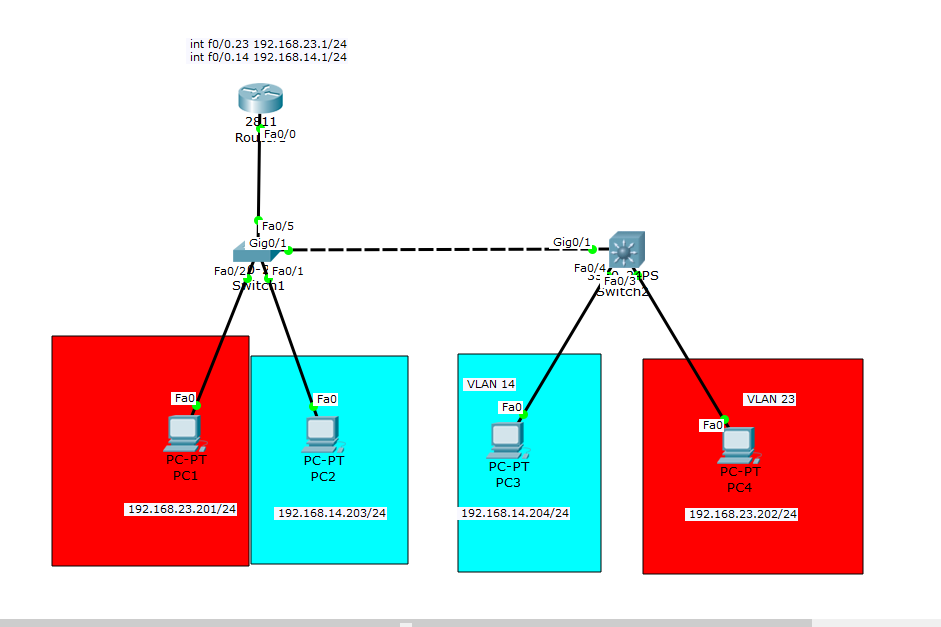
1. VLANs, VTP, and IEEE 802.1q (NOTE: Make sure you accomplish this ***without*** going into VLAN Database mode and show the syntax for both 2960 and 3560 switches!)

**Description:**

This question, I configured Vlans, VTP, and IEEE 802.1q on a topology including 2 switches(both a 2960 and 3560 switch), 4 PC’s and a router. First, I assigned each PC and interface with an address and the router with a sub interface. Configuring two separate Vlans splits our 4 PC’s into two separate subnetworks. This allows each PC to only be able to communicate with the other PC in their respective Vlans. VTP and 802.1q will both be configured on the 3560 switch to have it serve as our VTP client. The 2960 switch will only need a VTP configuration because 802.1q is already installed on the device.

**Topology:**



**Commands:**

|  |  |
| --- | --- |
| **Device commands** | **Description** |
| Router>enable | Sets user into privilege mode |
| Router#Configure Terminal | Sets user into configure mode |
| Router(config)#Hostname “name” | Sets the name of the router |
| Router(config)#enable secret cisco | Sets a password for when the user signs into privilege mode |
| Router(config)#line con 0 | Sets user into the configuration mode of the lines |
| Router(config-line)#exec-timeout 0 0 | Sets the unlimited time for the timeout |
| Router(config-line)#password cisco | Sets the password at the base line |
| Router(config-line)#logging-synchronous | Prevents the logging output from interrupting your console session |
| Router(config-line)#login | Enables a login prompt at base line |
| Router(config)#line vty 0 4 | Sets the user in configuration mode for the lines of 0-4 in TelNET |
| Router(config-line)#exec-timeout 60 0 | Sets timeout to 60 minutes |
| Router(config)#int f0/0.1 | Sets the user into the subinterface of 0.1 |
| Router(config-if)#[ip address] [subnet mask] | Sets the ip address and subnet mask of the interface |
| Router(config-if)#no[ip address][subnetmask] | Deletes the inputted ip address/subnet mask from the interface |
| Router(config-if)#no shutdown | Turns the interface on |
| **VLAN Commands** | **Description** |
| Switch(config-if)#switchport mode access | Enables the device to carry messages into the VLAN |
| Switch(config-if)#switchport access VLAN # | Sets the interface to a VLAN |
| Switch(config)#int VLAN # | Sets user in a specified VLAN |
| Switch(config-if)#ip address [ip address] [subnet mask] | Sets an ip address to the VLAN |
| Switch(config-if)#no shutdown | Enables the VLAN |
| Switch(config)#VLAN # | Sets user in a VLAN group |
| Switch(config-VLAN)#name “” | Sets a name for the VLAN |
| **IEEE 802.1q** | **Description** |
| Switch(config-if)#switchport trunk encapsulation dot1q | Sets trunking encapsulation to dot1q standards |
| Switch(config-if)#Switchport mode trunk | Enables the current interface to trunk |
| **VTP commands** | **Descriptions** |
| Switch(config)#vtp mode [server/client] | Sets the vtp mode to server or client |
| Switch(config)# vtp domain [INETLAB] | Sets the vtp domain |
| Switch(config)#vtp password cisco | Sets the vtp password to cisco |

|  |  |
| --- | --- |
| **Show Commands** | **Description** |
| Switch#show ip int brief | Shows the interfaces enables and information about each interface like ip address |
| Switch#show vlan | Shows a table of vlans and information about them |
| Switch#show run | Shows running configuration |
| PC-Ping [ip address] | Pings selected ip address |

Each show command in this list was helpful for configuring this topology. Each gave me insight on the devices and I was able to see what was configured already and if anything was configured wrong. To test the topology I would ping from PC to PC and if the PC could only reach the PC in the same vlan then it was working.

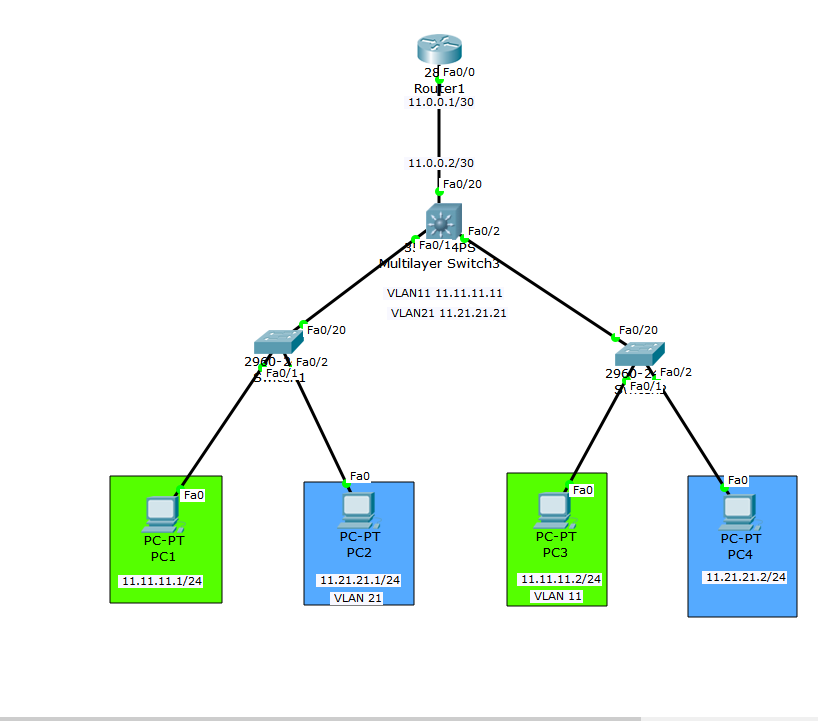
**Debug:**

For this topology, the show run and show vlan commands were useful to debug. Show run gave me an insight on the configuration of my switches and router. I was able to check if the correct interfaces were configured with the right information. Show vlan gave me insight into each vlan to make sure they were configured correctly with the numbers, names and interfaces.

1. InterVLAN Routing with a Router-on-a-stick and SVIs

**Description:**

In this question, we are asked to configure interVLAN routing with a “router on a stick” and SVIs. In this topology the interVLANs will serve as sub networks that will split the 4 PCs into VLAN21 and VLAN11. InterVLAN will allow the different VLANs to communicate with each other. The SVI was configured on a 3560 switch which will enable a switch to route connections. The 3560 is needed for its multilayer standards.

**Topology:**

**Commands:**

|  |  |
| --- | --- |
| **SVI Commands** | **Descriptions** |
| Switch(config)#ip routing | Sets ip routing on the multilayer |
| Switch(config)#int vlan # | Sets user in the specified vlan #’s interface |
| Switch(config-if)#ip address[ip address] [subnet mask] | Sets Vlan ip address |
| Switch(config)#int f0/20 | Sets user in f0/20’s interface |
| Switch(config-if)#ip address [ip address] [subnet mask] | Sets ip address for the interface f0/20 |
| Switch(config-if)#no switchport | Turns off switchport |
| **Show commands** | **Descriptions** |
| Switch#show vlan | Shows a table of the VLANS and information about each |
| Switch#show int vlan # | Shows current configurations on a vlan # |
| Switch#show run | Shows the configuration of a device |
| PC-Ping [ip address] | Pings the selected ip address |

The above show commands helped with the configuration and completion of this topology. Being able to see each interface and information in that interface helped when I was trying to ping from different VLANs. Each PC should now be able to ping and receive a reply from the host no matter which VLAN it was on.

**Debug:**

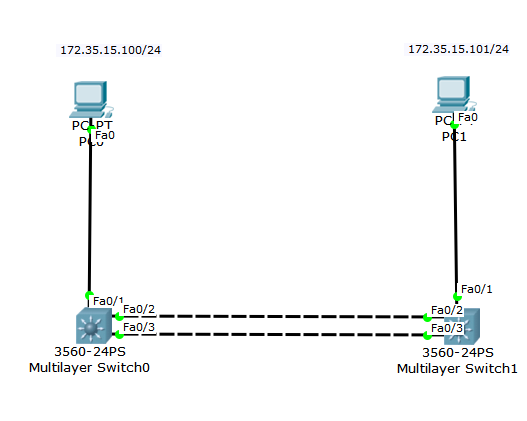
To debug this configuration I used the ping command and the show vlan commands. When trying to make sure the PC’s could communicate I would ping back and forth and if there was a problem with my VLANs I could use show vlan to see and verify that the configuration of each vlan was correct

1. First Hop Redundancy Protocol with HSRP using Link Aggregation between two 3560 multilayer switches. Demonstrate your knowledge of LACP and PAgP.

**Description:**

In this question, we are asked to use HSRP and the channel protocols PAgP and LACP while communicating through two 3560 multilayer switches. The addition of two PC’s will enable communication over the switches to each other.

**Topology:**



**Commands:**

|  |  |
| --- | --- |
| **HSRP Commands** | **Description** |
| Switch(config)#ip routing | Enables routing on a switch |
| Switch(config)#int vlan # | Sets user in the interface of the vlan |
| Switch(config-if)#standby [vlan #] ip [ip address] | Sets the standby ip address |
| Switch(config-if)#standby [vlan #] priority [105/115] | Sets the standby priority by number listed |
| Switch(config-if)#standby [vlan #] preempt | Enables HSRP with highest priority to become the active router |
| **PAGP commands** | **description** |
| Switch(config-if)#channel-protocol pagp | Declares pagp as the protocol |
| Switch(config-if)#channel-group # mode desirable | Assigns an interface to an etherchannel and declares the protocol |
| Switch(config)#int port-channel# | Sets user into the selected port channel, makes one if one does not already exist |
| Switch(config-if)#switchport trunk encapsulation dot1q | Assigns 802.1q to portchannel |
| Switch(config-if)#Switchport mode trunk | Sets the etherchannel to trunking mode |
| Switch(config-if)#switchport native vlan # | Sets the native vlan to the selected # |
| **LACP commands** | **Description** |
| Switch(config-if)#channel-protocol lacp | Declares lacp as the protocol |
| Switch(config-if)#channel-group # mode active | Assigns an interface to an etherchannel and declares what protocol to use |
| Switch(config-if)#spanning-tree vlan # root [primary/secondary] | Enables spanning tree protocol to be primary or secondary |
| **Show commands** | **Descriptions** |
| Switch#show int trunk | Shows a table of the interfaces/portchannels that are trunking |
| Switch#show standby | Shows the standby for HSRP |
| Switch#show etherchannel sum | Shows a brief summary of all etherchannels |

In this topology the goal is to be able to ping from PC to PC. When using LACP or PAgP in the topology they both were able to work. The above show commands were useful in making sure the configurations of either LACP or PAgP were correct.

**Debug:**

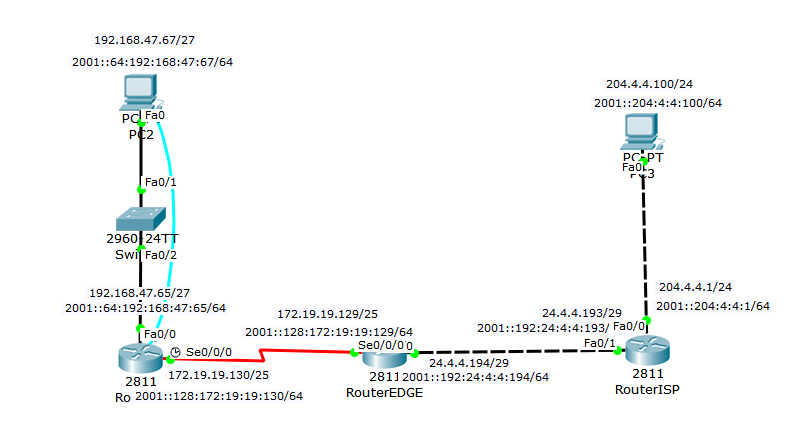
Ping was used to make sure that connectivity through the switches was working. Show run helped me look into the whole configuration of the switches when I was changing back and forth from LACP and PAgP. It made the process much smoother because I could see the whole configuration.

1. IPv4 and IPv6 Static routing (NOTE: With AND Without the use of a default static route.)

**Description:**

This question was the configuration of static routing for both ipv4 and ipv6. Static routing is manually putting in the ip address to specific routers to configure connections. This will enable us to have connectivity over multiple networks. I used three routers, one switch and two PC’s. The static routing will be from my router1 to my edge router and the default static route is used on my edge router to my isp router and back. This will enable the two networks to have connectivity and complete the ultimate goal of having the two PC’s be able to ping each other.

**Topology:**

**Commands:**

|  |  |
| --- | --- |
| **Ip routing commands(ipv4/ipv6)** | **Description** |
| Router(config)#ip route [dest N.A.] [subnet mask] [next hop] | Statically routes an Ipv4 address |
| Router(config)#ip route 0.0.0.0 0.0.0.0 24.4.4.193 | Sets a default static route out of a router |
| Router(config)#ipv6 unicast-routing | Enables ipv6 addressing on the router |
| Router(config)#ipv6 route [dest. Address/subnetmask] [Outgoing int] | Statically routes an IPv6 address |
| Router(config)#ipv6 route ::/0 [outgoing ip address] | Sets a default static route for ipv6 |
| **Show commands** | **Description** |
| Router#show ip route | Shows table of routes on the router |
| Router#show ipv6 route | Shows table of ipv6 routes on the router |

Using these show commands I was able to verify where my routes were being routed to and seeing the connections being made. When looking at the routing tables it was easy to differentiate the default static routes and static routes. Default static routes are identified as S\* and static routes are identified as S. At the end of the topology I was able to get full connectivity from the two PCs.

**Debug:**

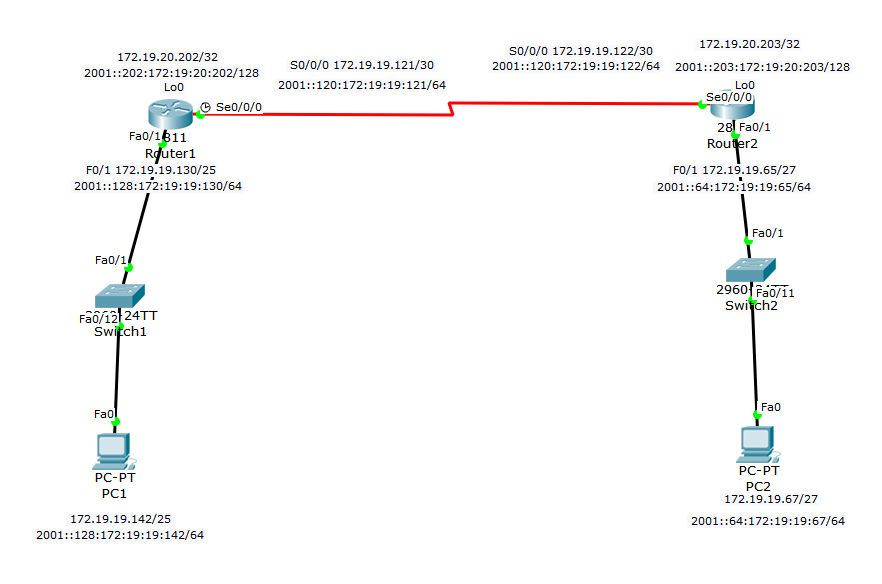
During this question, the two show commands along with ping and tracert commands helped me the most. Being able to see the full routing table for both ipv6 and ipv4 made it easy for me to see what was connected and what wasn’t in each router. The pings would verify the connection between routers and if the connection was lost I could use a tracert to see where the connection would be lost.

1. EIGRP and EIGRP for IPv6 (NOTE: Make sure you utilize VLSM in your topology and configuration without the use of automatic summarization.)

**Descriptions:**

In this question we start EIGRP and dynamic routing. With EIGRP I used 2 routers, PCs and switches. With EIGRP it is different than static in the sense of the networks are advertised to each other instead of manually put in. The routers communicate with each other and connect the networks attached. The ultimate goal will be to get each PC to connect to one another.

**Topology:**

**Commands:**

|  |  |
| --- | --- |
| **EIGRP Commands** | **Description** |
| Router(config)#router eigrp # | Sets eigrp routing |
| Router(config-router)#router-id [ip address] | Sets router id |
| Router(config-router)#network [ip address] [wildcard mask] | Sets the network for EIGRP |
| Router(config-router)#no auto-summary | Turns off auto summarization |
| Router(config)#ipv6 unicast-routing | Enables Ipv6 addressing and routing |
| Router(config)#ipv6 router eigrp # | Enters ipv6 eigrp routing |
| Router(config-router)#no shutdown | Turns on eigrp on the router |
| Router(config)#int [interface] | Sets user in interface |
| Router(config-if)#ipv6 eigrp # | Enables ipv6 eigrp |
| **Show commands** | **Description** |
| Router#show ip route | Shows ipv4 routing table |
| Router#show ipv6 route | Shows ipv6 routing table |
| Router#show run | Shows the configuration of the device |

In this topology, we used the ip route commands for ipv4 and ipv6 to configure eigrp on both routers and advertise their networks. Once both networks were advertised and connected there was full connectivity between the PC’s. The show commands helped me analyze the routing tables and see where eigrp was and where I needed it to be so I could connect the networks.

**Debug:**

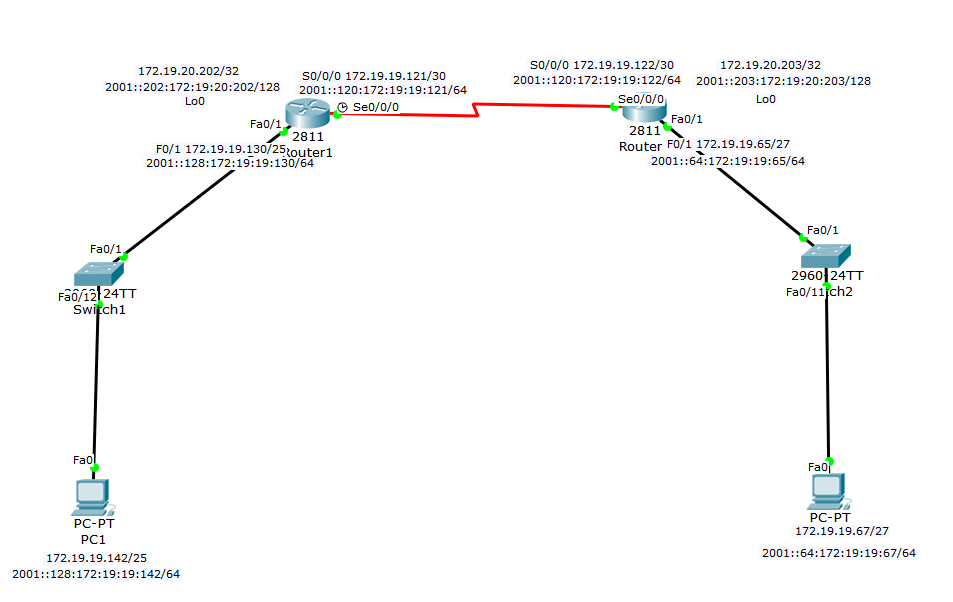
For debugging this topology, I used my show commands and ping. The show ip route commands would allow me to see where I needed to implement eigrp and the networks that were already connected. After I configured the eigrp, I would ping from PC to PC to verify the connectivity.

1. OSPFv2 and OSPFv3 (NOTE: Make sure you utilize loopback interfaces within your topology and configuration…and make sure you specify why you are utilizing them.)

**Description:**

In this question, we again use a dynamic routing protocol. Instead of eigrp we use ospf. Most of the topology is relatively the same, with 2 routers, switches and PCs. OSPFv2 is used for ipv4 and OSPFv3 is used for ipv6. Lastly, we implement a loopback address on our routers that is our router-id. Then end goal of this question is to be able to ping the PC’s.

**Topology:**



**Commands:**

|  |  |
| --- | --- |
| **OSPF Commands** | **Descriptions** |
| Router(config)#router ospf # area # | Senters ospf routing using a # and area # |
| Router(config-router)#router-id [ip address] | Sets the router id for ospf |
| Router(config-router)#network [ip address][wildcard mask] area # | Sets network address for routing protocol |
| Router(config)#ipv6 unicast-routing | Enables ipv6 configurations |
| Router(config)#ipv6 router ospf #area # | Enters ipv6 ospf |
| Router(config)#int [interface] | Enters the specified interface |
| Router(config-if)#ipv6 ospf # area # | Enables ospf ipv6 on specified interface |
| **Show Commands** | **Descriptions** |
| Router#show ip route | Shows routing table for ipv4 |
| Router#show ipv6 route | Shows routing table for ipv6 |
| Router#show run | Shows the configuration of the device |

For this question, the most used commands were again the show ip route commands and show run. They allowed me to see how ospf was working throughout the topology and how to get the routers to connect so my PC’s would have access to each other. The final test showed I had complete connectivity from both PC’s to one another.

**Debug:**

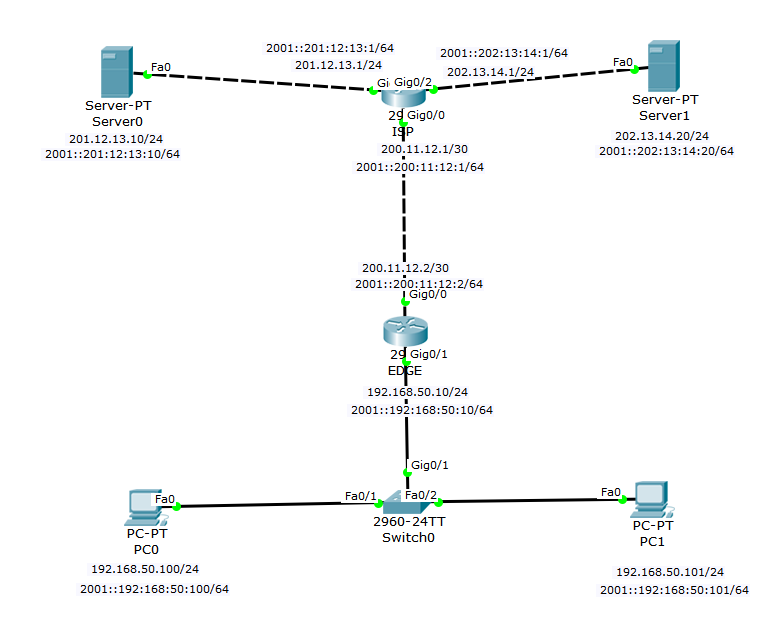
The show route commands allowed me to track ospf throughout the network and see where it was accessing and where it wasn’t. The show run command would allow me to see the full configuration of my routers to double check that I inputted everything on each interface correctly.

1. Extended ACLs and Extended Named ACLs for IPv4 and IPv6 (NOTE: how, where and why you applying these…)

**Description:**

In this question we are needed to use Extended ACL and Extended name ACL, both of which help us contrl traffic to and from the user. In this topology we used edge and isp routers, DNS,HTTP, HTTPS servers, a PC, and a switch. We controlled different websites in the servers and used ACl so that only certain devices could access them.

**Topology:**



**Commands:**

|  |  |
| --- | --- |
| **ACL Commands** | **Descriptions** |
| Router(config)#ip access-list extended [name] | Creates an extended named acl |
| Router(config)#permit tcp [source address] 0.0.0.0 [dest address] 0.0.0.0 | Adds a permit for traffic to the access list being configured |
| Router(config)#deny ip any any | Denys any ip address that wasnt specified in the ACL |
| Router(config-if)#ip access-group [name/num] [in/out] | Applies ACL inside or outside an interface |
| Router(config)#ipv6 access-list [100-199] | Creates an extended numbered ACL in ipv6 |
| Router(config)#[permit/deny] [source ipv6 address] [dest ipv6 address] | Creates a statement in the ACL for an ipv6 address |
| Router(config-if)#no switchport | Changes from layer 2 mode to layer 3 mode |
| Router(config-if)#ipv6 traffic-filter [ACL number/name] [in/out] | Applies ipv6 access list inside or outside on an interface |
| **Show Commands** | **Descriptions** |
| Router#show access-lists | Shows ipv4 ACL’s |
| Router#show ipv6 access-lists | Shows ipv6 ACL’s |
| Router#show run | Shows the configuration of the device |

For this question I configured the topology so that the DNS server was able to only access the HTTP server and the PC could only reach the HTTPS server. Using two ACL configuration I made this connection possible and have the functionality of a website be only accessable from certain devices.

**Debug:**

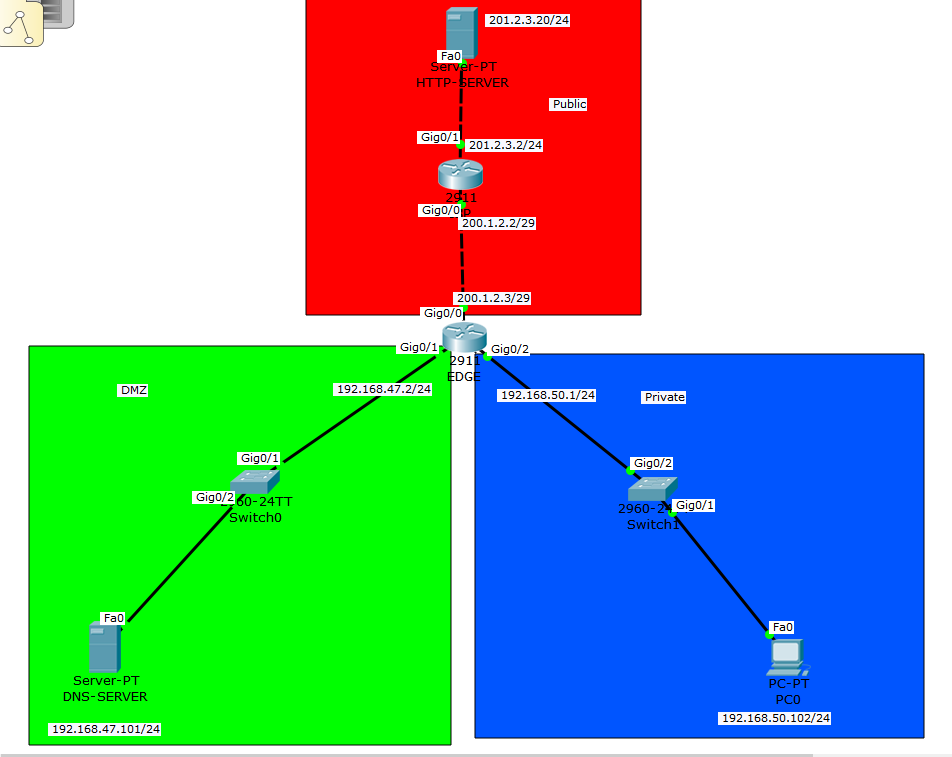
Debugging this problem, the show run command helped me view my configurations on my routers to make sure my ACL’s were correct and running along with using the webpage tab in the pc and DNS server to trouble shoot if the configurations worked.

1. Static NAT and Dynamic NAT/PAT for IPv4 (NOTE: Make sure you show a configuration with static NAT for a web server in the DMZ, then PAT using a pool of multiple addresses for internal hosts connecting to the Internet. You should also demonstrate your knowledge of using just the interface IP Address when this is the only available address presented to you.)

**Description:**

In this question we are asked to configure a network using static NAT and dynamic PAT while also having a DMZ network consisting of a switch and server. I also had a private network of a switch and PC and a public network of a router and server. In the DMZ we used static NAT without being given the end devices ip address, and we used PAT at the Edge router to pool public address and assign private ones to in our private network.

**Topology:**



**Commands:**

|  |  |
| --- | --- |
| **Nat/PatCommands** | **Descriptions** |
| Router(config)#ip access-list standard [name] | Creates an access list for the private network |
| Router(config)#permit tcp [source address] 0.0.0.0 [dest address] 0.0.0.0 | Adds a permit for traffic to the access list being configured |
| Router(config)#deny any | Denys any ip address that wasnt specified in the ACL |
| Router(config-if)#ip access-group [name/num] [in/out] | Applies ACL inside or outside an interface |
| Router(config)#ip nat pool [name][start address][end address]netmask[submask] | Creates a pool of address for NAT |
| Router(config)#ip natinside source list [name] pool [name] overload | Combines a pool and access list and allows multiple addresses to one user |
| Router(config-if)#ip nat [inside/outside] | Sets the interface as an inside or outside |
| **Show Commands** | **Descriptions** |
| Router#show ip nat translations | Show nat translations |
| Router#show run | Shows the configuration of the device |

In this problem we configured three different networks of a DMZ, private network and public network. We were able to configure nat and pat on the edge router and ultimately be able to ping from end device to end device.

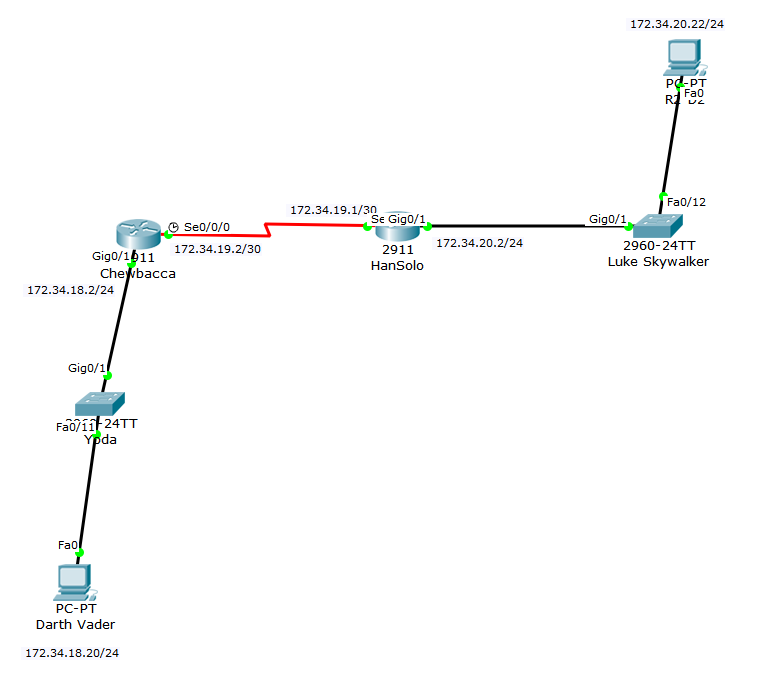
**Debug:**

In the debugging part of this question using the show ip nat translations command was very useful because it gave us insight on what was happening and nat and we were able to see and make adjustments necessary. Also having a show run on our end devices would show us the configurations for the networks and see if we needed to change anything

1. DHCP Server on the Router and a local DHCP client and a remote DHCP client (ie: using DHCP Relay.)

**Description:**

In this problem we are configuring DHCP servers and clients to a a DHCP server. In this topology we have a DHCP server, and a remote as well as a lock DHCP client. Since they are two different networks we use DHCP relay to connect the networks. Once the topology is done we should have our server, and two clients all using DHCP relay.

**Topology:**

**Commands:**

|  |  |
| --- | --- |
| **DHCP Commands** | **Descriptions** |
| Router(config)#ip dhcp excluded-address [ip address start] [ip address finish] | Sets the excluded addresses that DHCP can’t use to auto assign |
| Router(config)#ip dhcp pool [name] | Creates DHCP pool |
| Router(dhcp-config)# network [network address] [subnet mask] | Sets DHCP pool |
| Router(dhcp-config)#default router [ip address] | Sets the default router for the DHCP pool |
| Router(config)#router ospf 1 | Enters ospf |
| Router(config-router)#network [ip address] [WC mask] [area #] | Sets network to route to |
| Router(config-if)# ip helper-address [ip address] | Routes all DHCP requests to address listed |
| **Show Commands** | **Descriptions** |
| Router#show ip dhcp pool | Shows the dhcp pools configured |
| Router#show ip dhcp binding | Shows the dhcp bindings |

In this problem we had to set DHCP throughout the routers. To test the configuration was working we would go into each PC and run the DHCP and see if it would give an address or not. By the end of the question I got both PC’s configured using DHCP

**Debug:**

To debug this problem I used the show cases above to make sure I was on the right track but for the most part I used the ip configuration tab in the PC’s to see if the DHCP actually worked.

1. PPP using PPP CHAP

**Description:**

For this problem we are asked to use PPP chap. To do so, I used two routers and connected them through a DCE cable. From there I configured PPP using chap.

**Topology:**



**Commands:**

|  |  |
| --- | --- |
| **PPP Commands** | **Descriptions** |
| Router(config)#username [username] password cisco | Sets the username and password to cisco |
| Router(config)#int s0/0/0 | Sets user in the s0/0/0 interface |
| Router(config)#encapsulation ppp | Sets PPP on interface |
| Router(config)#ppp authentication chap | Specifies for chap to be used |
| **Show Commands** | **Descriptions** |
| Router(config)#debug ppp authentication | Enables debugging for the current ppp authentication |
| Router(config)#debug ppp negotiation | Enables debugging between the PPP process |

Using the show commands above we were able to verify and check that PPP chap was working correctly. This part was definitely the least time consuming and easiest part of the project.

**Debug:**

To debug this problem using a show run or the debugging commands would help. The debugging commands will show the configurations on the router and from there you can check to see if PPP I being used correctly.