**Name:** Maitri Savalia  **Roll No.:**22BCP014

**Div/Group:** D1-G1

Class B Tech CSE 3rd Year **Sub:** Computer Network

**EXPERIMENT 1**

|  |  |
| --- | --- |
| |  | | --- | | **Aim:** Simulation of Various Networking Topologies | |
| **Prerequisite:** Nil |
| **Outcome:** To impart knowledge of Computer Networking Technology |
| **Theory:**  Network topologies refer to the arrangement of various elements (links, nodes, etc.) in a computer network. The choice of topology affects the network's performance and reliability. Here are the primary types of network topologies:  **1. Bus Topology**   * Structure: All devices are connected to a single central cable, called the bus or backbone. * Advantages:   + Easy to implement and extend.   + Requires less cable compared to other topologies. * Disadvantages:   + Difficult to troubleshoot.   + Limited cable length and number of stations.   + A break in the central cable disrupts the entire network.   **2. Star Topology**   * Structure: All devices are connected to a central hub or switch. * Advantages:   + Easy to install and manage.   + Failure of one node doesn't affect the rest of the network.   + Easy to detect and troubleshoot faults. * Disadvantages:   + Requires more cable compared to bus topology.   + If the central hub fails, the entire network is disrupted.   **3. Ring Topology**   * Structure: Each device is connected to two other devices, forming a circular data path. * Advantages:   + Data is transferred in a sequential manner, reducing the chance of collision.   + Can be more efficient than bus topology in handling high traffic. * Disadvantages:   + A failure in any cable or device breaks the loop and can disrupt the entire network.   + More difficult to install and reconfigure.   **4. Mesh Topology**   * Structure: Every device is connected to every other device in the network. * Advantages:   + Highly reliable as multiple paths exist for data transmission.   + Failure of one device doesn't affect the network. * Disadvantages:   + Expensive due to the high number of cables and ports required.   + Complex to install and manage.   **5. Hybrid Topology**   * Structure: Hybrid topology is a network topology that combines elements of two or more basic topologies (like star, bus, ring, or mesh) to leverage the strengths of each. It is designed to meet specific needs and provide flexibility. * Advantages: * By combining different topologies, hybrid networks can offer greater reliability and fault tolerance. For example, a star-bus hybrid network can ensure that failure in one star network doesn’t affect the others * Disadvantages: * Designing and implementing a hybrid network can be complex due to the need to integrate multiple topologies and manage their interactions * The initial setup and configuration of a hybrid network can be more expensive compared to simpler topologies due to the need for additional hardware and configuration efforts. |
| **Procedure:**   1. Open Cisco Packet Tracer and simulate the topologies of required size. 2. Assign the IP Addresses to the system. 3. Check the connectivity between the devices. |
| **Steps:**  1. Bus Topology (Using Switch)   1. Add Devices:    * Drag the desired number of PCs onto the workspace. 2. Add Switches:    * Drag the necessary number of switches onto the workspace, arranging them in a linear sequence. 3. Connect Devices:    * Use Copper Straight-Through cables to connect each PC to the nearest switch.    * Use Copper Cross-Over cables to connect each switch to the next switch in the sequence, forming a linear bus-like structure.   2. Star Topology   1. Add Devices:    * Drag the desired number of PCs onto the workspace. 2. Add a Switch:    * Drag one switch onto the workspace, placing it at the centre. 3. Connect Devices:    * Use Copper Straight-Through cables to connect each PC to the central switch.   3. Ring Topology (Using Switches)   1. Add Devices:    * Drag the desired number of PCs onto the workspace. 2. Add Switches:    * Drag the necessary number of switches onto the workspace, arranging them in a circular layout. 3. Connect Switches:    * Use Copper Cross-Over cables to connect each switch to the next switch in the circle, and connect the last switch back to the first switch to complete the ring. 4. Connect Devices:    * Use Copper Straight-Through cables to connect each PC to one of the switches.   4. Mesh Topology   1. Add Devices:    * Drag the desired number of PCs onto the workspace. 2. Add Switches:    * Drag the necessary number of switches onto the workspace. 3. Connect Switches:    * Use Copper Cross-Over cables to connect each switch to multiple other switches. For a full mesh, every switch should be connected to every other switch. For a partial mesh, only some of the switches will be interconnected. 4. Connect Devices:    * Use Copper Straight-Through cables to connect each PC to one of the switches.   5. Hybrid Topology   1. Add Devices:    * Drag the desired number of PCs onto the workspace. 2. Add Switches:    * Drag the necessary number of switches onto the workspace, arranging them in a linear sequence. 3. Connect Devices:    * Use Copper Straight-Through cables to connect each PC to the nearest switch.    * Use Copper Cross-Over cables to connect each switch to the desired PC or switch based on the desired pattern. |
| **Output:**  **Star Topology:**    **Ring Topology:**    **Bus Topology:**    **Mesh Topology:**    **Hybrid Topology:** |
| **Observation & Learning:**   * **Bus Topology:** * Connectivity: All PCs were able to communicate through the sequence of switches, simulating a linear bus structure. * Failure Impact: A failure in any switch disrupts communication for the PCs connected through that switch, simulating the single point of failure characteristic of bus topologies. * Traffic Handling: Increased network traffic can lead to collisions and network slowdowns, emphasizing the limitations of bus topology in larger networks. * **Star Topology:** * Connectivity: All PCs were able to communicate effectively through the central switch. * Failure Impact: A failure in the central switch leads to complete network failure, highlighting the dependency on the central hub. * Ease of Troubleshooting: The central point of connectivity makes it easy to manage and troubleshoot the network. * Performance: Network performance was generally better compared to bus topology due to the direct connection to the switch. * **Ring Topology:** * Connectivity: PCs were able to communicate in a circular fashion, maintaining a continuous loop. * Failure Impact: A break in any link or switch caused disruption, but the overall network could still function due to the closed loop. * Traffic Handling: The use of a token-passing protocol helps to manage network traffic and prevent collisions. * Redundancy: Provides a level of redundancy as data can travel in either direction around the ring, allowing for continued operation if one path fails. * **Mesh Topology:** * Connectivity: PCs had multiple paths to communicate, ensuring robust connectivity. * Failure Impact: The failure of a single switch or link had minimal impact due to the presence of multiple alternative paths. * Redundancy and Reliability: High redundancy and reliability due to the interconnected nature of switches. * Complexity and Cost: Increased complexity in setting up and managing the network. Higher cost due to the number of switches and connections required. * Performance: Excellent performance due to multiple paths for data to travel, reducing bottlenecks. * **Hybrid Topology:** * Connectivity: Hybrid topology allows for multiple paths between devices by combining different topologies. This ensures that there are multiple routes for data to travel, enhancing connectivity. * Failure Impact: The interconnected nature of hybrid topologies minimizes the impact of single point failures. For example, in a star-bus hybrid, failure in one star network segment does not affect the others. * Redundancy and Reliability: Multiple paths and interconnected segments provide redundancy. If one path fails, alternative routes can be used, which enhances overall network reliability. * Complexity and Cost: Designing and setting up a hybrid network can be complex due to the need to integrate multiple topologies and ensure their compatibility. * Performance: Multiple pathways and a well-integrated structure allow for efficient data transmission, reducing potential bottlenecks and improving overall performance. |
| **Conclusion:**  The experiment demonstrated that each network topology—bus, star, ring, and mesh—has distinct advantages and disadvantages, making the choice dependent on specific network requirements:   1. Bus Topology:    1. Strengths: Simple and cost-effective for small networks.    2. Weaknesses: Prone to collisions and single point of failure, not suitable for larger networks. 2. Star Topology:    1. Strengths: Easy to manage and troubleshoot, good performance.    2. Weaknesses: Central switch failure can disrupt the entire network. 3. Ring Topology:    1. Strengths: Balanced traffic management and some redundancy.    2. Weaknesses: Complex setup, individual link/switch failures can cause issues. 4. Mesh Topology:    1. Strengths: Highest redundancy and fault tolerance, excellent performance.    2. Weaknesses: High complexity and cost. 5. Hybrid Topology:    1. Strengths: Highly adaptable network design.    2. Weaknesses: High complexity and cost.   In summary, simpler topologies (bus, star) are suitable for small networks with limited budgets, while more complex topologies (ring, mesh) offer better performance and reliability for larger, critical network environments. |
| **Questions:** |
| **Q: Which is the most efficient topology in a LAN environment and why?**  A: The most efficient topology in a LAN environment is the Star topology.   * + Centralized Management: The central switch makes it easy to manage and troubleshoot the network.   + Performance: Each device has a dedicated connection to the switch, reducing collisions and ensuring high performance.   + Scalability: It is easy to add or remove devices without disrupting the network.   + Reliability: While the central switch is a single point of failure, redundancy can be added by using multiple switches to enhance reliability.   **Q: How can we test the connectivity between the terminals?**  A: Connectivity between terminals can be tested using the following:   * Ping Command: Use the ping command to send ICMP echo requests to another terminal and check for replies, indicating successful connectivity.   + Example: ping 10.0.0.2   **Q: What are the two categories of cables? In what type of connection are they used?**  A: The two categories of cables are:   * + Copper Straight-Through Cable:     - Usage: Connecting different types of devices, such as a PC to a switch or a router to a switch.   + Copper Cross-Over Cable:     - Usage: Connecting similar types of devices, such as switch to switch, PC to PC, or router to router. |

**EXPERIMENT 2**

|  |
| --- |
| **Part A** |
| **Class B Tech CSE 3rd Year Sub : Computer Networks** |
| |  | | --- | | **Aim:** Simulation of Virtual Local Area Network | |
| **Prerequisite:** Nil |
| **Outcome:** To impart knowledge of Computer Networking Technology |
| **Theory:** A VLAN, or Virtual Local Area Network, is something I can use to segment a physical network into multiple logical networks. Even if the devices are connected to the same physical switch, I can create separate networks for different purposes. This is useful for managing the network more efficiently, improving security, and reducing unnecessary broadcast traffic.   1. Segmentation:    * I can segment the network based on different criteria like function, department, or application. For example, I could create one VLAN for the HR department and another for the IT department, even though all the devices might be connected to the same physical switch. 2. Broadcast Control:    * By using VLANs, I limit broadcast traffic to just the devices within the same VLAN. This helps prevent the network from getting bogged down by unnecessary traffic, which is especially important in larger networks. 3. Security:    * VLANs also enhance security by isolating different parts of the network. For example, I can make sure that sensitive data in one VLAN is not accessible to devices in another VLAN. |
| **Procedure:**   1. Open Cisco Packet Tracer and simulate the sample topologies with required size of VLAN. 2. Perform Necessary Operation on Switch to create and configure VLAN. 3. Check the connectivity between the devices. |

|  |
| --- |
| **Part B** |
| **Steps:**   1. Open Cisco Packet Tracer 2. Configure the Switches and connect the PC’s 3. Create VLANs :   enable  configure terminal  vlan 2 name Office  exit  vlan 3 name Home  exit   1. Assign VLANs to Switch Ports:   interface fastEthernet 0/1  switchport access vlan 2  exit  interface fastEthernet 0/2  switchport access vlan 2  exit  interface fastEthernet 0/3  switchport access vlan 3  exit  interface fastEthernet 0/4  switchport access vlan 3  exit   1. Verify VLAN Configuration:   show vlan |
| **Output:** |
| **Observation & Learning:**  **Observations:**   1. **VLAN Segmentation**: Achieved effective network segmentation, isolating traffic between different VLANs. 2. **Reduced Broadcast Traffic**: VLANs limited broadcasts to devices within the same VLAN, enhancing network efficiency. 3. **Inter-VLAN Communication**: Devices in different VLANs couldn’t communicate without Inter-VLAN Routing, highlighting the need for a router   **Learnings:**   1. VLANs are powerful tools for organizing and securing networks. 2. Proper configuration and persistence are key to maintaining network stability. |
| **Conclusion:** The VLAN implementation experiment successfully demonstrated how to segment a network into distinct logical groups, improving security and network management. By isolating traffic within each VLAN, broadcast domains were effectively reduced, which led to enhanced network performance. This experiment showed that VLANs are an efficient method for organizing and controlling network traffic, making them a valuable tool in network design and administration. |
| **Questions:** |
| 1. **What is the maximum number of VLAN can be created in a network?**   **Ans**: The maximum number of VLANs that can be created in a network is 4,096. This limit is defined by the 12-bit VLAN ID field in the IEEE 802.1Q standard, which allows VLAN IDs ranging from 0 to 4,095.   1. **What is mean by MTU? What is the value of MTU in Ethernet?**   **Ans**: MTU (Maximum Transmission Unit) refers to the largest size of a packet or frame that can be sent over a network interface without needing to be fragmented. It defines the maximum amount of data that can be transmitted in a single frame or packet on the network layer. The standard MTU size for Ethernet is 1,500 bytes. This value represents the maximum size of the payload (or data) portion of an Ethernet frame, excluding the Ethernet header and trailer.   1. **What happen when the broadcast operation is performed from a system in certain VLAN?**   **Ans**: When a broadcast is sent from a system within a VLAN, it is contained within that VLAN. Only devices in the same VLAN receive the broadcast, which helps isolate traffic and improve network efficiency. This containment prevents broadcast messages from affecting devices in other VLANs, reducing overall network load and enhancing security. |

**EXPERIMENT 3**

|  |
| --- |
| **Part A** |
| **Class B Tech CSE 3rd Year Sub : Computer Networks** |
| |  | | --- | | **Aim:** Simulation of Spanning Tree Protocol (STP) | |
| **Prerequisite:** Nil |
| **Outcome:** To impart knowledge of Computer Networking Technology |
| **Theory:** In this simulation, my primary objective is to thoroughly understand the operation of the Spanning Tree Protocol (STP) and its crucial role in preventing network loops within Ethernet networks. By simulating STP, I aim to gain a deeper insight into how it dynamically identifies and disables redundant paths that could potentially lead to network loops. This process is essential for maintaining a loop-free topology at Layer 2, which is critical for the stability and efficiency of the network. Through this hands-on experience, I expect to visualize the step-by-step mechanisms STP uses to optimize network traffic flow and prevent disruptions. |
| **Procedure:**   1. Open Cisco Packet Tracer and simulate the sample topologies for STP. 2. Perform Necessary Operation on Switch to create and configure STP. 3. Check the connectivity between the devices. |

|  |
| --- |
| **Part B** |
| **Steps:**  Step1: Open Cisco Packet Tracer on your computer.  Setp2: Drag and drop three switches onto the workspace. This setup will help you observe how STP handles redundant paths.  Step3: Use the “Copper Straight-Through” cable to connect the switches. Ensure each switch is connected to the other two switches, forming a triangle (a loop).  Step4: Drag and drop two PCs onto the workspace. Connect the first PC to the first switch and the second PC to the second switch using “Copper Straight-Through” cables.  Step5: Open CLI for one of the switch and write command “show spanning-tree” this will show the interfaces allowed or blocked for that switch.  Step6: Now send packet from one pc to another and the packet will only follow the path which is not blocked.  Step7: Now open CLI of all three switches and write command “no spanning-tree” to disable spanning tree from all the switches.  Step8: Now if you try to send a packet then the packet will travel in a loop infinitely.  Step9: Changing the priority of a switch to a higher priority than the root, such that the switch becomes the root. |
| **Output:**        Changing switch 2 to the root by giving priority higher than the previous root. |
| **Observation & Learning:**  **Observation**  During the simulation, I observed that with STP enabled, one of the links between the switches was automatically blocked to prevent network loops, which was visually indicated by a red dot on the blocked path. When I transmitted a packet from one PC to another, the packet followed the active, non-blocked path, ensuring seamless communication without any looping issues. Upon disabling STP on all switches, the packet began traveling in an infinite loop, highlighting the critical role STP plays in preventing network disruptions. Additionally, by altering the priority of a switch to make it the root, I could see how STP dynamically reconfigures the network to maintain an optimal loop-free topology.  **Learnings**  From this simulation, I learned how essential the Spanning Tree Protocol is in ensuring a loop-free environment in Ethernet networks, particularly in complex topologies with redundant paths. STP effectively prevents broadcast storms and packet looping, which can severely impact network performance. I also gained insights into how STP elects a root bridge and adjusts port states based on priority, allowing for flexibility and resilience in network design. Disabling STP demonstrated the potential risks of loops in the network, reinforcing the importance of STP in maintaining stability and efficient data transmission. |
| **Conclusion:**  The simulation of the Spanning Tree Protocol (STP) provided valuable insights into the crucial role STP plays in maintaining a stable and efficient network by preventing loops. Through observing how STP automatically blocks redundant paths and reconfigures the network based on root bridge priority, I gained a deeper understanding of its functionality. The experiment clearly demonstrated that without STP, network loops can occur, leading to continuous packet circulation and potential network failure. This emphasizes the importance of STP in real-world network environments, where it ensures loop-free communication, enhances network reliability, and supports optimal data flow across connected devices. |
| **Questions:** |
| 1. **What happen if STP is disabled on all the switches in the network?**   Ans: If STP is disabled on all switches in the network, it can lead to network loops where packets circulate indefinitely, causing broadcast storms, network congestion, and degraded performance. This results in increased latency, packet loss, and potential network outages due to overwhelmed switches and excessive traffic.   1. **What is mean by Designated, Root and Blk Interface?**   Ans: **Designated Interface:**   * The port on a network segment (or collision domain) that has the lowest path cost to the root bridge. This port is responsible for forwarding traffic to and from the segment and ensuring that it is the primary path for that segment.   **Root Interface:**   * The port on a switch that has the lowest path cost to the root bridge. It is the most efficient path for reaching the root bridge. Each non-root switch has one root port, and it is the port used for communicating with the root bridge.   **Blocked (Blk) Interface:**   * The port that STP has put into a blocking state to prevent network loops. This port does not forward traffic but listens for BPDUs to ensure the network topology remains loop-free. It is blocked to avoid creating a redundant path that could cause a loop.  1. **What are the different ways to identify the root in the STP?**   Ans:  **Bridge ID:**   * The primary method to identify the root bridge is by comparing the Bridge IDs of all switches. The Bridge ID consists of the bridge priority (a configurable value) and the MAC address of the switch. The switch with the lowest Bridge ID is elected as the root bridge.   **Show spanning-tree Command:**   * On Cisco switches, the show spanning-tree command can be used to view the current root bridge. The command output displays the root bridge ID along with other details such as port roles and STP states. |

**EXPERIMENT 4**

|  |  |
| --- | --- |
| **Aim:** | **Simulation of Static Routing protocol in small network** |
| **Prerequisite:** Nil | |
| **Outcome:** To impart knowledge of Computer Networking Technology | |
| **Theory:** | **Static Routing is a network routing technique where a network administrator manually sets specific routes to reach particular network destinations.** It is used in small, stable networks where routes are manually configured to direct traffic between different networks. It ensures predictable traffic flow without the need for dynamic updates.   * **Key Points:**  1. **Manually Configured Routes:**     * Static routing requires administrators to manually add and update routes, ensuring direct and controlled traffic paths. 2. **Next-Hop Address:** o The next-hop IP address is specified to indicate the next device in the path towards the destination network. 3. **Route Entries:**   o **Default Route:** Used when no specific route matches the destination.  o **Static Route:** A direct path to a specified network or subnet.  o **Administrative Distance:** Defines the priority of routes, with static routes usually having a lower value.   1. **Route Verification:**   o Use commands like show ip route to display the routing table and confirm routes are correctly set.   1. **No Automatic Updates:**     * Static routes don’t adjust automatically when the network changes, so administrators must manually reconfigure routes when necessary.     • **Variants:**  o **Floating Static Routes:** Backup routes with higher administrative distance.  o **Recursive Static Routes:** Routes pointing to another route for the next hop. |
| **Procedure:** | 1. Open Cisco Packet Tracer and simulate the sample topologies for static routing. 2. Perform Necessary Operation on router to create and configure static routing. 3. Check the connectivity between the devices. |

|  |  |
| --- | --- |
| **Part-B** | |
| **Steps:** | 1. **Create the Network Topology:**     * Drag and drop **routers**, **switches**, and **PCs**.    * Connect them using **cables** (straight-through for PC-to-switch, crossover for router-to- router).      1. **Assign IP Addresses:**     * Configure IP addresses for each **PC** and **router interface**.    * Example:   o PC1: 192.168.1.2 /24, Gateway: 192.168.1.1 o Router1 (G0/0): 192.168.1.1 /24 o Router1 (S0/0/0): 10.0.0.1 /30 o Router2 (S0/0/0): 10.0.0.2 /30 o Router2 (G0/0): 192.168.2.1 /24 o PC2: 192.168.2.2 /24, Gateway: 192.168.2.1     1. **Configure Static Routes:**     * **Router1**: Configure a static route to **Router2**'s network.     Router1> enable  Router1# configure terminal  Router1(config)# ip route 192.168.2.0 255.255.255.0 10.0.0.2     * + **Router2**: Configure a static route to **Router1**'s network.     Router2> enable  Router2# configure terminal  Router2(config)# ip route 192.168.1.0 255.255.255.0 10.0.0.1     1. **Verify Routes:**     * Use the show ip route command on both routers to verify the static routes.     Copy code  Router# show ip route     1. **Test Connectivity:**     * **Ping** from PC1 to PC2 and vice versa to ensure communication.     Copy code ping 192.168.2.2 ping 192.168.1.2 |
| **Output:** |  |
| **Observation &**  **Learnings:** | * **Impact of Static Routing:** Static routing simplifies traffic management in small networks by using manually configured routes. Misconfigurations can lead to data not reaching its destination, highlighting the importance of careful route setup.      * **Route Management:** Understanding next-hop addresses, destination networks, and interface configurations is essential for effective traffic flow. Each static route ensures that data travels through the intended path without relying on automatic adjustments.      * **Identifying Routes:** Static routes are manually defined by specifying the destination network and next-hop IP address. Using commands like show ip route helps verify and manage these routes, ensuring proper traffic flow between networks.      * **Static Routing's Role in Network Design:** Static routing provides control over traffic paths and is useful for stable, predictable networks. Properly configuring and monitoring static routes leads to a reliable, efficient network design without the complexity of dynamic protocols. |
| **Conclusion:** | Static Routing is essential for directing traffic in small, stable networks by manually configuring routes. Without it, data may not reach its destination, causing disruptions. Key elements include **Next-Hop Addresses**, **Destination Networks**, and **Interface**  **Configurations**. Static routing simplifies the network but requires careful planning, as routes must be manually managed. Verifying routes with commands like show ip route ensures proper data flow, resulting in efficient and reliable network performance without relying on dynamic routing protocols. |

**EXPERIMENT 5**

|  |  |
| --- | --- |
| **Aim:** | **Simulation of RIP and OSPF routing protocol in small network** |
| **Prerequisite:** Nil | |
| **Outcome:** To impart knowledge of Computer Networking Technology | |
| **Theory:** | * **Routing Information Protocol (RIP) and Open Shortest Path First (OSPF)** are dynamic routing protocols used to manage routes and ensure efficient data forwarding in small networks. They automatically update routes and adjust to network changes, optimizing the network’s traffic flow. * **Key Points:**      1. **RIP (Routing Information Protocol):**  * **Distance Vector Protocol:** RIP uses hop count as a metric to determine the best route. * **Max Hop Limit:** RIP limits the maximum hop count to 15, making it suitable for smaller networks. * **Periodic Updates:** RIP sends routing updates every 30 seconds, which can lead to slow convergence.  1. **OSPF (Open Shortest Path First):**  * **Link-State Protocol:** OSPF calculates the shortest path based on cost, taking bandwidth into account. * **Area-Based Design:** OSPF organizes networks into areas for scalability. * **Fast Convergence:** OSPF recalculates routes quickly when changes occur, making it more efficient for larger networks.  1. **Route Learning:**  * **RIP:** Shares entire routing table with neighbours periodically, which can lead to more traffic. * **OSPF:** Only shares network changes, reducing overhead and improving efficiency.  1. **Route Calculation:**  * **RIP:** Chooses routes based on the lowest hop count. * **OSPF:** Uses Dijkstra’s algorithm to calculate the shortest path based on link cost.  1. **Convergence:**  * **RIP:** Slower convergence due to periodic updates. * **OSPF:** Faster convergence, making it more responsive to network changes.     • **Variants:** o **RIPng:** Supports IPv6. o **OSPFv3:** OSPF version for IPv6. |
| **Procedure:** |  |

|  |  |
| --- | --- |
|  | **Part-B** |
| **Steps:** | 1. **Create the Topology:**      * + Open **Cisco Packet Tracer**.   + Drag and drop the required **routers**, **switches**, and **PCs** to create your network topology.      1. **Connect Devices:**      * + Use appropriate cables to connect the devices: o **Straight-through cables** for PC-to-switch and switch-to-router connections.   o **Serial or crossover cables** to connect router-to-router.     1. **Enable OSPF on Routers:**      * + Access the **router's CLI** by clicking on the router.   + Enter the following commands to enable OSPF on the router:     bash Copy code  Router> enable  Router# configure terminal  Router(config)# router ospf 1  Router(config-router)# network [network-address] [wildcard-mask] area [area-number]  Router(config-router)# exit  Router(config)# exit     * + Example:     bash Copy code  Router(config-router)# network 192.168.1.0 0.0.0.255 area 0 Router(config-router)# network 10.0.0.0 0.0.0.3 area 0     1. **Disable OSPF (if needed):**      * + To remove OSPF from the router, use the following commands:     Router> enable  Router# configure terminal  Router(config)# no router ospf 1  Router(config)# exit |

|  |  |
| --- | --- |
|  | 1. **Shutdown Ports (if needed):**      * + To shut down a port on a router or switch, use the following commands:     Router(config)# interface fastEthernet 0/1  Router(config-if)# shutdown Router(config-if)# exit     * + To re-enable the port, use:     Router(config)# interface fastEthernet 0/1  Router(config-if)# no shutdown  Router(config-if)# exit     1. **Verify OSPF Configuration:**      * + Use the following commands to check the OSPF status and verify the configuration:     bash  Copy code  Router# show ip ospf neighbor  Router# show ip route |

|  |  |
| --- | --- |
| **Output:** |  |
| **Observation &**  **Learnings:** | **Impact of OSPF:**    Disabling OSPF can lead to inefficient routing and suboptimal performance, emphasizing its critical role in dynamically managing routes and adapting to network changes to ensure effective data transmission.    **OSPF Components:**    Understanding the roles of **Areas**, **Router Types** (such as **Internal Routers**, **Backbone Routers**, and **ABRs**), and **Adjacency** is vital for effective network traffic management. Each component contributes to the overall efficiency and scalability of OSPF in complex network topologies.    **Identifying OSPF Neighbours:** |
|  | OSPF neighbors are determined through the exchange of **Hello packets**. Analyzing **OSPF database** and using commands such as show ip ospf neighbor allows administrators to identify and verify neighbor relationships, ensuring proper communication and routing updates between routers.    **OSPF's Role in Network Design:**    OSPFs ability to calculate the shortest path using the **Dijkstra algorithm** and its support for hierarchical design through areas are crucial for maintaining network efficiency and reliability. Proper configuration and monitoring of OSPF contribute to a robust, scalable network architecture capable of adapting to growth and changes. |
| **Conclusion:** | OSPF (Open Shortest Path First) is essential for maintaining efficient and stable network routing by dynamically calculating the best paths for data transmission. Disabling OSPF can lead to inefficient routing, network delays, and potential routing loops. Understanding OSPF concepts such as \*\*Areas\*\*, \*\*Router Types\*\* (Internal, Area Border Routers), and \*\*LinkState Advertisements (LSAs)\*\* is crucial for optimizing traffic flow and ensuring a loop-free, hierarchical network. Identifying OSPF neighbours involves exchanging \*\*Hello packets\*\* and building adjacencies to form a consistent routing topology. Properly configuring and monitoring OSPF ensures a scalable and robust network capable of adapting to changes and growing without compromising performance. |

**EXPERIMENT 6**

|  |
| --- |
| **Part A** |
| **Class B Tech CSE 3rd Year Sub : Computer Networks** |
| |  | | --- | | **Aim:** |   Introduction to Socket Programming- Design and Implement client-server elements of a few network applications e.g. Echo client and server, Time client and server, Online Quiz and Buzzer Application, etc. using TCP sockets. |
| **Prerequisite:** Nil |
| **Outcome:** To impart knowledge of Socket Programming |
| **Theory:** Socket programming enables communication between two systems over a network, allowing processes to exchange data via IP addresses and port numbers. It follows a client-server architecture, where the client initiates a connection to the server, which listens for incoming requests. Sockets can use either TCP, which provides reliable, connection-oriented communication, or UDP, which is faster but connectionless and less reliable. Common socket functions include socket() to create a socket, bind() to assign an IP and port, listen() for the server to wait for connections, and accept() to establish communication with a client. |
| **Procedure:**   1. Write Simple Client Server Program using Java/Python Programming Language 2. Execute the program using appropriate compiler. 3. Verify the working of the program. |

|  |
| --- |
| **Part B** |
| **Steps:**  Client.java  import javax.swing.\*;  import java.awt.\*;  import java.awt.event.ActionEvent;  import java.awt.event.ActionListener;  import java.io.BufferedReader;  import java.io.PrintWriter;  import java.net.Socket;  import java.io.InputStreamReader;  public class Chat\_Client extends JFrame {  private JTextArea chatArea;  private JTextField messageField;  private JButton sendButton;  private Socket socket;  private PrintWriter out;  private BufferedReader in;  private String serverAddress = "localhost";  private int serverPort = 12346;  public Chat\_Client() {  setTitle("Chat Client");  setSize(400, 400);  setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  chatArea = new JTextArea();  chatArea.setEditable(false);  messageField = new JTextField(25);  sendButton = new JButton("Send");  JPanel panel = new JPanel();  panel.setLayout(new BorderLayout());  panel.add(new JScrollPane(chatArea), BorderLayout.CENTER);  JPanel inputPanel = new JPanel();  inputPanel.setLayout(new FlowLayout());  inputPanel.add(messageField);  inputPanel.add(sendButton);  panel.add(inputPanel, BorderLayout.SOUTH);  add(panel);  setVisible(true);  try {  socket = new Socket(serverAddress, serverPort);  out = new PrintWriter(socket.getOutputStream(), true);  in = new BufferedReader(new InputStreamReader(socket.getInputStream()));  new Thread(this::receiveMessages).start();  } catch (Exception e) {  e.printStackTrace();  }  sendButton.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  sendMessage();  }  });  messageField.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  sendMessage();  }  });  }  private void sendMessage() {  try {  String message = messageField.getText();  out.println(message);  chatArea.append("Client: " + message + "\n");  messageField.setText("");  } catch (Exception e) {  e.printStackTrace();  }  }  private void receiveMessages() {  try {  String message;  while ((message = in.readLine()) != null) {  chatArea.append("Server: " + message + "\n");  }  } catch (Exception e) {  e.printStackTrace();  }  }  public static void main(String[] args) {  new Chat\_Client();  }  }  **Server.java**  import javax.swing.\*;  import java.awt.\*;  import java.awt.event.ActionEvent;  import java.awt.event.ActionListener;  import java.io.BufferedReader;  import java.io.PrintWriter;  import java.net.ServerSocket;  import java.net.Socket;  import java.io.InputStreamReader;  public class Chat\_Server extends JFrame {  private JTextArea chatArea;  private JTextField messageField;  private JButton sendButton;  private ServerSocket serverSocket;  private Socket clientSocket;  private PrintWriter out;  private BufferedReader in;  private int port = 12346;  public Chat\_Server() {  setTitle("Chat Server");  setSize(400, 400);  setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  chatArea = new JTextArea();  chatArea.setEditable(false);  messageField = new JTextField(25);  sendButton = new JButton("Send");  JPanel panel = new JPanel();  panel.setLayout(new BorderLayout());  panel.add(new JScrollPane(chatArea), BorderLayout.CENTER);  JPanel inputPanel = new JPanel();  inputPanel.setLayout(new FlowLayout());  inputPanel.add(messageField);  inputPanel.add(sendButton);  panel.add(inputPanel, BorderLayout.SOUTH);  add(panel);  setVisible(true);  try {  serverSocket = new ServerSocket(port);  clientSocket = serverSocket.accept(); // Wait for a client to connect  out = new PrintWriter(clientSocket.getOutputStream(), true);  in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));  new Thread(this::receiveMessages).start();  } catch (Exception e) {  e.printStackTrace();  }  sendButton.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  sendMessage();  }  });  messageField.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  sendMessage();  }  });  }  private void sendMessage() {  try {  String message = messageField.getText();  out.println(message);  chatArea.append("Server: " + message + "\n");  messageField.setText("");  } catch (Exception e) {  e.printStackTrace();  }  }  private void receiveMessages() {  try {  String message;  while ((message = in.readLine()) != null) {  chatArea.append("Client: " + message + "\n");  }  } catch (Exception e) {  e.printStackTrace();  }  }  public static void main(String[] args) {  new Chat\_Server();  }  } |
| **Output:** |
| **Observation & Learning:**  **Observations:**  In this experiment, we observe the establishment of a **TCP connection** between a client and a server using socket programming in Java. The server listens for incoming client connections through a specified port (ServerSocket.accept()), and once connected, both the client and server can exchange messages in real-time. The graphical user interface (GUI) allows for interactive communication, with messages sent from both sides being displayed in the respective text areas. We also notice that event-driven programming is used, as the message transmission is triggered by user actions, like clicking the "Send" button or pressing Enter. Additionally, multiple threads are employed for continuous message reception, allowing the application to listen for incoming messages without interrupting other actions.  **Learnings:**  From this experiment, we learn how to implement basic **client-server architecture** using Java sockets. It highlights key networking concepts such as **IP addresses and port numbers** for identifying endpoints, and the use of **TCP** for reliable, connection-oriented communication. The project illustrates how to manage input and output streams to send and receive data between networked applications. We also gain insights into using **multi-threading** to handle simultaneous tasks (e.g., sending and receiving messages) without blocking the user interface. This experiment reinforces the importance of using **sockets** for creating real-time, interactive networked applications, and demonstrates how data is transmitted and received efficiently over a network. |
| **Conclusion:** In conclusion, this experiment successfully demonstrates the practical implementation of socket programming and client-server communication in Java. By building a real-time chat application, it reinforces the core concepts of networking, such as TCP-based communication, socket management, and event-driven programming. The use of multi-threading ensures smooth, non-blocking data transmission, providing a robust framework for interactive network applications. Overall, the experiment highlights the importance of understanding network protocols, reliable data exchange, and the architectural principles needed to establish and maintain seamless communication between connected systems in a network. |
| **Questions:** |
| What is Socket?  Ans- A socket is an endpoint for communication between two machines over a network. It provides a mechanism for exchanging data between a client and a server using network protocols like TCP (for reliable, connection-oriented communication) or UDP (for fast, connectionless communication). Sockets are identified by an IP address and a port number, allowing processes to send and receive data across different networks or locally on the same machine. In socket programming, sockets enable applications to communicate by creating, binding, listening, sending, and receiving data in a structured manner.  Which socket is used for the communication between the client and server?  Ans- The **TCP (Transmission Control Protocol) socket** is commonly used for communication between a client and a server. This type of socket ensures reliable, connection-oriented communication, meaning that data is sent and received in the correct order, and any lost packets are retransmitted. TCP sockets establish a connection before data exchange begins, making them ideal for applications like web servers, email services, and chat applications where reliability and data integrity are crucial. This socket type is used when the client and server need to maintain a continuous, error-checked stream of communication.  What are the different operation supported on sockets?   * **Creating a Socket**: The socket() function initializes a new socket, specifying the address family (IPv4 or IPv6), socket type (TCP or UDP), and protocol. * **Binding**: The bind() operation assigns a specific IP address and port number to the socket, making it accessible for incoming connections. * **Listening**: In server sockets, the listen() function allows the socket to enter a passive mode, waiting for client connection requests. * **Accepting Connections**: The accept() function is used by the server to accept a connection from a client, establishing a communication channel. * **Sending Data**: The send() or write() functions transmit data from the sender to the receiver through the socket. * **Receiving Data**: The recv() or read() functions allow the socket to receive data sent by the other end of the connection. * **Closing the Socket**: The close() function terminates the connection and releases the resources associated with the socket. |

**EXPERIMENT 7**

|  |
| --- |
| **Part A** |
| **Class B Tech CSE 3rd Year Sub : Computer Networks** |
| |  | | --- | | **Aim:** |   Design and Implement client-server application using UDP sockets. |
| **Prerequisite:** Nil |
| **Outcome:** To impart knowledge of Socket Programming |
| **Theory:** Socket programming is a way to enable communication between different processes over a network using sockets as the endpoints for data exchange. It involves creating a server that listens for incoming client connections and a client that initiates a connection to the server. Socket programming typically uses TCP (for reliable communication) or UDP (for faster, connectionless communication) protocols. The process involves several key operations: creating a socket, binding it to an IP address and port, listening for connections (on the server side), accepting connections, and sending or receiving data. Through socket programming, developers can build applications that facilitate real-time data transfer, such as chat applications, web servers, and multiplayer games, allowing for interactive and dynamic networked experiences. |
| **Procedure:**   1. Write Simple Client Server Program using Java/Python Programming Language 2. Execute the program using appropriate compiler. 3. Verify the working of the program. |

|  |
| --- |
| **Part B** |
| **Steps:**  **Client.java**  import javax.swing.\*;  import java.awt.\*;  import java.awt.event.ActionEvent;  import java.awt.event.ActionListener;  import java.net.DatagramPacket;  import java.net.DatagramSocket;  import java.net.InetAddress;  public class Chat\_Client extends JFrame {  private JTextArea chatArea;  private JTextField messageField;  private JButton sendButton;  private DatagramSocket socket;  private InetAddress serverAddress;  private int serverPort = 12346;  public Chat\_Client() {  setTitle("Chat Client");  setSize(400, 400);  setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  chatArea = new JTextArea();  chatArea.setEditable(false);  messageField = new JTextField(25);  sendButton = new JButton("Send");  JPanel panel = new JPanel();  panel.setLayout(new BorderLayout());  panel.add(new JScrollPane(chatArea), BorderLayout.CENTER);  JPanel inputPanel = new JPanel();  inputPanel.setLayout(new FlowLayout());  inputPanel.add(messageField);  inputPanel.add(sendButton);  panel.add(inputPanel, BorderLayout.SOUTH);  add(panel);  setVisible(true);  try {  socket = new DatagramSocket();  serverAddress = InetAddress.getByName("localhost");  new Thread(this::receiveMessages).start();  } catch (Exception e) {  e.printStackTrace();  }  sendButton.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  sendMessage();  }  });  messageField.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  sendMessage();  }  });  }  private void sendMessage() {  try {  String message = messageField.getText();  byte[] buffer = message.getBytes();  DatagramPacket packet = new DatagramPacket(buffer, buffer.length, serverAddress, serverPort);  socket.send(packet);  chatArea.append("Client: " + message + "\n");  messageField.setText("");  } catch (Exception e) {  e.printStackTrace();  }  }  private void receiveMessages() {  try {  byte[] buffer = new byte[1024];  while (true) {  DatagramPacket packet = new DatagramPacket(buffer, buffer.length);  socket.receive(packet);  String message = new String(packet.getData(), 0, packet.getLength());  chatArea.append("Server: " + message + "\n");  }  } catch (Exception e) {  e.printStackTrace();  }  }  public static void main(String[] args) {  new Chat\_Client();  }  }  **Server.java**  import javax.swing.\*;  import java.awt.\*;  import java.awt.event.ActionEvent;  import java.awt.event.ActionListener;  import java.net.DatagramPacket;  import java.net.DatagramSocket;  import java.net.InetAddress;  public class Chat\_Server extends JFrame {  private JTextArea chatArea;  private JTextField messageField;  private JButton sendButton;  private DatagramSocket socket;  private InetAddress clientAddress;  private int clientPort;  public Chat\_Server() {  setTitle("Chat Server");  setSize(400, 400);  setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  chatArea = new JTextArea();  chatArea.setEditable(false);  messageField = new JTextField(25);  sendButton = new JButton("Send");  JPanel panel = new JPanel();  panel.setLayout(new BorderLayout());  panel.add(new JScrollPane(chatArea), BorderLayout.CENTER);  JPanel inputPanel = new JPanel();  inputPanel.setLayout(new FlowLayout());  inputPanel.add(messageField);  inputPanel.add(sendButton);  panel.add(inputPanel, BorderLayout.SOUTH);  add(panel);  setVisible(true);  try {  socket = new DatagramSocket(12346);  new Thread(this::receiveMessages).start();  } catch (Exception e) {  e.printStackTrace();  }  sendButton.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  sendMessage();  }  });  messageField.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  sendMessage();  }  });  }  private void sendMessage() {  try {  String message = messageField.getText();  byte[] buffer = message.getBytes();  DatagramPacket packet = new DatagramPacket(buffer, buffer.length, clientAddress, clientPort);  socket.send(packet);  chatArea.append("Server: " + message + "\n");  messageField.setText("");  } catch (Exception e) {  e.printStackTrace();  }  }  private void receiveMessages() {  try {  byte[] buffer = new byte[1024];  while (true) {  DatagramPacket packet = new DatagramPacket(buffer, buffer.length);  socket.receive(packet);  clientAddress = packet.getAddress();  clientPort = packet.getPort();  String message = new String(packet.getData(), 0, packet.getLength());  chatArea.append("Client: " + message + "\n");  }  } catch (Exception e) {  e.printStackTrace();  }  }  public static void main(String[] args) {  new Chat\_Server();  }  } |
| **Output:** |
| **Observation & Learning:**  **Observations**  This experiment illustrates key concepts in computer networks through the implementation of UDP socket communication between a client and server. By utilizing the UDP protocol, data is transmitted without the overhead of establishing a dedicated connection, highlighting the protocol's connectionless nature. The design demonstrates how messages can be sent and received concurrently using threading, which is crucial for real-time applications where latency is a factor. The dynamic handling of client addresses and ports by the server enables flexible communication, as it can interact with multiple clients without prior knowledge of their details. Furthermore, the user interface provides an engaging way to visualize network interactions, showing how data flows between entities in a networked environment.  **Learning**  This experiment reinforces fundamental networking principles, particularly those related to UDP and its characteristics. Understanding that UDP does not guarantee delivery or order of packets enhances awareness of potential pitfalls in network communication, especially for applications requiring reliable message delivery. The use of threads for concurrent message processing deepens the understanding of how network applications manage multiple data streams simultaneously, a critical aspect in real-time communications. Additionally, this project emphasizes the importance of error handling in network applications, as network instability can lead to lost packets or connection issues. Overall, the experience gained from building a UDP client-server architecture highlights essential concepts in computer networks, such as addressing, packet structure, and the trade-offs between different communication protocols. |
| **Conclusion:** In conclusion, this experiment successfully demonstrates the principles of computer networks through the implementation of a UDP client-server architecture. By utilizing the connectionless nature of UDP, we explored efficient data transmission without the need for a dedicated connection, which is crucial for applications requiring low latency. The incorporation of threading for concurrent message handling emphasized the importance of managing multiple data streams in real-time communications. Additionally, the dynamic addressing of clients showcased the flexibility and scalability of network interactions. Overall, the experience provided valuable insights into the practical challenges and considerations in network programming, reinforcing key networking concepts such as packet structure, addressing, and the trade-offs between different communication protocols. |

**EXPERIMENT 8**

|  |
| --- |
| **Part A** |
| **Class B Tech CSE 3rd Year Sub : Computer Networks** |
| **Aim:** Simulation of HTTP and DHCP in small network. |
| **Prerequisite:** Nil |
| **Outcome:** To impart knowledge of application layer of OSI architecture |
| **Theory:**  **HTTP (Hypertext Transfer Protocol)** is an application-layer protocol used for transmitting hypertext over the internet. It facilitates communication between clients (like web browsers) and servers by enabling the transfer of web pages, images, and other resources. HTTP operates using a request-response model, where a client sends a request to a server, and the server responds with the requested content. It is stateless, meaning each request is independent, and it operates over TCP, typically using port 80 (or HTTPS on port 443 for secure connections).  **DHCP (Dynamic Host Configuration Protocol)** is a network management protocol that automatically assigns IP addresses and other network configuration parameters to devices on a network. This enables devices to communicate on an IP network without the need for manual configuration. When a device connects to the network, it sends a DHCP request to a DHCP server, which then assigns an available IP address and provides additional information such as the subnet mask, default gateway, and DNS server addresses. DHCP simplifies network management and reduces configuration errors. |
| **Procedure:**   1. Simulate the small client server topology in cisco packet tracer. 2. Configure the HTTP and DHCP services on the server. 3. Verify the working of the network. |

|  |
| --- |
| **Part B** |
| **Steps:**   1. Open Cisco Packet Tracer 2. **Add Devices**  * **Add a Router**: Drag and drop a router onto the workspace. * **Add Switches**: Drag and drop one or more switches. * **Add End Devices**: Drag and drop PCs or laptops connected to the switch.  1. **Set Up DHCP**   **Add a DHCP Server**:   * + Drag and drop a server onto the workspace.   + Click on the server, and navigate to the Config tab.   + Select DHCP from the left menu.  1. **Configure DHCP Settings**:  * **IP Address Pool**: Define the DHCP IP address pool.   + **IP Address**: The starting IP address of the range.   + **Subnet Mask**: Typically 255.255.255.0.   + **Default Gateway**: Set it to the router’s interface IP address that connects to the DHCP clients.   + **DNS Server**: You can specify a DNS server IP (or leave it blank for testing). * Click on Add to add the IP address range.  1. **Connect Devices**:  * Use a straight-through cable to connect the router to the switch. * Connect the PCs to the switch.  1. **Set Up the Router** 2. **Configure Router Interfaces**:    * Click on the router and go to the CLI tab.    * Enter privileged mode by typing enable.    * Enter global configuration mode with configure terminal.    * Configure the interfaces (example):    * interface GigabitEthernet0/0    * ip address 192.168.1.1 255.255.255.0    * no shutdown    * exit 3. **Set Up HTTP Server** 4. **Configure HTTP on the Server**:    * Click on the server and go to the Config tab.    * Under the HTTP option, enable the HTTP server by checking the box.    * Optionally, configure the website files or HTML content you want to serve. |
| **Output:** |
| **Observation & Learning:**  **Observation**  During the simulation of HTTP and DHCP in Cisco Packet Tracer, it was observed that the DHCP server successfully assigned IP addresses to the connected PCs within the defined range. The configuration of the router interfaces and the DHCP settings facilitated seamless communication between devices in the network. Additionally, when accessing the HTTP server from a web browser on the PCs, the expected web page was displayed, indicating that the server was properly configured and accessible. Overall, the network functioned as intended, demonstrating the effectiveness of DHCP in managing IP address assignments and HTTP in serving web content.  **Learning**  This experiment provided valuable insights into the roles of DHCP and HTTP in network communication. Learning how DHCP automates the IP address allocation process highlighted the importance of efficient network management, reducing the need for manual configurations. The HTTP setup reinforced the concept of client-server architecture, illustrating how clients can request and retrieve web content from a server. Additionally, the hands-on experience in Cisco Packet Tracer enhanced practical networking skills, deepening the understanding of configuring and troubleshooting network devices, which is crucial for real-world networking scenarios. |
| **Conclusion:** In conclusion, the simulation of HTTP and DHCP in Cisco Packet Tracer effectively demonstrated the fundamental concepts of network communication and management. The successful configuration of the DHCP server illustrated the automated process of IP address allocation, enhancing network efficiency. Meanwhile, the HTTP server setup highlighted the client-server interaction, showcasing how web content is served to users. This experiment not only reinforced theoretical knowledge but also provided practical experience in configuring and troubleshooting network devices, which is essential for aspiring networking professionals. Overall, the experiment underscored the importance of these protocols in facilitating seamless communication within modern networks. |

**EXPERIMENT 9**

|  |
| --- |
| **Part A** |
| **Class B Tech CSE 3rd Year Sub : Computer Networks** |
| **Aim:** Simulation of DNS and FTP in small network. |
| **Prerequisite:** Nil |
| **Outcome:** To impart knowledge of application layer of OSI architecture |
| **Theory:** DNS (Domain Name System) is a system that translates human-readable domain names (like www.example.com) into IP addresses (like 192.0.2.1) that computers use to identify each other on the network. It functions like a phone book for the internet, ensuring that users can access websites using easily remembered names instead of numerical IP addresses. DNS servers perform this lookup and are crucial for the functioning of the internet.  FTP (File Transfer Protocol) is a standard network protocol used to transfer files between a client and a server over a TCP-based network like the internet. FTP enables users to upload, download, or manipulate files on a remote server. It's commonly used for website file management, data backup, and other file-sharing tasks. FTP operates in two modes: active and passive, depending on how the connection is established. |
| **Procedure:**   1. Simulate the small client server topology in cisco packet tracer. 2. Configure the DNS and FTP services on the server. 3. Verify the working of the network. |

|  |
| --- |
| **Part B** |
| **Steps:**  **FTP:**   1. **Open Cisco Packet Tracer** and create a new project. 2. **Add Devices**:  * Add one **FTP server** (e.g., a server device). * Add two **PCs** to act as clients (e.g., PC-0 and PC-1). * Add a **Switch** to connect all the devices.  1. **Connect the Devices**:  * Use **Copper Straight-Through** cables to connect the PCs and the server to the switch.  1. **Configure IP Addresses** 2. **Set up the FTP Server**   **Configure the FTP Service**:   * + Click on the server, go to the **Services** tab, and select **FTP** from the list.   + Turn on the **FTP service**.   + Optionally, add **FTP usernames and passwords** for secure access.  1. **Access the FTP Server**   **Open the FTP Client**:   * + On **PC-0**, go to the **Desktop** tab and open the **Command Prompt**.   + Type the command to connect to the FTP server: ftp 192.168.1.1   + Enter the **FTP username and password** (if configured).   **Transfer Files**:   * + Use FTP commands like get to download a file or put to upload a file.   **DNS:**   1. Set Up the Network   Open Cisco Packet Tracer and create a new project.  Add Devices:   * + Add a DNS Server (e.g., a server device).   + Add two PCs to act as clients (e.g., PC-0 and PC-1).   + Add a Switch to connect the devices.   Connect the Devices:   * + Use Copper Straight-Through cables to connect the PCs and the server to the switch**.**  1. Configure IP Addresses 2. Configure the DNS Server   **Set up the DNS Service**:   * Click on the server, go to the **Services** tab, and select **DNS**. * Turn on the **DNS service** by clicking the **ON** button.   **Add DNS Entries (Domain Name to IP Mappings)**:   * In the **DNS configuration window**, add a domain name and corresponding IP address.   + Example:     - Name: www.example.com     - Address: 192.168.1.1 (or any other server IP) * Click **Add** to save the entry.  1. Test Connectivity   **Ping the Server**:   * On **PC-0**, open the **Command Prompt** (Desktop tab) and ping the server using its **IP address** to ensure network connectivity: ping 192.168.1.1 * If the ping is successful, proceed to the next step.  1. Test DNS Resolution   **Check DNS Resolution from the Clients**:   * On **PC-0**, go to the **Command Prompt** and type the following command to test DNS resolution: ping www.example.com * The **ping** should resolve the domain name to the IP address (e.g., 192.168.1.1).   **Repeat the DNS Resolution Test on PC-1** to ensure consistency. |
| **Output:**  **FTP**        **DNS:** |
| **Observation & Learning:**  **DNS Observation and Learning:**  During the DNS simulation in Cisco Packet Tracer, we observe that domain names can successfully be resolved to their corresponding IP addresses through the DNS server. The DNS server enables clients to access services using easily remembered names instead of numerical IP addresses. Through this simulation, we learn the importance of DNS in facilitating seamless communication on the internet, eliminating the need for users to memorize IP addresses. Additionally, we understand the basic configuration steps of setting up DNS services and the concept of DNS resolution.  **FTP Observation and Learning:**  In the FTP simulation, we observe that the File Transfer Protocol enables the uploading and downloading of files between a client and server. Clients can interact with the FTP server through commands like put and get for file transfers. This simulation highlights the practical application of FTP for transferring data across networks. We learn how to configure an FTP server, establish FTP connections from clients, and utilize various FTP commands to manage files on remote servers, which is crucial for network file sharing and website management. |
| **Conclusion:** In conclusion, the DNS and FTP simulations in Cisco Packet Tracer demonstrate the fundamental roles these protocols play in network communication and file management. DNS simplifies internet usage by resolving domain names into IP addresses, making navigation more user-friendly, while FTP allows efficient file transfers between clients and servers, facilitating data sharing and management across networks. Through these simulations, we gained hands-on experience in configuring DNS and FTP services, understanding their practical applications, and reinforcing key networking concepts that are essential in real-world environments. |

**EXPERIMENT 10**

|  |
| --- |
| **Part A** |
| **Class B Tech CSE 3rd Year Sub : Computer Networks** |
| |  | | --- | | **Aim:** |   Simulation of SMTP in small network. |
| **Prerequisite:** Nil |
| **Outcome:** To impart knowledge of application layer of OSI architecture |
| **Theory:** SMTP (Simple Mail Transfer Protocol) is a communication protocol used for sending emails across networks. It is responsible for transferring messages from a sender's email client to the recipient's email server. SMTP operates at the application layer and uses a process of connecting to a mail server, authenticating the sender, and relaying the message to the recipient's server. It typically works with TCP port 25. SMTP does not handle email retrieval, which is done by protocols like POP3 or IMAP. It's widely used and essential for email communication across the internet. |
| **Procedure:**   1. Simulate the small client server topology in cisco packet tracer. 2. Configure the SMTP services on the server. 3. Verify the working of the network. |

|  |
| --- |
| **Part B** |
| **Steps:**  **1. Set Up the Network**   * **Drag and Drop Devices**: Place at least two PCs, a server, and a router on the workspace. * **Connect Devices**: Use cables (copper straight-through) to connect the PCs and server to the router.   **2. Configure IP Addresses**   * **Assign IPs to PCs and Server**:   + Click on each PC, go to the "Desktop" tab, then "IP Configuration" and assign an IP address, subnet mask, and default gateway.   + For the server, follow the same steps but ensure the IP is on the same network.   **3. Configure the Router**   * Click on the router, configure the router interfaces with IP addresses that match the network of the PCs and server. * Use commands in the CLI * Router> enable * Router# configure terminal * Router(config)# interface [interface\_name] * Router(config-if)# ip address [ip\_address] [subnet\_mask] * Router(config-if)# no shutdown * Router(config-if)# exit   **4. Configure the Email Server**   * Click on the server and go to the "Services" tab. * Select "Email" from the left panel. * **Enable SMTP** and configure the domain name (e.g., mydomain.com). * Set usernames and passwords for at least two email accounts for testing (e.g., user1@mydomain.com and user2@mydomain.com).   **5. Configure the PCs (Email Clients)**   * On each PC, click "Desktop" and open the **Email** client. * Configure the email client with the following:   + **SMTP Server**: The server’s IP address.   + **Username**: The email accounts created on the server (e.g., user1@mydomain.com).   + **Password**: The password set for the user on the server.   **6. Send a Test Email**   * From one PC, compose and send an email to the other user account. * Go to the second PC, open the email client, and receive the email. |
| **Output:** |
| **Observation & Learning:**  **Observation:**  In this SMTP simulation, the network was successfully set up with PCs and a server configured for email communication. The email was transmitted from one PC to the server and then to the recipient PC without issues, confirming that the SMTP protocol was functioning correctly. Using simulation mode, the packet flow was visualized, showing how the message is transmitted across the network, ensuring proper routing and communication between devices.  **Learning:**  Through this experiment, I learned how SMTP operates within a network to send and receive emails. I gained a deeper understanding of how email clients communicate with mail servers using the protocol and how network components like routers and IP addressing contribute to successful email transmission. Additionally, configuring and troubleshooting network devices in Cisco Packet Tracer enhanced my understanding of practical networking concepts. |
| **Conclusion:** The SMTP simulation in Cisco Packet Tracer demonstrated a practical application of email communication over a network. By successfully configuring network devices, email clients, and a mail server, the experiment highlighted how SMTP facilitates the transfer of emails between devices. This exercise enhanced understanding of both the SMTP protocol and network configuration, showcasing the importance of proper IP addressing and device setup for seamless communication in real-world networks. The experiment effectively reinforced key networking and protocol concepts essential for understanding email services. |