Data Communications and Networking

Coursework

Daniyar Nazarbayev

H00204990

Part 1.

Lab 1.

Lab 1 gave me a few basic ideas on how to use GNS3.

It had the following commands – **ip, show ip, clear ip, ping**.

It also tried to teach subnetting. It took me a while to understand it, but I feel like I do now (all thanks to the Venn diagrams and sets).

Before I used to think that both PCs had to have the same network id to be able to ping each other. That is not true. I could have one VPC with an ip of 192.168.1.2/8 and another with 192.168.1.1/24.

One of them has a network id of 192.0.0.0 and another 192.168.1.0, but both can ping each other. I like to think of subnets as "sets". 192.0.0.0 is the bigger set. 192.168.1.0 is its subset. To be able to ping each other both ips have to be in both sets, or to simplify, just the subset.

The main purpose of subnetting seems to be efficiency. On the web it is said it was created to fix the problem with the hubs. 192.0.0.0 family has much more addresses than 192.168.1.0, and since hub sends data to all addresses of its network, it would be wise to use subnets.

The lab also had VLANs. After setting them up, I noticed that I could not ping to PCs of different VLANs.

After looking up what a VLAN is, I found out that VLAN basically creates multiple switches. Physically there is one, but virtually there can be many, and none of them are not connected to each other and act separately. Each virtually separated switch can only see hosts assigned to its VLAN. They are the equivalent of subnets, and require routers to communicate with each other (unless it is a layer 3 switch which is a router in itself).

Lab 2.

Lab 2 showed us how to set up a router in GNS3. The model we used for the router was c2691.

Routers basically allow hosts to communicate when they are not part of the same set. We were thought about how to set up interfaces (ports) of the router, and the necessity of the gateway ip for each PC. What I noticed is that a router's interface has to be connected to a unique network.

As for the gateway, it is basically the ip address of the interface of the router that goes to the "outside world". It has to be of the same set as the PC that is connected to it.

Lab 3.

Lab 3 thought us routing. Part 1 was about setting static routing and part 2 was about dynamic routing (RIP protocol).

For static routing, we had to hardcode a routing protocol for each network that was not part of the router's networks. For example, router 1 has 2 interfaces, one has ip 192.168.1.254/24 and the other 10.0.0.1/24. The router has access to 2 networks – 192.168.1.0 and 10.0.0.0. It however does not have

access to networks 10.0.2.0 and 200.0.0.0, and I have to hardcode the path it has to take to reach both of them.

I can either do it like this.

ip route 10.0.2.0 255.255.255.0 10.0.0.2

ip route 200.0.0.0 255.255.255.0 10.0.0.2

10.0.0.2 because it is the address the router is directly connected to.

Or I could do it like this.

ip route 0.0.0.0 0.0.0.0 10.0.0.2

Because they both go through the same path, and 0.0.0.0 is basically a wildcard for any network.

Router 2 has access to networks 10.0.0.0 and 10.0.2.0.

Router 3 has access to networks 200.0.0.0 and 10.0.2.0.

The process is the same.

For dynamic routing, everything is much simpler. When you configure the router, you just tell which networks your router is connected to, like this:

router RIP

version 2

network 192.168.1.0

network 10.0.0.0

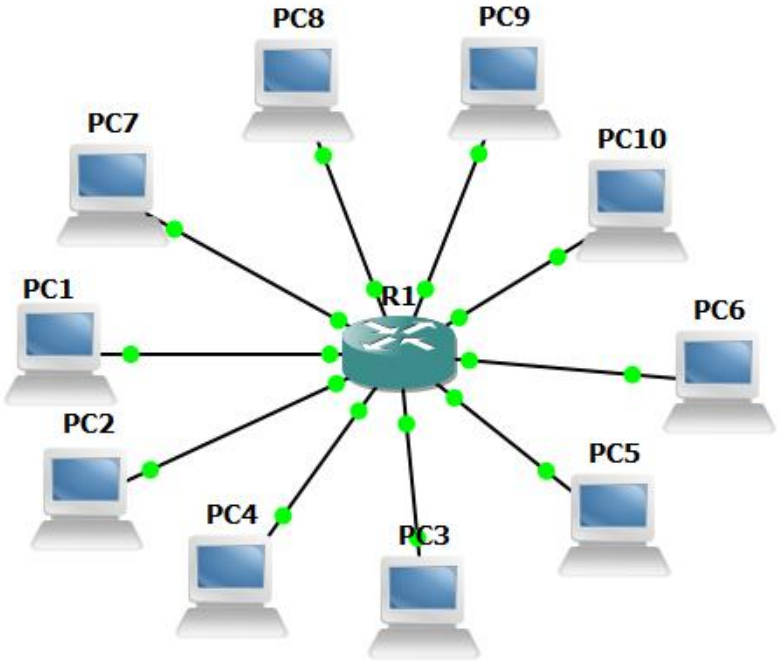
That is all a person has to do to set it up. The protocol does the rest.

The default subnet mask in RIP is /8, so in the second router, I only have to add:

network 10.0.0.0

Part 2.

Star topology.



This router has a NM-16ESW module, with an intent to use it as a switch.

To make it run as a switch, I had to put these lines:

conf t

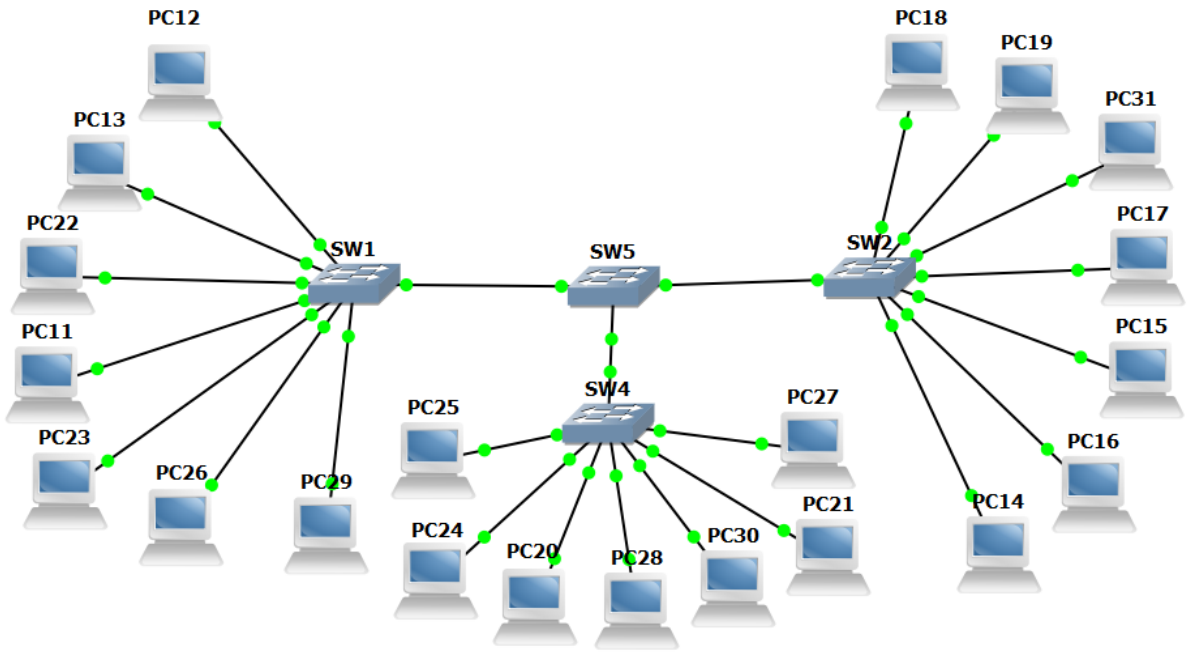
no ip routing

interface range fa1/0 – 15

no shutdown

switchport

Tree Topology.



I used normal 8 port switches for this topology. There are 4 switches and 21 hosts.

I encountered a problem while making it by the way. The tree has to be a spanning tree, meaning no loops, otherwise it will not work. In contrary, routers can have loops, because packets have TTL.

Advantages and disadvantages.

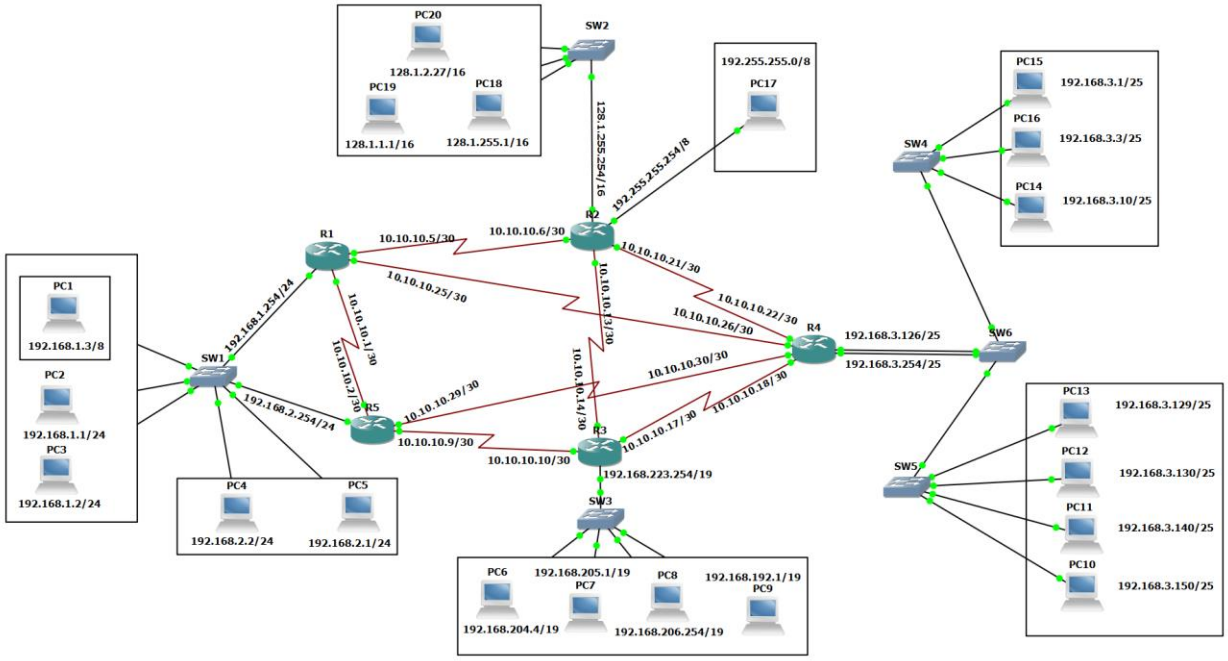
I looked at other topologies before making this, and I found that the star and tree topologies were the only ones that had switches. Everything else like a ring, line, bus, mesh topologies relied on host-to-host connections, and if one host were to go down, it would disable the connection for multiple hosts.

Star topology does not suffer from this. That said, all hosts might lose connection if the switch goes down, but I feel that typically switches can be monitored and easily replaced, since it is some company's property rather than personal property, meaning the switch is in their zone of control. Also, in cases that involve host-to-host connections, it is much harder to figure the exact cause, and they might have to recheck all the wiring.

I feel that the star topology can suffer from "DDOS". Imagine if a single switch had 2 billion ports, with 2 billion devices connected to it, that are set to a /1 subnet mask. If they were all to send a request all at once, the switch will not be able to handle it. Its CPU would not be fast enough to process all the requests, so it will have to queue them up in RAM or buffer and when it will hit that max, it will start denying requests.

I feel that that is why tree topology exists. Tree topology is also known as extended start topology. By definition, it allows more than one switch in its topology. That basically solves the problem with the star topology. That said, there might arise a problem with bottlenecks if the topology is designed unequally.

Part 3.



I added a link for the full-sized image on google drive.

https://drive.google.com/open?id=1zUB2wIh32ym4B3gAmx3Vh-LsB5vEADHA

Honestly, I do not really have much to add. The image speaks for itself.

There are 21 hosts, a few switches and 5 routers. I have implemented an OSPF routing protocol for them.

I have added a NM-4T module to c2691 to have 4 extra serial connections.

I used a /30 subnet mask for the router-to-router connections, so that I would not waste addresses.

The OSPF implementation was straight forward.

router ospf *<id>*

network *<network-id>* *<inverse-subnet-mask>* **area** *<area-number>*

This is the syntax for setting one. Unlike EIGRP, the id in OSPF does not have to be the same for every router. Also, what I noticed was that it did not matter whether I put a normal subnet mask or inverse. Subnet masks are a collection of 1's that have to start from left most side. Router noticed if I put a normal mask or inversed, and applied the logical operations necessary, before adding it to the config.

For area, I decided not to bother myself and make everything be in area 0, but in theory, every other area has to be directly connected to area 0. Area 0 is known as the backbone.

As you see, in my case it is not really necessary to have multiple areas, since all my routers are directly connected to hosts.

I tried doing a lost link experiment. Sent a ping -t, and upon deactivating a few routers that were used in that route, it took about 6 timeouts till it managed to ping the host again.

I have researched a bit on the theory behind OSPF, EIGRP and RIP.

As we already know, RIP uses hop counts, and it passes on the information it has to its neighbors, presumably through flooding. That is how the entire network gets updated. If a router gets disconnected for example, the hop count starts infinitely increasing. There will not be an overflow, since the max count is 15, but it is a redundant use of resources, which is why there are ways of countering this bug, like the split horizon technique. The protocol also supports load balancing over paths that have the same hop count. It works on a timer, every 30 seconds it checks whether connections are still up and waits for the ACK. RIP is considered a dead protocol though, mostly because the max hop count is 15, severely limiting the size of the network.

EIGRP is of the same caliber. It uses the neighbors for information, but the protocol is exclusive to CISCO hardware. It has advanced costing features that included time delays and link bandwidth. Whenever a router gets updated, it sends the update to all neighbors, and then ones who received the update will send it to their neighbors and so on.

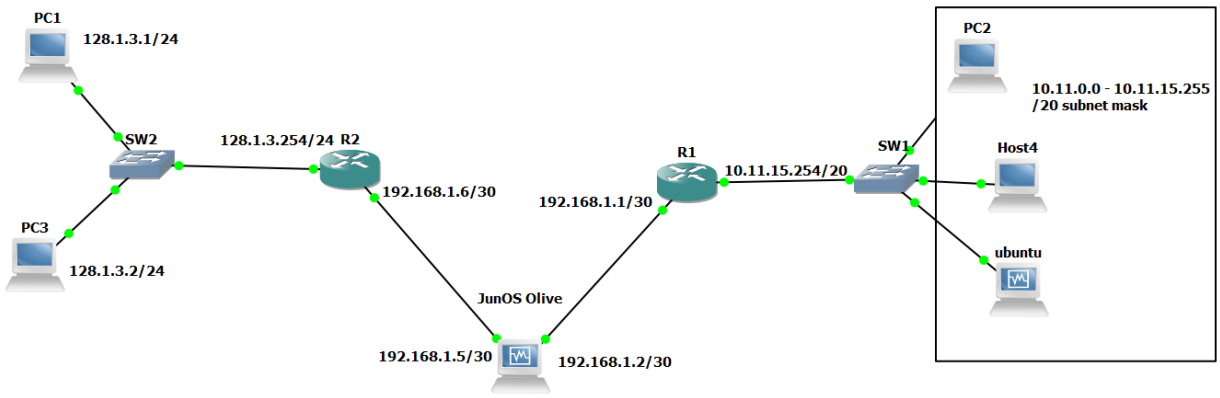
OSPF on other hand keeps track of the whole topology. Each router has the table of the whole topology, and a table for quickest paths to each node using Dijkstra's algorithm. A router finds all his neighbors, and sends the neighbor data and the costs to get to them through a multicast ip to all routers. The multicast ip is in 224.0.0.0 range and apparently you cannot set any ip from that range to an interface.

Every time a change to the topology is made, every router has to implement that change, and recalculate all the routes. I suppose that is why it takes so long for OSPF to recover and ping again.

Part 4.

For part 4, I did four things.

1) Juniper router.
2) Added my own pc to GNS3.
3) Added a ubuntu VM to GNS3.
4) Implemented EIGRP to the topology from part 3.



EIGRP was straight forward too.

router eigrp id

network <network id> <subnet-mask>

Unlike OSPF, id has to be identical for all routers. Subnet mask can be inverted or normal. Default mask is /24.

For other tasks, I decided to create an extra topology.

I implemented these tasks pretty hastily, in last 1-2 days before submission, so while things work, there are still some things that are not ironed out.

First of all, juniper cannot be directly connected to a VPC for some reason, cause it cannot ping them, but if you put a CISCO router in between, it will ping.

The reason why I needed a separate topology is because I was using serial ports in the previous topology, and this VM copy does not provide such option. I can only use fast ethernet connections.

Implementing juniper was not all that hard. Here is pretty much all the code that was used.

set interfaces em1 unit 0 family inet 192.168.1.2/30

set interfaces em0 unit 0 family inet 192.168.1.5/30

set protocols ospf area 0.0.0.0 interface em0

set protocols ospf area 0.0.0.0 interface em1

Unit keyword was pretty irrelevant, but necessary for some reason. Inet is required for ipv4.

As for routing, OSPF does not really require an id, so a lot of things are skipped in juniper.

Also, as you see, when we configure the routing, we just point to the interface to determine the network instead of typing it twice like in CISCO. The only other thing we have to add is the area.

For VM implementation (ubuntu), the only iso I had on hand was ubuntu, so I used it. It does not work in headless VM mode. It has to be opened in VirtualBox. The VM has to be fully stopped before that, and VirtualBox closed. I used this command to change ip.

sudo ifconfig eth0 10.0.0.100 netmask 255.255.255.0

use ifconfig to find the name (ex: eth0, esp0s3). Set any desired ip and mask.

Host4 is my pc connected. I can open cmd and ping PC2 for example if I put them on the same network. I connected my “wifi adapter” to the switch. It was one of the few network devices with an ip set (it was in 10.11.0.0-10.11.15.255 range, already released and renewed). I have only tested it with pings to the same network though. I did not want to temper with my settings.