Lab 1:

GNS3

- GNS3 stands for Graphical Network Simulator-3. It's a popular network simulation software used by networking professionals, students, and instructors to design, configure, and test complex network topologies without the need for physical hardware.

- GNS3 allows users to emulate different types of network devices, such as routers, switches, and firewalls, using real IOS (Internetwork Operating System) images provided by Cisco or other vendors. Users can create virtual networks by dragging and dropping devices onto a simulated workspace, then connect them together using virtual cables.

- One of the key advantages of GNS3 is its flexibility and scalability. Users can create and test large-scale network scenarios, experiment with different configurations, and troubleshoot network issues in a safe virtual environment. GNS3 also supports integration with external applications and tools, such as Wireshark for packet analysis and VirtualBox for virtual machine integration.

Class Work:

As this was an Intro class, sir didn't give us any lab work. But sir gave us an Intro to GNS3. After launching GNS3, we will make a new project and add as many nodes we need (if we need 2 PCs, we will add VPCs), then we add a switch and connect the PCs using a cable to some ports in the Switch.

After that, we turn on the PCs and right click on it to open a console of the PC. After opening consoles of both the PCs, a terminal appears. So, now, we can run commands in this. We can set IP Addresses of the PCs.

So, to do that, we will run the command - "ip <some IP Address>"

This command will check if the IP Address is empty and assign this to the virtual PC.

After this, we will run the command - "save"

to save this IP to the PC.

To see what IP is set to it, we can always run the command "show ip".

E.g., We can assign an IP to the PC1 in this manner, "ip 10.0.0.1/8", here 8 is the subnet mask.

We can assign an IP to the PC2 in this manner, "ip 10.0.0.2/8", here 8 is the subnet mask.

Now, we mentioned 8 as the subnet, so both the PCs have the same Network Address in IP. So, the digit before the first decimal point is our Network Address ('10' in 8 bits would be '00001010').

So, for systems to communicate via a switch, the network part of the IP Address should be equal as Switches work on Layer 2 and don't route a data packet. Routing a data packets are done by Routers only.

So, now, we can ping one device from another using the command - "ping <IP Address>"

IF THE NETWORK PART IS DIFFERENT, WE CANNOT COMMUNICATE BETWEEN DEVICES AS IT MEANS THAT THE DEVICES ARE IN DIFFERENT NETWORKS. AND PINGING SUCH IP WILL GIVE US AN ERROR.

SWITCHES ONLY FORWARD PACKETS IN THE LOCAL NETWORK.

To communicate between devices belonging to different networks, we will need a Router.

To show this, sir changed the IP of one of the PCs and we couldn't ping them.

Then we changed the switch to a router.

After that, we turned on the router and opened its console.

In that, we can see a console and we can make the changes to make a Gateway using to connect the two Network Addresses. In that console, we can run a command "?" to access the help section.

Now, as we need to configure the router, we need to open the configure terminal in that,

to do that, we can simply run the command, "configure terminal"

This will start the configure mode.

After that, we need to run the command, "interface FastEthernet0/0", to set the IP Address of the 1st Port which is connected to the PC1.

After this, we will run the command, "ip address 10.0.0.254 255.0.0.0", to mention the IP Address that the PC1 accepts and the subnet mask of it. (NOTE - GNS3 uses a pretty old Cisco Router which understands Subnets only when written like this 255.255.255.255 format, i.e., writing the subnet after the IP address using a '/' won't be understandable by the router).

After that, we will run the command, "no shutdown", so that the port and connection of the PC1 does not go to sleep or shutdown if there is no data transfer.

Similarly, we have to do the same thing for the second port of the router which is connected to PC2, using the command "interface FastEthernet0/1"

After this, set the IP of PC2 using the command. "ip address 11.0.0.254 255.0.0.0" as the PC2 only accepts IP Addresses with '11' as the network address.

Again, run the command "no shutdown" so that the port and the connection of PC2 does not go to sleep or shutdown if there is no data transfer.

Now, enter "exit" to exit from the config mode of the Router's Terminal.

Now, go to any PCs console and type the following command to make a Gateway in it,

"ip <IP Address of the PC> <Last IP Address of that network>"

e.g., for the 1st PC type the following command: "ip 10.0.0.1/8 10.0.0.254"

for the 2nd PC type this: "ip 11.0.0.1/8 11.0.0.254"

This is for setting a default gateway that the PC has an IP 10.0.0.1 (for PC1) and it has a gateway to the IP 10.0.0.254 in the router and it can communicate through that.

This makes the communication between both the PCs possible even though they are technically in different networks.

Now, pinging these IP addresses won't give us any kind of error.

Sir also taught us about ARP Tables.

Writing the command:

"arp"

in any console belonging to the PCs shows us it's ARP table which has stored the MAC address of the node in that IP Address.

Note - The entries in ARP tables are not permanent at all and they are stored for around 2 and a half minutes only because if it does not delete entries, the table might fill up soon and become too long to be used for traversal in a cache. So, the table keeps on clearing the stored contents after sometime.

Note - We can also make some static entries in an ARP table to avoid the broadcasting and delay in first data delivery. These static entries are used between devices which communicate regularly. Static ARP Entries also help us in preventing Security Issues like Man in the Middle and all.

Lab2:

HUBS Vs. SWITCHES

Hubs and switches are both devices used in computer networks to connect multiple devices together, such as computers, printers, and servers. However, they operate in different ways and have different functionalities:

Hub:

-> A hub is a basic networking device that operates at the physical layer (Layer 1) of the OSI model.

-> It works by broadcasting data to all devices connected to it, regardless of whether the data is intended for a specific device.

-> Hubs are essentially multi-port repeaters. When they receive data on one port, they broadcast that data out on all other ports.

-> They are simple and inexpensive but less efficient compared to switches, as they can cause network congestion and collisions, especially in larger networks.

Switch:

-> A switch is a more advanced networking device that operates at the data link layer (Layer 2) of the OSI model.

-> Unlike hubs, switches maintain a MAC address table, which maps MAC addresses to the physical ports on the switch.

-> When a switch receives data, it checks the destination MAC address and forwards the data only to the port associated with that MAC address, rather than broadcasting it to all ports. This makes switches more efficient and reduces network congestion.

-> Switches can also support full-duplex communication, allowing data to be transmitted and received simultaneously on each port.

-> They are more expensive than hubs but offer better performance and security, particularly in larger networks.

WIRESHARK

Wireshark is a widely-used network protocol analyzer. It is an open-source software application that allows you to capture and analyze the data traveling back and forth on a network in real-time. Here are some key features and uses of Wireshark:

1. Packet Capture: Wireshark captures packets (data units) as they travel through the network. It supports capturing packets from a wide range of network interfaces, including Ethernet, Wi-Fi, Bluetooth, and more.

2. Protocol Analysis: Wireshark supports a vast array of network protocols, ranging from common ones like TCP, UDP, HTTP, and DNS to more specialized or proprietary protocols. It can dissect and decode these protocols, allowing you to see the details of each packet, such as headers, payloads, and flags.

3. Live Capture and Offline Analysis: Wireshark can capture packets in real-time from a live network interface, or it can analyze packet capture files that have been saved previously. This makes it useful for troubleshooting network issues as they occur or for analyzing historical network traffic.

4. Filtering and Search: Wireshark provides powerful filtering capabilities, allowing you to focus on specific types of traffic or packets of interest. You can create complex filters based on criteria such as protocol, source or destination IP address, port number, packet size, and more.

5. Statistics and Graphing: Wireshark offers various statistical tools and graphing capabilities to help you analyze network traffic patterns, identify trends, and pinpoint anomalies. You can view statistics such as packet counts, protocol distribution, packet size distribution, and round-trip times.

6. Protocol Decryption: Wireshark can decrypt encrypted traffic if you have access to the necessary encryption keys. This feature is particularly useful for analyzing secure protocols like SSL/TLS or SSH.

7. Cross-Platform: Wireshark is available for multiple operating systems, including Windows, macOS, and various Linux distributions, making it widely accessible to users across different platforms.

CLASS WORK

1. Place a Hub and 3 VPCs and then connect them using ethernet.

2. Then, open the console of the three PCs and assign them IP address using the command: "ip <IP address>"

i. Assign the IP address "10.0.0.1/24" to PC1 using the commands:

ip 10.0.0.1/24

save

ii. Assign the IP address "10.0.0.2/24" to PC2 using the commands:

ip 10.0.0.2/24

save

iii. Assign the IP address "10.0.0.3/24" to PC3 using the commands:

ip 10.0.0.3/24

save

3. Then, click on the wire between the Hub and PC1 and right click and then choose the option "Start Capture" and Wireshark will launch.

4. After that, open the terminal of PC1 and ping PC2 using the command: "ping 10.0.0.2/24".

5. After pinging, we can see in the wireshark, some entries related to the switching of packets, i.e., it will show us entries like how a broadcast was made from PC1 and after that there was a private message from PC2 to PC1 which was in the ARP Protocol,i.e., it sent it's MAC Address to PC1.

6. Then close the terminal of PC3 and delete it.

7. After that, add a "Cloud" from the Devices menu into the area and connect that tp the Hub.

(This Cloud is not like AWS, Azure or GCP. This connects the simulator to the actual physical network of the VM. We can see the route of actual data packets and all using wireshark here.)

8. Now, you can open the terminal in the VM and type the command: "ifconfig", and we can see the Cloud's port on it.

9. Now, open wireshark from "Applications">"Internet">"Wireshark".

10. After that, open the Port of the Cloud in Wireshark and observe it as it's our PC connected to the network and interacting with it. You can see the route of the Data Packets in it.

11. After this, you can ping the PC2 from PC1 and still see the Packets being sent in the Wireshark (which is now sniffing the packets of the cloud) because the Hub broadcasts the data packets to every other node and the Cloud is in fact one of the device which is receiving the broadcast.

LAB WORK

1. Repeat the above exp by replacing the hub with a built in Switch amd observe the packets captured by Wireshark.

-> Steps: The steps are same as above in the Class work.

-> Result: (Only one entry of ARP broadcasting is available in the Wireshark of the cloud node, no other packets are displayed like before because the Switch does not broadcast every other data packet to all the nodes it is connected to, unlike Hub. You can compare the results by opening another Wireshark between PC1 and the Switch, it will show all the packets but the wireshark of the cloud won't display all the others because a switch creates a personal and singular connection between the devices and nothing is broadcasted except the first ARP Broadcast.)

Lab 3:

Notes of 23rd of February, 2024

DHCP

- DHCP stands for Dynamic Host Configuration Protocol.
- It's a network management protocol used on IP networks where a DHCP server dynamically assigns IP addresses and other network configuration parameters to devices on the network.
- This eliminates the need for a network administrator to manually assign IP addresses to every device, making it more efficient and scalable, especially in large networks.
- DHCP operates on the client-server model, where the DHCP server manages a pool of IP addresses and leases them to clients for a specific period of time.

Why is DHCP used?

- DHCP is used for several reasons:
  - Dynamic IP Address Allocation: DHCP allows for the automatic assignment of IP addresses to devices on a network. This dynamic allocation means that devices can join or leave the network without requiring manual intervention from network administrators to assign or reassign IP addresses.
  - Efficiency: DHCP simplifies the process of managing IP addresses in large networks. Instead of having to manually configure IP addresses for each device, DHCP automates this process, saving time and reducing the potential for human error.
  - Centralized Management: With DHCP, network administrators can centrally manage and configure IP address settings from a single DHCP server. This centralized management makes it easier to maintain and update network configurations.

- Address Conservation: DHCP leases IP addresses to devices for a specific period of time. Once a device no longer needs an IP address (e.g., it leaves the network or shuts down), the IP address can be reclaimed by the DHCP server and assigned to another device. This helps conserve IP address space by preventing the permanent allocation of addresses to devices that are not actively using them.

- IP Address Reuse: DHCP allows for the reuse of IP addresses as devices join and leave the network. This flexibility ensures that IP addresses are used efficiently and reduces the likelihood of running out of available addresses.

Evolution of DHCP

- The evolution of DHCP has been driven by the increasing complexity and scale of networks, as well as the need for more efficient and flexible IP address management. Here's a brief overview of the evolution of DHCP:

- DHCPv1: The original version of DHCP, defined in RFC 1531 in 1993. It provided basic automatic IP address assignment and configuration parameters for devices on a network.

- DHCPv2: This version introduced enhancements and improvements over DHCPv1 but was never widely deployed or standardized.

- DHCPv3: DHCPv3 was developed as part of the Dynamic Host Configuration working group within the IETF (Internet Engineering Task Force). It aimed to address limitations and issues with previous versions, such as better support for IPv6 and improved security features. However, DHCPv3 was never officially standardized or widely adopted.

- DHCPv4: DHCPv4, also known as DHCP for IPv4, is the most widely deployed version of DHCP. It is defined in RFC 2131 and RFC 2132 (with updates and extensions in subsequent RFCs) and provides automatic IP address assignment, configuration parameters (such as subnet mask, default gateway, DNS servers), and lease management for IPv4 networks.

- DHCPv6: With the adoption of IPv6, DHCPv6 was developed to provide similar functionality for IPv6 networks. It is defined in RFC 3315 and provides automatic IPv6 address assignment, configuration parameters, and other network settings for devices on an IPv6 network.

- DHCPv6 Prefix Delegation: This extension to DHCPv6 enables the delegation of IPv6 address prefixes to routers, allowing them to dynamically assign prefixes to downstream networks or devices.

- DHCPv6 Failover: DHCPv6 failover mechanisms, defined in RFC 7031, provide redundancy and high availability for DHCPv6 servers, ensuring continuous service even if one server fails.

Evolution of DHCP (from RARP to BOOTP to DHCP)

- The evolution of DHCP can be traced through its predecessors: RARP (Reverse Address Resolution Protocol) and BOOTP (Bootstrap Protocol), which laid the groundwork for DHCP's development:

- RARP (Reverse Address Resolution Protocol):

- RARP was designed to allow a computer to obtain its IP address dynamically by broadcasting its MAC address and requesting an IP address assignment from a RARP server.

- However, RARP had limitations. It only provided IP address assignment and required configuration of each device's MAC address in the RARP server's table, which was cumbersome and not scalable.

- BOOTP (Bootstrap Protocol):

- BOOTP was developed to address some of the limitations of RARP. It extended the functionality of RARP by providing additional parameters such as the IP address of the client's gateway and the IP address of the server hosting the operating system image (bootstrap file).

- Unlike RARP, BOOTP used a client-server architecture where the client (typically a diskless workstation) broadcasted a request, and a BOOTP server responded with the necessary configuration information.

- While an improvement over RARP, BOOTP still required manual configuration of client information on the server, which was not ideal for large networks or environments with frequently changing devices.

- DHCP (Dynamic Host Configuration Protocol):

- DHCP further refined the dynamic address assignment process introduced by RARP and BOOTP.

- Unlike BOOTP, DHCP introduced the concept of address leasing, allowing the DHCP server to dynamically assign IP addresses to clients from a pool of available addresses for a limited time period.

- DHCP also automated the configuration process by allowing for the dynamic allocation of other network configuration parameters, such as subnet mask, default gateway, DNS server, and more.

- Additionally, DHCP introduced the concept of DHCP relay agents, enabling DHCP messages to be forwarded across different network segments, thus facilitating the deployment of DHCP in large, segmented networks.

(What is a thin client?

- A thin client is a lightweight computer or software application that relies heavily on a central server or cloud infrastructure for its computational resources and applications. Unlike traditional desktop computers or thick clients, which have significant processing power, storage, and software installed locally, thin clients are designed to be simpler and more streamlined.

Here are some key characteristics of thin clients:

- Minimal Local Processing: Thin clients typically have limited processing power, memory, and storage capacity. They are designed to offload most of their processing tasks to a central server or cloud infrastructure.

- Remote Access: Thin clients primarily rely on remote access protocols such as Remote Desktop Protocol (RDP), Virtual Network Computing (VNC), or web-based interfaces to access applications and data stored on servers or in the cloud.

- Centralized Management: Thin clients are centrally managed by administrators who can remotely configure, update, and monitor them from a centralized management console. This simplifies maintenance and reduces the administrative overhead compared to managing individual desktop computers.

- Cost-Effective: Thin clients are often more cost-effective than traditional desktop computers because they have lower hardware requirements and longer lifespans. Additionally, since most of the processing occurs on central servers, thin clients can extend the lifespan of older hardware.

- Security: Because thin clients rely on centralized servers for data storage and processing, they can offer enhanced security compared to traditional desktop computers. Centralized management allows for easier implementation of security policies, updates, and access controls.

)

DHCP Pool

- A DHCP pool refers to a range of IP addresses that a DHCP server is configured to dynamically assign to client devices on a network. When a client device requests an IP address via DHCP, the DHCP server selects an available IP address from the pool and leases it to the client for a specified period of time.

The DHCP pool configuration typically includes the following parameters:

- IP Address Range: Specifies the range of IP addresses that are available for assignment to client devices. For example, a pool might include addresses from 192.168.1.100 to 192.168.1.200.

- Subnet Mask: Defines the subnet mask associated with the IP addresses in the pool, indicating the network portion of the IP address.

- Gateway: Specifies the default gateway or router IP address that client devices should use to access other networks.

- DNS Servers: Provides the IP addresses of DNS servers that client devices should use for domain name resolution.

- Lease Time: Specifies the duration for which IP addresses are leased to client devices from the pool.

- Additional Options: May include additional configuration parameters such as domain name, NTP (Network Time Protocol) servers, and more.

DHCP Lease time

- DHCP lease time refers to the duration for which an IP address is assigned to a client device by a DHCP server.

- When a client device connects to a network and requests an IP address via DHCP, the DHCP server assigns an IP address from its pool of available addresses and specifies a lease time for that address.

- During the lease period, the client device is allowed to use the assigned IP address and network configuration parameters (such as subnet mask, default gateway, DNS servers) provided by the DHCP server. Once the lease time expires, the client device must renew its lease by requesting the same IP address from the DHCP server. If the lease is not renewed, the IP address may be reclaimed by the DHCP server and assigned to another client device.

- The lease time is configurable by network administrators and can vary depending on network requirements and policies. Shorter lease times may be used in dynamic environments where devices frequently join and leave the network, ensuring efficient use of IP address space and facilitating IP address reclamation. Longer lease times may be preferred in more stable environments to reduce DHCP traffic and overhead.

What is DORA?

- DORA is an acronym that represents the four-step process involved in DHCP (Dynamic Host Configuration Protocol) for dynamically assigning IP addresses to client devices on a network.

Each letter in "DORA" stands for one of the steps in this process:

- Discover: In the first step, the client device broadcasts a DHCP Discover message to discover DHCP servers available on the network. This message is sent out by a client when it initially connects to the network or when it needs to renew its IP address lease.

- Offer: Upon receiving the DHCP Discover message, DHCP servers on the network respond with a DHCP Offer message. This message contains an available IP address that the server is willing to lease to the client, along with other network configuration parameters such as subnet mask, default gateway, DNS servers, and lease duration.

- Request: After receiving one or more DHCP Offer messages, the client selects one of the offered IP addresses and sends a DHCP Request message to the chosen DHCP server. This message formally requests the offered IP address and confirms the selection.

- Acknowledge: Finally, the DHCP server responds to the client's DHCP Request message with a DHCP Acknowledgment (Ack) message. This message confirms that the client's request has been accepted, and it includes the leased IP address and other configuration parameters. The client can now use the assigned IP address and network settings for the duration of the lease.

LAB WORK

- Demonstrate the exchange of DORA Messages for allocation of IP Address to the client using VPS Facility and using WIreshark.

- Configuring a Router to act as a DHCP Server.

- SETUP

- Place a router and a VPC in the workspace and connect them using an Ethernet.

- Now start the Router and Open it's console.

- After that, run the following commands:

$ configure terminal (this opens the terminal in configure options, allowing us to modify stuff)

$ interface FastEthernet 0/0 (as this selects the 0 port and uses that)

$ ip address 10.0.0.1 255.255.255.0 (This sets the given IP address to that Port)

$ no shutdown (This allows the port to not sleep and keep it active all the time)

$ exit (exits the Interface of the 0 port)

$ ip dhcp? (shows the help settings of DHCP)

$ ip dhcp excluded-address 10.0.0.1 10.0.0.10 (This excludes the provided range of IP Addresses to be assigned to any of its clients)

[ip dhcp excl 10.0.0.1 10.0.0.10, is the short of the above command]

$ ip dhcp pool testdhcp (it opens an option to create a DHCP Pool of the name 'testdhcp' [The name can be changed of course])

$ default-router 10.0.0.1 (sets the default router)

$ network 10.0.0.0 255.255.255.0 (sets the network)

$ dns-server 10.0.0.2 (sets the dns server)

$ lease 1 10 15 (it sets the lease time as 1D 10H and 15M)

$ domain-name home.net (sets the domain name as "home.net")

$ end (ends the dhcp menu)

$ write (writes the DHCP settings that we provided)

$ show ip dhcp pool (shows us the dhcp pools that we have set)

- Now start the VPC and open it's console and write the following commands:

$ ip dhcp -d (It sends a message to DHCP and asks for an IP Address and the DHCP Server sets an IP Address of 10.0.0.11 to the VPC1)

$ show ip (Now it will show the IP Address and all assigned to it by the DHCP)

- Now, turn off the PC and open a wireshark between the Router VHCP Connection and run the command"

$ ip dhcp -d

(Now this will display the DORA Transactions that take place when an IP is assigned from the DHCP within the wireshark)

- After the DHCP assigns an IP, there is an ARP Broadcast of the IP Address of the VPC. This shares the IP and also checks for any collision in IP Addresses.

- QUESTION

  - Configure a Router to act as a DHCP Server, with the following configuration:

    DHCP Pool Name: (any name)

    Server IP Address: 192.168.0.1/24

    Excluded Ip Address Range: 192.168.0.1 to 192.168.0.10

    Default Router: 192.168.0.1

    DNS Server: 192.168.0.2

    Domain Name: jecassam.ac.in

    Lease Time: 2h 30M

  - Display the DORA Message with the help of VPCS Facility, as well as using Wireshark.

  ANSWER -

  Same as before, just change the values.

Lab 4:

Notes of 1st of March, 2024.

VLAN

- A Virtual Local Area Network (VLAN) is a method of logically segmenting a physical network into multiple distinct broadcast domains. This segmentation allows you to isolate traffic and improve network performance, security, and manageability. VLANs are typically configured in switches by grouping together ports and assigning them to specific VLAN IDs.

- Devices within the same VLAN can communicate with each other as if they were connected to the same physical network, even if they are physically located in different parts of the network. VLANs are widely used in enterprise networks to control traffic flow and enforce security policies.

Why is VLAN important?

- VLANs are important for several reasons:

-> Network Segmentation: VLANs allow you to logically segment a network, separating different types of traffic (such as voice, data, or video) or different user groups (like departments or teams) into separate broadcast domains. This segmentation helps improve network performance and security by reducing broadcast traffic and isolating network issues.

-> Security: VLANs provide a level of security by isolating traffic within specific VLANs. By controlling access between VLANs through routers or firewalls, you can restrict communication between different parts of the network, helping to prevent unauthorized access or attacks.

-> Simplified Management: VLANs make network management easier by allowing administrators to group devices logically rather than physically. This flexibility enables easier administration of network policies, such as access control lists (ACLs) and Quality of Service (QoS) settings, across multiple devices.

-> Cost Savings: VLANs can help reduce costs by enabling the consolidation of network devices. Instead of having separate physical switches for each department or function, VLANs allow you to use a single physical switch while maintaining separate logical networks.

-> Flexibility and Scalability: VLANs provide flexibility and scalability in network design. You can easily add, remove, or modify VLANs as needed without physically rewiring the network. This makes it easier to adapt to changes in organizational structure or network requirements.

Why is Segregating the networks important?

- Segregating networks is important for several reasons:

-> Security: Segregation helps contain security breaches. If one segment of the network is compromised, it's harder for an attacker to move laterally to other parts of the network if they are segregated. This containment limits the potential impact of security incidents and reduces the likelihood of unauthorized access to sensitive data or systems.

-> Compliance: Many industries have regulatory requirements that mandate network segregation to protect sensitive information. Segregating networks helps organizations comply with regulations such as the Health Insurance Portability and Accountability Act (HIPAA) or the Payment Card Industry Data Security Standard (PCI DSS).

-> Performance: By segregating different types of traffic into separate networks, you can optimize network performance. For example, separating voice traffic from data traffic can ensure that real-time applications like VoIP receive sufficient bandwidth and low latency, enhancing overall network performance.

-> Resource Management: Segregation allows for more efficient resource allocation and management. By isolating specific types of traffic or user groups into separate networks, you can apply different policies, quality of service (QoS) settings, and bandwidth allocation strategies tailored to the needs of each segment.

-> Troubleshooting and Maintenance: Segregated networks make troubleshooting and maintenance easier. When issues arise, administrators can focus their efforts on the affected segment without disrupting the entire network. Similarly, performing maintenance tasks, such as software updates or configuration changes, can be done more safely and efficiently when network segments are isolated.

LAB WORK:

- Open a new project.

- Then add 2 Ethernet Switches and 6 VPCs.

- Now, we need to connect Switch 1 to 4 PCs, and it should have 10 VLANs assigned to PC1 and PC2, and 20 VLANs to PC3 and PC4.

- And, we need to connect the Switch 2 to the other 2 PCs and it should have 10 VLANs assigned to PC5 and 20 VLANs assigned to PC6.

- And then, we have to create a trunk connection between the two switches.

- So, to do this, right click on the switch and open the "Configure" option.

- After this, for the Switch 1, delete all the preset port settings, and select port 0 and set the VLANs to 10 and the type to "access" and then press "add". Do the same for Port 1. This will give 10 VLANs to the port 0 and 1 which will be connected to PC1 and PC2.

- Now, set do the same and just set VLAN = 20 for ports 2 and 3, which will be connected to PC3 and PC4.

- Now, create a port 4 and set the VLAN to 1 and the type to "dot1q". This is the trunk part of the connection.

- Now, do the same for Switch 2, i.e., set the VLANs of Port 0 to 10, and set the VLANs of Port 1 to 20 and set the Port 2 as trunk and set the VLAN as 1.

- Now, assign the following IPs to the PCs, using the command:

  "ip <ip address>"

  and then

  "save"

IP	PC
10.0.0.1/24	1
10.0.0.2/24	2
10.0.0.3/24	3
10.0.0.4/24	4
10.0.0.5/24	5
10.0.0.6/24	6

- Now, you can ping between PCs with the same VLAN numbers as it apparently sets the VLAN and not the "number of VLAN". Pinging PCs with different VLAN will give us an error as the host is not reachable.

- Now open a new project.
- Add 2 L2 Switches (Cisco Switches) and 6 VPCs and connect it the way we did it before.
- Now, running the switch will give us an error as the Software a key to work.
- To get the key, open a normal terminal, and run the command:

  "python2 keygen.py"

  This will generate a key. Copy that key and paste it into the file named ".iourc"
- Now, restart the GNS3 app and reopen the project.
- Now start the Switch 1 and open its console.
- Now run the commands:

  "show vlan brief" (this shows the brief of the VLANs set up)

  "Configure terminal" (opens the terminal in config mode)

  "vlan 10" (sets the VLAN 10)

  "name red" (sets it to a name "red")

  "no shutdown" (sets the port and VLAN to not shutdown when inactive)

  "exit" (exits the vlan mode)

  "vlan 20"

  "name green"

  "no shutdown"

  "exit"

  "interface range ethernet 0/0 - 1"

  "switchport access vlan 10" (sets the vlan to 10 and mode to access for the ports in the range 0/0-0/1)

  "no shutdown"

  "interface range ethernet 0/2 - 3"

  "switchport access vlan 20"

  "no shutdown"

"interface ethernet 3/3"

"switchport trunk encapsulation dot1q" (apparently you cant set the mode to trunk in Cisco Switches. So, we need to use an encapsulation method to set it as dot1q)

"switchport mode trunk" (sets the mode to trunk now)

"switchport trunk allowed vlan 10,20"

"no shutdown"

"end"

This has set the configuration given above for the 1st switch.

Now, set the config for the 2nd one yourself (THE LAB WORK FOR TODAY).

(ANSWER FOR THE LAB WORK IS THE SAME AS ABOVE)

Lab 5:

08th of March, 2024

Spanning Tree Protocol (STP)

- The Spanning Tree Protocol (STP) is a network protocol that ensures a loop-free topology in Ethernet networks. When multiple paths exist between switches, there's a risk of loops, which can lead to broadcast storms and network congestion. STP prevents these loops by identifying redundant links and blocking some of them, thus creating a loop-free tree structure.

Here's a simplified overview of how STP works:

-> Bridge ID (BID) Selection: Each switch in the network is assigned a unique Bridge ID, consisting of a priority value and the switch's MAC address. The switch with the lowest BID becomes the root bridge, which is the reference point for the spanning tree.

-> Root Bridge Election: Initially, all switches consider themselves as the root bridge. They exchange Bridge Protocol Data Units (BPDU) to determine which switch has the lowest BID. The switch with the lowest BID becomes the root bridge.

-> Designated Ports Selection: Once the root bridge is elected, each non-root switch determines the shortest path to reach the root bridge. The ports through which these paths are established are designated as forwarding ports, while all other ports on the switches are placed in a blocking state.

-> Blocking Ports: If there are redundant links between switches, STP identifies these links and blocks them to prevent loops. These blocked ports remain in a listening and learning state, ready to be activated if a failure occurs on the active path.

-> Topology Changes: STP continually monitors the network for changes in the topology. If a link fails or a new link is added, STP recalculates the spanning tree and adjusts port states accordingly.

-> Spanning Tree Variants: There are multiple variants of STP, including Rapid Spanning Tree Protocol (RSTP) and Multiple Spanning Tree Protocol (MSTP), which improve upon the original STP by reducing convergence times and providing support for multiple VLANs, respectively.

Why STP?

- Spanning Tree Protocol (STP) is essential for several reasons:

-> Loop Prevention: One of the primary reasons for using STP is to prevent loops in Ethernet networks. Without STP, loops can lead to broadcast storms, where broadcast packets continuously circulate the network, consuming bandwidth and causing network congestion. STP identifies and disables redundant links to ensure a loop-free topology.

-> Redundancy: While loops are undesirable, having redundant links between switches can be beneficial for fault tolerance and improved network performance. STP allows the network to utilize these redundant links effectively while preventing loops. If one link fails, STP reconverges the network to activate an alternate path, minimizing downtime.

-> Stability: STP contributes to the stability of Ethernet networks by ensuring a consistent and predictable topology. By establishing a single, optimal path between switches, STP helps in maintaining network stability and preventing unexpected network behaviors.

-> Scalability: Ethernet networks often grow in size and complexity over time. STP provides a scalable solution for managing network topology changes by automatically recalculating the spanning tree in response to changes such as link failures or additions. This scalability allows Ethernet networks to adapt to evolving requirements without manual intervention.

-> Standardization: STP is a widely adopted standard protocol defined by the IEEE 802.1D specification. Its widespread adoption ensures interoperability between networking equipment from different vendors, allowing for the creation of heterogeneous networks that consist of switches from various manufacturers.

-> Variants and Enhancements: Over time, STP has evolved into variants such as Rapid Spanning Tree Protocol (RSTP) and Multiple Spanning Tree Protocol (MSTP), which offer improvements in convergence times, support for multiple VLANs, and better utilization of redundant links. These enhancements make STP even more effective in modern network environments.

What are the problems with loops in an Ethernet Network?

- Loops in an Ethernet network can lead to several significant problems:

-> Broadcast Storms: Broadcast packets are messages sent to all devices on a network segment. In a looped network, these broadcast packets can circulate endlessly, creating what is known as a broadcast storm. As the packets continuously traverse the loop, they consume network bandwidth and overwhelm network devices, leading to degraded network performance and potentially causing network downtime.

-> Packet Duplication: In a looped network, packets may be duplicated as they traverse the loop multiple times. This duplication can result in inefficiencies, increased network latency, and potential data corruption if different versions of the same packet arrive at the destination out of order.

-> MAC Address Table Instability: Switches in Ethernet networks maintain MAC address tables to facilitate the forwarding of packets to their intended destinations. In a looped network, MAC address table entries may become unstable as switches receive packets with the same MAC address arriving on different ports due to the loop. This instability can lead to incorrect packet forwarding and network connectivity issues.

-> Unpredictable Behavior: Loops can cause unpredictable behavior in Ethernet networks, making it challenging for network administrators to troubleshoot and diagnose network problems. Devices may experience intermittent connectivity issues, and network traffic patterns may vary unpredictably, making it difficult to identify the root cause of performance problems.

-> Resource Exhaustion: In extreme cases, loops can lead to resource exhaustion on network devices such as switches and routers. As devices become overwhelmed by the continuous circulation of packets, their processing capabilities may be strained, leading to device crashes or failures and disrupting network services.

STP Standards

- Spanning Tree Protocol (STP) has several standards and variants developed over time to address different aspects of network topology management and improve its efficiency. Here are some of the key STP standards:

-> IEEE 802.1D: The original STP standard, defined by the Institute of Electrical and Electronics Engineers (IEEE), specifies the basic operation of STP. It defines how switches exchange Bridge Protocol Data Units (BPDUs) to establish a loop-free topology and elect a root bridge.

-> IEEE 802.1w (Rapid Spanning Tree Protocol - RSTP): RSTP is an enhancement of the original STP designed to reduce convergence times in larger networks. It achieves faster convergence by introducing new port roles (e.g., discarding, learning, and forwarding) and by using faster BPDU propagation mechanisms.

-> IEEE 802.1s (Multiple Spanning Tree Protocol - MSTP): MSTP extends the capabilities of STP to support multiple VLANs (Virtual Local Area Networks) within a single spanning tree instance. It allows network administrators to group VLANs into common spanning tree instances, reducing the overhead associated with maintaining separate spanning trees for each VLAN.

-> IEEE 802.1Q (VLAN Tagging): While not directly part of STP, IEEE 802.1Q is a standard for VLAN tagging, which is often used in conjunction with STP. VLAN tagging allows switches to identify and differentiate traffic belonging to different VLANs, enabling the creation of separate broadcast domains and improving network efficiency.

-> IEEE 802.1D-2004 (Rapid Reconfiguration of Spanning Tree Protocol - RSTP): This amendment to the original 802.1D standard introduced enhancements to STP, bringing it closer to the capabilities of RSTP. It includes features such as Proposal/Agreement mechanism and Port Roles that help reduce convergence times.

-> IEEE 802.1Q-2014 (Multiple Spanning Trees - MST): This amendment to the 802.1Q standard further extends the capabilities of MSTP, providing improvements in scalability and interoperability. It introduces the concept of Multiple Instances Spanning Tree Protocol (MISTP), allowing for finer control over the configuration of spanning tree instances.

Components of STP

- The Spanning Tree Protocol (STP) involves several components that work together to establish and maintain a loop-free topology in Ethernet networks:

-> Bridge: A bridge, also known as a switch, is a networking device that operates at the data link layer (Layer 2) of the OSI model. Bridges participate in STP by exchanging Bridge Protocol Data Units (BPDUs) with other bridges to determine the network topology and elect the root bridge.

-> Bridge ID (BID): Each bridge in the network is assigned a unique Bridge ID, which consists of a priority value and the bridge's MAC address. The priority value is a configurable parameter that determines a bridge's eligibility to become the root bridge.

-> Root Bridge: The root bridge is the central point of reference for the spanning tree topology. It is the bridge with the lowest Bridge ID in the network. All other bridges use the root bridge as the starting point for calculating the shortest paths to reach all other bridges in the network.

-> Ports: Ports on a bridge connect it to other devices in the network, such as switches or end devices. Each port is assigned a role in the spanning tree topology, which determines how it participates in the forwarding of data packets. The possible port roles include root port, designated port, and blocked port.

-> Bridge Protocol Data Units (BPDUs): BPDUs are messages exchanged between bridges to convey information about the spanning tree topology. BPDUs contain information such as the sender's Bridge ID, the root bridge's ID, and the cost of the path to reach the root bridge. BPDUs are used by bridges to elect the root bridge, determine port roles, and detect changes in the network topology.

-> Root Port: On each non-root bridge, the root port is the port that provides the shortest path to reach the root bridge. The root port is responsible for forwarding data packets towards the root bridge.

-> Designated Port: On each network segment, one port is designated as the designated port. The designated port is responsible for forwarding data packets away from the segment, towards the root bridge. All other ports on the segment are blocked to prevent loops.

-> Blocked Port: Ports that are not designated as root ports or designated ports are placed in a blocked state. Blocked ports do not forward data packets but continue to participate in the STP topology by receiving and transmitting BPDUs.

BPDU Messages

- Bridge Protocol Data Units (BPDUs) are messages exchanged between switches participating in the Spanning Tree Protocol (STP). These messages contain critical information about the spanning tree topology and help switches elect the root bridge, calculate paths to the root bridge, and determine the roles of ports in the network. Here's an overview of the information contained in BPDU messages:

-> Bridge ID: This identifies the sending bridge (switch). It consists of a priority value and the MAC address of the sending switch. The priority value is used to determine the root bridge, with lower values being preferred.

-> Root Bridge ID: This indicates the identity of the root bridge in the network. Initially, each switch advertises itself as the root bridge, but as BPDUs propagate through the network, switches converge on a common root bridge.

-> Path Cost: This is the cumulative cost of the path from the sending switch to the root bridge. It's calculated based on the cumulative costs of all links along the path. Each link has an associated cost, typically based on the link speed, with faster links having lower costs.

-> Port ID: This identifies the port through which the BPDU was sent. It's used to determine the role of the port on the receiving switch in the spanning tree topology.

-> Message Age: This indicates the time elapsed since the BPDU was originated by the root bridge. It helps switches determine the freshness of received BPDUs and detect changes in the network topology.

-> Max Age: This is the maximum age allowed for a BPDU before it's considered stale and discarded. If a switch doesn't receive BPDUs from the root bridge within the max age interval, it assumes that the root bridge is no longer reachable and starts the process of electing a new root bridge.

-> Hello Time: This is the interval between the transmission of BPDUs by the root bridge. BPDUs are sent out periodically to maintain the stability of the spanning tree topology.

-> Forward Delay: This is the time a port spends in the listening and learning states before transitioning to the forwarding state. It helps prevent transient loops during topology changes.

-> Flags and Flags Field: This field contains flags that indicate various properties of the BPDU, such as whether the sending switch is the root bridge or whether the port is designated or non-designated.

-> BPDU Message Timings:

i. BPDU Hello Messages: sent every 2 seconds

ii. Max Age (dead): 20 seconds

iii. Listening Time: 15 seconds

iv. Learning Time: 15 seconds

STP Operation (Using BPDU Messages)

- The operation of the Spanning Tree Protocol (STP) relies heavily on the exchange of Bridge Protocol Data Units (BPDUs) between switches. Here's an overview of how STP operates using BPDU messages:

-> Initialization: When switches are powered on or when network topology changes occur, STP goes through an initialization process. During this phase, each switch sends out BPDUs to announce its presence and gather information about neighboring switches.

-> Bridge ID Election: Each switch has a unique Bridge ID, composed of a priority value and the switch's MAC address. Initially, all switches consider themselves as potential root bridges and advertise their Bridge IDs in BPDUs. Switches compare received BPDUs to determine which switch has the lowest Bridge ID, and thus, should become the root bridge.

-> Root Bridge Election: The switch with the lowest Bridge ID becomes the root bridge. Once elected, the root bridge periodically sends out BPDUs with itself as the root bridge. All other switches forward these BPDUs to ensure consistent information throughout the network.

-> Path Cost Calculation: Switches calculate the cost of the path to reach the root bridge based on the cumulative cost of each link along the path. The path cost is included in BPDUs and helps switches determine the shortest path to reach the root bridge.

-> Port Roles Determination: Each switch determines the role of its ports in the spanning tree topology based on the information received in BPDUs. This includes identifying the root port (the port with the lowest path cost to the root bridge) and designated ports (ports on each segment with the lowest path cost to the root bridge).

-> BPDU Propagation: Switches continuously exchange BPDUs with neighboring switches to update information about the spanning tree topology. BPDUs contain information about the sender's Bridge ID, the root bridge's ID, the sender's port cost, and other relevant details.

-> Loop Detection and Blocking: If a switch detects a loop in the network, it takes action to block one or more ports to prevent the loop. This is achieved by placing redundant ports in a blocking state, effectively shutting down certain links to maintain a loop-free topology.

-> Topology Changes Detection: STP constantly monitors the network for changes in topology, such as link failures or additions. When a change is detected, switches update their spanning tree information accordingly and may adjust port roles or block/unblock ports as needed to maintain a loop-free topology.

STP Port States

- Spanning Tree Protocol (STP) defines several port states that switches can transition through as they participate in the spanning tree algorithm. These states determine the role of a port in the network topology and its ability to forward or block traffic. The main port states in STP are:

-> Blocking: In the blocking state, a port does not forward frames but listens to BPDUs to determine the topology of the network. This state prevents loops by disabling ports that would create them. All ports start in the blocking state when the switch is initially powered on.

-> Listening: After the blocking state, a port transitions to the listening state. In this state, the port prepares to participate in the spanning tree algorithm by actively listening for BPDUs. While in the listening state, the port still does not forward frames.

-> Learning: After the listening state, a port enters the learning state. In this state, the port continues to listen to BPDUs and also starts to learn MAC addresses by examining the source address of frames received on the port. However, the port still does not forward frames.

-> Forwarding: Once the port has completed the listening and learning states, it transitions to the forwarding state. In this state, the port is fully operational and forwards frames according to the

spanning tree topology. Frames are forwarded to their destination ports based on the MAC address table learned during the learning state.

-> Disabled: In some cases, a port may be administratively disabled by the network administrator. A disabled port does not participate in the spanning tree algorithm and does not forward frames.

STP Port Roles

- Spanning Tree Protocol (STP) assigns different roles to ports on switches to establish a loop-free topology and efficiently forward traffic within an Ethernet network. The main port roles in STP are:

-> Root Port (RP): Each non-root switch in the network selects one of its ports as the root port. The root port is the port that provides the shortest path (lowest cost) to the root bridge. Traffic from the switch to the root bridge is forwarded through the root port.

-> Designated Port (DP): On each network segment, one switch port is selected as the designated port. The designated port is responsible for forwarding traffic from that segment toward the root bridge. The switch that has the lowest path cost to the root bridge on a particular segment becomes the designated switch for that segment, and its port on that segment becomes the designated port.

-> Alternate Port (AP): An alternate port is a backup path to reach the root bridge in case the root port fails. An alternate port is always in a discarding state, meaning it does not forward traffic. If the root port fails, the alternate port can quickly transition to the forwarding state to maintain connectivity.

-> Backup Port (BP): Similar to an alternate port, a backup port is also a backup path to reach the root bridge. However, backup ports are designated ports that are in a blocking state (rather than a discarding state like alternate ports). Backup ports become designated ports if the current designated port fails.

-> Disabled Port: A port that has been administratively disabled by the network administrator. Disabled ports do not participate in the spanning tree algorithm and do not forward traffic.

LAB WORK:

- Create a new project.

- Add 3 Switches (L2 Switch - Cisco Switches).

- Connect them using cables on any port you want.

- Now, start the switches and open the console of the 1st L2 Switch.

- Now run the command "show spanning-tree", this will show us the details of the STP of the network.

- For switch 2, do the same thing. Now we can see the root bridge from the details printed after running that command.

- Do the same thing for switch 3 too.

LAB WORK QUESTION:

Connect 4 manages switches in a full mesh topology and find out:

1. Which one is the root bridge?

2. The root port of each of the non-root switches.

3. The Blocking Port(s).

Instructions -

- Do the same as before.

- Add 4 L2 Switches.

- Start them.

- Run the command "show spanning-tree" to get the details of the ports and all.

Lab 6:

Notes of 15th of March, 2024

What is FQDN?

- FQDN stands for Fully Qualified Domain Name. It represents the complete domain name for a specific host on the internet. An FQDN consists of two parts: the hostname and the domain name.

For example, consider the FQDN "www.example.com":

"www" is the hostname, representing a specific host or service.

"example.com" is the domain name, indicating the domain to which the host belongs.

Together, "www.example.com" forms the FQDN, providing the complete and specific address for accessing a particular host on the internet.

- FQDNs are used extensively in networking, server administration, and web browsing to uniquely identify resources on the internet and facilitate communication between devices and services.

- They are particularly useful in environments where multiple hosts may share the same domain name but have different hostnames or subdomains.

DNS (Domain Name System)

- The Domain Name System (DNS) is like the internet's phonebook. It's a decentralized system that translates human-readable domain names (like example.com) into numerical IP addresses that computers use to identify each other on the network.

- When you type a website address into your browser, your computer contacts a DNS server to find out the IP address associated with that domain name, allowing you to connect to the desired website. DNS is crucial for the functioning of the internet as we know it, enabling users to access websites using easily memorable names rather than complicated numerical addresses.

Why use DNS Server?

- Using a DNS server offers several benefits:

-> Human Readability: DNS servers allow us to use human-readable domain names (like google.com) instead of remembering complex IP addresses (like 172.217.4.206).

-> Centralized Management: DNS servers provide a centralized system for managing domain name mappings, making it easier to update and maintain records for large numbers of domain names.

-> Redundancy and Load Balancing: DNS servers can be set up in a redundant and load-balanced manner, ensuring that even if one server goes down, there are others available to handle requests. This improves reliability and availability.

-> Performance Optimization: DNS servers can be strategically placed around the world to reduce latency and improve the speed of DNS resolution for users accessing websites from different locations.

-> Security: DNS servers can implement security measures such as DNSSEC (Domain Name System Security Extensions) to help protect against DNS spoofing, cache poisoning, and other types of attacks.

-> Content Filtering and Blocking: DNS servers can be configured to filter or block access to certain websites or types of content, providing additional control over network traffic.

[=> To see the stored IP Addresses, we can write the command "cat /etc/hosts". This is a local file especially made by the OS and not shared to anyone. Apparently it shows the same IP Addresses, i.e., "192.168.1.10 foo.mydomain.org foo" and "192.168.1.13 bar.mydomain.org bar"]

=> DNS is designed as a hierarchical, distributed database which can resolve names to IP addresses.

=> DNS Server is based on the UDP Protocol and it's port number is 53.

Parts of FQDN

- A Fully Qualified Domain Name (FQDN) consists of several parts:

-> Host Name: This is the name of the specific host or server within the domain. For example, in the FQDN "www.example.com," "www" is the hostname.

-> Subdomain (Optional): A subdomain is a hierarchical part of the domain name that precedes the domain name itself. It can represent a specific branch or subdivision within the domain. In the FQDN "www.example.com," there's no subdomain, but in "blog.example.com," "blog" would be the subdomain.

-> Domain Name: This is the primary name of the domain under which the host is registered. In "www.example.com," "example" is the domain name.

-> Top-Level Domain (TLD): This is the highest level of the domain hierarchy and often indicates the type or purpose of the domain. Common TLDs include ".com," ".org," ".net," ".edu," etc. In "www.example.com," ".com" is the TLD.

Looking up IP Address for a given domain

- Using the "nslookup" command. (does a forward lookup)

-> Type the "nslookup" command and then press enter. Then input the FQDN to get it's IP Address.

- Using the "dig" command. (does a forward lookup)

-> Type the "dig" command in the given format "dig <Address or link of the site>"

- We can also get the name of the site from the IP Address by using the "reverse dig" command. (does a reverse lookup)

-> Type the "reverse dig" command in the given format "dig -x <IP Address>". But do note that it might show some wrong info as many famous sites might change their IP due to loadbalancers. So, it might not be correct all the time.

Can we access a website by using its IP Address?

- The answer is not quite constant. It varies as the security infrastructure has increased these days. So, sometime, the webpage might appear, and sometime it might not because some sites have security features which can disable the access when accessed using their IP Addresses.

Will the URL work if we append a dot character at the end?

- Yes, it will work because the protocol by default adds a dot character at the end. So, manually adding this does not give us any error.

Root Servers

- Root servers are a crucial part of the Domain Name System (DNS) infrastructure. They are a set of authoritative DNS servers that are responsible for the initial step in the DNS lookup process. There are 13 sets of root server clusters worldwide, labeled A through M, operated by various organizations.

- These root servers store the DNS root zone file, which contains the authoritative list of top-level domain (TLD) nameservers and their IP addresses. When a DNS resolver receives a query for a domain name (e.g., www.example.com) that it doesn't already have in its cache, it starts the resolution process by querying one of the root servers.

- However, the root servers don't have information about every domain on the internet. Instead, they provide referrals to TLD servers responsible for specific top-level domains like .com, .org, .net, and country-code TLDs like .uk, .fr, etc. Once the resolver receives the referral from the root servers, it then queries the appropriate TLD server, which, in turn, directs it to the authoritative nameserver for the domain in question.

=> We can make DNS Queries using the command "dig +trace <URL>". This will give us an idea of how the website names are associated to their IP Addresses.

=> The website name - IP Addresses are stored in 13 Servers in the world called as "Root Servers".

=> Tracing the DNS Query, we got to know that first, we check for the TLD of the FQDN in the Root Servers. After we find the TLD, we then look for the Domain Name in the servers. After we get the domain name, we check for the host name and shows us the result, i.e., IP Address of the URL.

=> So, all the data is scattered or distributed across various servers around the world

Common Terms associated with DNS

- Name Servers - Name servers, also known as DNS servers, are essential components of the Domain Name System (DNS) infrastructure. They are specialized servers responsible for translating domain names into their corresponding IP addresses and vice versa. Name servers store DNS records that map domain names to IP addresses and provide this information to DNS resolvers upon request.

- Authoritative Name Servers: These servers hold the original and official DNS records for specific domains. They are responsible for providing answers to DNS queries regarding those domains. Authoritative name servers can be primary (master) or secondary (slave), depending on their role in the DNS zone.

- Caching Name Servers: Caching name servers, also known as recursive name servers or caching resolvers, are a type of DNS server that plays a crucial role in the DNS resolution process. Their primary function is to perform DNS resolution on behalf of client devices and cache the results to improve performance and reduce the load on authoritative name servers.

- Recursive Query: A recursive query is a type of DNS query in which a DNS resolver, typically a caching name server or a client device, requests DNS resolution for a domain name and expects a complete answer from the DNS server.

- DNS Resolvers: DNS resolvers, also known as DNS clients or DNS resolv.conf, are components of the Domain Name System (DNS) infrastructure responsible for resolving domain names to their corresponding IP addresses. They are typically implemented in software libraries or as part of the operating system's networking stack.

- Reverse DNS Lookup: Reverse DNS lookup, also known as reverse DNS resolution or reverse DNS mapping, is the process of retrieving a domain name (usually a hostname) associated with a given IP address. While traditional DNS resolution translates domain names into IP addresses, reverse DNS lookup performs the opposite operation by translating IP addresses into domain names.

Common types of records stored in a DNS server

- DNS servers store various types of resource records (RRs), each serving a specific purpose in the Domain Name System (DNS). Some common types of DNS records include:

-> A (Address) Record: Associates a domain name with an IPv4 address. For example, "example.com" maps to "192.0.2.1".

-> AAAA (IPv6 Address) Record: Similar to an A record but maps a domain name to an IPv6 address.

-> CNAME (Canonical Name) Record: Maps an alias or subdomain to another domain's canonical name (the true or canonical name). For example, "www.example.com" can be a CNAME for "example.com".

-> MX (Mail Exchange) Record: Specifies the mail servers responsible for receiving email messages for a domain. It points to the domain name of the email server. For example, "mail.example.com".

-> TXT (Text) Record: Stores arbitrary text data associated with a domain. It can be used for various purposes, such as SPF (Sender Policy Framework) records for email authentication or domain verification.

-> NS (Name Server) Record: Specifies the authoritative name servers for a domain. It indicates which DNS servers are responsible for handling DNS queries for the domain.

-> PTR (Pointer) Record: Used for reverse DNS lookup, mapping an IP address to a domain name. It's the opposite of an A record.

-> SOA (Start of Authority) Record: Contains authoritative information about a DNS zone, including the primary name server, email address of the domain administrator, serial number, and other zone-related parameters.

-> SRV (Service) Record: Specifies the location (hostname and port number) of servers for specific services, such as SIP (Session Initiation Protocol) or LDAP (Lightweight Directory Access Protocol).

-> DNSKEY (DNS Key) Record: Contains cryptographic public keys used in DNSSEC (Domain Name System Security Extensions) for DNS security and authentication.

=> To get the Name Servers of a domain, use the command "dig -t ns <URL>".

=> To get the Mail Exchange of a domain, use the command "dig -t mx <URL>".

=> To get the Start of Authority of a domain, use the command "dig -t soa <URL>".

=> If we want to find the IP of an URL using different DNS Server, use the command "dig <URL> @<DNS Server>".

Lab 7:

Notes of 5th of April, 2024.

INSTALLING AND SETTING UP A DNS SERVER

- Setting up a DNS (Domain Name System) server can be a crucial step in managing your network infrastructure, enabling the translation of domain names into IP addresses. Here's a basic guide to installing and setting up a DNS server on a Linux system using BIND (Berkeley Internet Name Domain), one of the most commonly used DNS servers:

-> Install BIND: First, you need to install the BIND package on your Linux system. The installation process can vary depending on your Linux distribution. For example, on Ubuntu, you can use the following command:

sudo apt-get update

sudo apt-get install bind9

[in GNS3 it is "sudo dnf install bind"]

[-> After that, you can write the command "hostname" will show who is the host of the node]

[-> Then, we need the hostname to be a FQDN. So, to do this, we need to change a config file using the commands:

"vim /etc/hostname"

Then enter the new hostname as "cse.jecassam.ac.in" and save the file. After that, we will need to restart the PC for the changes to apply. So, restart the device.

If you cannot restart the server right away, use the command "hostname cse.jecassam.ac.in" which will set the new hostname for that session only.

Then, install some other packages using the command:

"sudo dnf install bind bind-utils"

After that, look for the PCs IP Address using the command:

"ifconfig"

Then, add the IP and the present Hostname in the file using the command:

"vim /etc/hosts"

Now, you can ping the Hostname in the terminal using the command:

"ping cse.jecassam.ac.in"

So, the hostname has now been assigned an IP Address]

-> Configuration Files: After installing BIND, you'll need to configure its main configuration file named named.conf, zone files, and other settings. Here's a basic overview:

= named.conf: This file contains global configuration options for BIND. The location of this file can vary depending on your Linux distribution. Common locations include /etc/named.conf or /etc/bind/named.conf.

= Zone Files: These files define the DNS zones hosted by your server, including forward and reverse lookup zones. Each zone file typically contains records mapping domain names to IP addresses (forward lookup) or IP addresses to domain names (reverse lookup).

-> Edit named.conf: Open the named.conf file in a text editor, using the command: "vim /etc/named.conf", and configure it according to your requirements. Here's a minimal example of what it might look like:

options {

directory "/var/named";

recursion no; [set the recursion to no because otherwise, it can be used to set a DDoS attack]

};

zone "." {

type hint;

file "named.ca";

};

include "/etc/named.rfc1912.zones";

[You can look up the "/etc/named/named.ca" file and see that it has the IP Addresses of those central DNS Servers]

-> Create Zone Files: Create zone files for each domain you want to host. These files typically reside in the same directory specified in the named.conf file, using the command "vim /etc/named.rfc1912.zones". Here's a basic example of a zone file for a domain named example.com:

zone "jecassam.ac.in" IN {

type primary;

file "jecassam.forward.zone";

allow-update { none; }'

};

//jecassam.ac.in in reverse zone

zone "<Remove the last bit from the IP Address and then write the IP Address in reverse>" IN {

type "primary";

file "jecassam.reverse.zone";

allow-update { none; };

};

Then, write the command "named-checkconf". This will give us an error if we made an error while changing the zone file. No output means no error.

Now, write the commands :

"cd/var/named"

"cp named.localhost jecassam.forward.zone"

"vim jecassa.forward.zone" and then change the contents to this

NOTE: EITHER USE ONLY TABS OR SPACES. DO NOT MIX THEM

```
$TTL 8h
@   IN  SOA ns1.jecassam.ac.in. root.jecassam.ac.in. (
            2024040501  ; Serial
            1D      ; Refresh
            1H      ; Retry
            1W      ; Expire
            3H      ; Minimum TTL
        )
    IN  NS  ns1.jecassa.ac.in.
    IN  MX 10 mx1.jecassam.ac.in.  //This is the Mail Server

    cse IN  A  10.0.2.15  //we can also write cse.jecassam.ac.in. But still writing CSE will automatically do it.
    ns1 IN  A  10.0.2.53 //A means that it is an Address.
    max1 IN  A  10.0.2.15
    www IN  CNAME  cse.jecassam.ac.in.  //this sets a www domain on this
```

Now the commands:

"cp jecassam.forward.zone jecassam.reverse.zone"

"vim jecassam.reverse.zone" and then change the contents to this

NOTE: EITHER USE ONLY TABS OR SPACES. DO NOT MIX THEM

```
$TTL 8h
@   IN  SOA ns1.jecassam.ac.in. root.jecassam.ac.in. (
            2024040501  ; Serial
            1D      ; Refresh
            1H      ; Retry
            1W      ; Expire
            3H      ; Minimum TTL
```

)

IN NS ns1.jecassa.ac.in.

15 IN PTR cse //PTR means that is a Pointer

53 IN PTR ns1

15 IN PTR mx1

[I guess, the reverse zone links the IP address back to the Domain Names. Not sure though.]

Now, check if we have written correctly or not using the command:

"named-checkzone jecassam.ac.in jecassam.forward.zone"

Do the same for the reverse zone file

-> Start BIND Service: After configuring BIND and creating zone files, start the BIND service using the following command:

sudo systemctl start named

You can also enable BIND to start automatically on system boot using:

sudo systemctl enable named

-> After that, run this command:

"chown root:named jecassam.*"

-> Testing: Test your DNS server configuration by querying it for domain records using tools like dig or nslookup. For example:

dig cse.jecassam.ac.in in @127.0.0.1

This should return the IP address associated with the domain example.com.

-> Firewall Configuration: Ensure that your firewall allows DNS traffic (UDP port 53) to reach your DNS server.

Lab 8:

NOTES OF 26TH OF APRIL, 2024

SOCKET PROGRAMMING

Books for Socket Programming - Beginning Linux Programming by Richard Stones and Neil Matthew

- Socket Porgramming is done in C because it allows us to clear out our concepts and make a clear idea of what is actually happening.

What is Socket Porgramming?

- Socket programming is a way of enabling communication between processes on different computers or within the same computer. It's a fundamental concept in networking and is widely used in various applications like web servers, email clients, chat applications, and more.

Here's a brief overview of how it works:

= Socket: A socket is an endpoint for communication between two machines. It's like a door through which information can be sent and received.

= Client-Server Model: In socket programming, communication typically happens between a client and a server. The server waits for incoming connections from clients, and the client initiates communication by connecting to the server.

= Types of Sockets: There are two main types of sockets:

+ TCP (Transmission Control Protocol) sockets and UDP (User Datagram Protocol) sockets. TCP provides reliable, connection-oriented communication, while UDP provides connectionless, unreliable communication.

= Steps in Socket Programming:

a. Creating a Socket: Both the client and server need to create sockets to establish communication.

b. Binding: The server binds to a specific address and port, while the client typically doesn't bind to any specific address or port.

c. Listening (Server): The server socket listens for incoming connections.

d. Connecting (Client): The client socket connects to the server socket.

e. Accepting (Server): Once a client connects, the server accepts the connection and creates a new socket for communication with that client.

f. Sending and Receiving Data: Both the client and server can send and receive data through their respective sockets.

g. Closing the Connection: After communication is complete, both the client and server close their sockets.

= Socket APIs: Most programming languages provide socket APIs that allow developers to work with sockets easily. For example, in Python, you can use the socket module to create and manage sockets.

NOTE - TCP is called a Stream Socket and UDP is called the Data Socket (or DGRAM).

Sockets vs IPC (Inter-Process Communication)

[IPC is a type of Sockets, kinda]

- Sockets and Inter-Process Communication (IPC) are both mechanisms used for communication between processes, but they serve different purposes and operate at different levels of abstraction.

= Socket:

> Purpose: Sockets are primarily used for communication between processes over a network. They enable communication between processes running on different machines or on the same machine but through a network protocol stack.

> Abstraction Level: Sockets operate at a higher level of abstraction compared to IPC. They abstract away the complexities of network communication, providing a convenient interface for sending and receiving data over a network.

> Typical Use Cases: Sockets are commonly used in client-server architectures for building networked applications such as web servers, chat applications, and distributed systems.

= Inter-Process Communication (IPC):

> Purpose: IPC is used for communication between processes running on the same machine. It allows processes to exchange data and synchronize their actions.

> Abstraction Level: IPC operates at a lower level of abstraction compared to sockets. It provides mechanisms for processes to directly share memory, pass messages, or synchronize using semaphores or other synchronization primitives.

> Typical Use Cases: IPC is commonly used in multi-threaded applications, microservices architectures, and operating systems for communication between different components of a system.

Necessary things required for a port connection

- To establish a socket connection, you typically need the following:

= IP Address and Port Number: For a client to connect to a server, it needs to know the IP address and port number of the server it wants to communicate with. The server also needs to bind to a specific IP address and port to listen for incoming connections.

= Socket Creation: Both the client and server need to create sockets. In most programming languages, you can use socket APIs to create sockets. For example, in Python, you can use the socket module.

= Binding (for Server): If you're building a server, it needs to bind to a specific IP address and port number so that clients can connect to it. This step associates the socket with a specific network interface and port.

= Listening (for Server): After binding, the server socket needs to listen for incoming connections. It enters a listening state, waiting for clients to connect.

= Connection (for Client): The client socket needs to initiate a connection to the server by specifying the server's IP address and port number.

= Accepting (for Server): When a client tries to connect, the server socket accepts the connection, creating a new socket specifically for communication with that client.

= Sending and Receiving Data: Once the connection is established, both the client and server can send and receive data through their respective sockets. This involves using methods provided by the socket API to send data from one end and receive it at the other end.

= Closing the Connection: After communication is complete, both the client and server should close their sockets to release resources and properly terminate the connection.

Port Numbers

- In networking, ports are virtual endpoints used to identify specific processes or services running on a host device. They facilitate communication by allowing multiple services or processes to operate concurrently on the same device. Ports are categorized into different types based on their functionality and usage:

= Well-Known Ports (0-1023):

These ports are also known as system ports or privileged ports.

Reserved for specific, well-known services. For example, HTTP typically uses port 80, HTTPS uses port 443, FTP uses port 21, SSH uses port 22, etc.

Access to well-known ports is restricted on most systems, typically requiring administrative privileges to bind services to these ports.

= Registered Ports (1024-49151):

Registered ports are used by applications or services that have been registered with the Internet Assigned Numbers Authority (IANA).

They are typically used by applications and services that are not widely deployed but still need a reserved port number for their operation.

Examples include various database systems, web servers with non-standard configurations, and proprietary services.

= Dynamic or Private Ports (49152-65535):

Also known as ephemeral ports or private ports.

These ports are available for dynamic allocation by client applications and are not reserved for any specific service.

When a client initiates communication with a server, it typically uses a dynamic port as its source port. The server responds to the client's dynamic port.

After the communication is completed, the dynamic port is released and can be reused by other applications.

- Well-Known Ports vs. Ephemeral Ports:

Well-known ports are used by servers to offer specific services, while ephemeral ports are used by clients to initiate communication.

Well-known ports are static and predefined, whereas ephemeral ports are dynamically allocated by the operating system to client applications.

Well-known ports have fixed numbers (0-1023), while ephemeral ports have a much larger range (49152-65535).

Unix Domain Sockets

- Unix domain sockets, also known as IPC (Inter-Process Communication) sockets or local sockets, are a type of inter-process communication mechanism used for communication between processes on the same Unix-like operating system.

Unlike traditional network sockets that communicate over a network, Unix domain sockets communicate entirely within the operating system's kernel.

Here are some key points about Unix domain sockets:

= Communication Within a Single Host: Unix domain sockets facilitate communication between processes running on the same host. They are ideal for scenarios where processes need to exchange data locally without the overhead of network communication.

= File System-Based: Unix domain sockets are implemented using special file types within the file system. They are represented as special files that processes can use to establish communication channels.

= High Performance: Unix domain sockets offer high-performance communication because data is transferred directly between processes in the kernel space, without the overhead associated with network communication. This makes them suitable for scenarios where low latency and high throughput are important.

= Synchronous and Reliable: Communication over Unix domain sockets is synchronous and reliable. Data sent by one process is guaranteed to be received in the same order by the receiving process, and the sender is blocked until the receiver acknowledges receipt.

= Socket Addressing: Like network sockets, Unix domain sockets are identified by an address. The address is typically a file path within the file system, though abstract socket addresses are also supported on some systems.

= Socket API: Unix domain sockets are accessed using the same socket API as network sockets. This means that processes can use familiar socket functions like socket(), bind(), listen(), connect(), send(), and recv() to establish and manage Unix domain socket connections.

= Security and Permissions: Since Unix domain sockets operate entirely within the host system, they inherit the permissions and security model of the underlying file system. Access to Unix domain sockets can be controlled using file system permissions, making them suitable for securing local communication channels.

Netstat in Socket Programming

- In socket programming, netstat is a command-line tool used for displaying network-related information such as active network connections, routing tables, interface statistics, masquerade connections, and multicast memberships.

It is available on most Unix-like operating systems, including Linux, macOS, and BSD variants. Netstat can be particularly useful when developing or debugging socket-based applications as it provides insight into the current state of network connections on the system.

Here are some common uses of netstat in the context of socket programming:

= Displaying Active Network Connections:

netstat -a

This command displays all active network connections, both listening and established, along with the associated protocol, local and foreign addresses, and state.

= Displaying Listening Sockets:

netstat -l

This command lists all sockets that are listening for incoming connections. It shows the protocol, local address, and port number for each listening socket.

= Displaying PID and Program Name:

netstat -p

This command displays the PID (Process ID) and program name associated with each network connection or listening socket. This information can be helpful in identifying which processes are responsible for specific network activity.

= Displaying Network Statistics:

netstat -s

This command provides various network statistics such as TCP, UDP, ICMP, and IP traffic counts, errors, and packet statistics.

= Filtering Output:You can combine netstat with other commands or tools to filter and process its output further. For example, you can use grep to search for specific patterns in the output or awk to extract specific fields.

netstat -a | grep "ESTABLISHED"

This command displays only the established TCP connections.

TCP Sockets

- Server Side

= socket(): create a socket (an OS resource file or file descriptor)

= bind(): name the socket (local sockets have a filename. Network sockets have a port number)

= listen(): create a queue for incoming requests (client connects)

= accept(): a new socket is created (for the specific connecting client)

= read()/write(): communicate using socket

- Client Side

= socket(): create a socket (an OS resource file or file descriptor)

= connect(): establish a connection with the server

= read()/write(): communicate using socket

UDP Sockets

- Server Side

= socket(): create a socket (an OS resource file or file descriptor)

= bind(): name the socket (local sockets have a filename. Network sockets have a port number)

= recvirom()/sendto(): communicate with the socket

- Client Side

= socket(): create a socket (an OS resource file or file descriptor)

= recvirom()/sendto(): communicate with the socket