SRM Institute of Science and Technology

Department of Computer Applications

Delhi – Meerut Road, Sikri Kalan, Ghaziabad, Uttar Pradesh – 201204

Circular – 2020-21

MCA 2nd semester
COMPUTER NETWORKS (PCA20C05J)

Lab Manual

Lab 1: Familiarization with Configuring and Installing a LAN using Packet Tracer

Title: Familiarization with Configuring and Installing a LAN using Packet Tracer

Aim: To familiarize with the process of configuring and installing a Local Area Network (LAN) using Cisco Packet Tracer.

Procedure:

1. Open Cisco Packet Tracer.
2. Drag and drop a few end devices (e.g., PCs, Laptops) onto the workspace.
3. Add a network device, such as a Switch (e.g., 2960 series), to connect the end devices.
4. Connect the end devices to the switch using appropriate cables (e.g., Copper Straight-Through cable).
5. Configure IP addresses and subnet masks for each end device within the same network segment.
6. Verify connectivity between the end devices using the `ping` command from the command prompt of one PC to another.
7. Observe the packet flow in simulation mode to understand how data travels within the LAN.

Source Code:

- N/A (This lab involves graphical configuration and command-line interface within Cisco Packet Tracer. No traditional programming source code is involved.)

Input:

- IP addresses: e.g., PC1: `192.168.1.10`, PC2: `192.168.1.11`
- Subnet Mask: `255.255.255.0`

Expected Output:

- Successful `ping` replies between all connected end devices, indicating proper LAN connectivity.
- Observation of data packets flowing through the switch in simulation mode.

Lab 2: Experimenting with Network Protocols for Achieving Communication between Computers using Packet Tracer

Title: Experimenting with Network Protocols for Achieving Communication between Computers using Packet Tracer

Aim: To experiment with various network protocols (e.g., ARP, ICMP, DNS, HTTP) to achieve and observe communication between computers in Cisco Packet Tracer.

Procedure:

1. Set up a basic LAN as in Lab 1.
2. **ICMP (Ping):** Use the `ping` command between two PCs to observe ICMP packet exchange in simulation mode.
3. **ARP:** Clear the ARP cache on a PC (`arp -d *`) and then ping another PC. Observe ARP requests and replies in simulation mode.
4. **DNS:** Add a server (e.g., DNS Server) and configure DNS services. Configure a PC to use this DNS server. Attempt to ping a hostname and observe DNS queries/replies.
5. **HTTP:** Add a server (e.g., HTTP Server) and enable HTTP services. Configure a PC to access the web server via its browser. Observe HTTP requests and responses.
6. Analyze the PDU (Protocol Data Unit) details for each protocol in simulation mode.

Source Code:

- N/A (This lab involves network configuration and observation within Cisco Packet Tracer.)

Input:

- IP addresses and subnet masks for devices.
- DNS server configuration (IP address, domain names).
- HTTP server configuration (web page content).
- Commands like `ping`, `arp -d *`, web browser requests.

Expected Output:

- Successful communication observed for each protocol (e.g., successful pings, web page loading).
- Detailed PDU information showing the headers and data for each protocol in simulation mode.
- Understanding of how different protocols contribute to network communication.

Lab 3: Creating a LAN using Packet Tracer

Title: Creating a LAN using Packet Tracer

Aim: To design, implement, and verify the functionality of a Local Area Network (LAN) using Cisco Packet Tracer.

Procedure:

1. **Design Phase:** Draw a logical topology for a small LAN (e.g., 5 PCs, 1 server, 1 switch). Assign IP addresses and subnet masks logically.
2. **Implementation Phase:**
    - Place the chosen devices (PCs, Server, Switch) on the Packet Tracer workspace.
    - Connect the devices using appropriate cables (e.g., Straight-Through for PC to Switch, Cross-Over for Switch to Switch if expanding).
    - Configure IP addresses, subnet masks, and default gateways (if applicable) on all end devices and server.
    - Configure the switch (e.g., VLANs, port security - optional for a basic LAN).
3. **Verification Phase:**
    - Use `ping` commands to test connectivity between all devices in the LAN.
    - Test services (e.g., if a server is included, test HTTP or DNS access from PCs).
    - Use `ipconfig` (on PCs) and `show ip interface brief` (on switch CLI) to verify configurations.

Source Code:

- N/A (This lab focuses on network design and configuration within Cisco Packet Tracer.)

Input:

- Logical network design.
- Specific IP addressing scheme (e.g., `192.168.1.0/24`).
- Device types and connections.

Expected Output:

- A functional LAN topology in Packet Tracer where all devices can communicate with each other.
- Successful verification of connectivity and services.
- A clear understanding of LAN design and implementation principles.

Lab 4: To Study Different Types of Transmission Media

Title: Study of Different Types of Transmission Media

Aim: To understand and identify various types of transmission media used in computer networks, including their characteristics, advantages, and disadvantages.

Procedure:

1. **Guided Media Study:**
   - **Twisted-Pair Cable:** Research and identify characteristics of UTP and STP cables (categories, connectors, applications, limitations).
   - **Coaxial Cable:** Research and identify characteristics of thinnet and thicknet (types, connectors, applications, limitations).
   - **Fiber Optic Cable:** Research and identify characteristics of single-mode and multi-mode fiber (core/cladding, light sources, applications, advantages/disadvantages).
2. **Unguided Media Study:**
   - **Radio Waves:** Research characteristics (frequency bands, applications like Wi-Fi, Bluetooth, satellite communication).
   - **Microwaves:** Research characteristics (line-of-sight, applications like terrestrial microwave, satellite communication).
   - **Infrared:** Research characteristics (short-range, line-of-sight, applications like remote controls).
3. **Comparison:** Create a comparative table summarizing the key features, bandwidth, distance, cost, and typical applications of each medium.

Source Code:

- N/A (This is a theoretical and research-based lab.)

Input:

- Research materials (textbooks, online resources, datasheets).

Expected Output:

- A comprehensive report or presentation detailing the different types of transmission media.
- A clear understanding of the physical layer aspects of networking.
- Ability to differentiate between various media based on their properties and suitability for different network scenarios.

Lab 5: Interconnection Software for Communication between Two Different Network Architectures using Packet Tracer

Title: Interconnection Software for Communication between Two Different Network Architectures using Packet Tracer

Aim: To simulate and understand the role of interconnection devices (e.g., routers, multi-layer switches) in enabling communication between two different network architectures (e.g., two distinct LANs) using Cisco Packet Tracer.

Procedure:

1. **Setup Two LANs:** Create two separate LANs, each with its own unique network address (e.g., LAN A: `192.168.1.0/24`, LAN B: `192.168.2.0/24`).
2. **Introduce a Router:** Add a router to the workspace.
3. **Connect LANs to Router:** Connect one interface of the router to LAN A's switch and another interface to LAN B's switch.
4. **Configure Router Interfaces:** Assign IP addresses to the router interfaces that belong to their respective LANs (e.g., Router Fa0/0: `192.168.1.1`, Router Fa0/1: `192.168.2.1`).
5. **Configure Default Gateways:** Set the default gateway on all end devices in LAN A to the router's interface IP in LAN A (`192.168.1.1`), and similarly for LAN B.
6. **Test Connectivity:** Attempt to `ping` a PC in LAN B from a PC in LAN A, and vice versa.
7. **Observe Routing:** In simulation mode, observe how packets are forwarded by the router between the two different networks.

Source Code:

- N/A (This lab involves network configuration and observation within Cisco Packet Tracer.)

Input:

- Two distinct network address ranges.
- Router interface IP addresses.
- Default gateway configurations on end devices.

Expected Output:

- Successful `ping` replies between devices in different LANs, demonstrating inter-network communication.
- Observation of the router forwarding packets between the two networks, illustrating its role as an interconnection device.
- Understanding of how routers enable communication across different IP subnets.

Lab 6: Using Packet Tracer to Connect a Network with Different Types of Media Connection

Title: Using Packet Tracer to Connect a Network with Different Types of Media Connection

Aim: To connect network devices using various types of transmission media (e.g., copper, fiber, wireless) within a single network topology in Cisco Packet Tracer.

Procedure:

1. **Mixed Media LAN:**
   - Start with a central switch.
   - Connect some PCs to the switch using **Copper Straight-Through** cables.
   - Add a server or another switch that requires a **Fiber Optic** connection. Configure the switch with a fiber module and connect it using a fiber cable.
   - Add a Wireless Access Point (WAP) and a laptop with a wireless adapter. Connect the WAP to the switch via copper. Configure the wireless connection for the laptop.
2. **WAN Connection (Optional but good for media):**
   - Connect two routers.
   - Use a **Serial DTE/DCE** cable to connect their serial interfaces, simulating a WAN link.
   - Configure IP addresses on the serial interfaces and routing protocols (e.g., static routes) to allow communication over the serial link.
3. **Verify Connectivity:** Test communication between devices connected via different media types (e.g., wired PC to wireless laptop, wired PC to device over fiber link).

Source Code:

- N/A (This lab focuses on physical layer connections and basic configuration within Cisco Packet Tracer.)

Input:

- Various cable types available in Packet Tracer.
- Appropriate network modules for devices (e.g., fiber modules for switches/routers, wireless adapters for end devices).
- IP addresses and basic configurations for devices.

Expected Output:

- A network topology in Packet Tracer demonstrating the use of copper, fiber, and wireless media.
- Successful communication between devices connected through different media types.
- Practical understanding of how different physical media are integrated into a network.

Lab 7: Error Detecting Code Using CRC-CCITT (16-bit) - Java/C/C++ Program

Title: Error Detecting Code Using CRC-CCITT (16-bit)

Aim: To implement an error-detecting code using the CRC-CCITT (16-bit) algorithm in a programming language (Java/C/C++).

Procedure:

1. **Understand CRC:** Study the Cyclic Redundancy Check (CRC) algorithm, focusing on polynomial division using XOR operations.
2. **CRC-CCITT (16-bit) Polynomial:** Identify the standard generator polynomial for CRC-CCITT (16-bit), which is x16+x12+x5+1 (binary: `10001000000100001`).
3. **Implementation Steps:**
   - Append `n` zeros to the data word (where `n` is the degree of the generator polynomial, i.e., 16 for CRC-CCITT 16-bit).
   - Perform binary division (XOR operation) of the augmented data word by the generator polynomial.
   - The remainder obtained is the CRC checksum.
   - Append this checksum to the original data word to form the codeword.
   - At the receiver, divide the received codeword by the same generator polynomial. If the remainder is zero, no error is detected.

Source Code (C++ Example):

```
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>

// Function to perform XOR operation on two binary strings
std::string xor_op(const std::string& a, const std::string& b) {
    std::string result = "";
    int n = b.length();
    for (int i = 1; i < n; i++) {
        if (a[i] == b[i])
            result += "0";
        else
            result += "1";
    }
    return result;
}

// Function to perform binary division for CRC calculation
std::string divide(std::string dividend, std::string divisor) {
    int divisor_len = divisor.length();
    int dividend_len = dividend.length();
    std::string temp = dividend.substr(0, divisor_len);
    std::string remainder = "";

    for (int i = 0; i < dividend_len - divisor_len + 1; i++) {
        if (temp[0] == '1') {
            remainder = xor_op(temp, divisor);
        } else {
            remainder = xor_op(temp, std::string(divisor_len, '0'));
        }
```

```
        if (i < dividend_len - divisor_len) {
            remainder += dividend[divisor_len + i];
        }
        temp = remainder;
    }
    return remainder;
}

// Function to calculate CRC
std::string calculate_crc(std::string data, std::string generator) {
    int generator_len = generator.length();
    std::string augmented_data = data + std::string(generator_len - 1, '0');
    std::string remainder = divide(augmented_data, generator);
    return remainder;
}

int main() {
    std::string data_word;
    std::string generator_polynomial = "10001000000100001"; // CRC-CCITT (16-
bit) polynomial

    std::cout << "Enter the data word (binary string): ";
    std::cin >> data_word;

    std::string crc_checksum = calculate_crc(data_word, generator_polynomial);
    std::cout << "CRC Checksum: " << crc_checksum << std::endl;

    std::string transmitted_codeword = data_word + crc_checksum;
    std::cout << "Transmitted Codeword: " << transmitted_codeword << std::endl;

    // Simulate reception and error checking
    std::string received_codeword;
    std::cout << "\nEnter the received codeword (binary string, can be same as
transmitted for no error): ";
    std::cin >> received_codeword;

    std::string received_remainder = divide(received_codeword,
generator_polynomial);

    if (std::all_of(received_remainder.begin(), received_remainder.end(),
[](char c){ return c == '0'; })) {
        std::cout << "No error detected in the received codeword." << std::endl;
    } else {
        std::cout << "Error detected in the received codeword. Remainder: " <<
received_remainder << std::endl;
    }

    return 0;
}
```

Input:

- **Data Word (Binary):** `1101011011`
- **Generator Polynomial (Binary):** `10001000000100001` (Hardcoded in the example for CRC-CCITT 16-bit)

Expected Output (for Data Word `1101011011`):

- **CRC Checksum:** (Calculated by the program, e.g., `0101110000000000`)
- **Transmitted Codeword:** (Data Word + CRC Checksum, e.g., `11010110110101110000000000`)

- **Received Codeword Check:** "No error detected" if the received codeword is correct, or "Error detected" with the non-zero remainder if an error is introduced.

Lab 8: Case Study Submission for: Sliding-Window Flow Control & Stop-And-Wait Flow Control

Title: Case Study: Sliding-Window Flow Control & Stop-And-Wait Flow Control

Aim: To analyze and compare the principles, advantages, and disadvantages of Sliding-Window Flow Control and Stop-And-Wait Flow Control protocols, and to identify scenarios where each protocol is most suitable.

Procedure:

1. **Research Stop-and-Wait:**
   - Understand its mechanism: sender sends one frame, waits for ACK before sending the next.
   - Analyze its efficiency (low utilization in high-latency/high-bandwidth links).
   - Discuss its simplicity and error handling.
2. **Research Sliding Window:**
   - Understand its mechanism: sender can transmit multiple frames within a window before waiting for ACKs.
   - Differentiate between Go-Back-N and Selective Repeat.
   - Analyze its efficiency (better utilization of bandwidth).
   - Discuss its complexity and error handling.
3. **Comparative Analysis:**
   - Create a table comparing key parameters: throughput, buffer requirements, complexity, error recovery mechanisms, suitability for different network conditions (e.g., high vs. low latency, high vs. low error rate).
4. **Scenario Analysis:**
   - Identify network scenarios where Stop-and-Wait is appropriate (e.g., very short links, low bandwidth).
   - Identify network scenarios where Sliding Window is necessary (e.g., long-distance links, high bandwidth, satellite communication).
5. **Conclusion:** Summarize the findings and provide a reasoned conclusion on the trade-offs between the two protocols.

Source Code:

- N/A (This is a theoretical case study and research-based lab.)

Input:

- Academic papers, textbooks, and online resources on flow control protocols.
- Network scenario descriptions for analysis.

Expected Output:

- A well-structured case study document (report) that clearly explains both protocols.
- A comprehensive comparison highlighting their strengths and weaknesses.
- Examples of practical applications for each flow control mechanism.

Lab 9: SIMULATION OF STOP AND WAIT PROTOCOL using NS/2 or any other tool

Title: Simulation of Stop-and-Wait Protocol using NS/2 or any other tool

Aim: To simulate the Stop-and-Wait protocol to understand its operation, performance characteristics, and limitations (e.g., low throughput in high-latency networks) using a network simulation tool like NS/2.

Procedure:

1. **Tool Familiarization:** If using NS/2, ensure it is installed and basic commands are understood.
2. **Scenario Design:**
   - Create a simple network topology (e.g., two nodes connected by a duplex link).
   - Define link characteristics (bandwidth, propagation delay).
   - Define packet size and number of packets to transmit.
3. **Protocol Implementation (in NS/2 Tcl script):**
   - Define agents for sender and receiver.
   - Implement the Stop-and-Wait logic:
     - Sender sends one data packet.
     - Sender sets a timer.
     - Sender waits for ACK.
     - If ACK received before timeout, send next packet.
     - If timeout, retransmit the current packet.
     - Receiver sends ACK upon receiving a valid data packet.
4. **Simulation Execution:** Run the NS/2 Tcl script.
5. **Trace Analysis:** Analyze the generated trace file (e.g., using `nam` for visualization or a custom script for parsing) to observe:
   - Packet transmission and reception times.
   - ACK transmissions.
   - Retransmissions due to timeouts.
   - Calculate throughput and end-to-end delay.

Source Code:

- N/A (This lab involves creating a simulation script in a tool like NS/2. A typical NS/2 script is written in Tcl.)

Input:

- NS/2 Tcl script defining the network topology, link parameters, and Stop-and-Wait logic.
- Configuration parameters within the script (e.g., `set val(bw) 1Mb`, `set val(delay) 100ms`).

Expected Output:

- Visualization of packet flow in `nam` showing the Stop-and-Wait behavior.
- Trace file analysis demonstrating:
  - Sequential transmission of data packets.
  - ACKs for each packet.
  - Potential retransmissions if errors or delays are introduced.
- Calculated performance metrics (e.g., low throughput for high delays).

Lab 10: Study of Switches, Bridges using Cisco Packet Tracer

Title: Study of Switches and Bridges using Cisco Packet Tracer

Aim: To understand the functionality, differences, and configuration of network switches and bridges, including how they forward frames and build MAC address tables, using Cisco Packet Tracer.

Procedure:

1. **Bridge Simulation:**
   - Create a simple network with two segments connected by a bridge (e.g., two hubs, each with a PC, connected by a bridge).
   - Observe how the bridge learns MAC addresses and forwards/filters frames based on its MAC address table.
   - Send traffic between PCs on the same segment and different segments.
2. **Switch Simulation:**
   - Create a network with multiple PCs connected to a single switch.
   - Observe how the switch learns MAC addresses and forwards frames only to the destination port (unlike a hub).
   - Use `show mac address-table` command on the switch CLI to view the learned MAC addresses.
   - Introduce a broadcast storm (e.g., by connecting two switch ports with a single cable, if spanning tree is not active) and observe its effect.
3. **VLAN Configuration (Optional):**
   - Configure VLANs on a switch to segment the network logically.
   - Assign ports to different VLANs and test communication within and between VLANs (if routing is configured).

Source Code:

- N/A (This lab involves network configuration and observation within Cisco Packet Tracer.)

Input:

- Network topology with bridges and switches.
- `ping` commands to generate traffic.
- CLI commands for switch configuration and verification (e.g., `show mac address-table`).

Expected Output:

- Observation of bridges forwarding frames based on MAC addresses and segmenting collision domains.
- Observation of switches intelligently forwarding frames to specific ports, reducing unnecessary traffic.
- Verification of MAC address learning on switches.
- Understanding of the role of switches and bridges in local area networks.

Lab 11: To Configure Network Security using Two Routers by Blocking ICMP Ping Request - CISCO Packet Tracer

Title: Configuring Network Security by Blocking ICMP Ping Request using Cisco Packet Tracer

Aim: To configure network security using Access Control Lists (ACLs) on two routers to block ICMP ping requests between specific networks or hosts in Cisco Packet Tracer.

Procedure:

1. **Network Setup:**
    - Create a topology with two routers (Router A and Router B) connected via a serial link.
    - Connect a LAN segment to each router (e.g., LAN A to Router A, LAN B to Router B).
    - Assign IP addresses to all interfaces and configure basic routing (e.g., static routes or a routing protocol like RIP/OSPF) to ensure full connectivity between LAN A and LAN B.
2. **Initial Connectivity Test:**
    - From a PC in LAN A, `ping` a PC in LAN B. Verify successful pings.
3. **ACL Configuration (on Router A, for example):**
    - Decide whether to use a Standard or Extended ACL. For blocking specific traffic types like ICMP, an Extended ACL is generally preferred.
    - Define an ACL to deny ICMP traffic from LAN A to LAN B (or vice versa, or specific hosts).
    - Permit all other necessary traffic.
    - Apply the ACL to the appropriate interface and direction (e.g., `in` or `out`).
    - Example commands:
    - `RouterA(config)# access-list 101 deny icmp 192.168.1.0 0.0.0.255 any echo`
    - `RouterA(config)# access-list 101 permit ip any any`
    - `RouterA(config)# interface GigabitEthernet0/0 // (or relevant interface)`
    - `RouterA(config-if)# ip access-group 101 out // (or in, depending on desired effect)`

4. **Verification:**
    - From a PC in LAN A, attempt to `ping` a PC in LAN B. The pings should now fail.
    - Attempt to access other services (e.g., HTTP) if configured, to ensure only ICMP is blocked.
    - Use `show access-lists` and `show ip interface <interface>` to verify ACL application.

Source Code:

- N/A (This lab involves router configuration commands within Cisco Packet Tracer CLI.)

Input:

- Network topology with two routers and two LANs.
- IP addressing scheme.
- Cisco IOS CLI commands for router configuration and ACLs.

Expected Output:

- Initial successful pings between LANs.
- After ACL configuration, pings between the specified source/destination should fail.
- Other allowed traffic should continue to flow normally.
- Understanding of how ACLs can be used to implement basic network security policies.

Lab 12: Case Study Submission for Routing

Title: Case Study: Routing Protocols

Aim: To analyze and compare different routing protocols (e.g., RIP, OSPF, EIGRP, BGP), their operational principles, convergence mechanisms, scalability, and suitability for various network sizes and types.

Procedure:

1. **Research Distance-Vector Protocols (e.g., RIP):**
   - Understand hop count metric, Bellman-Ford algorithm.
   - Analyze convergence issues (count-to-infinity), split horizon, poison reverse.
   - Discuss suitability for small networks.
2. **Research Link-State Protocols (e.g., OSPF):**
   - Understand SPF algorithm, LSA types, areas.
   - Analyze fast convergence, hierarchical design, scalability.
   - Discuss suitability for large, complex networks.
3. **Research Hybrid/Advanced Protocols (e.g., EIGRP, BGP):**
   - **EIGRP:** Understand DUAL algorithm, feasible successor, unequal cost load balancing.
   - **BGP:** Understand path-vector, AS concept, policy-based routing, suitability for inter-domain routing (Internet).
4. **Comparative Analysis:**
   - Create a detailed comparison table covering:
     - Algorithm (Distance Vector, Link State, Path Vector)
     - Metric used
     - Convergence speed
     - Scalability
     - Resource consumption (CPU, memory, bandwidth)
     - Administrative distance
     - Typical deployment scenarios
5. **Scenario Application:**
   - Propose which routing protocol would be best suited for different network scenarios (e.g., a small office, a large enterprise, an ISP). Justify your choices.

Source Code:

- N/A (This is a theoretical case study and research-based lab.)

Input:

- Academic papers, textbooks, and online resources on routing protocols.
- Descriptions of various network topologies and requirements.

Expected Output:

- A comprehensive case study document detailing the researched routing protocols.
- A clear comparative analysis of their features and performance.
- Recommendations for routing protocol selection based on specific network scenarios.

Lab 13: Designing Various Topologies using Cisco Packet Tracer

Title: Designing Various Network Topologies using Cisco Packet Tracer

Aim: To design and implement different fundamental network topologies (e.g., Bus, Star, Ring, Mesh, Hybrid) using Cisco Packet Tracer and understand their characteristics, advantages, and disadvantages.

Procedure:

1. **Bus Topology:**
   - Place multiple PCs and connect them to a single coaxial cable (or simulate with a hub/switch for simplicity, though true bus is shared medium).
   - Test communication.
2. **Star Topology:**
   - Place a central switch (or hub).
   - Connect multiple PCs to the central switch.
   - Test communication and observe the single point of failure (the central device).
3. **Ring Topology (Simulated):**
   - While true token ring is less common, simulate a logical ring using switches or routers where each device connects to exactly two others, forming a closed loop.
   - Test communication paths.
4. **Mesh Topology (Partial/Full):**
   - For a small number of routers, connect each router directly to every other router (full mesh).
   - For a larger number, create a partial mesh.
   - Test redundancy and multiple paths.
5. **Hybrid Topology:**
   - Combine two or more basic topologies (e.g., a star topology LAN connected to a bus topology backbone, or multiple star LANs connected via a mesh of routers).
   - Test overall connectivity and observe how different topologies are integrated.
6. **Analysis:** For each topology, identify its pros (e.g., ease of installation, fault tolerance) and cons (e.g., single point of failure, cabling complexity).

Source Code:

- N/A (This lab focuses on network design and configuration within Cisco Packet Tracer.)

Input:

- Various network devices (PCs, servers, switches, hubs, routers).
- Different cable types.
- IP addressing schemes.

Expected Output:

- Packet Tracer files demonstrating the visual representation and functional connectivity of each topology.
- A clear understanding of the physical and logical characteristics of different network topologies.

- Ability to choose appropriate topologies for different network requirements.

Lab 14: To Configure Internet Access/Implementation using CISCO Packet Tracer

Title: Configuring Internet Access/Implementation using Cisco Packet Tracer

Aim: To configure and simulate Internet access within a network environment using Cisco Packet Tracer, including concepts like Network Address Translation (NAT) and default routing.

Procedure:

1. **Basic Network Setup:**
   - Create a local network (LAN) with PCs and a switch.
   - Connect a router to this LAN. This router will act as the "edge" router for your local network.
2. **ISP Simulation:**
   - Add another router to represent the "ISP Router."
   - Connect your edge router to the ISP router via a serial link (simulating a WAN connection).
   - Add a server to the ISP side to represent a "Web Server" (e.g., `www.example.com`).
3. **IP Addressing:**
   - Assign private IP addresses to your LAN (e.g., `192.168.1.0/24`).
   - Assign public-like IP addresses to the WAN link between your router and the ISP router (e.g., `203.0.113.0/30`).
   - Assign a public IP to the ISP's web server.
4. **Default Routing:**
   - On your edge router, configure a default route pointing towards the ISP router (e.g., `ip route 0.0.0.0 0.0.0.0 [ISP_Router_Interface_IP]`).
   - On the ISP router, configure a route back to your public IP range (or a default route if it's the only client).
5. **NAT Configuration:**
   - Configure NAT (Network Address Translation) on your edge router to translate private LAN IP addresses to a public IP address when traffic leaves your network towards the ISP.
   - Use either Static NAT (for a server) or Dynamic NAT/PAT (for multiple users).
   - Example PAT commands:
   - `Router(config)# ip nat inside source list 1 interface Serial0/0/0 overload`
   - `Router(config)# access-list 1 permit 192.168.1.0 0.0.0.255`
   - `Router(config)# interface GigabitEthernet0/0 // (LAN interface)`
   - `Router(config-if)# ip nat inside`
   - `Router(config-if)# interface Serial0/0/0 // (WAN interface)`
   - `Router(config-if)# ip nat outside`

6. **Verification:**
   - From a PC in your LAN, attempt to `ping` the ISP's web server.
   - From a PC in your LAN, attempt to access the web server via its browser.
   - Use `show ip nat translations` on your edge router to see active NAT entries.

Source Code:

- N/A (This lab involves router configuration commands within Cisco Packet Tracer CLI.)

Input:

- Network topology with LAN, edge router, ISP router, and web server.
- Private and public IP addressing schemes.
- Cisco IOS CLI commands for static/default routing and NAT configuration.

Expected Output:

- Successful `ping` replies from LAN PCs to the ISP's web server.
- Successful access to the web server from LAN PCs using a web browser.
- Verification of NAT translations occurring on the edge router.
- Understanding of how NAT and default routing enable private networks to access the Internet.

Lab 15: Web Programming using HTML

Title: Web Programming using HTML

Aim: To create basic web pages using HTML to understand the fundamental structure, elements, and attributes of web development.

Procedure:

1. **Basic HTML Structure:** Understand the essential tags: `<!DOCTYPE html>`, `<html>`, `<head>`, `<title>`, `<body>`.
2. **Headings and Paragraphs:** Use `<h1>` to `<h6>` for headings and `<p>` for paragraphs.
3. **Text Formatting:** Experiment with `<b>` (bold), `<i>` (italic), `<u>` (underline), `<strong>`, `<em>`.
4. **Lists:** Create ordered lists (`<ol>`) and unordered lists (`<ul>`) with list items (`<li>`).
5. **Links:** Create hyperlinks (`<a>` tag) to external websites and internal sections.
6. **Images:** Insert images (`<img>` tag) using `src` and `alt` attributes.
7. **Tables:** Create simple tables (`<table>`, `<tr>`, `<th>`, `<td>`).
8. **Forms (Basic):** Create a simple form (`<form>`, `<input type="text">`, `<input type="submit">`, `<label>`).
9. **Save and View:** Save the HTML file with a `.html` extension and open it in a web browser to view the output.

Source Code (Example: `index.html`):

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>My First HTML Page</title>
    <style>
        /* Basic styling for readability */
        body {
            font-family: Arial, sans-serif;
            margin: 20px;
            line-height: 1.6;
        }
        h1 {
            color: #333;
        }
        table, th, td {
            border: 1px solid #ccc;
            border-collapse: collapse;
            padding: 8px;
        }
        form {
            margin-top: 20px;
            padding: 15px;
            border: 1px solid #eee;
            border-radius: 5px;
            background-color: #f9f9f9;
        }
        input[type="text"] {
            padding: 8px;
            margin-bottom: 10px;
            border: 1px solid #ddd;
```

```
            border-radius: 3px;
        }
        input[type="submit"] {
            background-color: #4CAF50;
            color: white;
            padding: 10px 15px;
            border: none;
            border-radius: 4px;
            cursor: pointer;
        }
        input[type="submit"]:hover {
            background-color: #45a049;
        }
    </style>
</head>
<body>

    <h1>Welcome to My Web Page!</h1>

    <p>This is a **paragraph** of text. We can use <i>italic</i>, <b>bold</b>,
and <u>underline</u> formatting.</p>

    <h2>About Me</h2>
    <p>I am learning web development. Here are some things I enjoy:</p>

    <h3>My Favorite Hobbies (Unordered List)</h3>
    <ul>
        <li>Reading books</li>
        <li>Playing sports</li>
        <li>Learning new technologies</li>
    </ul>

    <h3>My Top 3 Programming Languages (Ordered List)</h3>
    <ol>
        <li>Python</li>
        <li>JavaScript</li>
        <li>C++</li>
    </ol>

    <h2>Useful Links</h2>
    <p>Visit <a href="https://www.google.com" target="_blank">Google</a> for
searching or check out the <a href="#contact">Contact Section</a> below.</p>

    <h2>An Example Image</h2>
    <img src="https://placehold.co/300x200/007bff/ffffff?text=Web+Image"
alt="Placeholder Web Image" style="max-width: 100%; height: auto; border-radius:
8px;">
    <p><i>A simple placeholder image.</i></p>

    <h2>My Schedule</h2>
    <table>
        <thead>
            <tr>
                <th>Day</th>
                <th>Activity</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>Monday</td>
                <td>Study HTML</td>
            </tr>
            <tr>
                <td>Tuesday</td>
                <td>Practice CSS</td>
            </tr>
        </tbody>
```

```
    </table>

    <h2 id="contact">Contact Me</h2>
    <form action="#" method="post">
        <label for="name">Name:</label><br>
        <input type="text" id="name" name="name" placeholder="Your Name"><br>
        <label for="email">Email:</label><br>
        <input type="text" id="email" name="email"
placeholder="your.email@example.com"><br><br>
        <input type="submit" value="Submit">
    </form>

</body>
</html>
```

Input:

- The HTML code itself, written in a text editor.

Expected Output:

- A rendered web page in a browser displaying:
    - A main heading "Welcome to My Web Page!".
    - A paragraph with various text formatting.
    - Headings "About Me", "My Favorite Hobbies", "My Top 3 Programming Languages".
    - An unordered list of hobbies.
    - An ordered list of programming languages.
    - Hyperlinks to Google and an internal section.
    - A placeholder image.
    - A simple table with a schedule.
    - A basic contact form with input fields and a submit button.