Iis (DI01016021) - Problem Solver

Antigravity AI

June 4, 2026

Iis

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

Basics of Information System

1.1 Importance of Information Technology

Methodology:

IT Investment ROI Calculation To quantify the importance and value of Information Technology in a business environment we often calculate the Return on Investment (ROI) and Payback Period.

1. Identify Total Initial Investment (I): Sum of all costs for IT hardware software and training.
2. Identify Annual Net Profit (P): Expected revenue increase or cost savings generated annually due to the IT system.
3. Calculate Return on Investment (ROI) in percentage:

$$ROI = \left(\frac{P}{I}\right) \times 100$$

4. Calculate Payback Period in years:

$$\text{Payback Period} = \frac{I}{P}$$

Worked Example:

Calculate ROI for IT Infrastructure A company invests $50000 in new IT infrastructure to automate its workflow. This IT system generates an additional $15000 in net profit annually due to increased efficiency. Calculate the Return on Investment (ROI) and the Payback Period.

Solution:

1. Identify given values:
   - Initial Investment $I = 50000$
   - Annual Net Profit $P = 15000$
2. Calculate ROI:

$$ROI = \left(\frac{15000}{50000}\right) \times 100$$

$$ROI = 0.3 \times 100 = 30\%$$

3. Calculate Payback Period:

$$\text{Payback Period} = \frac{50000}{15000} = 3.33 \text{ years}$$

Therefore the ROI is 30% and the IT system will pay for itself in approximately 3.33 years.

1.2 Hardware Components

Methodology:

Storage Capacity and CPU Performance Calculation Evaluating hardware components involves calculating storage capacity in different digital units and estimating CPU instruction execution time based on clock frequency.

1. **Storage Conversion:**
   - 1 Byte (B) = 8 bits
   - 1 Kilobyte (KB) = 1024 Bytes
   - 1 Megabyte (MB) = 1024 KB
   - 1 Gigabyte (GB) = 1024 MB
2. **CPU Clock Cycle Time (T):**
$$T = \frac{1}{f}$$
   where f is the CPU clock frequency in Hertz (Hz).
3. **Execution Time (E):**
$$E = \text{Number of Clock Cycles} \times T$$

Worked Example:

Evaluate Hardware Specifications A computer has a 2.5 GHz processor and a 500 GB Hard Disk Drive. 1. Calculate the clock cycle time of the CPU in nanoseconds. 2. Determine the total hard disk capacity in Megabytes (MB).
Solution:

1. **Calculate Clock Cycle Time:**
   - CPU Frequency $f = 2.5$ GHz $= 2.5 \times 10^9$ Hz
   - Clock Cycle Time $T = \frac{1}{2.5 \times 10^9}$ seconds
   - $T = 0.4 \times 10^{-9}$ seconds $= 0.4$ ns
2. **Calculate Storage Capacity in MB:**
   - Given Capacity = 500 GB
   - Since 1 GB = 1024 MB
   - Capacity in MB $= 500 \times 1024 = 512000$ MB

The CPU clock cycle time is 0.4 nanoseconds and the storage capacity is 512000 MB.

1.3 Applications of Internet Digital Platforms

Methodology:

Network Transfer Time Estimation Digital platforms heavily rely on network data transfer. Estimating the time required to upload or download data over the internet is crucial for platform performance analysis.

1. Identify the File Size (S) in Megabytes (MB).
2. Identify the Network Bandwidth (B) in Megabits per second (Mbps).

3. Convert the File Size to Megabits (Mb):

$$S_{Mb} = S \times 8$$

4. Calculate Transfer Time (t) in seconds:

$$t = \frac{S_{Mb}}{B}$$

5. Convert Transfer Time t to minutes or hours if required (divide by 60 for minutes).

Worked Example:

Download Time for E-Learning Platform An e-learning digital platform requires a student to download a video lecture of size 450 MB. The student's internet connection has a stable bandwidth of 15 Mbps. Calculate the estimated time required to download the video in minutes.

Solution:

1. Identify given values:
   - File Size $S = 450$ MB
   - Bandwidth $B = 15$ Mbps
2. Convert File Size from Megabytes to Megabits:

$$S_{Mb} = 450 \times 8 = 3600 \text{ Mb}$$

3. Calculate Transfer Time in seconds:

$$t = \frac{3600}{15} = 240 \text{ seconds}$$

4. Convert Transfer Time to minutes:

$$t_{minutes} = \frac{240}{60} = 4 \text{ minutes}$$

It will take exactly 4 minutes to download the video lecture from the digital platform.

CHAPTER 2

Digital Logic

2.1 Introduction to Digital Computers and Number System

Digital computers operate using discrete signals representing binary states (0 and 1). Computational problems in digital logic frequently involve converting numbers between different bases, specifically Decimal (Base 10), Binary (Base 2), Octal (Base 8) and Hexadecimal (Base 16).

Methodology:

Decimal to Binary Conversion To convert a decimal integer to a binary number, use the repeated division-by-2 method:

1. Divide the given decimal number by 2.
2. Record the remainder (which will be either 0 or 1).
3. Use the integer quotient as the new dividend.
4. Repeat steps 1 through 3 until the quotient becomes 0.
5. The binary equivalent is formed by reading the remainders in reverse order (from the last remainder to the first).

Worked Example:

Convert Decimal 45 to Binary Convert the decimal number 45_{10} to its binary equivalent.
Step 1: Divide 45 by 2. $45 \div 2 = 22$ with remainder 1 (Least Significant Bit)
Step 2: Divide 22 by 2. $22 \div 2 = 11$ with remainder 0
Step 3: Divide 11 by 2. $11 \div 2 = 5$ with remainder 1
Step 4: Divide 5 by 2. $5 \div 2 = 2$ with remainder 1
Step 5: Divide 2 by 2. $2 \div 2 = 1$ with remainder 0
Step 6: Divide 1 by 2. $1 \div 2 = 0$ with remainder 1 (Most Significant Bit)
Step 7: Read the remainders from bottom to top. Result: 101101_2

Methodology:

Binary to Decimal Conversion To convert a binary number to decimal, expand the number into a sum of powers of 2:

1. Assign a positional weight to each bit, starting from 2^0 for the rightmost bit (Least Significant Bit), 2^1 for the next, and so on.
2. Multiply each binary digit (0 or 1) by its corresponding positional weight.
3. Sum all the resulting products to get the decimal equivalent.

Worked Example:

Convert Binary 110110 to Decimal Convert the binary number 110110_2 to its decimal equivalent.
Step 1: Identify the positional weights for each bit from right to left. The bits are 1, 1, 0, 1, 1, 0. The weights are $2^5, 2^4, 2^3, 2^2, 2^1, 2^0$.
Step 2: Multiply each bit by its weight and sum them up.

$$(1 \times 2^5) + (1 \times 2^4) + (0 \times 2^3) + (1 \times 2^2) + (1 \times 2^1) + (0 \times 2^0)$$

Step 3: Evaluate the powers of 2.

$$(1 \times 32) + (1 \times 16) + (0 \times 8) + (1 \times 4) + (1 \times 2) + (0 \times 1)$$

Step 4: Sum the results.

$$32 + 16 + 0 + 4 + 2 + 0 = 54_{10}$$

Methodology:

Binary to Octal and Hexadecimal Conversions Binary numbers can be directly converted to Octal (Base 8) or Hexadecimal (Base 16) by grouping bits:

1. **Binary to Octal:** Group the binary digits into sets of three, starting from the rightmost bit. If the leftmost group has fewer than three bits, pad it with leading zeros. Convert each 3-bit group to its corresponding decimal digit (0-7).
2. **Binary to Hexadecimal:** Group the binary digits into sets of four, starting from the rightmost bit. Pad the leftmost group with leading zeros if necessary. Convert each 4-bit group to its corresponding hex digit (0-9, A-F).

Worked Example:

Convert Binary 101110101 to Octal and Hexadecimal Convert 101110101_2 to Octal and Hexadecimal.
Part A: Binary to Octal Step 1: Group bits into threes from right to left: 101 110 101
Step 2: Convert each group to a decimal digit: $101_2 = 5_{10}$ $110_2 = 6_{10}$ $101_2 = 5_{10}$
Step 3: Combine digits: 565_8
Part B: Binary to Hexadecimal Step 1: Group bits into fours from right to left. Pad with leading zeros: 0001 0111 0101
Step 2: Convert each group to a hex digit: $0001_2 = 1_{10} = 1_{16}$ $0111_2 = 7_{10} = 7_{16}$ $0101_2 = 5_{10} = 5_{16}$
Step 3: Combine digits: 175_{16}

2.2 Working of Logic Gates

Logic gates are the basic building blocks of digital circuits. They perform fundamental logical operations based on Boolean algebra.

Methodology:

Evaluating Basic Logic Gate Expressions The three fundamental logic operations are AND, OR, and NOT.

1. **AND Gate ($\cdot$):** The output is 1 ONLY if all inputs are 1. Mathematically represented as $Y = A \cdot B$.
2. **OR Gate ($+$):** The output is 1 if AT LEAST ONE input is 1. Mathematically represented as $Y = A + B$.
3. **NOT Gate ($'$ or $\bar{A}$):** The output is the inversion of the input. Mathematically represented as $Y = A'$.

A	B	NOT A (A')	AND ($A \cdot B$)	OR ($A + B$)
0	0	1	0	0
0	1	1	0	1
1	0	0	0	1
1	1	0	1	1

Worked Example:

Evaluate Boolean Expression for Specific Inputs Given the Boolean expression $Y = (A \cdot B) + (A' \cdot C)$, evaluate the output Y when $A = 0$, $B = 1$, and $C = 1$.
Step 1: Substitute the given input values into the expression. $Y = (0 \cdot 1) + (0' \cdot 1)$
Step 2: Evaluate the NOT operation ($0'$). Since $0' = 1$, the expression becomes: $Y = (0 \cdot 1) + (1 \cdot 1)$
Step 3: Evaluate the AND operations. $(0 \cdot 1) = 0$ $(1 \cdot 1) = 1$ Substitute these results back into the expression: $Y = 0 + 1$
Step 4: Evaluate the OR operation. $0 + 1 = 1$
Result: The final output is $Y = 1$.

Methodology:

Evaluating XOR and XNOR Gates Exclusive gates are derived from basic gates and are heavily used in arithmetic circuits.

1. **XOR Gate ($\oplus$):** Output is 1 if the inputs are DIFFERENT (odd number of 1s). The Boolean expression is $Y = A \oplus B = A'B + AB'$.
2. **XNOR Gate ($\odot$):** Output is 1 if the inputs are THE SAME (even number of 1s). The Boolean expression is $Y = A \odot B = A'B' + AB$.

A	B	XOR ($A \oplus B$)	XNOR ($A \odot B$)
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

Worked Example:

Implement XOR Logic using Basic Gates Prove mathematically that $A'B + AB'$ behaves as an XOR gate by evaluating the output for $A = 1$, $B = 0$.
Step 1: Substitute the inputs $A = 1$ and $B = 0$ into the expression $A'B + AB'$. $Y = (1') \cdot 0 + 1 \cdot (0')$
Step 2: Evaluate the NOT operations. $1' = 0$ and $0' = 1$. Substitute these values: $Y = (0 \cdot 0) + (1 \cdot 1)$
Step 3: Evaluate the AND operations. $0 \cdot 0 = 0$ and $1 \cdot 1 = 1$. $Y = 0 + 1$
Step 4: Evaluate the OR operation. $Y = 1$
Conclusion: For inputs $A = 1$ and $B = 0$ (different inputs), the output is 1, which matches the behavior of an XOR gate.

2.3 Working of Universal Gates

NAND and NOR gates are called Universal Gates because any logical function, including the basic gates (AND, OR, NOT), can be implemented using only NAND gates or only NOR gates.

Methodology:

Implementing Basic Gates using NAND Gates A NAND gate is an AND gate followed by a NOT gate: $Y = (A \cdot B)'$. To build basic gates using only NAND:

1. **NOT Gate:** Tie all inputs of a NAND gate together. $A \cdot A = A$, so $(A \cdot A)' = A'$.

2. **AND Gate:** Pass the inputs through a NAND gate, then invert the result using a second NAND gate (configured as a NOT gate). $((A \cdot B)')' = A \cdot B$.

3. **OR Gate:** Invert both individual inputs A and B using two NAND gates. Pass these inverted signals (A' and B') into a third NAND gate. By De Morgan's Law: $(A' \cdot B')' = A + B$.

Worked Example:

Implement the OR Gate Operation using Only NAND Gates Show the step-by-step Boolean algebra proof that combining NAND gates produces an OR gate operation for inputs $A = 0$ and $B = 1$.
Step 1: Apply $A = 0$ to a NAND gate configured as a NOT gate. Output 1: $A' = (0 \cdot 0)' = 0' = 1$.
Step 2: Apply $B = 1$ to a second NAND gate configured as a NOT gate. Output 2: $B' = (1 \cdot 1)' = 1' = 0$.
Step 3: Feed Output 1 ($A' = 1$) and Output 2 ($B' = 0$) into a third NAND gate. Final Output $Y = (A' \cdot B')'$
$Y = (1 \cdot 0)'$
Step 4: Evaluate the final NAND operation. First, $1 \cdot 0 = 0$. Then, invert it: $0' = 1$.
Result: For inputs $A = 0$ and $B = 1$, the output $Y = 1$, which is the correct behavior for $0 + 1$ in an OR gate.

Methodology:

Implementing Basic Gates using NOR Gates A NOR gate is an OR gate followed by a NOT gate: $Y = (A+B)'$. To build basic gates using only NOR:

1. **NOT Gate:** Tie all inputs of a NOR gate together. $A + A = A$, so $(A + A)' = A'$.

2. **OR Gate:** Pass the inputs through a NOR gate, then invert the result using a second NOR gate (configured as a NOT gate). $((A + B)')' = A + B$.

3. **AND Gate:** Invert both individual inputs A and B using two NOR gates. Pass these inverted signals (A' and B') into a third NOR gate. By De Morgan's Law: $(A' + B')' = A \cdot B$.

Worked Example:

Implement the AND Gate Operation using Only NOR Gates Show the step-by-step Boolean algebra proof that combining NOR gates produces an AND gate operation for inputs $A = 1$ and $B = 1$.
Step 1: Apply $A = 1$ to a NOR gate configured as a NOT gate. Output 1: $A' = (1 + 1)' = 1' = 0$.
Step 2: Apply $B = 1$ to a second NOR gate configured as a NOT gate. Output 2: $B' = (1 + 1)' = 1' = 0$.
Step 3: Feed Output 1 ($A' = 0$) and Output 2 ($B' = 0$) into a third NOR gate. Final Output $Y = (A' + B')'$
$Y = (0 + 0)'$
Step 4: Evaluate the final NOR operation. First, $0 + 0 = 0$. Then, invert it: $0' = 1$.
Result: For inputs $A = 1$ and $B = 1$, the output $Y = 1$, which is the correct behavior for $1 \cdot 1$ in an AND gate.

CHAPTER 3

Operating System

3.1 General features of OS

An operating system (OS) provides several critical features, including process scheduling, memory management, file handling, and security. In modern computing, managing memory via paging is a vital computational feature of an OS.

Methodology:

Calculating Page Number and Offset One of the essential features of an OS is memory management using paging. Given a logical address and a fixed page size, the OS calculates the corresponding page number and the offset within that page.

1. **Identify Given Values:** Let the logical address be L (in bytes) and the page size be P (in bytes).
2. **Calculate Page Number (p):** Use integer division to find the page number:

$$p = \lfloor \frac{L}{P} \rfloor$$

3. **Calculate Offset (d):** Use the modulo operator to find the offset within the page:

$$d = L \pmod{P}$$

4. **Result:** The logical address L translates to Page p with Offset d.

Worked Example:

Memory Management Address Translation An operating system uses a page size of 1024 bytes. A process generates a logical address of 2500. Calculate the page number and the offset for this address.
Step 1: Identify Given Values
Logical address $L = 2500$
Page size $P = 1024$
Step 2: Calculate Page Number

$$p = \lfloor \frac{2500}{1024} \rfloor = \lfloor 2.441 \rfloor = 2$$

Step 3: Calculate Offset

$$d = 2500 \pmod{1024} = 2500 - (2 \times 1024) = 2500 - 2048 = 452$$

Step 4: Result
The logical address 2500 corresponds to Page number 2 with an Offset of 452.

Methodology:

Identifying OS Features via Function Mapping To determine the primary operating system feature required for a specific computing scenario, evaluate the task using a feature mapping algorithm:

1. **Analyze the User Request:** Identify the primary action required by the system.
2. **Map to Core Functions:**
   - Multiple applications simultaneously → **Multitasking/Process Management**.
   - Organizing data on a disk → **File Management**.
   - Allocating RAM to active processes → **Memory Management**.
   - Protecting data via passwords or access controls → **Security**.
   - Communicating with hardware components → **Device Management**.
3. **Select Primary Feature:** Choose the feature that best aligns with the core function identified.

Worked Example:

Feature Identification for Server Setup A company wants to set up a system that assigns strict permissions to different employee accounts to prevent unauthorized access to financial data. Identify the primary OS feature utilized here.

Step 1: Analyze the User Request

The system needs to assign strict permissions and prevent unauthorized access to data.

Step 2: Map to Core Functions

The action of protecting data and enforcing access permissions directly aligns with OS security rather than file organization or memory allocation.

Step 3: Select Primary Feature

The primary OS feature utilized is **Security**.

3.2 Types of OS

Operating systems are categorized based on their processing methods, user access, and time constraints. A common type is the Time-Sharing OS, where CPU time is divided among users.

Methodology:

Time-Sharing CPU Allocation In a Time-Sharing Operating System, the CPU allocates a fixed time quantum (slice) to each user or process in a round-robin fashion. To evaluate system efficiency, we calculate the CPU utilization.

1. **Identify Given Values:** Let N be the number of active processes. Let Q be the time quantum allocated per process. Let S be the context switch time (overhead).
2. **Calculate Total Cycle Time (T):** The time required to complete one full round for all processes is:
$$T = N \times (Q + S)$$
3. **Calculate CPU Utilization (U):** The percentage of time the CPU spends executing actual process work versus total cycle time:
$$U = \left(\frac{N \times Q}{T}\right) \times 100\%$$

Worked Example:

Calculating Time-Sharing CPU Utilization A Time-Sharing OS has 5 active user processes. The OS allocates a time quantum of 20 ms per process. Each context switch takes 5 ms. Calculate the total cycle time and the CPU utilization percentage.

Step 1: Identify Given Values

Number of processes $N = 5$

Time quantum $Q = 20$ ms

Context switch time $S = 5$ ms

Step 2: Calculate Total Cycle Time (T)

$$T = 5 \times (20 + 5) = 5 \times 25 = 125 \text{ ms}$$

Step 3: Calculate CPU Utilization (U)

$$U = \left(\frac{5 \times 20}{125}\right) \times 100\% = \left(\frac{100}{125}\right) \times 100\% = 0.8 \times 100\% = 80\%$$

Step 4: Result

The total cycle time is 125 ms and the CPU utilization is 80%.

Methodology:

Classification of OS Types To classify an operating system into its correct operational category, verify its characteristics against the following rule set:

1. **Assess Input Mode:** Does the system group similar jobs and process them without user interaction? If yes → **Batch Operating System**.
2. **Check Time Constraints:** Must the system respond within a strict time limit? If yes → **Real-Time Operating System**.
3. **Examine User Access:** Are multiple users sharing the system simultaneously using time slices? If yes → **Time-Sharing Operating System**.
4. **Determine Hardware Distribution:** Are multiple independent CPUs physically distributed but appearing as a single system? If yes → **Distributed Operating System**.

Worked Example:

Selecting the Appropriate OS Type A manufacturing plant uses a robotic arm to weld car parts on an assembly line. The robotic arm must react to sensor inputs within 5 milliseconds to prevent damaging the chassis. Classify the type of operating system required.

Step 1: Assess Input Mode

The system responds to real-time sensor inputs, not a batch of similar jobs.

Step 2: Check Time Constraints

The system must respond within a strict time limit (5 milliseconds). Failure to do so results in physical damage.

Step 3: Final Classification

Because the system has strict deadline requirements, it is classified as a **Real-Time Operating System**.

3.3 Proprietary and Open-source software

Software can be categorized into Proprietary or Open-Source based on its licensing, code availability, and redistribution rights. Decision-makers often evaluate the financial impact of these software models using Total Cost of Ownership (TCO).

Methodology:

Total Cost of Ownership Comparison When choosing between Proprietary and Open-Source software models, organizations compute the Total Cost of Ownership (TCO) over a given period (Y years).

1. **Identify Costs for Proprietary Software (TCO_P):** Let L be the initial license cost. Let M_P be the annual maintenance and support cost.

$$TCO_P = L + (M_P \times Y)$$

2. **Identify Costs for Open-Source Software (TCO_O):** Let I be the initial integration and setup cost (license is usually \$0). Let M_O be the annual support and training cost.

$$TCO_O = I + (M_O \times Y)$$

3. **Decision Rule:** Compare TCO_P and TCO_O. Select the software model with the lowest computed TCO.

Worked Example:

Comparing Proprietary and Open-Source TCO A company is evaluating two database solutions for a 5-year project.

- **Proprietary Solution:** Requires an initial license of \$10000 and annual support fees of \$2000.
- **Open-Source Solution:** The software is free to download, but initial setup and integration will cost \$4000. Annual training and community support management will cost \$3500.

Calculate the TCO for both and determine which is more cost-effective.
Step 1: Calculate Proprietary TCO
$L = 10000$, $M_P = 2000$, $Y = 5$

$$TCO_P = 10000 + (2000 \times 5) = 10000 + 10000 = \$20000$$

Step 2: Calculate Open-Source TCO
$I = 4000$, $M_O = 3500$, $Y = 5$

$$TCO_O = 4000 + (3500 \times 5) = 4000 + 17500 = \$21500$$

Step 3: Decision
Since \$20000 $<$ \$21500, the Proprietary solution has a lower TCO over 5 years and is recommended.

Methodology:

Evaluating Software Licensing Models To determine whether a software product is Proprietary or Open-Source, evaluate the licensing constraints using a Boolean scoring algorithm:

1. **Source Code Availability (S):** If source code is publicly accessible and modifiable, $S = 1$. If restricted, $S = 0$.
2. **Redistribution Rights (R):** If users are legally allowed to share or redistribute the software freely, $R = 1$. If prohibited, $R = 0$.
3. **Compute Total Score (T):** $T = S + R$
4. **Decision Rule:**
   - If $T = 2$, the software is primarily **Open-Source**.
   - If $T < 2$, the software is primarily **Proprietary**.

Worked Example:

Software Model Evaluation A team wants to adopt a new database called "DataVault Pro". The vendor provides compiled binaries but explicitly states in the EULA that reverse-engineering or sharing the software is illegal. Evaluate the software model.

Step 1: Evaluate Source Code Availability (S)

The vendor provides compiled binaries and forbids reverse-engineering. Source code is restricted. $S = 0$.

Step 2: Evaluate Redistribution Rights (R)

The EULA states that sharing the software is strictly illegal. $R = 0$.

Step 3: Compute Total Score (T)

$T = 0 + 0 = 0$.

Step 4: Decision Rule

Since $T = 0$ ($T < 2$), "DataVault Pro" is classified as **Proprietary** software.

CHAPTER 4

Information Communication and Networking

4.1 Basic terminology

In computer networking and data communication, analyzing the performance of a network requires calculating fundamental metrics such as data rate, bandwidth, and various delays.

Methodology:

Calculating Data Transmission Metrics To evaluate network performance, apply the following fundamental formulas:

1. **Transmission Delay (T_t)**: The time required to push all the bits of a packet onto the link.

$$T_t = \frac{L}{B}$$

   where L is the packet length in bits and B is the bandwidth or transmission rate in bits per second (bps).

2. **Propagation Delay (T_p)**: The time required for a bit to travel from the source to the destination.

$$T_p = \frac{d}{v}$$

   where d is the distance of the physical link and v is the propagation speed in the medium (typically 2×10^8 to 3×10^8 m/s).

3. **Total Latency**: Total time for a message to arrive at the destination.

$$\text{Latency} = T_t + T_p + \text{Queuing Delay} + \text{Processing Delay}$$

Worked Example:

Compute Latency for a Simple Network Calculate the total transmission and propagation delay for a 5 Megabyte file sent over a 100 Mbps link. The distance between the sender and receiver is 2000 km and the propagation speed is 2×10^8 m/s. Ignore queuing and processing delays.

Step 1: Convert units to bits and bits per second. File size $L = 5$ MB $= 5 \times 10^6$ Bytes $\times 8$ bits/Byte $= 40 \times 10^6$ bits. Bandwidth $B = 100$ Mbps $= 100 \times 10^6$ bps.

Step 2: Calculate Transmission Delay (T_t).

$$T_t = \frac{L}{B} = \frac{40 \times 10^6}{100 \times 10^6} = 0.4 \text{ seconds}$$

Step 3: Calculate Propagation Delay (T_p). Distance $d = 2000$ km $= 2 \times 10^6$ m.

$$T_p = \frac{d}{v} = \frac{2 \times 10^6}{2 \times 10^8} = 0.01 \text{ seconds}$$

Step 4: Calculate Total Latency.

$$\text{Latency} = 0.4 + 0.01 = 0.41 \text{ seconds} = 410 \text{ milliseconds}$$

4.2 Transmission media

Transmission media carry information from a sender to a receiver. They are categorized into guided (wired) and unguided (wireless) media. The data carrying capacity and signal strength degradation are key computational aspects.

Methodology:

Shannon Channel Capacity Formula The maximum theoretical data rate of a noisy channel is calculated using Shannon's theorem:

1. Identify the bandwidth of the channel B in Hertz (Hz).
2. Identify the Signal-to-Noise Ratio (SNR). If given in decibels (dB), convert it to an absolute ratio:

$$\text{SNR}_{dB} = 10 \log_{10}(\text{SNR}) \implies \text{SNR} = 10^{\frac{\text{SNR}_{dB}}{10}}$$

3. Calculate the capacity C in bits per second (bps) using the formula:

$$C = B \log_2(1 + \text{SNR})$$

Worked Example:

Calculate Maximum Data Rate of a Channel A transmission channel has a bandwidth of 3000 Hz and a Signal-to-Noise Ratio of 30 dB. Calculate the maximum theoretical data rate.
Step 1: Convert SNR from dB to absolute ratio.

$$30 = 10 \log_{10}(\text{SNR}) \implies \log_{10}(\text{SNR}) = 3 \implies \text{SNR} = 10^3 = 1000$$

Step 2: Apply Shannon's capacity formula.

$$C = B \log_2(1 + \text{SNR}) = 3000 \log_2(1 + 1000) = 3000 \log_2(1001)$$

Since $\log_2(1024) = 10$, $\log_2(1001)$ is approximately 9.97.

$$C \approx 3000 \times 9.97 = 29910 \text{ bps} \approx 29.9 \text{ kbps}$$

Methodology:

Attenuation in Transmission Media Attenuation measures the loss of signal strength as it propagates through a medium. It is typically expressed in decibels (dB).

1. Note the transmitted power P_{in} and the received power P_{out}.
2. Calculate attenuation A using the formula:

$$A = 10 \log_{10} \left(\frac{P_{in}}{P_{out}} \right)$$

3. For attenuation per kilometer, divide the total attenuation by the distance d in kilometers.

Worked Example:

Calculate Signal Attenuation A signal is transmitted with a power of 100 mW over a 5 km fiber optic cable. The received power is 1 mW. Calculate the total attenuation and the attenuation per kilometer.

Step 1: Calculate total attenuation. $P_{in} = 100$ mW, $P_{out} = 1$ mW.

$$A = 10 \log_{10} \left(\frac{100}{1} \right) = 10 \log_{10}(100) = 10 \times 2 = 20 \text{ dB}$$

Step 2: Calculate attenuation per kilometer. Distance $d = 5$ km.

$$\text{Attenuation per km} = \frac{20 \text{ dB}}{5 \text{ km}} = 4 \text{ dB/km}$$

4.3 OSI Model

The Open Systems Interconnection (OSI) model standardizes networking functions into seven layers. As data passes from upper to lower layers, each layer adds its own control information (header/trailer), increasing the total size of the transmitted data.

Methodology:

Header Encapsulation Calculation To calculate the total size of data transmitted over the physical medium and the protocol overhead:

1. Start with the application payload size D.
2. Add the header sizes for each layer: Transport (H_T), Network (H_N), and Data Link (H_D). Add Data Link trailer (T_D) if applicable.
3. Calculate the total transmitted size: $S = D + H_T + H_N + H_D + T_D$.
4. Calculate overhead percentage:

$$\text{Overhead } \% = \left(\frac{S - D}{S} \right) \times 100$$

Worked Example:

Compute Protocol Overhead for a Message An application generates a 400-byte message. It is sent using TCP (20-byte header), IP (20-byte header), and Ethernet (14-byte header and 4-byte trailer). Calculate the total transmitted frame size and the overhead percentage.

Step 1: Identify sizes. Payload $D = 400$ bytes. Headers: $H_T = 20$ bytes, $H_N = 20$ bytes, $H_D = 14$ bytes. Trailer: $T_D = 4$ bytes.

Step 2: Calculate total size.

$$S = 400 + 20 + 20 + 14 + 4 = 458 \text{ bytes}$$

Step 3: Calculate overhead percentage.

$$\text{Overhead } \% = \left(\frac{458 - 400}{458} \right) \times 100 = \left(\frac{58}{458} \right) \times 100 \approx 12.66\%$$

4.4 Network Topologies

Network topology defines the physical or logical arrangement of nodes. The choice of topology affects the number of links and ports required, which impacts cost and fault tolerance.

Methodology:

Calculating Links in Network Topologies To determine the number of physical connections (links) and ports required for n devices:

1. **Mesh Topology**: Every device connects to every other device.

$$\text{Links} = \frac{n(n-1)}{2}$$

$$\text{Ports per device} = n - 1$$

2. **Star Topology**: Every device connects to a central hub.

$$\text{Links} = n$$

3. **Ring Topology**: Each device connects to exactly two other devices, forming a closed loop.

$$\text{Links} = n$$

Worked Example:

Determine Hardware Requirements for Topologies A company needs to connect 15 computers. Calculate the number of cables (links) required if they use a Mesh topology versus a Star topology.
Step 1: Calculate links for Mesh topology. Here, $n = 15$.

$$\text{Links}_{\text{mesh}} = \frac{15 \times (15-1)}{2} = \frac{15 \times 14}{2} = 15 \times 7 = 105 \text{ cables}$$

Step 2: Calculate links for Star topology.

$$\text{Links}_{\text{star}} = n = 15 \text{ cables}$$

Step 3: Conclusion. A Mesh topology requires 105 cables, whereas a Star topology requires only 15 cables, making the Star topology significantly cheaper to wire.

4.5 Types of Networks

Networks are classified by their geographic span (LAN, MAN, WAN). At the network layer, large networks are logically divided into subnets to improve addressing efficiency and routing.

Methodology:

IP Address Subnetting Basics Given an IPv4 address and a subnet mask (or CIDR notation):

1. Convert the subnet mask to binary to find the number of network bits (N) and host bits (H). Note that $N + H = 32$.
2. **Network Address**: Perform a bitwise AND between the IP address and the subnet mask.
3. **Broadcast Address**: Set all host bits of the network address to 1.
4. **Valid Host Range**: From (Network Address +1) to (Broadcast Address −1).
5. **Number of Hosts**: Calculated as $2^H - 2$.

Worked Example:

Calculate Subnet Parameters Given the IP address 192.168.1.130 with a subnet mask of 255.255.255.192 (or /26), calculate the network address, broadcast address, and number of valid hosts.
Step 1: Determine Network and Host bits. Subnet mask 255.255.255.192 in binary is: 11111111.11111111.11111111.11000000 This gives $N = 26$ and $H = 6$.
Step 2: Calculate Number of Hosts.

$$\text{Valid Hosts} = 2^6 - 2 = 64 - 2 = 62$$

Step 3: Calculate Network Address. Focus on the 4th octet. IP octet is 130 (10000010_2). Mask octet is 192 (11000000_2). Bitwise AND: 10000010 AND 11000000 = $10000000_2 = 128$. Network Address: 192.168.1.128.
Step 4: Calculate Broadcast Address. Set the 6 host bits to 1 in the 4th octet: $10111111_2 = 128 + 63 = 191$. Broadcast Address: 192.168.1.191.
Step 5: Define Valid Host Range. From 192.168.1.129 to 192.168.1.190.

4.6 Internet and Intranet

The Internet is a global network of networks, while an Intranet is a private network. IP addressing is fundamental to navigating both. Legacy classful addressing provides a basic framework for identifying network sizes based on the first octet.

Methodology:

IP Address Class Identification To identify the class of an IPv4 address and its default subnet mask:

1. Look at the first octet (the first decimal number) of the IP address.
2. Class A: 1 to 126 (Default Mask: 255.0.0.0)
3. Class B: 128 to 191 (Default Mask: 255.255.0.0)
4. Class C: 192 to 223 (Default Mask: 255.255.255.0)
5. Class D: 224 to 239 (Used for Multicasting)
6. Class E: 240 to 255 (Experimental)

Worked Example:

Identify IP Address Classes Determine the class and default subnet mask for the following IP addresses: (a) 10.5.2.1 (b) 172.16.0.50 (c) 192.168.1.1
Step 1: Analyze IP (a) 10.5.2.1 First octet is 10. Since 10 falls in the range $1 - 126$, this is a **Class A** address. Default Subnet Mask: 255.0.0.0.
Step 2: Analyze IP (b) 172.16.0.50 First octet is 172. Since 172 falls in the range $128 - 191$, this is a **Class B** address. Default Subnet Mask: 255.255.0.0.
Step 3: Analyze IP (c) 192.168.1.1 First octet is 192. Since 192 falls in the range $192 - 223$, this is a **Class C** address. Default Subnet Mask: 255.255.255.0.

4.7 Networking Devices

Networking devices such as hubs, switches, and routers segment networks differently. Understanding how they manage traffic mathematically defines the capacity and performance of the network.

Methodology:

Determining Broadcast and Collision Domains To count domains in a network topology, apply the behavior of each device:

1. **Hub**: All ports on a hub belong to exactly 1 Collision Domain and 1 Broadcast Domain.
2. **Switch**: Every port on a switch is a separate Collision Domain. All ports belong to 1 Broadcast Domain.
3. **Router**: Every port on a router is a separate Collision Domain and a separate Broadcast Domain.

Worked Example:

Count Domains for a Network Layout A network consists of a Router with two interfaces. Interface A connects to a 10-port Switch, and Interface B connects to a 5-port Hub. There is a computer connected to every available port on the switch and the hub. Determine the total number of collision domains and broadcast domains.

Step 1: Count Broadcast Domains. Routers break broadcast domains. The router has 2 active interfaces. Therefore, there are **2 Broadcast Domains**.

Step 2: Count Collision Domains on the Hub network. A hub does not break collision domains. All 5 computers connected to the hub, plus the router interface, share **1 Collision Domain**.

Step 3: Count Collision Domains on the Switch network. A switch creates a separate collision domain per port. The switch connects to 10 computers and 1 router interface (using 11 ports in total). However, only the links between the devices constitute the collision domains. - 10 links from the switch to computers = 10 collision domains. - 1 link from the switch to the router = 1 collision domain. Total for this segment = **11 Collision Domains**.

Step 4: Calculate Total Collision Domains. Total = 1 (Hub segment) + 11 (Switch segment) = **12 Collision Domains**.

Methodology:

Evaluating Device Capacity The backplane capacity (or switch fabric capacity) defines how much data a switch can process simultaneously across all its ports.

1. Note the number of ports N and the bandwidth of each port B.
2. Because switches support Full-Duplex communication (sending and receiving simultaneously), each port can handle $2 \times B$ data rate.
3. Calculate total capacity:

$$\text{Total Capacity} = N \times B \times 2$$

Worked Example:

Calculate Switch Backplane Capacity A network switch has 24 ports, and each port operates at Gigabit speed (1000 Mbps). Calculate the required backplane capacity to support non-blocking wire-speed forwarding on all ports simultaneously.

Step 1: Identify parameters. Number of ports $N = 24$. Port bandwidth $B = 1000$ Mbps $= 1$ Gbps.

Step 2: Calculate full-duplex capacity per port. Capacity per port $= 2 \times 1$ Gbps $= 2$ Gbps.

Step 3: Calculate total backplane capacity.

$$\text{Total Capacity} = 24 \times 2 \text{ Gbps} = 48 \text{ Gbps}$$

The switch needs a backplane capacity of 48 Gbps to operate without blocking traffic.

CHAPTER 5

IP Addressing Scheme and DNS

5.1 Network Addressing (IPv4) & Frame Format

Methodology:

Subnetting an IPv4 Network Subnetting divides a single large network into multiple smaller logical networks (subnets). To compute subnet parameters from a given IP address and CIDR prefix (e.g. /26):

1. **Identify the default mask**: Based on the IP class (A=/8, B=/16, C=/24).
2. **Calculate Subnet Bits (n)**: n = CIDR Prefix − Default Prefix.
3. **Number of Subnets**: 2^n.
4. **Calculate Host Bits (h)**: $h = 32 -$ CIDR Prefix.
5. **Number of Valid Hosts per Subnet**: $2^h - 2$ (subtracting Network and Broadcast addresses).
6. **Block Size**: 256 − subnet mask value in the interesting octet.

Worked Example:

Calculate subnets and host ranges for 192.168.10.0/27 **Step 1: Identify Class and Default Mask**
The address starts with 192 which is Class C. Default prefix is /24.
Step 2: Calculate Subnet Bits (n)
$n = 27 - 24 = 3$ bits borrowed for subnetting.
Number of subnets $= 2^3 = 8$ subnets.
Step 3: Calculate Host Bits (h)
$h = 32 - 27 = 5$ bits for hosts.
Valid hosts per subnet $= 2^5 - 2 = 32 - 2 = 30$ hosts.
Step 4: Block Size and Ranges
The mask for /27 is 255.255.255.224.
Block size $= 256 - 224 = 32$.
The subnets increment by 32 in the fourth octet:

- Subnet 0: 192.168.10.0 (Range: 192.168.10.1 to 192.168.10.30) Broadcast: 192.168.10.31
- Subnet 1: 192.168.10.32 (Range: 192.168.10.33 to 192.168.10.62) Broadcast: 192.168.10.63
- Subnet 2: 192.168.10.64 (Range: 192.168.10.65 to 192.168.10.94) Broadcast: 192.168.10.95

Methodology:

IPv4 Fragmentation Calculations When an IPv4 packet exceeds the Maximum Transmission Unit (MTU) of a network link it must be fragmented.

1. **Identify Payload**: Subtract the standard IP header (20 bytes) from the total packet size to get the

original data payload.

2. **Determine Max Fragment Payload**: MTU − 20 bytes. This value must be a multiple of 8. If it is not then round down to the nearest multiple of 8.
3. **Calculate Number of Fragments**: Divide the original payload by the max fragment payload and round up.
4. **Calculate Offset**: For any fragment k (starting at 0), Offset = $(k \times \text{Fragment Payload})/8$.
5. **More Fragments (MF) Flag**: MF = 1 for all fragments except the very last one (MF = 0).

Worked Example:

Fragment a 4000 byte IPv4 packet over an MTU of 1500 bytes **Step 1: Identify Payload**
Original Packet Size = 4000 bytes. Header = 20 bytes.
Original Payload = 4000 − 20 = 3980 bytes.
Step 2: Determine Max Fragment Payload
MTU = 1500 bytes. Header = 20 bytes.
Max Fragment Payload = 1500 − 20 = 1480 bytes.
Check if 1480 is a multiple of 8: 1480/8 = 185. (Yes it is).
Step 3: Fragments Needed
Fragment 1 Payload = 1480 bytes
Fragment 2 Payload = 1480 bytes
Remaining Payload = 3980 − 1480 − 1480 = 1020 bytes.
Fragment 3 Payload = 1020 bytes.
Step 4: Calculate Offsets and Flags

- **Fragment 1:** Total Size = 1500, Offset = 0, MF = 1.
- **Fragment 2:** Total Size = 1500, Offset = 1480/8 = 185, MF = 1.
- **Fragment 3:** Total Size = 1020 + 20 = 1040, Offset = (1480 + 1480)/8 = 370, MF = 0.

5.2 Comparison IPv4 & IPv6

Methodology:

IPv6 Address Compression Rules IPv6 addresses are 128 bits long and represented as eight groups of four hexadecimal digits. To simplify and shorten them:

1. **Leading Zeros**: Within any 16-bit block (four hex digits) leading zeros can be omitted. E.g. `00AB` becomes `AB`. `0000` becomes `0`.
2. **Double Colon**: A contiguous sequence of two or more blocks of zeros can be replaced by a double colon (`::`). This rule can only be applied **once** per address to avoid ambiguity.

Worked Example:

Compress the IPv6 address 2001:0DB8:0000:0000:0008:0800:200C:417A **Step 1: Omit Leading Zeros**
Break down into 8 blocks:

- `2001` → `2001`
- `0DB8` → `DB8`
- `0000` → `0`

- `0000` → `0`
- `0008` → `8`
- `0800` → `800` (trailing zeros are NOT omitted)
- `200C` → `200C`
- `417A` → `417A`

Intermediate result: `2001:DB8:0:0:8:800:200C:417A`
Step 2: Apply Double Colon
Find the longest contiguous sequence of zero blocks. Here we have two back-to-back zero blocks (`0:0`). Replace them with `::`.
Final Compressed Address: `2001:DB8::8:800:200C:417A`

Summary Comparison Table (IPv4 vs IPv6)

Feature	IPv4	IPv6
Address Length	32 bits	128 bits
Format	Dotted Decimal	Hexadecimal
Address Space	2^{32} (approx 4.3 billion)	2^{128} (undepletable)
Header Size	20 to 60 bytes (variable)	40 bytes (fixed)
IPsec Support	Optional	Built-in
Configuration	Manual or DHCP	SLAAC or DHCPv6

5.3 DNS (Domain Name System)

Methodology:

DNS Resolution and FQDN Constraints DNS translates human-readable domain names into IP addresses. Fully Qualified Domain Names (FQDNs) follow strict sizing constraints that can be calculated:

1. **Maximum Length of FQDN**: The absolute total length of a domain name cannot exceed 253 characters (excluding the final dot).
2. **Maximum Label Length**: Each label (separated by dots) can have a maximum of 63 characters.
3. **Valid Characters**: Labels can only contain letters (A-Z or a-z) digits (0-9) and hyphens (-). Hyphens cannot be at the start or end of a label.

To determine if a domain name is valid for DNS query formulation verify each label length and the aggregate sum including dots.

Worked Example:

Validate the FQDN test-server.subdomain.example.com **Step 1: Validate individual label lengths** Split the FQDN into labels using the dots as separators:

- Label 1: `test-server` (11 characters) ≤ 63 (Valid)
- Label 2: `subdomain` (9 characters) ≤ 63 (Valid)
- Label 3: `example` (7 characters) ≤ 63 (Valid)
- Label 4: `com` (3 characters) ≤ 63 (Valid)

Step 2: Validate total FQDN length
Sum of label characters $= 11 + 9 + 7 + 3 = 30$.
Number of dots $= 3$.
Total length $= 30 + 3 = 33$ characters.
Since $33 \leq 253$ the FQDN length is mathematically valid.
Step 3: Check character rules
All characters are letters and hyphens. No label starts or ends with a hyphen. The DNS resolution constraints are completely satisfied.

CHAPTER 6

Information Security

6.1 Need for Information Security

Information security is fundamentally about protecting data and systems from unauthorized access or disruption. In practice, the need for security is evaluated using computational risk models and financial metrics to justify security investments.

Methodology:

Return on Security Investment Calculation To mathematically justify the need for information security, organizations use the Return on Security Investment (ROSI) model.

1. **Calculate Single Loss Expectancy (SLE):** The financial loss from a single security incident.
2. **Calculate Annualized Rate of Occurrence (ARO):** The estimated number of times the incident occurs per year.
3. **Calculate Annualized Loss Expectancy (ALE):**

$$ALE = SLE \times ARO$$

4. **Calculate Mitigated Risk:** Multiply the ALE by the percentage of risk mitigated by the proposed security solution.
5. **Calculate ROSI:**

$$ROSI = \frac{\text{Mitigated Risk} - \text{Cost of Solution}}{\text{Cost of Solution}} \times 100\%$$

Worked Example:

Calculating ROSI for Data Breach Mitigation An IT company faces a projected 2 data breaches per year (ARO = 2). The estimated cost per breach is \$50000. The company plans to buy an Intrusion Detection System (IDS) that costs \$20000 annually and reduces the risk of breaches by 80%. Calculate the ROSI.

Step 1: Calculate the expected loss without the IDS.

$$SLE = 50000$$

$$ARO = 2$$

$$ALE = SLE \times ARO = 50000 \times 2 = 100000$$

Step 2: Calculate the mitigated risk. The IDS reduces the risk by 80%.

$$\text{Mitigated Risk} = 100000 \times 0.80 = 80000$$

Step 3: Calculate ROSI. The cost of the solution is \$20000.

$$ROSI = \frac{80000 - 20000}{20000} = \frac{60000}{20000} = 3$$

Expressed as a percentage, the ROSI is 300%. For every dollar spent on the IDS, the company saves three dollars in breach damages.

6.2 The Principles of Security

The core principles of information security form the CIA Triad: Confidentiality, Integrity, and Availability. These principles can be quantified and assessed using availability formulas and vulnerability scoring matrices.

Methodology:

System Availability and Quantitative Risk Assessment Availability is measured as the percentage of time a system is operational. Risk Assessment assigns numerical values to threat matrices.

1. **Calculate System Availability (Uptime Percentage):**

$$\text{Availability } (\%) = \left(\frac{\text{Total Uptime}}{\text{Total Uptime} + \text{Total Downtime}} \right) \times 100$$

2. **Calculate Quantitative Risk Score:**

$$\text{Risk Score} = \text{Asset Value} \times \text{Threat Probability} \times \text{Vulnerability Factor}$$

Where Threat Probability and Vulnerability Factor are values between 0.0 and 1.0.

Worked Example:

Assessing Availability for a Critical Web Server A web server is required to maintain 99.9% availability over a 30-day month (720 hours). It experiences an unexpected downtime of 45 minutes due to a configuration error. Determine the actual availability and check if it meets the principle of availability requirement.
Step 1: Convert all times to identical units (minutes).

$$\text{Total Time} = 720 \text{ hours} \times 60 \text{ minutes/hour} = 43200 \text{ minutes}$$

$$\text{Downtime} = 45 \text{ minutes}$$

$$\text{Uptime} = \text{Total Time} - \text{Downtime} = 43200 - 45 = 43155 \text{ minutes}$$

Step 2: Calculate Availability Percentage.

$$\text{Availability } (\%) = \left(\frac{43155}{43200} \right) \times 100 = 99.8958\%$$

Step 3: Compare with requirements. The calculated availability is 99.896%, which falls slightly below the required 99.9%. The system fails the required availability principle.

6.3 Cyber attacks

Cyber attacks exploit vulnerabilities to compromise systems. Understanding these attacks requires calculating password strength against brute-force attacks and assessing network bandwidth consumption during Denial of Service (DoS) attacks.

Methodology:

Password Entropy and Brute Force Attack Analysis Password entropy measures the computational difficulty required to guess a password.

1. **Determine Character Pool Size (R):** Number of possible characters (e.g. 26 for lowercase letters, 10 for digits, 62 for alphanumeric).

2. **Calculate Password Entropy (E):**

$$E = L \times \log_2(R)$$

Where L is the length of the password.

3. **Calculate Search Space (Total Combinations C):**

$$C = R^L$$

4. **Calculate Maximum Time to Crack (T):**

$$T = \frac{C}{\text{Guesses per second}}$$

Worked Example:

Estimating Brute Force Attack Duration An attacker targets an employee's 6-character password which consists entirely of numeric digits. The attacker's hardware can process 10^4 password guesses per second. Calculate the password entropy and the maximum time required to crack it.

Step 1: Identify Pool Size and Length.

$$R = 10 \text{ (digits 0-9)}$$

$$L = 6$$

Step 2: Calculate Total Combinations (C).

$$C = 10^6 = 1000000$$

Step 3: Calculate Time to Crack (T).

$$T = \frac{1000000}{10000} = 100 \text{ seconds}$$

Step 4: Calculate Entropy (E).

$$E = 6 \times \log_2(10) \approx 6 \times 3.32 = 19.92 \text{ bits}$$

With an entropy of just under 20 bits, this password is highly vulnerable to brute-force cyber attacks.

Methodology:

Distributed Denial of Service Traffic Calculation A DDoS attack floods a network with malicious traffic. We evaluate its impact by calculating the total bandwidth generated.

1. **Calculate Aggregate Packet Rate (P):**

$$P = \text{Number of Botnet Nodes} \times \text{Packets per Node per Second}$$

2. **Calculate Attack Bandwidth (B):**

$$B = P \times \text{Average Packet Size in Bits}$$

3. **Convert Bandwidth to Mbps or Gbps:** Divide by 10^6 for Mbps, or 10^9 for Gbps.

Worked Example:

Mitigating a Botnet UDP Flood Attack A botnet composed of 5000 infected IoT devices targets a university network. Each device sends 200 UDP packets per second, with an average packet size of 1500 bytes. The university firewall can handle a maximum of 10 Gbps of traffic. Determine if the firewall will be overwhelmed.

Step 1: Calculate Aggregate Packet Rate.

$$P = 5000 \times 200 = 1000000 \text{ packets per second}$$

Step 2: Convert Packet Size to Bits.

$$\text{Size in bits} = 1500 \text{ bytes} \times 8 \text{ bits/byte} = 12000 \text{ bits}$$

Step 3: Calculate Total Attack Bandwidth.

$$B = 1000000 \times 12000 = 12000000000 \text{ bits per second}$$

Step 4: Convert to Gbps and Compare.

$$B \text{ in Gbps} = \frac{12 \times 10^9}{10^9} = 12 \text{ Gbps}$$

The attack generates 12 Gbps of traffic, which exceeds the firewall's 10 Gbps maximum capacity. The system will be overwhelmed.

6.4 Cyber Law

Cyber Law outlines the legal consequences and penalty calculations for cybercrimes. A rule-based adjudication method evaluates the impact of an incident and maps it to specific statutory penalties.

Methodology:

Cyber Incident Penalty Adjudication Algorithm We use an algorithmic decision matrix to assess damages under standard IT Law frameworks (e.g. Information Technology Act).

1. **Quantify Direct Data Loss Value (D):** Number of records compromised × Unit cost per record.
2. **Quantify Operational Disruption Cost (O):** Hours of downtime × Hourly operational cost.
3. **Determine the Base Fine (F_{base}):** Assigned by the specific Cyber Law Section violated (e.g. Section 43 for unauthorized access).
4. **Calculate Total Penalty Liability (P_{total}):**

$$P_{total} = F_{base} + D + O$$

Worked Example:

Calculating Cyber Law Penalties for Data Theft An insider illegally accesses a database, violating Cyber Law norms against unauthorized access (which carries a base fine of $5000). The attacker steals 2000 customer records. Industry regulators value a compromised record at $15. Furthermore, investigating the breach forces the database to go offline for 4 hours, during which the business loses $1200 per hour. Calculate the total penalty liability imposed on the attacker.

Step 1: Quantify Direct Data Loss Value (D).

$$D = 2000 \text{ records} \times \$15/\text{record} = \$30000$$

Step 2: Quantify Operational Disruption Cost (O).

$$O = 4 \text{ hours} \times \$1200/\text{hour} = \$4800$$

Step 3: Identify Base Fine (F_{base}).

$$F_{base} = \$5000$$

Step 4: Calculate Total Penalty Liability (P_{total}).

$$P_{total} = \$5000 + \$30000 + \$4800 = \$39800$$

The total penalty liability assessed against the attacker under cyber law frameworks is \$39800.

Appendix: Key Formulae

This appendix provides a quick reference to the key formulae and mathematical relationships discussed in this book.

Unit 1: System Performance and Financials

- **Execution Time (E):**
$$E = \text{Number of Clock Cycles} \times T$$
Where T is the clock period.
- **Clock Period (T):**
$$T = \frac{1}{f}$$
Where f is the clock frequency.
- **File Size in Megabits (S_{Mb}):**
$$S_{Mb} = S \times 8$$
Where S is the file size in Megabytes.
- **Transfer Time (t):**
$$t = \frac{S_{Mb}}{B}$$
Where S_{Mb} is the size in Megabits and B is the network bandwidth.
- **Return on Investment (ROI):**
$$ROI = \left(\frac{P}{I}\right) \times 100$$
Where P is the net profit or savings and I is the initial investment.
- **Payback Period:**
$$\text{Payback Period} = \frac{I}{P}$$
Where I is the initial investment and P is the periodic (e.g., annual) cash flow or savings.

Unit 2: Boolean Algebra and Digital Logic

- **Logical OR:**
$$Y = A + B$$
- **Sum of Products Expression:**
$$Y = (A \cdot B) + (A' \cdot C)$$
- **De Morgan's Theorem / NOR Gate:**
$$Y = (A' \cdot B')'$$
- **Idempotent Law (with Complement):**
$$(A \cdot A)' = A'$$

Unit 3: Cloud Computing and Storage

- **Total Cost of Ownership (On-Premises) (TCO_O):**

$$TCO_O = I + (M_O \times Y)$$

Where I is the initial hardware cost, M_O is the annual maintenance cost, and Y is the number of years.

- **Total Cost of Ownership (Cloud/Provider) (TCO_P):**

$$TCO_P = L + (M_P \times Y)$$

Where L is the upfront licensing or lease cost, and M_P is the annual subscription or maintenance cost.

- **Paging – Page Number (p) and Offset (d):**

$$p = \lfloor \frac{L}{P} \rfloor, \quad d = L \pmod{P}$$

Where L is the logical address and P is the page size.

- **Task Execution Time (T):**

$$T = N \times (Q + S)$$

Where N is the number of tasks, Q is the execution quantum/time per task, and S is the context switching or setup overhead.

- **Resource Utilization (U):**

$$U = \left(\frac{N \times Q}{T} \right) \times 100\%$$

Where $N \times Q$ represents the total useful processing time and T is the total elapsed time.

Unit 4: Computer Networks and Data Communication

- **Mesh Topology Links:**

$$\text{Links} = \frac{n(n-1)}{2}$$

Where n is the number of nodes. The number of ports required per device is $n - 1$.

- **Star Topology Links:**

$$\text{Links} = n$$

- **Network Latency:**

$$\text{Latency} = T_t + T_p + \text{Queuing Delay} + \text{Processing Delay}$$

- **Transmission Delay (T_t):**

$$T_t = \frac{L}{B}$$

Where L is the message length (bits) and B is the bandwidth (bps).

- **Propagation Delay (T_p):**

$$T_p = \frac{d}{v}$$

Where d is the distance and v is the signal propagation velocity.

- **Total Network Capacity (Full-Duplex):**

$$\text{Total Capacity} = N \times B \times 2$$

Where N is the number of bidirectional links.

- **Protocol Overhead Percentage:**

$$\text{Overhead } \% = \left(\frac{S - D}{S}\right) \times 100$$

Where S is the total frame size and D is the payload data size.

- **Shannon Channel Capacity (C):**

$$C = B \log_2(1 + \text{SNR})$$

Where B is the bandwidth (Hz) and SNR is the Signal-to-Noise Ratio.

- **Signal-to-Noise Ratio (in dB):**

$$\text{SNR}_{dB} = 10 \log_{10}(\text{SNR}) \implies \text{SNR} = 10^{\frac{\text{SNR}_{dB}}{10}}$$

- **Signal Attenuation (A):**

$$A = 10 \log_{10}\left(\frac{P_{in}}{P_{out}}\right)$$

Where P_{in} is the input power and P_{out} is the output power.

Unit 5: IP Addressing and Subnetting

- **Valid Hosts per Subnet:**

$$\text{Valid Hosts} = 2^h - 2$$

Where h is the number of host bits ($32 - \text{Prefix Length}$).

- **Borrowed Subnet Bits (n):**

$$n = \text{CIDR Prefix} - \text{Default Prefix}$$

- **IP Fragmentation Offset:**

$$\text{Offset} = \frac{k \times \text{Fragment Payload}}{8}$$

Where k is the fragment index (starting at 0 for the first fragment).

Unit 6: Information Security and Risk Management

- **Password Entropy (E):**

$$E = L \times \log_2(R)$$

Where L is the password length and R is the size of the character set.

- **Password Combinations (C):**

$$C = R^L$$

- **Time to Crack Password (T):**

$$T = \frac{C}{\text{Guesses per second}}$$

- **DDoS Attack Volume:**

$$P = \text{Number of Botnet Nodes} \times \text{Packets per Node per Second}$$

$$B = P \times \text{Average Packet Size in Bits}$$

Where P is the total packets per second and B is the total attack bandwidth.

- **Total Breach Cost (P_{total}):**

$$P_{total} = F_{base} + D + O$$

  Where F_{base} is the base regulatory fine, D is the cost of compromised data records, and O represents other operational and mitigation costs.

- **Annualized Loss Expectancy (ALE):**

$$ALE = SLE \times ARO$$

  Where SLE is the Single Loss Expectancy and ARO is the Annualized Rate of Occurrence.

- **Return on Security Investment (ROSI):**

$$ROSI = \left(\frac{\text{Mitigated Risk} - \text{Cost of Solution}}{\text{Cost of Solution}} \right) \times 100\%$$

- **Risk Score:**

$$\text{Risk Score} = \text{Asset Value} \times \text{Threat Probability} \times \text{Vulnerability Factor}$$

- **System Availability:**

$$\text{Availability (\%)} = \left(\frac{\text{Total Uptime}}{\text{Total Uptime} + \text{Total Downtime}} \right) \times 100$$