indirect TCP (I-TCP):** It completely isolates the fixed network from the wireless link. The AP handles all wireless retransmissions, shielding the fixed sender from the slow wireless performance.

Feature	Indirect TCP	Snooping TCP	Mobile TCP
Connection	Split connection	End-to-end	Split connection
End-to-End Semantics	Lost	Maintained	Maintained
Data Encryption	Fails	Works (Header snooping)	Fails
Primary Goal	Isolate wireless errors	Prevent sender backoff	Handle disconnections

4.3 TCP over 3G Mobile Networks

3G mobile networks (like UMTS and CDMA2000) introduce specific characteristics such as high data rates, large delay variations, and built-in link-layer error recovery (ARQ - Automatic Repeat reQuest). TCP must be tuned to operate efficiently over these networks.

Methodology:

TCP Optimization for 3G Networks Standard TCP can be optimized for 3G environments by modifying specific parameters:

1. **Large Initial Window:** Set the initial congestion window to a larger value (e.g. 2 to 4 segments) to reduce the time spent in the slow-start phase.

2. **Window Scale Option:** Enable TCP Window Scaling to support large bandwidth-delay products, preventing the sender from being limited by the default 64 KB window.

3. **Selective Acknowledgements (SACK):** Implement SACK so the receiver can acknowledge non-contiguous blocks of data, allowing the sender to retransmit only the missing packets rather than the whole window.

4. **Timestamp Option:** Use timestamps to accurately calculate the Round Trip Time (RTT), preventing spurious timeouts caused by the variable delays in 3G radio access networks.

Worked Example:

Calculating Bandwidth-Delay Product for 3G **Problem:** A 3G UMTS connection has an effective bandwidth of 2 Mbps (2×10^6 bps) and a Round Trip Time (RTT) of 150 ms. 1. Calculate the Bandwidth-Delay Product (BDP) in bytes. 2. Determine if standard TCP (maximum window size of 65,535 bytes) can fully utilize this link.
Solution:

1. **Calculate BDP:**

$$BDP = \text{Bandwidth} \times \text{RTT}$$

$$BDP = 2,000,000 \text{ bps} \times 0.150 \text{ s} = 300,000 \text{ bits}$$

Convert to bytes:

$$BDP = \frac{300,000}{8} = 37,500 \text{ Bytes}$$

2. **Analyze Utilization:** Since the calculated BDP ($37,500$ bytes) is less than the standard maximum TCP window size of $65,535$ bytes, standard TCP without window scaling is sufficient to fully utilize the 2 Mbps link at an RTT of 150 ms. However, if bandwidth increases to 4 Mbps, BDP becomes $75,000$ bytes, which would require the TCP Window Scale option.

CHAPTER 5
Technologies in Trends

This chapter focuses on the computational and systematic understanding of modern wireless communication technologies including Wireless LAN (WLAN), Bluetooth, and cellular network generations. We will explore the architectural components and operational methods for deploying these technologies.

5.1 Explain Architecture of WLAN

A Wireless Local Area Network (WLAN) uses radio communication to provide mobility to network users while maintaining connectivity to the wired network.

Methodology:

WLAN Architecture Components and Configuration The IEEE 802.11 standard defines the fundamental building blocks of a WLAN.

1. **Basic Service Set (BSS)**: The fundamental building block consisting of a single Access Point (AP) and associated wireless stations (STAs).
2. **Extended Service Set (ESS)**: A set of one or more interconnected BSSs forming a single logical network.
3. **Distribution System (DS)**: The backbone (usually wired) that connects multiple APs to form an ESS.
4. **Configuration Algorithm**:
   (a) Determine the coverage area and identify the number of APs required.
   (b) Assign non-overlapping frequency channels (e.g. Channels 1, 6, 11 in 2.4 GHz) to adjacent APs to minimize interference.
   (c) Connect APs to the Distribution System (switches/routers).
   (d) Configure a unified Service Set Identifier (SSID) across all APs for seamless roaming.

Worked Example:

WLAN Channel Allocation for Office Floor A company needs to deploy 3 Access Points (AP1, AP2, AP3) linearly across an office floor to ensure full coverage. The APs use the 2.4 GHz band. Assign appropriate channels to avoid co-channel interference.
Solution:

1. **Identify Available Channels:** In the 2.4 GHz band, the non-overlapping channels are 1, 6, and 11.
2. **Allocate Channel to AP1:** Assign Channel 1 to AP1.
3. **Allocate Channel to AP2:** AP2 is physically adjacent to AP1. To prevent interference, assign a non-overlapping channel. Assign Channel 6 to AP2.

4. **Allocate Channel to AP3:** AP3 is physically adjacent to AP2. It must not use Channel 6. Since AP1 is relatively far, we could reuse Channel 1, but using Channel 11 provides maximum separation. Assign Channel 11 to AP3.

Final Assignment: AP1 → Ch 1, AP2 → Ch 6, AP3 → Ch 11.

5.2 Explain Bluetooth

Bluetooth is a wireless technology standard for exchanging data over short distances using short-wavelength UHF radio waves.

Methodology:

Bluetooth Network Topology Formation Bluetooth networks are formed dynamically using ad-hoc connections.

1. **Piconet Formation**:
   (a) One device acts as the **Master**.
   (b) Up to 7 devices can act as **Active Slaves**.
   (c) Up to 255 devices can be in a **Parked** state.
   (d) Communication occurs exclusively between the Master and the Slaves.
2. **Scatternet Formation**:
   (a) Multiple piconets can overlap to form a Scatternet.
   (b) A device can participate in multiple piconets by time-multiplexing.
   (c) A device can be a Slave in multiple piconets or a Master in one and a Slave in another.

Worked Example:

Calculating Capacity of a Bluetooth Scatternet A Bluetooth scatternet consists of 4 interconnected piconets. Piconet A has 1 Master and 5 Slaves. Piconet B has 1 Master and 7 Slaves. Piconet C has 1 Master and 3 Slaves. Piconet D has 1 Master and 6 Slaves. One device acts as a Slave in both Piconet A and Piconet B. Another device acts as the Master of Piconet C and a Slave in Piconet D. Calculate the total number of unique active devices in this scatternet.
Solution:

1. **Count devices in individual piconets:**
   - Piconet A: 1 (Master) + 5 (Slaves) = 6 devices
   - Piconet B: 1 (Master) + 7 (Slaves) = 8 devices
   - Piconet C: 1 (Master) + 3 (Slaves) = 4 devices
   - Piconet D: 1 (Master) + 6 (Slaves) = 7 devices

   Total non-unique devices = $6 + 8 + 4 + 7 = 25$
2. **Identify and subtract overlapping devices (bridge nodes):**
   - One device is shared between A and B (counted twice). Subtract 1.
   - One device is shared between C and D (Master of C, Slave of D; counted twice). Subtract 1.
3. **Calculate total unique devices:** Total = $25 - 1 - 1 = 23$ unique devices.

5.3 Differentiate 4G 5G and 6G Mobile Network

The progression of mobile network generations introduces significant improvements in data rates, latency, and device density.

Methodology:

Network Performance Evaluation Algorithm To differentiate between mobile networks based on computational requirements:

1. **Data Rate (R)**: Identify the maximum theoretical throughput (measured in Gbps).
2. **Latency (L)**: Identify the end-to-end delay (measured in milliseconds ms).
3. **Connection Density (D)**: Determine the maximum number of supported devices per square kilometer.
4. **Selection Criteria**:
   - If $L \approx 10 - 50$ ms and $R \approx 1$ Gbps, select **4G (LTE-A)**.
   - If $L \approx 1$ ms, $R \approx 10 - 20$ Gbps, and $D \approx 10^6$ devices/km^2, select **5G**.
   - If $L < 0.1$ ms, $R \geq 100$ Gbps, and $D \approx 10^7$ devices/km^2, select **6G**.

Parameter	4G	5G	6G
Max Data Rate	1 Gbps	10-20 Gbps	100+ Gbps
Latency	10-50 ms	1 ms	< 0.1 ms
Frequency Band	2 to 8 GHz	3 to 100 GHz	95 GHz to 3 THz
Key Technology	MIMO OFDM	Massive MIMO	Terahertz Comm

Worked Example:

Selecting the Appropriate Mobile Network Generation A smart city initiative plans to deploy a network of autonomous vehicles and high-definition remote surgery stations. The autonomous vehicles require a connection density of 500,000 devices/km^2 and latency of no more than 2 ms. The remote surgery requires a data rate of 5 Gbps and latency of less than 1 ms. Determine the minimum network generation required to support both applications.

Solution:

1. **Analyze Requirements for Autonomous Vehicles:**
   - Density Requirement: 5×10^5 devices/km^2.
   - Latency Requirement: ≤ 2 ms.
   - 4G supports $\approx 10^5$ devices/km^2 and > 10 ms latency (Fails).
   - 5G supports 10^6 devices/km^2 and 1 ms latency (Passes).
2. **Analyze Requirements for Remote Surgery:**
   - Data Rate Requirement: 5 Gbps.
   - Latency Requirement: < 1 ms.
   - 4G max data rate is 1 Gbps and latency > 10 ms (Fails).
   - 5G provides up to 20 Gbps and 1 ms latency (Marginal/Passes depending on exact 5G release).
   - 6G provides > 100 Gbps and < 0.1 ms latency (Passes comfortably).
3. **Conclusion:** While 5G meets the autonomous vehicle density and comes close to the remote surgery requirements, strict sub-millisecond latency for critical medical operations generally mandates **5G**

URLLC (Ultra-Reliable Low-Latency Communication) or **6G**. Therefore, the minimum required network generation is **5G**.

Appendix: Key Formulae

This appendix provides a quick reference to the key abstract formulae and mathematical relationships discussed in this book, categorized by topic.

Network Performance and Delay

Transmission Delay The time required to push all the packet's bits into the link.

$$T_d = \frac{S}{B}$$

- T_d: Transmission delay (seconds)
- S: Packet or data size (bits)
- B: Bandwidth or link capacity (bits per second)

Token Bucket and Traffic Shaping

Maximum Burst Duration The maximum time a token bucket can sustain transmitting at the maximum rate.

$$S = \frac{C}{M - R} \quad \text{(for } M > R\text{)}$$

- S: Maximum burst duration (seconds)
- C: Bucket capacity (bytes or tokens)
- M: Maximum output rate
- R: Token arrival rate

Maximum Burst Data The total amount of data transmitted during the maximum burst period.

$$D = M \times S$$

- D: Maximum data transmitted (bytes)
- M: Maximum output rate
- S: Maximum burst duration (seconds)

Time to Overflow The time taken for a leaky bucket to overflow when input exceeds capacity.

$$t = \frac{C}{A - R}$$

- t: Time to overflow (seconds)
- C: Bucket capacity
- A: Average arrival rate of incoming data
- R: Output/processing rate of the bucket

Discarded Data The amount of data lost when the bucket overflows.

$$\text{Discarded Data} = (T - t) \times (A - R)$$

- T: Total duration of data arrival (seconds)
- t: Time to overflow (seconds)
- A: Average arrival rate
- R: Processing rate

File Distribution Architecture

Client-Server Distribution Time The minimum time required to distribute a file to N clients using a single server.

$$D_{CS} = \max\left(\frac{N \cdot F}{u_s}, \frac{F}{d_{\min}}\right)$$

- D_{CS}: Client-Server distribution time
- N: Number of clients
- F: File size
- u_s: Server upload capacity
- $d_{\min}$: Minimum download rate among all clients

Peer-to-Peer (P2P) Distribution Time The minimum time required to distribute a file to N peers in a P2P network.

$$D_{P2P} = \max\left(\frac{F}{u_s}, \frac{F}{d_{\min}}, \frac{N \cdot F}{u_s + \sum_{i=1}^{N} u_i}\right)$$

- D_{P2P}: Peer-to-Peer distribution time
- N: Number of peers
- F: File size
- u_s: Server/Initial seeder upload capacity
- $d_{\min}$: Minimum download rate among all peers
- u_i: Upload capacity of the i-th peer

Packet Structure and Overhead

Total Packet Size The total size of a transmitted packet including all layer headers.

$$S = P + H_t + H_n + H_d$$

- S: Total packet size
- P: Payload size
- H_t: Transport layer header size
- H_n: Network layer header size
- H_d: Data Link layer header size

Protocol Overhead The percentage of a packet consumed by headers.

$$O = \left(\frac{H_t + H_n + H_d}{S}\right) \times 100\%$$

- O: Overhead percentage
- H_t, H_n, H_d: Header sizes
- S: Total packet size

Bluetooth Piconet Network

Devices in a Piconet The composition of devices in a single Bluetooth piconet.

$$N_{\text{total}} = 1 \text{ (Master)} + N_{\text{slaves}}$$

- N_{total}: Total number of active devices in the piconet (maximum 8)
- N_{slaves}: Number of active slave devices (maximum 7)