Course Code:	**CSE3006**	**Course Name:**	Computer Network
Course Type:	**Manual**	**Class Number:**	BL2024250400848
Credits:	**4**	**Semester/Year:**	INTERIM SEMESTER 2024-25

Lab Report

Submitted by

Name: Pranjal Kumar Shukla
Reg No: 23BAI10411

Bachelor in Technology
in
CSE (AIML)

Submitted to: Mohd Rafi Lone Sir

School of Computer Science and Enginerring, VIT Bhopal University

Date of Submission: 9 th December , 2024

INDEX

Faculty signature: .....

EXPERIMENT NO. - 1

Experiment Title

Experiment 1: Demo session of all networking hardware and Functionalities.

Objective

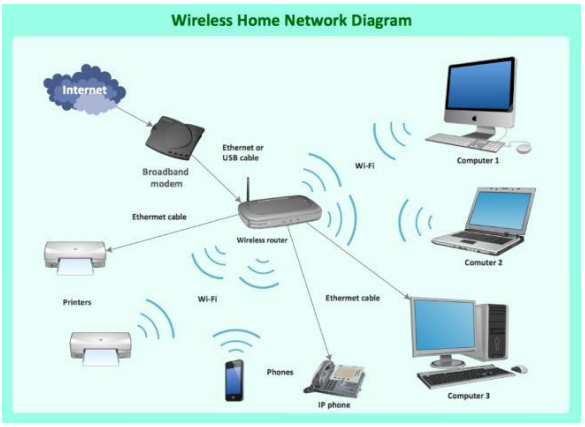
To demonstrate the functionality of various networking hardware such as routers, switches, hubs, and NICs, and understand their roles in network communication.

Theory/Background

Networking hardware refers to the physical devices that facilitate communication and data transfer within and between networks. Each piece of hardware plays a distinct role in ensuring the smooth functioning of a network. The major networking components include routers, switches, hubs, and Network Interface Cards (NICs).

1. Router

- **Definition**: A router is a networking device that connects multiple networks and directs data packets between them.

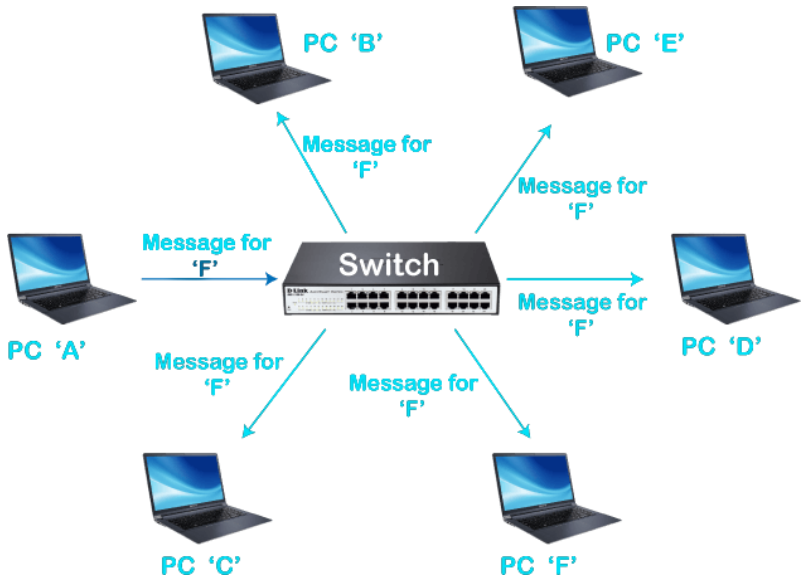


- **Functions**:
    1. Routes data based on IP addresses.
    2. Provides NAT (Network Address Translation) to enable private IP addresses to access the internet.
    3. Enables inter-network communication (e.g., LAN to WAN).
    4. Implements firewalls and security policies for traffic control.
- **Example**: When you access a website, the router in your home network sends the request to your Internet Service Provider (ISP), which routes it to the destination server.

2. Switch

- **Definition**: A switch is a networking device that connects devices within a LAN (Local Area Network) and forwards data based on MAC addresses.

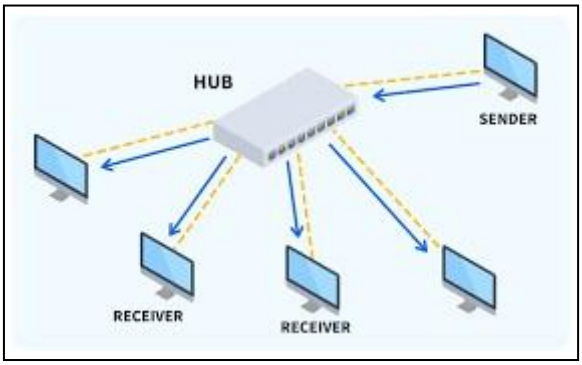
Faculty signature: .....



- **Functions**:
    1. Operates at Layer 2 (Data Link Layer) of the OSI model.
    2. Forwards packets to specific devices instead of broadcasting to all connected devices.
    3. Reduces network congestion by segmenting traffic.
    4. Provides higher bandwidth than hubs.
- **Example**: A switch in an office network connects employees' computers to printers, servers, and the internet.

3. Hub

- **Definition**: A hub is a basic networking device that connects multiple devices in a LAN and broadcasts incoming data to all connected devices.



- **Functions**:
    1. Operates at Layer 1 (Physical Layer) of the OSI model.
    2. Lacks intelligence; cannot differentiate the intended recipient of data.
    3. Causes network collisions in high traffic environments.
- **Limitations**: Due to inefficiency and lack of filtering, hubs are largely obsolete and replaced by switches.

4. Network Interface Card (NIC)

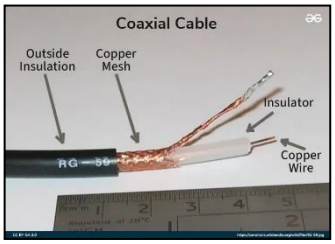
Faculty signature: .....

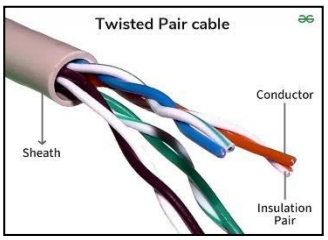
- **Definition**: A NIC is a hardware component (integrated or standalone) that connects a device to a network.

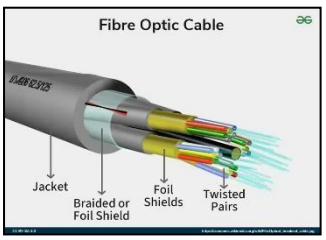
- **Functions**:
    1. Provides physical access to a networking medium (Ethernet, Wi-Fi, etc.).
    2. Assigns a unique MAC address to the device.
    3. Converts data into signals that can be transmitted over the network.
- **Types**: Wired (Ethernet-based) and Wireless (Wi-Fi-enabled).

5. Ethernet Cables

- **Definition**: Ethernet cables are the physical medium for transmitting data between devices. Common types include Coaxial Cables, Twisted Pair Cables, Fiber optic Cables







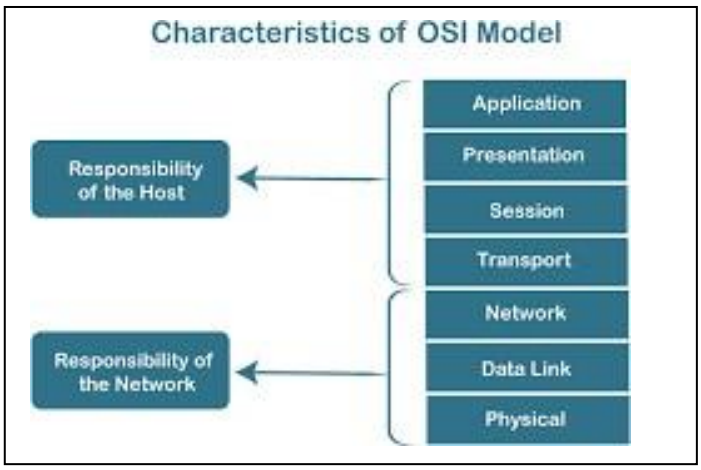
- **Functions**:
    1. Cat5e and Cat6 cables support high-speed data transfer (up to 1Gbps and 10Gbps, respectively).
    2. Used for creating wired connections in LAN setups.

6. OSI and TCP/IP Models

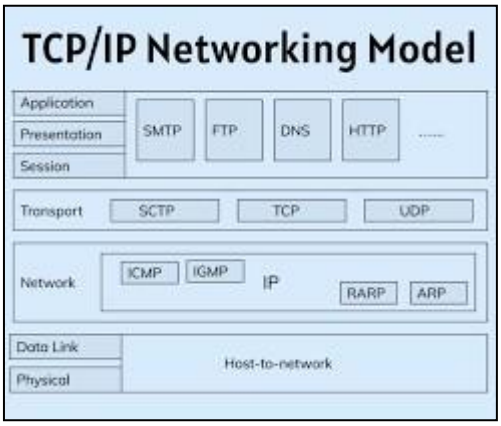
Understanding the OSI (Open Systems Interconnection) and TCP/IP models is crucial for comprehending how these devices interact.

- **OSI Model**: Defines 7 layers of communication. Relevant layers include:
    - **Layer 1: Physical Layer**: Hubs and cables operate here.
    - **Layer 2: Data Link Layer**: Switches work at this layer using MAC addresses.
    - **Layer 3: Network Layer**: Routers operate here using IP addresses.

Faculty signature: .....



- **TCP/IP Model**: A simplified 4-layer model. Devices typically operate in the Network Access and Internet layers.



Why Networking Hardware Matters?

1. **Scalability**: Enables networks to grow and accommodate more devices.
2. **Security**: Hardware like routers provides firewalls and encryption.
3. **Efficiency**: Switches and routers optimize data delivery, minimizing congestion.
4. **Interoperability**: Ensures seamless communication across devices and networks.

Apparatus/Software Required

1. Routers and switches
2. Ethernet cables (Cat5e or Cat6)
3. Hubs
4. Network Interface Cards
5. Cisco Packet Tracer (for simulations)

Faculty signature: .....

6. PCs or laptops

Procedure

1. Identify and label the hardware components.
2. Connect the devices using Ethernet cables.
3. Power on the devices and ensure connections are properly established.
4. Demonstrate the routing process using Cisco Packet Tracer:
    - Configure IP addresses for connected devices.
    - Ping between devices to verify connectivity.
5. Explain the packet transmission through a hub and a switch to show differences.
6. Observe and explain the router's role in directing traffic between two networks.

Observations/Results

- Data packets transmitted through a hub showed collisions due to broadcast nature.
- Switches efficiently managed packets by directing them to specific MAC addresses.
- Routers successfully routed data between two distinct networks.

Discussion/Analysis

1. Routers work at the network layer, while switches operate at the data link layer.
2. Hubs lack intelligence compared to switches and routers.
3. Packet Tracer provided insights into network traffic flow and device configurations.

Conclusion

This experiment provided hands-on exposure to networking hardware and their roles in a communication network. It highlighted the differences in functionality and efficiency between hubs, switches, and routers.

Viva Questions

1. What are the main differences between a hub and a switch?
2. Why is a router necessary in a network?
3. What is the role of a NIC in network communication?

References

1. "Computer Networking: Principles, Protocols, and Practice" by Olivier Bonaventure
2. Cisco Networking Academy materials

Faculty signature: .....

EXPERIMENT NO. - 2

Experiment Title

Experiment 2: **Introduction to Socket Programming and Basic Linux Commands.**

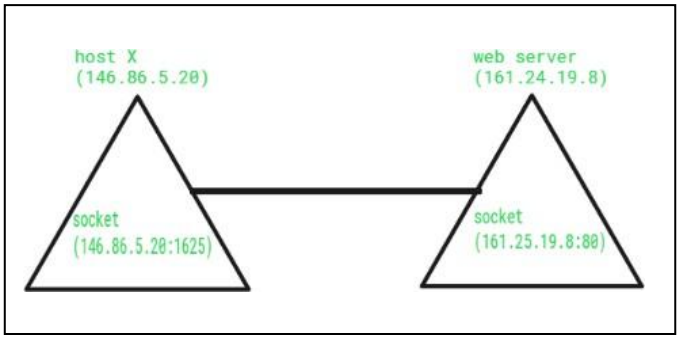
Objective

1. To understand the fundamentals of socket programming for establishing communication between client and server systems.
2. To learn and use basic Linux commands for file management, process control, and network configurations.

Theory/Background

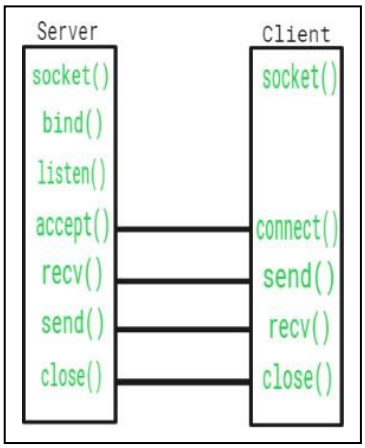
Socket Programming

- **Definition**: Socket programming is a method of enabling communication between two processes across a network, using the client-server architecture.



- **How it Works**:
    1. A socket is created at both ends (client and server).
    2. The server listens for incoming connections, while the client initiates a connection.
    3. Once connected, data is exchanged between the two systems.
- **Key Concepts**:
    1. **Socket Types**:
        - **Stream Socket (TCP)**: Reliable communication using TCP protocol.
        - **Datagram Socket (UDP)**: Connectionless communication using UDP protocol.

Faculty signature: .....



2. **Socket API Functions**:
    - **Server Side**: socket(), bind(), listen(), accept().
    - **Client Side**: socket(), connect().
    - **Data Exchange**: send(), recv().
3. **Protocols**: TCP (connection-oriented) and UDP (connectionless).

- **Applications**:
    - Web browsing (HTTP).
    - File transfers (FTP).
    - Video streaming (RTP).

Basic Linux Commands

- **Introduction**: Linux commands are used to interact with the operating system for managing files, processes, and networks.
- **Categories of Commands**:
    1. **File Management**:
        - ls: Lists files and directories.
        - cd: Changes directory.
        - cp: Copies files or directories.
        - mv: Moves or renames files.
        - rm: Deletes files or directories.
    2. **Process Management**:
        - ps: Displays running processes.
        - kill: Terminates a process.
        - top: Displays real-time system processes.

Faculty signature: .....

3. **Network Configuration**:
    - ping: Checks connectivity to a host.
    - ifconfig/ip: Displays or configures network interfaces.
    - netstat: Shows network connections.
4. **File Permissions**:
    - chmod: Changes file permissions.
    - chown: Changes file ownership.

- **Why Linux Commands are Important?**
    - Efficiently manage system resources.
    - Troubleshoot networking and system issues.
    - Automate tasks using scripts.

Apparatus/Software Required

1. Linux-based system (Ubuntu, Fedora, etc.).
2. Python/Java/C compiler for socket programming.
3. Text editor (e.g., Vim, Nano, VSCode).
4. Network connectivity.

Procedure

Socket Programming

1. **Server-Side Steps**:
    - Create a socket using socket() function.
    - Bind the socket to an IP address and port using bind().
    - Listen for incoming connections using listen().
    - Accept a connection using accept().
    - Exchange data using send() and recv().
2. **Client-Side Steps**:
    - Create a socket using socket().
    - Connect to the server using connect().
    - Exchange data using send() and recv().

Basic Linux Commands

1. Open a terminal and test basic file management commands:
    - Create directories and files using mkdir and touch.
    - Move files using mv.

Faculty signature: ……..

- Copy files using cp.

2. Use process management commands:
    - Check active processes using ps.
    - Kill a process using kill with the process ID.
3. Test network connectivity:
    - Use ping to check if a remote host is reachable.
    - View active network connections using netstat.

Observations/Results

1. Successfully established a connection between client and server using sockets.
2. Verified two-way communication by exchanging messages.
3. Understood how Linux commands simplify tasks like file management and process control.

Discussion/Analysis

1. Socket programming highlights the importance of protocols like TCP and UDP.
2. Understanding socket functions (bind, listen, accept, etc.) is crucial for building networked applications.
3. Linux commands like ping and netstat are essential tools for diagnosing network issues.

Conclusion

This experiment provided a hands-on understanding of socket programming and demonstrated the practical use of basic Linux commands. The concepts learned are fundamental for network communication and system management.

Viva Questions

1. What is the difference between TCP and UDP?
2. Explain the significance of the bind() function in socket programming.
3. How does the chmod command change file permissions?

References

1. "Unix Network Programming" by W. Richard Stevens.
2. Linux man pages (man socket, man chmod).
3. Online resources like GeeksforGeeks and TutorialsPoint for Linux commands.

Faculty signature: .....

EXPERIMENT NO. - 3

Experiment Title

Experiment 3: **Study of Various Types of Connectors.**

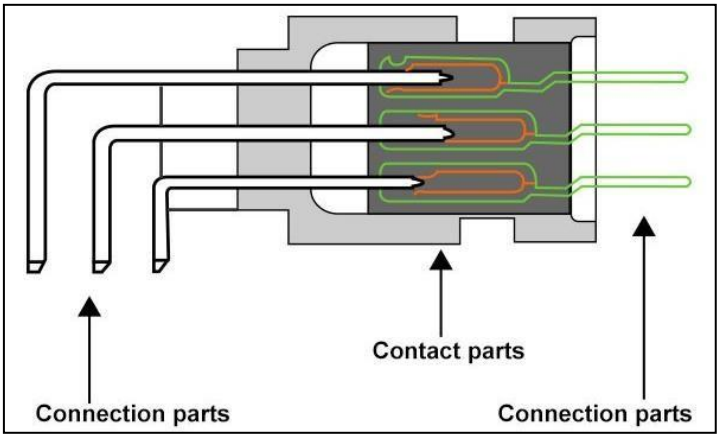
Objective

To identify and understand the different types of connectors used in networking, their structure, functionalities, and applications.

Theory/Background

What are Connectors?

Connectors are physical devices that link cables to network devices, enabling data transmission. They ensure secure and efficient connections in various networking setups. The type of connector used depends on the cable type and network requirements.



Types of Connectors

1. Board-to-Board connectors:

Description:
Board-to-board connectors are used to connect two Printed Circuit Boards (PCBs) without the need for additional cables. They are engineered with precise alignment and contact points to ensure a reliable mechanical and electrical connection. They are typically modular, allowing for customization to fit specific designs.

Applications:

- Compact Electronics: Used in compact electronic devices like smartphones and tablets.
- Embedded Systems: Facilitates connections in embedded systems, such as IoT devices and controllers.
- Industrial Equipment: Ensures communication between PCBs in complex machinery.
- Consumer Electronics: Found in TVs, gaming consoles, and other layered circuit designs.

2. Cable-to-Cable connectors:

Faculty signature: .....

- **Description**:
  Cable-to-board connectors are designed to link a cable to a PCB. They ensure secure connectivity while maintaining signal integrity. These connectors often include locking mechanisms to prevent accidental disconnection.

- **Applications**:

  - Consumer Electronics: Connects displays, batteries, or external ports to the PCB.
  - Automotive Systems: Links sensors or external wiring to car control units.
  - Medical Devices: Provides connections for cables in diagnostic and monitoring equipment.
  - Aerospace: Enables high-reliability connections in spacecraft and avionics systems.

3. Cable-to-Board connectors:

- **Description**:
  Cable-to-board connectors are designed to link a cable to a PCB. They ensure secure connectivity while maintaining signal integrity. These connectors often include locking mechanisms to prevent accidental disconnection.

- **Applications**:

  - Consumer Electronics: Connects displays, batteries, or external ports to the PCB.
  - Automotive Systems: Links sensors or external wiring to car control units.
  - Medical Devices: Provides connections for cables in diagnostic and monitoring equipment.
  - Aerospace: Enables high-reliability connections in spacecraft and avionics systems.

4. Chip to package connector:

- **Description**:
  Chip-to-package connectors are designed to establish connections between a semiconductor chip and its package (or substrate). These connectors are micro-scale and facilitate both mechanical support and electrical coupling.

- **Applications**:

  - Microprocessors: Ensures connections between CPU chips and their substrate.
  - Integrated Circuits (ICs): Links ICs to PCBs or other systems.
  - Memory Modules: Used in RAM and flash storage connections.
  - Wearable Devices: Miniaturized connectors for compact and lightweight designs.

5. Input and output connector:

- **Description**:
  Input/output (I/O) connectors enable connections between external devices and a system. They include ports such as USB, HDMI, VGA, and audio jacks, allowing for the transfer of data, power, or signals.

- **Applications**:

  - Computers and Peripherals: Connects keyboards, mice, printers, and monitors.
  - Home Entertainment: Links TVs, speakers, and gaming consoles.
  - Industrial Automation: Enables external sensors and actuators to communicate with control systems.
  - Medical Equipment: Connects imaging systems, diagnostic tools, and external interfaces.

Why Study Connectors?

1. Connectors are integral to establishing physical network connections.
2. Different applications and environments require specific connector types.

Faculty signature: .....

3. Understanding connectors ensures compatibility and optimized network performance.

Apparatus/Software Required

1. RJ45 connectors and crimping tool.
2. Fiber optic connectors (SC, LC, ST).
3. Coaxial cables and connectors (BNC, F-Type).
4. USB cables and connectors.
5. Network devices (switches, routers, PCs).

Procedure

1. **Identify the Connectors**:
    - Examine the physical structure of RJ45, RJ11, fiber optic, and coaxial connectors.
    - Note the differences in pin configuration and usage.
2. **Test RJ45 Connectors**:
    - Use a crimping tool to attach RJ45 connectors to Cat5e/Cat6 cables.
    - Test the connection using a cable tester.
3. **Inspect Fiber Optic Connectors**:
    - Observe the SC, LC, and ST connectors for their design and compatibility with fiber cables.
    - Test connectivity using a fiber network device.
4. **Coaxial Connector Setup**:
    - Connect a coaxial cable to a device using BNC or F-Type connectors.
    - Verify signal strength using a TV or broadband modem.
5. **USB Connector Usage**:
    - Connect USB peripherals to a computer and observe functionality.

Observations/Results

1. RJ45 connectors successfully transmitted data over twisted-pair cables.
2. Fiber optic connectors demonstrated high-speed, lossless data transfer.
3. Coaxial connectors provided strong signals for TV and broadband devices.
4. USB connectors supported seamless data transfer and device connectivity.

Discussion/Analysis

1. RJ45 connectors are standard for Ethernet networks due to their simplicity and compatibility.
2. Fiber optic connectors are crucial for high-speed, long-distance communication.

Faculty signature: ……..

3. Understanding coaxial and USB connectors helps in diverse network and device setups.

Conclusion

This experiment provided practical knowledge of networking connectors and their specific applications. RJ45 and fiber optic connectors are fundamental for modern networks, while coaxial and USB connectors support specialized applications.

Viva Questions

1. What is the difference between RJ45 and RJ11 connectors?
2. Why are fiber optic connectors preferred for high-speed communication?
3. Explain the wiring standards (T568A and T568B) used with RJ45 connectors.

References

1. "Network Cabling Handbook" by John J. McNamara.
2. Online resources like Cisco Networking Academy and TutorialsPoint.

Faculty signature: ……..

EXPERIMENT NO. - 4

Experiment Title

Experiment 4: **LAN Installations and Configuration.**

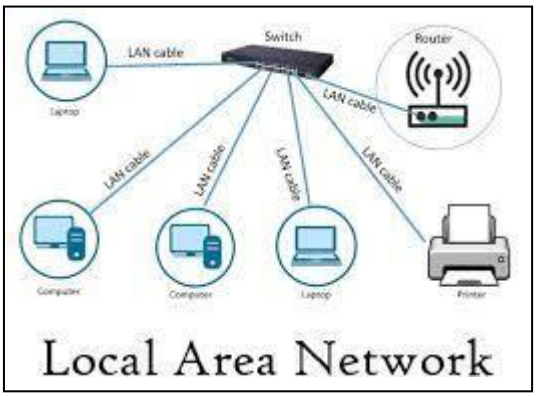
Objective

To set up and configure a Local Area Network (LAN) by connecting multiple devices, assigning IP addresses, and ensuring network connectivity for efficient data sharing and communication.

Theory/Background

What is a LAN?

A Local Area Network (LAN) is a network that connects devices within a limited area, such as an office, home, or campus. LANs are designed to share resources like printers, files, and internet connections among connected devices.



Components of LAN

1. **Switch**: Connects devices within the LAN and directs data packets based on MAC addresses.
2. **Router**: Provides internet access and interconnects different networks.
3. **Ethernet Cables**: Physical medium used to connect devices to the network.
4. **Computers/Devices**: Endpoints that communicate over the network.
5. **Network Interface Card (NIC)**: Ensures a device can communicate over the network.

Why Configure a LAN?

1. **Resource Sharing**: Enables shared access to printers, files, and applications.
2. **Efficient Communication**: Facilitates quick communication through emails or messaging systems within the network.
3. **Centralized Management**: Easier administration and monitoring of connected devices.

Apparatus/Software Required

Faculty signature: .....

1. Network Switch
2. Router
3. Ethernet Cables (Cat5e or Cat6)
4. PCs or Laptops with NICs
5. IP Addressing Scheme (Static or Dynamic using DHCP)
6. Network Configuration Tools (e.g., Windows Network Settings, Linux ifconfig/Netplan commands)

Procedure

1. **Hardware Setup**:
    - Connect the router to the switch using an Ethernet cable.
    - Connect all devices (PCs/laptops) to the switch using Ethernet cables.
2. **Assign IP Addresses**:
    - **Static IP**:
        - Open network settings on each device.
        - Manually enter an IP address, subnet mask, gateway, and DNS.
    - **Dynamic IP (DHCP)**:
        - Configure the router to act as a DHCP server.
        - Devices will automatically receive IP addresses upon connection.
3. **Verify Connectivity**:
    - Ping other devices in the LAN to test connectivity.
    - Example: ping 192.168.1.2 from a terminal or command prompt.
4. **Share Resources**:
    - Enable file and printer sharing on the devices.
    - Access shared resources using the device's network name or IP address.
5. **Test Internet Access** (if required):
    - Ensure the router is connected to the internet.
    - Verify all devices can browse the internet.

Observations/Results

1. Devices successfully connected within the LAN can communicate using IP addresses.
2. Resources such as files and printers were accessible across devices.
3. Internet access was successfully shared via the router.

Discussion/Analysis

Faculty signature: ……..

1. Static IP addressing is more secure and predictable but requires manual configuration.
2. Dynamic IP addressing using DHCP simplifies management in larger networks.
3. Effective LAN setup ensures seamless resource sharing and connectivity.

Conclusion

This experiment demonstrated the setup and configuration of a LAN, including IP address assignment, connectivity verification, and resource sharing. It emphasized the practical application of networking principles in local environments.

Viva Questions

1. What is the difference between a LAN and a WAN?
2. Why is DHCP preferred for large networks?
3. How can you troubleshoot connectivity issues in a LAN?

References

1. "Computer Networking: A Top-Down Approach" by Kurose and Ross.
2. Online resources such as Cisco Networking Academy and TutorialsPoint.

Faculty signature: ……..

EXPERIMENT NO. - 5

Experiment Title

Experiment 5: **Implementation of Various Error-Correcting Techniques.**

Objective

To study and implement various error-correcting techniques used in data transmission to detect and correct errors for reliable communication.

Theory/Background

What are Error-Correcting Techniques?

Error-correcting techniques are methods used to detect and correct errors that may occur during the transmission of data over a network. These techniques ensure data integrity and reliability in communication systems.

Types of Errors in Data Transmission

1. **Single-bit Error**: An error affecting only one bit in a data packet.
2. **Burst Error**: Multiple bits in a packet are altered during transmission.

Popular Error-Correcting Techniques

1. **Parity Check**:
    - Adds an extra parity bit to the data.
    - **Even Parity**: The total number of 1s in the data, including the parity bit, is even.
    - **Odd Parity**: The total number of 1s in the data, including the parity bit, is odd.
2. **Checksum**:
    - Computes a checksum value for the data and appends it to the packet.
    - The receiver recalculates the checksum to verify data integrity.
3. **Cyclic Redundancy Check (CRC)**:
    - Uses polynomial division to generate a CRC code appended to the data.
    - The receiver checks for remainders during division to detect errors.
4. **Hamming Code**:
    - Adds redundant bits (parity bits) to detect and correct errors.
    - Can detect up to two errors and correct a single-bit error.
5. **Reed-Solomon Code**:
    - Uses mathematical formulas for error correction, effective in burst error correction.

Faculty signature: .....

- Widely used in storage devices and communication protocols.

Apparatus/Software Required

1. A PC or laptop with a programming environment (Python, C, or Java).
2. Network simulation tools (e.g., Cisco Packet Tracer).
3. Sample data packets for testing.

Procedure

1. **Parity Check Implementation**:
   - Write a program to generate even/odd parity bits for given data.
   - Send data with parity bits and verify them at the receiver end.
2. **Checksum Calculation**:
   - Implement a program to calculate checksum values for data blocks.
   - Validate the data by recalculating the checksum at the destination.
3. **CRC Implementation**:
   - Use a polynomial divisor to calculate the CRC code.
   - Verify received data by recalculating CRC and checking for remainders.
4. **Hamming Code Implementation**:
   - Write a program to generate Hamming Code for given data.
   - Simulate a single-bit error in transmission and correct it using the received Hamming Code.
5. **Simulation of Errors**:
   - Introduce single-bit and burst errors in transmitted data.
   - Use error-correcting techniques to detect and correct them.

Observations/Results

1. Parity Check detected single-bit errors but failed for burst errors.
2. Checksum and CRC provided robust error detection for multiple-bit errors.
3. Hamming Code successfully corrected single-bit errors in data packets.
4. Reed-Solomon code proved effective in correcting burst errors.

Discussion/Analysis

1. Parity Checks are simple but insufficient for complex error patterns.
2. CRC is widely used in network communication due to its efficiency and reliability.
3. Hamming Code is optimal for applications where single-bit error correction suffices.

Faculty signature: ……..

4. Error-correcting codes like Reed-Solomon are essential for high-reliability systems such as CDs, DVDs, and satellite communications.

Conclusion

This experiment provided insights into the implementation of various error-correcting techniques and their effectiveness in different error scenarios. Understanding these techniques is crucial for designing robust communication systems.

Viva Questions

1. How does CRC differ from a simple checksum?
2. What are the limitations of a parity check?
3. Explain how Hamming Code corrects a single-bit error.
4. Why is Reed-Solomon widely used in data storage systems?

References

1. "Data and Computer Communications" by William Stallings.
2. Online resources such as GeeksforGeeks and TutorialsPoint for coding examples.

Faculty signature: .....

EXPERIMENT NO. - 6

Experiment Title

Experiment 6: **Implementation of Various Data Link Layer (DLL) Protocols.**

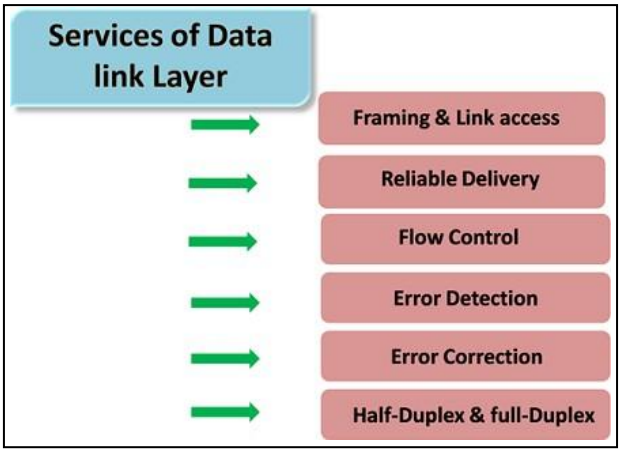
Objective

To study and implement different Data Link Layer protocols to understand their functionality in error detection, error correction, and efficient data transfer.

Theory/Background

What is the Data Link Layer (DLL)?

The Data Link Layer is the second layer in the OSI model, responsible for node-to-node data transfer, framing, error detection, and correction. It operates between the Network Layer and the Physical Layer, ensuring reliable communication.



Key Functions of DLL

1. **Framing**: Encapsulates network layer data into manageable frames.
2. **Error Control**: Detects and corrects errors during transmission.
3. **Flow Control**: Manages data rate between sender and receiver to prevent congestion.

Common DLL Protocols

1. **Synchronous Data Link Control (SDLC):**

   SDLC is basically a communication protocol of computer. It usually supports multipoint links even error recovery or error correction also. It is usually used to carry SNA (Systems Network Architecture) traffic and is present precursor to HDLC. It is also designed and developed by IBM in 1975. It is also used to connect all of the remote devices to mainframe computers at central locations may be in point-to-point (one-to-one) or point-to-multipoint (one-to-many) connections. It is also used to make sure that the data units should arrive correctly and with right flow from one network point to next network point.

Faculty signature: .....

- **I/O Channel Communication link –**
  These links are basically hard-wired among all co-located nodes. They also usually operate at almost high rate of data and generally error-free.

- **Non-I/O Channel Communication Link –**
  These links generally use some form of data communications equipment to transmit data among several nodes. SDLC protocol is basically required to ensure error-free performance over all these links.

SDLC also supports various types of architectures as given below :

- Normal Disconnected Mode (NDM) and Normal Response Mode (NRM)
- Two-way alternate (half-duplex) data flow
- Secondary station point-to-point, multipoint, and multi-multipoint configurations
- Primary station point-to-point and multipoint configurations
- Modulo 8 transmit-and-receive sequence counts
- Nonextended (single-byte) station address

Types of network nodes in SDLC :
SDLC generally identifies and describes two types of Network nodes as given below :

1. **Primary Node –**

The primary node is generally responsible for handling all secondary nodes and also responsible for controlling all operations and links. Data is processed only through this node. This node polls secondaries in predetermined order and then secondaries can then send if they have any data to be sent. This node also sets up and tears down links and manages link while it is operational.

2. **Secondary Node –**

A secondary node is generally responsible for sending all of data that is being received to primary node. This node is controlled by primary node i.e., secondaries will only transmit data to primary only if primary allows and grants permission to do so.

Basic Configurations followed by SDLC nodes :
SDLC primary and secondary node generally get connected in four different basic configurations as given below :

1. **Point-to-Point –**

As the name suggests, in Point-to-Point configuration, there are only two nodes. One node is primary and other one is secondary node.

2. **Multipoint –**

As the name suggests, in Multipoint configuration, there are multiple nodes. One node is primary and all of other are secondary nodes. It is also known as multidrop configuration. In this, secondaries are polled separately in predefined sequence.

3. **Loop –**

As the name suggests, in Loop configuration, primary node is connected to first secondary node and last secondary node in loop. In simple words, primary node is connected to secondary nodes and has two nodes on either side. In this, intermediate secondary nodes transmit data through one another as they give responses to primary requests.

4. **Hub go-ahead –**

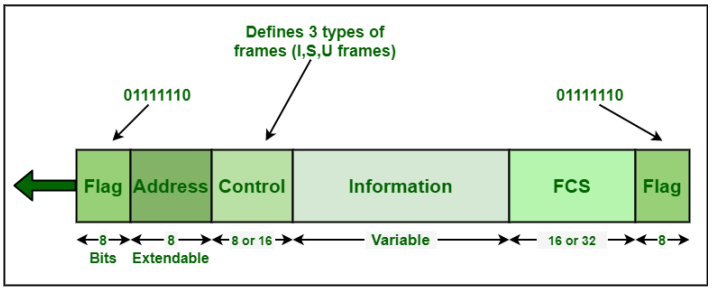
This configuration generally involves inbound and outbound channels. The primary node simply requires outbound channel to transmit data with secondary node whereas secondary node simply requires inbound channel to transmit data with primary node.

Faculty signature: ……..

2. **High-Level Data Link Control (HDLC)**:

High-Level Data Link Control (HDLC) generally uses term “frame” to indicate and represent an entity of data or a protocol of data unit often transmitted or transferred from one station to another station. Each and every frame on link should begin and end with Flag Sequence Field (F). Each of frames in HDLC includes mainly six fields. It begins with a flag field, an address field, a control field, an information field, an frame check sequence (FCS) field, and an ending flag field. The ending flag field of one frame can serve as beginning flag field of the next frame in multiple-frame transmissions.

The basic frame structure of HDLC protocol is shown below :



Basic Frame Structure

3. **Serial Line Internet Protocol**:

It is a TCP/IP implementation which was described under **RFC 1055 (Request for Comments)**. SLIP establishes point to point serial connections which can be used in dial-up connections, serial ports and routers. It frames the encapsulated IP packets across a serial line for establishing connection while using line speed between 12000 bps and 19.2 Kbps.

SLIP was introduced in 1984 when Rick Adams used it to connect 4.2 Berkeley Unix and Sun Microsystems workstations. It soon caught up with the rest of the world as a credible TCP/IP implementation. It has now become obsolete after being replaced by PPP (Point to Point Protocol) which solves many deficiencies present in it.

Characteristics

1. It introduces two special characters END(decimal 192) and ESC(decimal 129). Depending, on whether data byte code represents END or ESC character, the two byte sequence of ESC and octal 334 or ESC and octal 335 respectively is sent in data packet.

Faculty signature: .....

2. There is no maximum packet size in SLIP since it has no standard specification. However, the widely accepted value is 1006 bytes of datagram for both sending and receiving.

3. The sender and receiver should be aware of IP address for both ends while using SLIP.

4. It only supports static assignment during IP addressing.

5. It transfers data in synchronous form.

6. A SLIP frame consists of a payload (data) and a flag to act as a end delimiter.

Advantages

1. It can allow different combinations of network configurations such as host-host, host-router, router-router etc.

2. It can be easily used in microcontrollers because of small overhead.

3. It is easy to implement being a basic packet protocol and due to wide application of TCP/IP.

Disadvantages

1. It does not perform any authentication of data and IP addresses cannot be dynamically assigned while using SLIP.

2. SLIP provides no type identification method. The type of protocol sent cannot be detected. Hence, only one protocol can run over a SLIP connection.

3. It has no error detection or correction mechanism in data transmission.

4. A SLIP connection provides no mechanism for hosts to communicate addressing information.

5. SLIP provides no compression features to improve packet throughput. CSLIP was a variant used for same purpose but it could not achieve wide application.

4. Point-to-Point Protocol (PPP)

PPP stands for Point-to-Point Protocol. PPP is Windows default Remote Access Service (RAS) protocol and is Data Link Layer (DLL) protocol used to encapsulate higher network layer protocols to pass through synchronous and asynchronous lines of communication. Initially created as an encapsulation protocol to carry numerous layer of network traffic over point-to-point connections. In addition, PPP settled various measures including asynchronous and bit-oriented synchronous encapsulation, multiplexing of network protocols, negotiation of sessions, and negotiation of data-compression. PPP also supports non TCP / IP protocols, such as IPX / SPX and DECnet. A prior standard known as Serial Link Internet Protocol (SLIP) has been largely replace by it.

What is the PPP Authentication Protocol?

PPP supports two main authentication protocols:PPP supports two main authentication protocols:

- **Password Authentication Protocol (PAP):** A basic authentication scheme where the user's identity information such as the user name and password are transmitted in the plain format. Its disadvantage is that it is less secure because of the simplicity.
- **Challenge Handshake Authentication Protocol (CHAP):** CHAP is more secure as it periodically alert the identity of the remote client by using a three way hand shake method. The password is encrypted, thus it is much more secure than plaintext PAP.

Which Protocol is Replaced by PPP?

PPP took the place of the Serial Line Internet Protocol (SLIP). SLIP was one of the older protocols that was used for establishing connections for serial means to provide P-P communication contacts but it had some issues such

as, it did not support features like error detection and link multiplexing. PPP has certain advancements over the conventional protocols with respect to error detection, multiplexing and authentication and therefore is more appropriate for use on the modern internet connections.

Services Provided by PPP

PPP offers several essential services, including:PPP offers several essential services, including:

- **Data Encapsulation:** It enshrines network layer protocols such as IP and others for transmission over serial link.
- **Link Control Protocol (LCP):** It is employed to set up, initial and finalize the data link connection.
- **Network Control Protocols (NCP):** PPP has provisions for supporting many NCPs for the configuration of the network-layer protocols such as IP, IPX, and AppleTalk.
- **Authentication:** Offers both PAP and CHAP for the purpose of authentication.
- **Error Detection:** It also has methods of maintaining integrity of the data through error checking techniques.

History of Point-to-Point Protocol (PPP)

PPP returns to late 1980s, when true standard for sequential IP executions was SLIP. RFC 1134, distributed in 1989, was principal formal IETF report identified with PPP. This RFC isn't just standard yet proposal for what could be characterized as primary PPP standard, RFC 1171, in 1990. This early report has been re-examined numerous times and included few different records characterizing different protocols that contain entire PPP suite. Instead of trying to create PPP from scratch, IETF built PPP on basis of ISO High-Level Data Link Control (HDLC) protocol, which was initially developed by IBM. Developers of PPP adopted its framing mechanism from HDLC protocol and component of its general operation.

Features of PPP

- **Packet Framing –** Network layer data packet formulation within data link block.
- **Multi-Protocol –** Yield information from any NCP network layer upwards at same time as demultiplex.
- **Bit Transparency –** Should carry certain bit pattern in field of data.
- **Error Detection –** No modification.

Components of PPP

It uses three components to allow PPP to transmit data over serial point-to – point link. Each part has its own autonomous role and entails use of other two complete its tasks.

These three components are :

- **High-Level Data-Link Control (HDLC) protocol –** HDLC is method used to frame data over PPP links. On account of PPP, the standard version of OSI is used instead of proprietary version of Cisco. This standardization assists in ensuring that different vendors can properly communicate PPP executions.
- **Link Control Protocol (LCP) –** It is liable for formulating, configuring, testing, sustaining and terminating transmission links. Additionally, two endpoints of connections impart negotiation for setting up of alternatives and use of features.

Faculty signature: .....

- **Network Control Protocols (NCPs) –** NCP frames are used to communicate and customize protocols on Network layer that can be used over PPP session. There is one NCP for every higher-layer protocol that is upheld by PPP. NCPs enable PPP to work over analogous connection in consonance with many Network layer protocols.

Working of PPP

PPP jointly uses these three components to enable communication. There are four main steps to establish,

maintain and terminate PPP session:

- **Step-1:** Initial step of setting up PPP session between devices includes both sending LCP link-establishment frame for configuration and testing purposes. Such frames also characterize which alternatives, for instance compression, authentication, and multilink, given PPP host chooses. If authentication is established and needed it will take place during this step.
- **Step-2:** It uses LCP frames to test link 's nature. Assembled data can be used to evaluate if links is appropriate for dealing different protocols on upper layer.
- **Step-3:** NCP frames are sent over link to determine which network layer protocols need configuration. For instance, connection to use IP, IPX, AppleTalk and so on can need to be optimized.
- **Step-4:** In this step, when ending PPP session, LCP link-termination frames are used to cut connection. Third category of LCP frame (Link-Maintenance) is often used for leveraging and troubleshooting PPP links.

Advantages of PPP

- **Supports Multiple Protocols:** PPP can support various network layer protocol hence it is well suitable for any specific network.
- **Error Detection:** Such features as built-in error detection make it possible to have reliable data transmission.
- **Authentication:** These support features secure connection establishment with authentication protocols.
- **Compression:** PPP much backs data compression with the intention of improving the transmission rates.

Disadvantages of PPP

- **Overhead:** PPP brings about overhead since it involves packaging and managing different protocols.
- **Limited to Point-to-Point:** As it is clear form the name, PPP is only able to support direct point to point connections, and hence it is not very useful in large networks.
- **Complex Configuration:** PPP also requires configuration and management of the links which may make the configuration and management of PPP a bit tough compared to the other protocols like Ethernet.

5. **Link Control Protocol (LCP) –**
   It was originally developed and created by IEEE 802.2. It is also used to provide HDLC style services on LAN (Local Area Network). LCP is basically a PPP protocol that is used for establishing, configuring, testing, maintenance, and ending or terminating links for transmission of data frames.
6. **Link Access Procedure (LAP) –**
   LAP protocols are basically a data link layer protocols that are required for framing and transferring data across point-to-point links. It also includes some reliability service features. There are basically three types of LAP i.e. LAPB (Link Access Procedure Balanced), LAPD (Link Access Procedure D-Channel), and LAPF (Link Access Procedure Frame-Mode Bearer Services). It is actually originated from IBM SDLC, which is being submitted by IBM to the ISP simply for standardization.
7. **Network Control Protocol (NCP) –**
   NCP was also an older protocol that was implemented by ARPANET. It basically allows users to have access

Faculty signature: ……..

to use computers and some of the devices at remote locations and also to transfer files among two or more computers. It is generally a set of protocols that is forming a part of PPP. NCP is always available for each and every higher-layer protocol that is supported by PPP. NCP was replaced by TCP/IP in the 1980s.

Apparatus/Software Required

1. Programming Environment (Python, C, or Java).
2. Simulation Tools (e.g., Packet Tracer, NS3).
3. A PC or Laptop with network simulation capability.

Procedure

Stop-and-Wait ARQ

1. Write a program where the sender sends a frame and waits for an acknowledgment.
2. Simulate error by skipping an acknowledgment to observe retransmission.

Go-Back-N ARQ

1. Implement a program with a sliding window mechanism for sending multiple frames.
2. Simulate error in one frame and observe the retransmission of all subsequent frames.

Selective Repeat ARQ

1. Modify the Go-Back-N implementation to allow retransmission of only erroneous frames.
2. Use unique sequence numbers for each frame to manage selective acknowledgment.

Sliding Window Protocol

1. Implement a sender-receiver communication system with a dynamic window size.
2. Allow for continuous transmission and acknowledgment of data within the window.

Observations/Results

1. Stop-and-Wait ARQ was simple but inefficient due to high idle time.
2. Go-Back-N ARQ improved throughput but incurred overhead for retransmitting all frames after an error.
3. Selective Repeat ARQ optimized retransmissions but required additional memory and complexity.
4. Sliding Window Protocol provided seamless data transfer and better resource utilization.

Discussion/Analysis

1. **Efficiency**: Sliding Window Protocol is the most efficient due to continuous data flow.
2. **Complexity**: Selective Repeat ARQ is complex but minimizes retransmissions, making it suitable for high-speed networks.
3. **Applications**:

Faculty signature: .....

- Stop-and-Wait is ideal for low-speed, error-free channels.
- Sliding Window is widely used in protocols like TCP for reliable data transfer.

Conclusion

This experiment provided a practical understanding of Data Link Layer protocols and their effectiveness in managing reliable data transfer, error control, and flow control.

Viva Questions

1. Why is Stop-and-Wait ARQ considered inefficient?
2. How does Go-Back-N ARQ handle errors compared to Selective Repeat ARQ?
3. Explain the role of window size in Sliding Window Protocol.
4. Which DLL protocol is most commonly used in modern communication systems and why?

References

1. "Data Communications and Networking" by Behrouz A. Forouzan.
2. Online tutorials and documentation from GeeksforGeeks and TutorialsPoint.

Faculty signature: .....

EXPERIMENT NO. - 7

Experiment Title

Experiment 7: Communication Between a Web Browser (Client) and Web Server: Displaying Server's Date and Time in the Web Browser.

Objective

To establish communication between a web browser (client) and a web server, retrieve the server's current date and time, and display it on the web browser.

Theory/Background

Socket Programming

Socket programming is a way to enable communication between two devices (e.g., a client and a server) over a network. A socket is a software structure that allows data exchange through the network.

- **Client-Server Model**: In this model, the client sends a request to the server, and the server processes the request and sends back a response.
- **TCP Socket**: A connection-oriented communication method that ensures reliable data transmission.

Steps Involved

1. **Server Creation**: The server listens for incoming connections on a specified port. Upon receiving a connection request, the server sends a response (in this case, the current date and time) to the client.
2. **Client Creation**: The client (web browser) sends a request to the server. Once the server responds, the client displays the server's data.

Communication Process

1. **Web Browser (Client)**: It initiates the connection to the server using an HTTP request.
2. **Web Server**: Listens for requests, processes them, and sends a response containing the current date and time.
3. **Socket Communication**: The client and server communicate using a socket, with the client sending an HTTP request and the server returning the response with the current date and time.

Apparatus/Software Required

1. **Python** (for creating socket programming code).
2. **Web Browser** (for displaying server's response).
3. **Text Editor** (e.g., VS Code, Sublime Text) for writing code.
4. **Operating System**: Any OS supporting socket programming (Linux/Windows).

Faculty signature: .....

Procedure

Server-Side Program (Python)

1. **Create a server that listens on a specific port (e.g., port 8080).**
2. **Accept client connections** and send the current date and time to the connected client.

python

Copy code

```
import socket
from datetime import datetime

# Create a socket object
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.bind(('127.0.0.1', 8080))  # Bind to localhost and port 8080
server_socket.listen(1)  # Listen for one connection

print("Server is running on port 8080...")

while True:
    client_socket, client_address = server_socket.accept()  # Accept connection
    print(f"Connection from {client_address} established!")

    # Send current date and time to the client
    current_time = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    client_socket.sendall(f"HTTP/1.1 200 OK\nContent-Type: text/plain\n\nCurrent Date and Time: {current_time}".encode())

    client_socket.close()  # Close the connection
```

Client-Side Program (Web Browser)

1. **Open a web browser** and navigate to the server's address (e.g., http://127.0.0.1:8080).
2. **The client sends an HTTP request** to the server, and the server responds with the current date and time.

The client in this case is the web browser that makes an HTTP request to the server (localhost in this case). The server responds with the current date and time, which the browser displays.

Observations/Results

Faculty signature: .....

1. The server receives a connection from the client (web browser).
2. The current date and time from the server are displayed correctly in the browser after the server responds.
3. Example output in the web browser:
    - "Current Date and Time: 2024-12-08 14:30:22"

Discussion/Analysis

1. **Socket Communication**: This experiment demonstrates how socket programming can be used for client-server communication over a network. The server sends real-time data (the current date and time) in response to a client request.
2. **HTTP Request-Response Model**: The interaction follows the HTTP protocol for client-server communication, and the client can receive dynamic data from the server in real time.

Conclusion

This experiment demonstrated the basics of socket programming and how a client (web browser) can request and receive real-time data from a server. The server successfully communicated its current date and time to the client using a socket-based communication model.

Viva Questions

1. What is socket programming, and why is it used in network communication?
2. How does the client-server model work in socket programming?
3. What is the role of the HTTP protocol in client-server communication?
4. How does the server determine the current date and time to send to the client?
5. Explain the importance of the bind() and listen() methods in the server-side code.

References

1. "Python Networking Programming" by John Goerzen.
2. Python documentation for socket library: https://docs.python.org/3/library/socket.html.
3. "Computer Networking: A Top-Down Approach" by James Kurose and Keith Ross.

Faculty signature: .....

EXPERIMENT NO. - 8

Experiment Title

Experiment 8: **Network Communication Model with IP Address Transfer from Client to Server.**

Objective

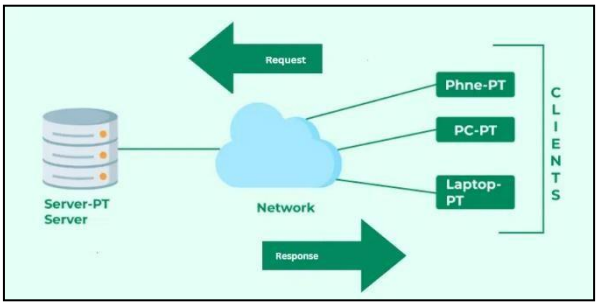
To establish a connection between a client and a server over a network, transfer the client's IP address to the server, and display the received IP address on the server side.

Theory/Background

Socket Programming in Networking

Socket programming is essential for network communication. It allows two devices to communicate over a network using IP addresses and ports. The client sends a request, and the server responds by processing that request. This experiment will demonstrate how the client's IP address can be sent to the server and displayed.

- **Socket**: A socket is the endpoint of a two-way communication link between two programs running on a network. It is used to send and receive data.
- **TCP/IP**: Transmission Control Protocol/Internet Protocol (TCP/IP) is the foundational communication protocol of the internet. It enables the exchange of data across different devices connected to a network.
- **Client-Server Communication**: The client initiates a connection to the server, sends a request (in this case, its IP address), and waits for the server to process and respond.



Communication Process

1. **Client Side**: The client connects to the server and retrieves its own IP address using the socket's properties. The client sends this IP address to the server.
2. **Server Side**: The server listens for incoming connections, accepts them, and processes the received IP address. It then displays this IP address for confirmation.

Apparatus/Software Required

1. **Python Programming Language** for socket programming.
2. **Text Editor** (e.g., Sublime Text, VS Code) to write the code.

Faculty signature: .....

3. **Operating System**: Linux or Windows supporting socket programming.
4. **Network Simulation Software** (optional): Packet Tracer or any network simulator for testing.

Procedure

Server-Side Code (Python)

1. **Create a server program** that listens on a specific port (e.g., port 12345).
2. **Accept client connections** and retrieve the IP address sent by the client.
3. **Display the received IP address** on the server side.

python

Copy code

```
import socket

# Create a socket object
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Bind the socket to localhost and port 12345
server_socket.bind(('127.0.0.1', 12345))

# Enable the server to listen for incoming connections
server_socket.listen(5)
print("Server listening on port 12345...")

# Accept connection from the client
client_socket, client_address = server_socket.accept()
print(f"Connection established with client: {client_address}")

# Receive and display the client's IP address
client_ip = client_address[0]  # The IP address of the client
print(f"Client IP Address: {client_ip}")

# Close the connection
client_socket.close()
```

Client-Side Code (Python)

1. **Create a client program** that connects to the server.

Faculty signature: ……..

2. **Retrieve the client's own IP address**.
3. **Send the IP address to the server**.

python

Copy code

```
import socket

# Create a socket object
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Connect to the server at localhost, port 12345
client_socket.connect(('127.0.0.1', 12345))

# Get the client's own IP address
client_ip = socket.gethostbyname(socket.gethostname())

# Send the client's IP address to the server
client_socket.send(client_ip.encode())

# Close the connection
client_socket.close()
```

Observations/Results

1. The client successfully connects to the server and retrieves its own IP address.
2. The client sends its IP address to the server, which is displayed on the server side.
3. Example output on the server side:
    - "Client IP Address: 127.0.0.1"

Discussion/Analysis

1. **Server's Role**: The server listens for incoming connections and processes the client's request. It successfully receives the IP address and displays it.
2. **Client's Role**: The client retrieves its own IP address and sends it to the server. This demonstrates how sockets can be used to exchange IP addresses in network communication.
3. **Socket Functionality**: By utilizing the socket library in Python, both the client and server can establish communication over the network. This experiment shows how the client-server model works in a practical scenario.

Faculty signature: .....

Conclusion

This experiment demonstrated the basic functionality of socket programming, with the client sending its own IP address to the server. It provided insight into how data can be transferred between systems over a network using sockets, allowing real-time communication.

Viva Questions

1. What is the significance of the bind() method in the server code?
2. How does the server identify the client's IP address?
3. What is the difference between client_socket.accept() and server_socket.accept()?
4. What would happen if the client tries to connect to an incorrect server IP?
5. How can you modify the client code to send additional information to the server?

References

1. "Python Network Programming" by John Goerzen.
2. Python official documentation for the socket library: https://docs.python.org/3/library/socket.html.
3. "Computer Networking: A Top-Down Approach" by James Kurose and Keith Ross.

Faculty signature: .....

EXPERIMENT NO. - 9

Experiment Title

Experiment 9: Client-Server Communication with Message Exchange.

Objective

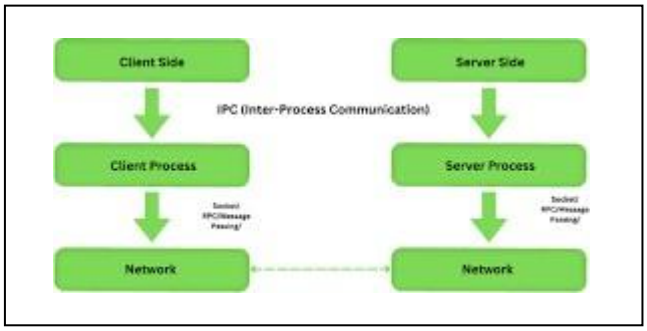
To create a client-server communication system where the client sends a message to the server, and the server responds by sending back the same message.

Theory/Background

Socket Programming

Socket programming is used for establishing communication between a client and a server over a network. This experiment focuses on how data can be sent from the client to the server and how the server processes and returns the data back to the client.

- **TCP (Transmission Control Protocol)** is typically used for this kind of communication, ensuring reliable, ordered delivery of data packets between the client and server.
- **Client-Server Communication**: The client sends a request message to the server. The server processes the message and sends the same message back to the client as a response.



Communication Process

1. **Client Side**: The client connects to the server, sends a message, and waits for a response.
2. **Server Side**: The server listens for incoming connections, receives the message, and sends the same message back to the client.

Apparatus/Software Required

1. **Python** programming language (for socket programming).
2. **Text Editor** (e.g., Sublime Text, VS Code).
3. **Operating System**: Linux or Windows with socket library support.

Faculty signature: .....

Procedure

Server-Side Code (Python)

1. **Create a server** that listens for incoming connections on a specific port (e.g., port 12345).
2. **Receive a message** from the client.
3. **Send the same message back to the client**.

python

Copy code

```
import socket


# Create a socket object
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)


# Bind the socket to localhost and port 12345
server_socket.bind(('127.0.0.1', 12345))


# Enable the server to listen for incoming connections
server_socket.listen(5)
print("Server listening on port 12345...")


# Accept connection from the client
client_socket, client_address = server_socket.accept()
print(f"Connection established with client: {client_address}")


# Receive the message from the client
client_message = client_socket.recv(1024).decode()
print(f"Received message: {client_message}")


# Send the same message back to the client
client_socket.sendall(client_message.encode())


# Close the connection
client_socket.close()
```

Client-Side Code (Python)

Faculty signature: ……..

1. **Create a client** that connects to the server.
2. **Send a message** to the server.
3. **Receive the same message** from the server.

python

Copy code

```
import socket

# Create a socket object
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Connect to the server at localhost, port 12345
client_socket.connect(('127.0.0.1', 12345))

# Send a message to the server
message = "Hello, Server!"
client_socket.sendall(message.encode())

# Receive the response from the server
server_response = client_socket.recv(1024).decode()
print(f"Server Response: {server_response}")

# Close the connection
client_socket.close()
```

Observations/Results

1. The client successfully connects to the server and sends a message.
2. The server receives the message and sends it back.
3. Example output on the client side:
    - "Server Response: Hello, Server!"

Discussion/Analysis

1. **Data Exchange**: This experiment demonstrates the basic functionality of socket programming for data exchange. The message sent by the client is returned by the server.

Faculty signature: .....

2. **TCP Communication**: The use of TCP ensures that the message is reliably transmitted, with no loss or corruption of data during the exchange.
3. **Server Response**: The server simply echoes the message, which is a common pattern used in testing and debugging network connections.

Conclusion

This experiment demonstrated the basics of socket programming for bi-directional communication between a client and a server. It shows how data can be sent from the client to the server and how the server can respond with the same data.

Viva Questions

1. How does the recv() method work in socket programming?
2. What would happen if the client sent more data than the buffer size in recv()?
3. What is the role of the encode() and decode() methods in the client-server communication?
4. Why is bind() used in the server program?
5. How would you modify the server to handle multiple clients at once?

References

1. "Python Network Programming" by John Goerzen.
2. Python documentation for socket library: https://docs.python.org/3/library/socket.html.
3. "Computer Networking: A Top-Down Approach" by James Kurose and Keith Ross.

Faculty signature: .....

EXPERIMENT NO. – 10

Experiment Title

Experiment 10: Introduction to TCL Commands.

Objective

To explore and experiment with the usage of TCL (Tool Command Language) commands and their application in network management and automation.

Theory/Background

TCL (Tool Command Language)

TCL is a high-level programming language used for automation, network management, and scripting. It is often used in testing environments, especially in network simulation tools such as Cisco Packet Tracer.

- **Scripting**: TCL allows for scripting commands to control and automate various network devices.
- **Integration**: TCL scripts can be integrated into network simulators or used in real devices to configure network settings or perform automated tasks.
- **Commands**: TCL commands include control flow commands, string manipulation, data types, and file handling, allowing for powerful automation in networking.

Common TCL Commands

1. **Control Flow**:
    - if, else: Used for decision-making in scripts.
    - while, for: Loops used for repeating commands.
2. **Data Manipulation**:
    - set: Used to assign values to variables.
    - append: Adds a value to a list or string.
    - string: Commands to manipulate strings (e.g., string length, string tolower).
3. **File Handling**:
    - open: Opens a file for reading or writing.
    - puts: Writes data to the file or console.
    - gets: Reads input from a file.
4. **Network Automation**:
    - **send**: Used in TCL-based tools like expect to send commands to network devices.
    - **expect**: Waits for specific output or prompts from network devices before continuing with the script.

Faculty signature: ……..

Apparatus/Software Required

1. **Cisco Packet Tracer** (or any other network simulator with TCL scripting support).
2. **TCL Interpreter**: Tools like ActiveTcl or Tclsh for executing TCL scripts.
3. **Basic Network Setup**: A router, switch, or other network devices that support TCL scripting for automation.

Procedure

1. **Open Cisco Packet Tracer** and create a simple network with routers, switches, and computers.
2. **Write a TCL script** to automate configuration tasks (e.g., configure IP addresses on routers).
3. **Use TCL commands** to check the status of devices, automate configurations, and perform simple troubleshooting.

Example of a simple TCL script in Packet Tracer:

tcl

Copy code

```
# Simple TCL script to configure IP address on a router

set router_ip 192.168.1.1

send "enable\r"

send "configure terminal\r"

send "interface g0/0\r"

send "ip address $router_ip 255.255.255.0\r"

send "no shutdown\r"

send "exit\r"
```

4. **Execute the script** in the network simulator to verify that the devices are configured as expected.

Observations/Results

1. The TCL script successfully configures network devices.
2. Automation of tasks using TCL commands reduces manual effort and time.
3. Example result: The router's interface is automatically configured with the specified IP address.

Discussion/Analysis

1. **Automation**: TCL provides an efficient way to automate repetitive network configurations and tasks.
2. **Network Management**: By using TCL in network simulators like Cisco Packet Tracer, administrators can script and automate device configurations, significantly improving network management efficiency.

Conclusion

Faculty signature: ……..

This experiment provided hands-on experience with TCL scripting and its application in network automation. It highlighted how TCL can simplify network configurations and improve the efficiency of network management tasks.

Viva Questions

1. What is TCL, and how is it used in networking?
2. How do you manipulate strings in TCL?
3. How can you automate router configuration using TCL commands?
4. What is the role of the send and expect commands in TCL scripts?
5. How would you troubleshoot a device using TCL scripts?

References

1. "TCL and the Tk Toolkit" by John Ousterhout.
2. Cisco Packet Tracer documentation.
3. TCL official documentation: https://www.tcl.tk/doc/.

Faculty signature: .....