INDEX

PREFACE

This book, "Advanced Data Structures Lab Manual", is designed to support postgraduate students in understanding, designing, and analyzing complex data structures and their applications in modern computing. Readers of this manual are expected to have fundamental knowledge of basic data structures and programming concepts. The study of Advanced Data Structures has become essential in the field of Computer Science & Engineering, as it forms the backbone of efficient algorithm design, large-scale system development, and high-performance applications.

To facilitate the development of strong analytical and implementation skills, this practical manual has been prepared with carefully structured laboratory exercises. Each experiment includes clear objectives, detailed explanations, and sample solutions to promote better comprehension and practical understanding. The manual aims to bridge the gap between theoretical concepts and real-world applications, enabling students to explore advanced topics such as dynamic data structures, balanced trees, hashing techniques, graph algorithms, and priority structures.

We hope that this manual will serve as a valuable resource for M.Tech CSE students in mastering advanced data structures from an applied perspective. There is always room for refinement and enhancement, and we warmly welcome constructive feedback and suggestions from readers and users for future improvements.

ACKNOWLEDGEMENT

It has been a rewarding experience working on the Advanced Computer Networks Laboratory Manual. We would like to express our sincere gratitude to **Dr.K.Abdul Basith**, Professor and Head of the Department of Computer Science and Engineering, Marri Laxman Reddy Institute of Technology & Management, for his constant encouragement, valuable guidance, and continuous support in the preparation of this manual.

We are deeply indebted and gratefully acknowledge the support and motivation provided by the **Dr.P.Sridhar** Director, Marri Laxman Reddy Institute of Technology & Management, for granting us the opportunity and necessary resources to develop this laboratory manual.

Our heartfelt thanks to **Dr. R. Murali Prasad**, Principal, Marri Laxman Reddy Institute of Technology & Management, for his insightful suggestions, timely feedback, and academic guidance throughout the preparation of this document.

Finally, we extend our sincere appreciation to all the faculty members of the CSE Department whose encouragement, cooperation, and constructive inputs have played a significant role in helping us accomplish this work successfully.

COMPUTER SCIENCE AND ENGINEERING

ADVANCED DATA STRUCTURES LAB MANUAL

GENERAL INSTRUCTIONS

- Students are instructed to attend the Advanced Data Structures laboratory on time. Late comers will not be entertained in the lab.
- Students must be punctual. Experiments conducted during the session will not be repeated for those who are absent or late.
- Students are expected to come prepared with the theory and procedure of the experiment scheduled for the day.
- Students must display their identity cards before entering the laboratory.
- The use of mobile phones inside the lab is strictly prohibited.
- Any damage or loss of system components such as keyboard, mouse, or other peripherals during the lab session will be the responsibility of the student, and a fine or penalty will be imposed accordingly.
- Students must update their lab records and observation books session-wise. Before leaving the lab, the student should get their observation book signed by the concerned faculty member.
- Lab records must be submitted in the next lab session to the respective faculty members in the staffroom for correction and return.
- Students should not move around or disturb others during the lab session.
- In case of any emergency, students must obtain written permission from the concerned faculty member before leaving the laboratory.
- Faculty members reserve the right to suspend any student from the lab session on disciplinary grounds.

COMPUTER SCIENCE AND ENGINEERING

ADVANCED DATA STRUCTURES LAB MANUAL

INSTITUTE VISION AND MISSION

Vision:

To be as an ideal academic institution by graduating talented engineers to be ethically strong, competent with quality research and technologies.

Mission:

- Utilize rigorous educational experiences to produce talented engineers
- Create an atmosphere that facilitates the success of students
- Programs that integrate global awareness, communication skills and Leadership qualities
- Education and Research partnership with institutions and industries to prepare the students for interdisciplinary research

COMPUTER SCIENCE AND ENGINEERING

ADVANCED DATA STRUCTURES LAB MANUAL

DEPARTMENT VISION AND MISSION

Vision:

To empower the students to be technologically adept, innovative, self-motivated and responsible global citizen possessing human values and contribute significantly towards high quality technical education with ever changing world.

Mission:

- To offer high-quality education in the computing fields by providing an environment where the knowledge is gained and applied to participate in research, for both students and faculty.
- To develop the problem solving skills in the students to be ready to deal with cutting edge technologies of the industry.
- To make the students and faculty excel in their professional fields by inculcating the communication skills, leadership skills, team building skills with the organization of various co-curricular and extra- curricular programmes.
- To provide the students with theoretical and applied knowledge, and adopt an education approach that promotes lifelong learning and ethical growth.

COMPUTER SCIENCE AND ENGINEERING

ADVANCED DATA STRUCTURES LAB MANUAL

Program Outcomes

PO No.	NBA Statement / Vital Features	No. of Vital Features
PO1.	**An ability to independently carry out research /investigation and development work to solve practical problems.** 1. Research problems in Computer Science and Engineering are clearly identified and defined. 2. Literature review highlights research gaps and suitable methods. 3. Experiments or simulations are conducted using appropriate tools. 4. Data collection, analyses, and interpretation systematically. 5. Innovative approaches are applied to engineering problem-solving. **6.** Results are validated against established theories and standards.	**6**
PO 2.	**An ability to write and present a substantial technical report/document** 1. Technical reports, dissertations, and papers are well-structured. 2. Referencing and academic integrity practices are properly maintained. 3. Content is presented with clarity, precision, and logical flow. 4. Oral communication and presentation skills are effectively demonstrated. 5. Digital tools are used for documentation and visualization. **6.** Research findings are communicated to both technical and non-technical audience.	**6**
PO 3.	**Students should be able to demonstrate advanced proficiency in Computer Science and allied emerging areas of Engineering.** i. In-Depth Technical Knowledge ii. Advanced Problem-Solving and Algorithmic Skills. iii. Hands-On Practical Expertise iv. Research and Innovation Aptitude **v.** Interdisciplinary Integration	**5**
PO 4.	**Students should be able to identify, analyze, and effectively solve complex real-world problems by applying advanced computing concepts, while considering solutions from a global perspective** i. Ability to break down multifaceted problems into manageable parts and develop	

	effective solutions using advanced computing techniques. ii. Analytical Thinking and Critical Reasoning iii. Advanced Computing Knowledge. iv. Capability to understand and model complex systems, considering interdependencies and dynamic behaviors within global contexts. v. Global and Societal Awareness. vi. Aptitude for developing novel approaches and innovative solutions to address complex challenges. vii. Ability to work effectively in diverse teams and integrate knowledge from multiple disciplines to solve global problems. viii. Skill in clearly presenting problem analyses, solutions, and their implications to technical and non-technical global audience.	**8**
PO 5.	**An ability to acquire and apply advanced technical knowledge, professional skills, and modern computing tools to develop sustainable solutions** i. Ability to continuously learn and apply cutting-edge technical knowledge in computing and related fields. ii. Proficiency with Modern Computing Tools. iii. Capability to design and implement solutions that are environmentally, and economically feasible. iv. Understanding of professional ethics and responsibility in creating technology solutions with long-term positive impact. v. Integration of Multidisciplinary Knowledge. vi. Adaptability to sustainable practices.	**6**
PO 6.	**An Ability to recognize the significance of lifelong learning and actively pursue continuous professional development by adapting technologies in emerging areas.** i. Self-Directed Learning. ii. Skill in quickly learning and integrating new technologies and tools as they emerge in the field. iii. Capability to assess the relevance and impact of emerging technologies and decide on their applicability. iv. Continuous Professional Development Planning. v. Ability to engage with professional communities, attend workshops, and collaborate to stay updated. vi. Habit of regularly reflecting on personal growth, learning experiences, and professional competencies to improve continuously.	**6**

COMPUTER SCIENCE AND ENGINEERING
ADVANCED DATA STRUCTURES LAB MANUAL

Program Educational Objectives

Sl. No.	PEOs Name	Program Education Objective Statements
1	PEO – 1	Graduates will achieve professional excellence and success in the field of Computer Science and Engineering by applying strong technical foundations and problem-solving skills to contribute effectively to industry, academia, and entrepreneurship.
2	PEO – 2	Graduates will demonstrate a commitment to lifelong learning by continuously enhancing their knowledge and skills through professional development and self-directed learning to effectively adapt to evolving global challenges.
3	PEO – 3	Graduates of the Computer Science and Engineering program will actively pursue advanced research, contributing to the development of solutions for complex problems and the generation of new knowledge to effectively address real-world challenges.
4	PEO – 4	Graduates will exhibit professionalism, effective communication, leadership skills, and ethical responsibility while working in multidisciplinary teams to deliver computing solutions that address societal needs and contribute to sustainable development.

COMPUTER SCIENCE AND ENGINEERING

ADVANCED COMPUTER NETWORKS LAB MANUAL

Program Specific Outcomes

PSO1: Applications of Computing: Ability to use knowledge in various domains to provide solution to new ideas and innovations.

PSO2: Programming Skills: Identify required data structures, design suitable algorithms, develop and maintain software for real world problems.

COMPUTER SCIENCE AND ENGINEERING

ADVANCED DATA STRUCTURES LAB MANUAL

Course Structure

Advanced computer networks lab will have a continuous evaluation during II semester for 40 sessional marks and 60 end semester examination marks.

Out of the 40 marks for internal evaluation, day-to-day work in the laboratory shall be evaluated for 20 marks and internal practical examination shall be evaluated for 10 marks conducted by the laboratory teacher concerned.

The end semester examination shall be conducted with an external examiner and internal examiner. The external examiner shall be appointed by the principal / Chief Controller of examinations

COMPUTER SCIENCE AND ENGINEERING

ADVANCED DATA STRUCTURES LAB MANUAL

OBJECTIVE:

- To be able to introduce core programming basics and program design with functions using Python programming language.
- To understand a range of Object-Oriented Programming, as well as in-depth data and information processing techniques.
- To understand the high-performance programs designed to strengthen the practical expertise.

OUTCOMES:

Upon the completion of Operating Systems practical course, the student will be able to:

- Student should be able to understand the basic concepts scripting and the contributions of scripting language
- Ability to explore python especially the object oriented concepts, and the built in objects of Python.

2425837: ADVANCED DATA STRUCTURES LAB (Professional Elective - III Lab)

M.Tech CSE I Year II Sem.

L T P C
0 0 4 2

Prerequisites: Data communication, Basic networking principles, Computer Networks

Course Objectives:

1. Understand and analyze the existing protocols
2. Understand the use of network packet capturing tools

Course Outcomes: Ability of acquiring the practical exposure to existing protocols

List of Experiments:

1. Implement the IP fragmentation and reassembly algorithm.
2. Implement the IP forwarding algorithm.
3. Implement the simplest sliding window protocol of TCP.
4. Connect two systems using a switch and configure private IP addresses to the systems and ping them from each other. Using Wireshark, capture packets and analyze all the header information in the packets captured.
5. Install Telnet on one of the systems connected by a switch and telnet to it from the other system. Using Wireshark, capture the packets and analyze the TCP 3-way Handshake for connection establishment and tear down.
6. Start packet capture in wireshark application and then open your web browser and type in an URL of the website of your choice. How long did it take from when the HTTP GET message was sent until the HTTP OK reply was received for the web page you visited in your web browser.

Experiment-1

1) Implement the IP fragmentation and reassembly algorithm.

```
   def fragment_packet(data, mtu):
   header_size = 20  # Assume IP header is 20 bytes
   max_data_size = mtu - header_size
   fragments = []
   offset = 0
   while data:
      chunk = data[:max_data_size]
      data = data[max_data_size:]
      mf = 1 if data else 0
      fragments.append({
         'offset': offset,
         'data': chunk,
         'mf': mf
      })
      offset += len(chunk)
   return fragments
def reassemble_fragments(fragments):
   fragments.sort(key=lambda x: x['offset'])
   data = ''.join([frag['data'] for frag in fragments])
   return data
packet_data                                                                    =
"ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890abcdefghijklmnopqrstuvwxyz"
mtu = 30
print("Original Data Length:", len(packet_data))
print("\n--- Fragmentation ---")
fragments = fragment_packet(packet_data, mtu)
for i, frag in enumerate(fragments):
   print(f"Fragment {i+1}: Offset={frag['offset']} | MF={frag['mf']} | Data={frag['data']}")
print("\n--- Reassembly ---")
reassembled = reassemble_fragments(fragments)
print("Reassembled Data:", reassembled)
print("Data Matches Original:", reassembled == packet_data)
```

OUTPUT:

```
Original Data Length: 62
--- Fragmentation ---
Fragment 1: Offset=0 | MF=1 | Data=ABCDEFGHIJKLMN
Fragment 2: Offset=14 | MF=1 | Data=OPQRSTUVWXYZ123
Fragment 3: Offset=28 | MF=1 | Data=4567890abcdefghi
Fragment 4: Offset=43 | MF=1 | Data=jklmnopqrstuv
Fragment 5: Offset=57 | MF=0 | Data=wxyz
```



```
--- Reassembly ---
```

Reassembled
Data: ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890abcdefghijklmnopqrstuvwxyz
Data Matches Original: True

Viva Questions

1. What is IP fragmentation?
2. Why is IP fragmentation required in computer networks?
3. What is MTU (Maximum Transmission Unit)?
4. Which IP header fields are involved in fragmentation?
5. What is the purpose of the Identification field in IP fragmentation?
6. What is the DF (Don't Fragment) flag?
7. What is the MF (More Fragments) flag?
8. What is the Fragment Offset field?
9. Why is the Fragment Offset measured in units of 8 bytes?
10. Where does IP reassembly take place?
11. How does the destination host identify fragments of the same IP packet?
12. What happens if one IP fragment is lost?
13. What is IP reassembly timeout?
14. Can routers perform IP reassembly? Why or why not?
15. Why is IP fragmentation considered inefficient in modern networks?

Experiment-2
Implement the IP forwarding algorithm.
Each router:

- Uses a routing table containing:
  - Network address
  - Subnet mask
  - Next hop
  - Interface
- Applies Longest Prefix Match (LPM) to forward the packet.

```
import ipaddress
routing_table = [
    ("192.168.1.0", "255.255.255.0", "192.168.1.1", "eth0"),
    ("192.168.0.0", "255.255.0.0", "192.168.0.1", "eth1"),
    ("10.0.0.0", "255.0.0.0", "10.0.0.1", "eth2"),
    ("0.0.0.0", "0.0.0.0", "192.168.100.1", "eth3")
]
def ip_forward(dest_ip):
    dest = ipaddress.IPv4Address(dest_ip)
    max_prefix = -1
    selected_route = None

    for net, mask, next_hop, iface in routing_table:
        network = ipaddress.IPv4Network(f"{net}/{mask}", strict=False)
        if dest in network:
            prefix_len = network.prefixlen
            if prefix_len > max_prefix:
                max_prefix = prefix_len
                selected_route = (net, mask, next_hop, iface)
    if selected_route:
        net, mask, next_hop, iface = selected_route
        print(f"Destination IP: {dest_ip}")
        print(f"Matched Route : {net}/{mask}")
        print(f"Next Hop      : {next_hop}")
        print(f"Outgoing Iface: {iface}")
    else:
        print(f"No route found for {dest_ip}")
print("=== IP Forwarding Simulation ===\n")
ip_forward("192.168.1.24")
print("\n")
ip_forward("192.168.10.5")
print("\n")
ip_forward("10.5.6.7")
print("\n")
```

```
ip_forward("8.8.8.8")
```

OUTPUT:

```
=== IP Forwarding Simulation ===
Destination IP: 192.168.1.24
Matched Route : 192.168.1.0/255.255.255.0
Next Hop      : 192.168.1.1
Outgoing Iface: eth0

Destination IP: 192.168.10.5
Matched Route : 192.168.0.0/255.255.0.0
Next Hop      : 192.168.0.1
Outgoing Iface: eth1
```

Viva Questions

1. What is IP forwarding?
2. What is the purpose of the IP forwarding algorithm?
3. What information does a router use to forward an IP packet?
4. What is a routing table?
5. What is the role of the destination IP address in forwarding?
6. What is longest prefix matching?
7. Why is longest prefix match used in IP forwarding?
8. What is the default route?
9. What happens if no matching route is found?
10. What is the role of the next-hop address?
11. What is the difference between direct and indirect routing?
12. How does TTL affect IP forwarding?
13. What action is taken when TTL becomes zero?
14. Does IP forwarding modify the IP header? If yes, which fields?
15. How is IP forwarding different from IP routing?

Experiment-3

Implement the simplest sliding window protocol of TCP.

Sender can send multiple packets within a window without waiting for ACK.

If a packet is lost or ACK is not received, sender retransmits all packets from the lost one (Go-Back-N).

Receiver only accepts packets in order.

Program:

```
import random
import time
class Sender:
    def __init__(self, total_packets, window_size):
        self.total_packets = total_packets
        self.window_size = window_size
        self.base = 0
        self.next_seq = 0
    def send_packets(self):
        print("=== Sender Side ===")
        while self.base < self.total_packets:
            # Send all packets in the window
                    while self.next_seq < self.base + self.window_size and self.next_seq < self.total_packets:
                print(f"Sender: Sending packet {self.next_seq}")
                self.next_seq += 1
            ack = Receiver.receive(self.base, self.next_seq)
            if ack == self.base:
                print(f"Sender: Timeout! Resending from packet {self.base}")
                time.sleep(1)  # simulate timeout delay
            else:
                print(f"Sender: Received ACK for packet {ack - 1}")
                self.base = ack
class Receiver:
    @staticmethod
    def receive(start, end):
        loss_chance = random.randint(0, 9)
        if loss_chance < 2:
            print(f"Receiver: Packet {start} lost or corrupted")
            return start  # simulate lost ACK for first in window
        else:
            for i in range(start, end):
                print(f"Receiver: Received packet {i}")
            return end  # acknowledge all
if __name__ == "__main__":
    total_packets = 10
    window_size = 4
    sender = Sender(total_packets, window_size)
    sender.send_packets()
```


OUTPUT:

=== Sender Side ===
Sender: Sending packet 0
Sender: Sending packet 1
Sender: Sending packet 2
Sender: Sending packet 3
Receiver: Received packet 0
Receiver: Received packet 1
Receiver: Received packet 2
Receiver: Received packet 3
Sender: Received ACK for packet 3
Sender: Sending packet 4
Sender: Sending packet 5
Sender: Sending packet 6
Sender: Sending packet 7
Receiver: Packet 4 lost or corrupted
Sender: Timeout! Resending from packet 4
Sender: Sending packet 4
Sender: Sending packet 5
Sender: Sending packet 6
Sender: Sending packet 7
Receiver: Received packet 4
Receiver: Received packet 5
Receiver: Received packet 6
Receiver: Received packet 7
Sender: Received ACK for packet 7
Sender: Sending packet 8
Sender: Sending packet 9
Receiver: Received packet 8
Receiver: Received packet 9
Sender: Received ACK for packet 9

Viva Questions

1. What is a sliding window protocol?
2. Why is sliding window protocol used in TCP?
3. What is meant by window size in TCP?
4. What is the simplest sliding window protocol?
5. How does the sender decide how many packets to send?
6. What is the role of acknowledgements in sliding window protocol?
7. What happens when an acknowledgement is lost?
8. How does sliding window protocol provide flow control?
9. What is the difference between stop-and-wait and sliding window protocol?
10. How does sliding window protocol improve network efficiency compared to stop-and-wait?

Experiment-4

Connect two systems using a switch and configure private IP addresses to the systems and ping them from each other. Using Wireshark, capture packets and analyze all the header information in the packets captured.

Step-by-Step Configuration

1. Connect Systems to Switch

- Use two Ethernet cables to connect both systems to the switch.

2. Assign Private IP Addresses

On System A:
Windows:
Control Panel → Network & Internet → Change adapter settings → Ethernet → Properties → IPv4 → Set:
IP address: 192.168.1.10
Subnet Mask: 255.255.255.0
On System B:
IP address: 192.168.1.20
Subnet Mask: 255.255.255.0

3. Ping Test

On System A, open Command Prompt:
ping 192.168.1.20

Expected Output:

Reply from 192.168.1.20: bytes=32 time<1ms TTL=128

4. Capture Packets Using Wireshark

1. Launch Wireshark on System A.
2. Select the active Ethernet interface.
3. Click Start Capture.
4. Run the ping command from System A to System B.
5. Stop the capture after a few seconds.

5. Analyze the Packet Headers

Captured Packets:

You'll see a sequence of:

- ICMP Echo Request (ping request)
- ICMP Echo Reply (ping response)

Click on any packet to expand its layers:

- Ethernet II
- Internet Protocol Version 4 (IPv4)
- Internet Control Message Protocol (ICMP)

Header Fields to Analyze

Ethernet Header:

- Source MAC: MAC address of System A
- Destination MAC: MAC address of System B
- Type: 0x0800 (IPv4)

IP Header:

- Source IP: 192.168.1.10
- Destination IP: 192.168.1.20
- Version: 4
- Header Length: 20 bytes
- Total Length: 84 (varies)
- TTL (Time To Live): usually 128 (Windows)

- Protocol: 1 (ICMP)

ICMP Header:

- Type: 8 (Echo Request) / 0 (Echo Reply)
- Code: 0
- Checksum: Auto calculated
- Identifier/Sequence Number: Helps match replies to requests

Viva Questions

1. What is the function of a network switch?
2. What is a private IP address?
3. Which private IP address ranges are defined by IPv4?
4. Why is a subnet mask required while configuring an IP address?
5. What protocol is used when one system pings another?
6. What is the purpose of the ARP protocol in this experiment?
7. Why are ARP packets captured before ICMP packets in Wireshark?
8. What are the main fields present in an Ethernet frame header?
9. What information is present in the IP header of the captured packet?
10. What is the role of Wireshark in network packet analysis?

Experiment-5

Install Telnet on one of the systems connected by a switch and telnet to it from the other system. Using Wireshark, capture the packets and analyze the TCP 3-way Handshake for connection establishment and tear down.

Install Telnet Server on System A (Telnet Server):

If **System A** is running **Windows**, follow these steps to enable the Telnet Server:

1. Open **Control Panel → Programs → Turn Windows Features On or Off**.
2. Scroll down and check **Telnet Server**.
3. Click **OK** and wait for the installation to complete.

If **System A** is running **Linux**, you can install Telnet using the following command:

```
sudo apt update
sudo apt install telnetd
sudo service inetd restart  # or 'service telnet restart' depending on your system
```

2. Install Telnet Client on System B (Telnet Client):

On **System B**, you need to have a **Telnet client** installed.

For **Windows**:

1. **Telnet Client** is usually enabled by default. If not, go to **Control Panel → Programs and Features → Turn Windows Features On or Off** and check **Telnet Client**.

For **Linux**:

```
sudo apt update
sudo apt install telnet
```

3. Configure Private IP Addresses:

Assign private IP addresses to both systems. For example:

- **System A (Telnet Server)**: 192.168.1.10
- **System B (Telnet Client)**: 192.168.1.20

Make sure they can **ping each other**:

On **System B**, run:

```
ping 192.168.1.10
```

4. Telnet Connection Setup:

1. On **System A** (Telnet Server), ensure the **Telnet service is running**.
2. On **System B** (Telnet Client), open the terminal or command prompt and use the following command to connect to **System A**:

```
telnet 192.168.1.10
```

Once connected, you can interact with the Telnet session.

Capture Packets Using Wireshark:

1. Open **Wireshark** on **System B** (or on any system where the packets can be captured).
2. Start a **packet capture** on the active network interface (Ethernet/Wi-Fi).
3. Filter the capture using the filter:

```
tcp.port == 23
```

This will only show the Telnet-related TCP packets, as Telnet runs on port 23.

4. Now, on **System B**, type the telnet command to initiate the Telnet session.

Analyze the TCP 3-Way Handshake:

The **TCP 3-way handshake** is used to establish a connection between a client and a server.

1. **SYN (System B to System A)**: The client (System B) sends a **SYN** message to initiate the connection.
2. **SYN-ACK (System A to System B)**: The server (System A) responds with a **SYN-ACK** to acknowledge the request.
3. **ACK (System B to System A)**: The client (System B) sends an **ACK** to complete the connection.

Connection Teardown:

Once the Telnet session is over, the connection is closed with a **4-way handshake**:

1. **FIN (System B to System A)**: The client (System B) sends a **FIN** request to close the connection.
2. **ACK (System A to System B)**: The server acknowledges the **FIN** with an **ACK**.
3. **FIN (System A to System B)**: The server sends its own **FIN** to close the connection.
4. **ACK (System B to System A)**: The client acknowledges the server's **FIN**.

Viva Questions

1. What is Telnet and why is it used?
2. Which transport layer protocol does Telnet use?
3. What is the default port number of Telnet?
4. What is a TCP 3-way handshake?
5. Which flags are used in the TCP 3-way handshake?
6. What is the purpose of the SYN flag?
7. What happens in the second step of the TCP 3-way handshake?
8. How is a TCP connection terminated?
9. What TCP flags are used during connection termination?
10. Why is Wireshark used to analyze the TCP handshake?

Experiment-6

Start packet capture in wireshark application and then open your web browser and type in an URL of the website of your choice. How long did it take from when the HTTP GET message was sent until the HTTP OK reply was received for the web page you visited in your web browser.

1. Launch Wireshark for Packet Capture:

1. Open **Wireshark** on your system.
2. Select the active **network interface** (Ethernet or Wi-Fi).
3. Start the packet capture by clicking on the **Start Capturing Packets** button (the blue shark fin icon).
4. Optionally, apply a filter to narrow down the capture to HTTP traffic:
   http
   This will filter out other types of traffic and focus on HTTP packets.

2. Open Your Web Browser:

1. In your web browser, type the URL of a website (e.g., http://www.example.com).
2. Hit **Enter** to visit the website.

3. Wait for the HTTP Request and Response:

- **HTTP GET request**: The browser will send a **GET** request to the server to retrieve the web page.
- **HTTP OK reply**: The server will respond with an **HTTP 200 OK** response, which indicates that the page was successfully retrieved.

4. Stop Packet Capture in Wireshark:

1. Once the page has loaded, go back to **Wireshark**.
2. Click on the **Stop Capturing Packets** button (the red square icon).

5. Locate the HTTP GET and HTTP OK Packets:

- In the Wireshark capture window, look for the **HTTP GET** request packet and the **HTTP OK** (HTTP 200) response packet.

6. Analyze the Time:

1. **Locate the GET request**:
   - The **GET request** is sent by the browser to the server. In the **Wireshark packet details**, you will see something like:

```
GET / HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0 ...
```

2. **Locate the HTTP OK response**:
   - The **HTTP 200 OK** response from the server can be found in a subsequent packet:

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
```

3. **Measure the time** between the **GET** request and the **HTTP OK** reply:
   - In Wireshark, when you click on the HTTP GET request, you can view the **Time** column, which shows the timestamp of when the packet was captured.
   - Do the same for the **HTTP OK** response.
   - The difference between these two timestamps gives you the **round-trip time** for the request and response.

Output Example:

GET Request Packet:

```
Frame 1: 162 bytes on wire
Ethernet II, Src: 00:14:22:58:3c:2f, Dst: 00:15:5d:3b:62:01
Internet Protocol Version 4, Src: 192.168.1.10, Dst: 93.184.216.34
Transmission Control Protocol, Src Port: 45678, Dst Port: 80, Seq: 1, Len: 146
Hypertext Transfer Protocol
 GET / HTTP/1.1\r\n
  Host: www.example.com\r\n
  User-Agent: Mozilla/5.0 ...
   Accept-Encoding: gzip, deflate
    \r\n
```

HTTP OK Response Packet:

```
Frame 5: 567 bytes on wire
Ethernet II, Src: 00:15:5d:3b:62:01, Dst: 00:14:22:58:3c:2f
Internet Protocol Version 4, Src: 93.184.216.34, Dst: 192.168.1.10
Transmission Control Protocol, Src Port: 80, Dst Port: 45678, Seq: 1, Ack: 147, Len: 521
Hypertext Transfer Protocol
HTTP/1.1 200 OK\r\n
Date: Tue, 15 Mar 2022 13:21:45 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 1256
 \r\n
```


Viva Questions

1. What is the purpose of using Wireshark in this experiment?
2. What happens when you type a URL in a web browser?
3. What is an HTTP GET message?
4. What does an HTTP 200 OK response indicate?
5. Which protocol is used to transfer web pages?
6. How can you identify the HTTP GET packet in Wireshark?
7. How can you identify the HTTP OK response in Wireshark?
8. What factors affect the time between HTTP GET and HTTP OK response?
9. Which transport layer protocol is used by HTTP?
10. How is the response time measured using Wireshark?