KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

## DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

# ADVANCED COMPUTER NETWORKS LABORATORY MANUAL

Subject Code : KGR2358213

Regulation : KGR23

Academic Year : 2023-2024

## M. Tech CSE/CS I Year II Sem.


**COMPUTER SCIENCE AND ENGINEERING**

**KG REDDY COLLEGE OF ENGINEERING AND TECHNOLOGY**

Affiliated o JNTUH, Chilkur,(V), Moinabad(M) R. R Dist, TS-501504

KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

# VISION AND MISSION OF THE INSTITUTION

## VISION:

| |
|---|
| To become an institution which is internationally recognized for its holistic approach to engineering, innovative teaching and learning culture, research and entrepreneurial ecosystem, and sustainable social impact in the community. |

## MISSION:

| |
|---|
| • To offer undergraduate and post-graduate programs which are supported through industry relevant curriculum and innovative teaching and learning processes that would help students succeed in their professional careers. |
| • To provide faculty and students with an ecosystem that fosters innovation, research, entrepreneurship, and international exposure through strategic partnerships with governmentorganizations and collaboration with industries. |
| • To provide holistic learning environment to students which will contribute to their personal and professional growth and enable them to become leaders in their respective fields. |
| • To contribute to the development of the region by using our technological expertise to work with nearby communities and support them in their social and economic development. |

\

KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

# DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

## VISION:

| |
|---|
| To be recognized as a department of excellence by stimulating a learning environment in which students and faculty will thrive and grow to achieve their professional, institutional and societal goals. |

## MISSION:

| |
|---|
| • To provide high quality technical education to students that will enable life-long learning and build expertise in advanced technologies in Computer Science and Engineering. |
| • To promote research and development by providing opportunities to solve complex engineering problems in collaboration with industry and government agencies. |
| • To encourage professional development of students that will inculcate ethical values and leadership skills through entrepreneurship while working with the community to address societalissues. |

## PROGRAM EDUCATIONAL OBJECTIVES

| |
|---|
| **PEO 1:** Graduates will provide solutions to difficult and challenging issues in their profession by applying computer science and engineering theory and principles. |
| **PEO 2:** Graduates have successful careers in computer science and engineering fields or will be able to successfully pursue advanced degrees. |
| **PEO 3:** Graduates will communicate effectively, work collaboratively and exhibit high levels of professionalism, moral and ethical responsibility. |
| **PEO 4:** Graduates will develop the ability to understand and analyze engineering issues in a broader perspective with ethical responsibility towards sustainable development. |

KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

# DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

## PROGRAM OUTCOMES

| |
|---|
| **PO I: Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems. |
| **PO II: Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences. |
| **PO III: Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations. |
| **PO IV: Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions. |
| **PO V: Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations. |
| **PO VI: The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice. |
| **PO VII: Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of and need for sustainable development. |
| **PO VIII: Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice. |
| **PO IX: Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings. |
| **PO X: Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions. |
| **PO XI: Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments. |
| **PO XII: Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change. |

KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

# DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

## PROGRAM SPECIFIC OUTCOMES

| |
|---|
| **PSO1:** The Computer Science and Engineering graduates are able to analyze, design, develop, test and apply management principles, mathematical foundations in the development of intelligent systems with computational solutions, make them to expert in designing the secure application and hardware prototype |
| **PSO2:** The graduating student will be analyze the contemporary research issues in different areasof computer science & engineering and explore research gaps, analyze and carry out research in the specialized/emerging areas. |
| **PSO3**: Develop their skills to solve problems in the broad area of programming concepts and appraise environmental and social issues with ethics and manage different projects in multi-disciplinary field to conducive in cultivating skills for successful career, entrepreneurship and higher studies. |

KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

# DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

## ADVANCED COMPUTER NETWORK LAB

**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:

| S.NO | 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. |

# 1Q. **Implement the IP fragmentation and reassembly algorithm.**

**.The IP fragmentation and reassembly algorithm** is a process that occurs when a packet is too large to be transmitted over a network in a single piece. The packet must be broken up into smaller pieces, called fragments, that can be transmitted and reassembled at the destination.

IP Fragmentation:

CODE:

```
def fragment_packet(packet, mtu):
    fragments = []
    offset = 0

    while len(packet) > mtu:
        fragment = packet[:mtu]
        packet = packet[mtu:]

        # set fragment flags
        flags = 0b001 if offset == 0 else 0b000
        more_fragments = 0b001 if len(packet) > 0 else 0b000
        fragment_offset = offset >> 3  # divide by 8 to get byte offset

        # create IP header for fragment
        ip_header = create_ip_header(flags, fragment_offset, more_fragments)

        # add IP header to fragment and append to list
        fragments.append(ip_header + fragment)

        # update offset
        offset += mtu

    # create IP header for last fragment
    flags = 0b000 if offset == 0 else 0b000
    more_fragments = 0b000
    fragment_offset = offset >> 3
    ip_header = create_ip_header(flags, fragment_offset, more_fragments)

    # add IP header to last fragment and append to list
    fragments.append(ip_header + packet)
    return fragments
```

**IP Reassembly**

CODE

```
def reassemble_packets(fragments):
    packets = {}

    for fragment in fragments:
        # extract IP header information
        flags = fragment[6] & 0b111
        fragment_offset = (fragment[6] & 0b1111111111110000) >> 3
        more_fragments = (fragment[6] & 0b1000) >> 2

        # get identification field from IP header
        id_field = (fragment[4] << 8) + fragment[5]

        if id_field not in packets:
            # first fragment of new packet
            packets[id_field] = {
                "fragments": [fragment],
                "more_fragments": more_fragments,
                "length": len(fragment) - 20  # subtract IP header length
            }
        else:
            # add fragment to existing packet
            packets[id_field]["fragments"].append(fragment)
            packets[id_field]["more_fragments"] = more_fragments
            packets[id_field]["length"] += len(fragment) - 20

        if not more_fragments:
            # all fragments received for packet
            reconstructed_packet = b"".join(packets[id_field]["fragments"])
            del packets[id_field]
            return reconstructed_packet

    # not all fragments received for any packet
    return None
```

## 2Q.. Implement the IP forwarding algorithm.

**IP FORWARDING**

```
def extract_dest_ip(packet):
    # extract destination IP address from packet header
    return packet[16:20]

def lookup_routing_table(dest_ip):
    # look up next hop in routing table based on destination IP address
    # return None if no next hop found
    # routing table might be represented as a dictionary with destination
    # network prefixes as keys and next hops as values
    return routing_table.get(dest_ip)

def encapsulate_packet(packet, next_hop, interface):
    # create new IP header with appropriate source and destination addresses
    # return encapsulated packet
    src_ip = get_interface_ip(interface)
    new_header = create_ip_header(src_ip, next_hop, len(packet))
    return new_header + packet

def send_packet(packet, next_hop):
    # send packet to next hop
    # this might involve sending the packet over a physical interface
    # or passing it to another layer of the network stack
    # depending on the implementation
    pass

def create_icmp_packet(packet, interface, dest_unreachable=False):
    # create ICMP packet with appropriate type and code based on input
    # for example, if dest_unreachable is True, create a destination
    # unreachable message with a code indicating that the host is unreachable
    # and include the original packet in the payload
    pass

def get_interface_ip(interface):
    # get the IP address associated with the specified interface
    # this might involve querying the operating system or a network API
    Pass
```

## 3Q. Implement the simplest sliding window protocol of TCP.

**The simplest sliding window protocol of TCP** is known as the "Stop-and-Wait" protocol. In this protocol, the sender sends a single packet and waits for an acknowledgement from the receiver before sending the next packet.

**CODE:**

```
# constants
MAX_PACKET_SIZE = 1024
TIMEOUT = 1.0  # in seconds

def send_data(data, dest_addr):
    # split data into packets of maximum size
    packets = [data[i:i+MAX_PACKET_SIZE] for i in range(0, len(data), MAX_PACKET_SIZE)]

    # initialize variables
    seq_num = 0
    ack_expected = 0

    while ack_expected < len(packets):
        # send next packet
        packet = create_packet(seq_num, packets[seq_num])
        send_packet(packet, dest_addr)

        # start timer
        start_time = time.time()

        # wait for acknowledgement or timeout
        while time.time() - start_time < TIMEOUT:
            ack_packet = receive_packet()
            if ack_packet and is_acknowledgement(ack_packet, ack_expected):
                # acknowledgement received
                ack_expected += 1
                break

        # timeout
        if ack_expected != seq_num + 1:
            continue

        # next sequence number
        seq_num += 1

def receive_data():
    # initialize variables
    next_seq_num = 0
    expected_seq_num = 0
```

```
  while True:
      # receive packet
      packet = receive_packet()

      # check sequence number
      if not packet or get_sequence_number(packet) != expected_seq_num:
          # ignore packet
          continue

      # deliver payload to application layer
      payload = get_payload(packet)
      deliver_data(payload)

      # send acknowledgement
      ack_packet = create_acknowledgement(expected_seq_num)
      send_packet(ack_packet, get_source_address(packet))

      # update sequence numbers
      expected_seq_num += 1
      next_seq_num += 1

      # return when all packets have been received
      if len(payload) < MAX_PACKET_SIZE:
          return
```

## SEND AND RECIEVE DATA CODE:

```
def create_packet(seq_num, payload):
    # create packet with sequence number and payload
    # packet might include a checksum or other metadata
    return (seq_num, payload)
def send_packet(packet, dest_addr):
    # send packet to specified destination address
    # this might involve sending the packet over a physical interface
    # or passing it to another layer of the network stack
    pass
def receive_packet():
    # receive next packet
    # this might involve waiting for a packet to arrive over a physical interface
    # or polling a socket for incoming data
    pass
def is_acknowledgement(packet, expected_seq_num):
    # check if
```

## 4Q. 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.

**To connect two systems using a switch and configure private IP addresses to the systems, follow these steps:**

1. Connect the two systems to the switch using Ethernet cables.
2. Assign private IP addresses to each system. For example, you could use the following addresses:
    - System 1: 192.168.1.1
    - System 2: 192.168.1.2
3. Make sure that the IP addresses are in the same subnet and that they are not already in use by another device on the network.

To capture packets using Wireshark, follow these steps:

1. Download and install Wireshark on one of the systems.
2. Start Wireshark and select the network interface that is connected to the switch. This is typically the Ethernet interface.
3. Start a capture session and initiate a ping from one system to the other.
4. Stop the capture session after a few seconds.

To analyze the header information in the captured packets, look for the following information:

- Source and destination IP addresses: These are the IP addresses of the sending and receiving systems.
- Protocol: This indicates the type of packet, such as ICMP for ping packets.
- Source and destination MAC addresses: These are the MAC addresses of the sending and receiving network interfaces.
- Timestamp: This indicates the time when the packet was captured.
- Length: This indicates the length of the packet in bytes.

You can also analyze the packet data to see the actual contents of the packets. This can be useful for troubleshooting network issues or identifying potential security threats.

5Q.. **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.**

To install Telnet on one of the systems connected by a switch and Telnet to it from the other system, follow these steps:

1. Install the Telnet client on the system that you want to use to connect to the other system. The Telnet client is typically included with most operating systems, but you may need to install it separately if it is not already installed.
2. On the system that you want to connect to, enable the Telnet service. This typically involves starting the Telnet server or enabling the Telnet service in the system's settings.
3. Determine the IP address of the system that you want to connect to. You can do this by opening a command prompt and entering the following command
4. ipconfig
   This will display the system's IP address and other network configuration information.

5. On the system that you want to use to connect to the other system, open a command prompt and enter the following command:

6. telnet <IP address of other system>
   For example, if the IP address of the other system is 192.168.1.2, enter
7. telnet 192.168.1.2
   This will initiate a Telnet session with the other system.

To capture packets using Wireshark and analyze the TCP 3-way handshake for connection establishment and tear down, follow these steps:

1. Download and install Wireshark on the system that you are using to connect to the other system.
2. Start Wireshark and select the network interface that is connected to the switch. This is typically the Ethernet interface.
3. Start a capture session and initiate a Telnet session with the other system.
4. Stop the capture session after the Telnet session has ended.
5. Analyze the captured packets to identify the TCP 3-way handshake for connection establishment and tear down. Look for the following information:
   - SYN packet: This is the first packet in the TCP 3-way handshake. It is sent by the client to request a connection with the server.
   - SYN-ACK packet: This is the second packet in the TCP 3-way handshake. It is sent by the server in response to the SYN packet to acknowledge the connection request and request a connection with the client.

- ACK packet: This is the third and final packet in the TCP 3-way handshake. It is sent by the client in response to the SYN-ACK packet to acknowledge the connection request and establish the connection.

6. You should also look for packets that indicate the end of the Telnet session, such as FIN or RST packets. These packets indicate that the connection has been closed.
   By analyzing the packet data, you can gain insight into the performance and behavior of the TCP protocol, as well as identify potential issues or vulnerabilities in the network.

## 6Q.. **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.**

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. Download and install Wireshark on your system.
2. Start Wireshark and select the network interface that is connected to your network.
3. Start a capture session in Wireshark by clicking the "Capture" button.
4. Open your web browser and enter the URL of the website that you want to visit.
5. Once the website has loaded, stop the capture session in Wireshark by clicking the "Stop" button.
6. Analyze the captured packets to identify the HTTP GET message and the HTTP OK reply. Look for the following information:

- HTTP GET message: This is the message that your web browser sends to the web server to request the web page. Look for packets that contain the GET request and the URL of the website that you visited.
- HTTP OK reply: This is the message that the web server sends back to your web browser to indicate that the web page has been successfully retrieved. Look for packets that contain the HTTP status code 200 OK and the content of the web page.

7. By analyzing the packet data, you can determine the time it takes for the web server to respond to the HTTP GET message and send the HTTP OK reply. Look for the time difference between the packet containing the HTTP GET message and the packet containing the HTTP OK reply. The time difference represents the round trip time (RTT) for the HTTP request and response.