Build your own Internet[1]

Current version: **2018-03-25**

1. Introduction

In this assignment, you will learn how to build and operate a layer-3 network using traditional distributed routing protocols, how different networks managed by different organizations interconnect with each other, and how protocols, configuration, and policy combine in Internet routing.

More specifically, you will first learn how to set up a valid forwarding state within an autonomous system (AS) using OSPF, an intra-domain routing protocol (Part A). Then, you will learn how to set up valid forwarding state between different ASes, so that an end-host in one AS (e.g., your laptop connected to the university wireless network) can communicate with an end-host in another AS (e.g., Google's server). To do that, you will need to use the only inter-domain routing protocol deployed today: BGP (Part B). After that, you will implement different BGP policies to reflect business relationships or traffic engineering that exist in the real Internet (Part C). You will configure both OSPF and BGP through the Quagga software routing suite, which runs on several virtual routers in your virtual machine (VM).

The rest of the document is organized as follows. We first give you a crash course on how to configure Quagga routers (Section 2), and then describe the setup you will have to use (Section 3). Then, we list the questions you should answer (Section 4), and finish with submission and other general information (Section 5).

1.1 Schedule

- The deadline for Part A of this project is Apr. 3rd, 23:59.
- The deadline for Part C is June 6th, 23:59.
- The deadline for Part D is July 4th, 23:59.
- In between Part A and Part C, we will have a classwide hackathon in which students peer their networks with each other (Part B). The hackathon will happen Apr. 30th.

[1] This project is based on one from the course "Communication Networks" taught by Prof. Laurent Vanbever and TA Thomas Holterbach at ETH Zurich. This project has also been used at USC's "Computer Networks" course taught by Prof. Ethan Katz-Bassett and TAs Yi-Ching Chiu and Pedro Silva. This project is also currently being used at Columbia's undergrad networking course. We appreciate their effort developing the assignment.

1.2 Groups and collaboration policy

This is an *individual project*, but you can discuss at a conceptual level with other students or consult Internet material, as long as the final code and configuration you submit is completely yours and as long as you do not share code or configuration. Before starting the project, be sure to read the collaboration policy at the end of this document.

2. Configuring IP Routers

In this assignment, you will need to configure IP routers through a Command Line Interface (CLI). Each IP router vendor (e.g., Cisco or Juniper) or software routing suite (e.g., Quagga) has its own CLI. Fortunately, those CLIs are similar, and if you know how to configure a router using the Quagga CLI, you can easily configure a Cisco or a Juniper router as well.

In this section, we briefly describe how to configure a Quagga router. However, this is a very short introduction, we strongly encourage you to take a look at the official documentation to get more information.

2.1 The Quagga Command Line Interface

When you enter the Quagga CLI (see Section 3.3 on how to access routers), you should see the following line:

router_name#

At any time when you are in Quagga CLI, you can use **?** to see all the possible commands that you can type at that point. For example, you should see the following (partial) output when you type **?** when you first enter the CLI.

router_name# ?

clear	**Reset functions**
configure	**Configuration from vty interface**
exit	**Exit current mode and down to previous mode**
no	**Negate a command or set its defaults**
ping	**Send echo messages**
quit	**Exit current mode and down to previous mode**
show	**Show running system information**
traceroute	**Trace route to destination**
write	**Write running configuration to memory, network, or terminal**

Showing configuration. The command **show** will print various snapshots of the router configuration. To see what types of information can be shown, just type **show ?**. For example, **show running-config** will print the configuration that is currently running on the router. You can shorten the commands when there is no ambiguity. For instance, **show run** is equivalent to **show running-config**. Similar to the Linux terminal, you can also use auto-completion by pressing the tab key.

Switching to configuration mode. To configure your router, you must enter the configuration mode with **configure terminal** (**conf t** for the short version). You can verify that you are in the configuration mode by looking for the "config" prefix in your CLI prompt. You should see the following when you are in configuration mode.

router_name(config)#

If you want to cancel parts of the configuration, you can prefix the command you want to remove with **no**.

2.2 Configuring router interfaces

A router interconnects IP networks through several router interfaces. When a router receives a packet from one interface, it forwards it to another interface based on some pre-calculated forwarding decisions. Each interface must have an IP address configured and must be in a different subnet. To configure the IP address for an interface, you will first enter the configuration mode, and then specify the name of the interface you want to configure. For example, you can use the following commands to configure interface <interface_name> to 1.0.0.1/24.

router_name# conf t
router_name(config)# interface <interface_name>
router_name(config-if)# ip address 1.0.0.1/24

There are several ways to verify the current configuration has been updated correctly. You can use **show run** to examine the current router configuration. You can also show detailed information for interfaces by command **show interface**. To look at a specific interface, use **show interface <interface_name>**.

Once you have configured an IP address and a subnet to an interface, the router knows that packets with a destination IP in this subnet must be forwarded to this interface. You can use the following commands to show the subnets that are directly connected to your router.

router_name# show ip route connected
C>* 1.0.0.0/24 is directly connected, <interface_name>

We see that 1.0.0.0/24 is directly connected and reachable with the interface <*interface_name*>. At this stage, any packet with a destination IP that is not in a directly connected subnet will be dropped. If you want your router to know where to forward packets with an IP destination in a remote subnet, you must use routing protocols, such as OSPF or BGP.

2.3 Configuring OSPF

OSPF routers flood IP routes over OSPF adjacencies. For Quagga routers, they continuously (and automatically) probe any OSPF-enable interface to discover new neighbors to establish adjacencies with. By default, the Quagga router will activate OSPF on any interface whose prefix is covered by a **network** command under the **router ospf** configuration. For instance, the following commands would activate OSPF on any interface whose IP address falls under 1.0.0.0/24 or 2.0.0.0/24:

router_name# conf t
router_name(config)# router ospf
router_name(config-router)# network 1.0.0.0/24 area 0
router_name(config-router)# network 2.0.0.0/24 area 0

OSPF has scalability issues when there is a large number of routers. To mitigate such issues, the router topology can be hierarchically divided into what is called “areas”. In this assignment, your network is small and you do not need more than one area: *you will only use the area 0*.

With OSPF, each link between two routers is configured with a weight, and only the shortest paths are used to forward packets. You can use the following commands to set the weight of a link connected to <interface_name> to 900:

router_name# conf t
router_name(config)# interface <interface_name>
router_name(config-if)# ip ospf cost 900

You can use the following command to check the OSPF neighbors of a router:

router_name# show ip ospf neighbor
Neighbor ID Pri State Dead Time Address Interface RXmtL RqstL DBsmL
1.0.0.2 1 Full/Backup 1.0.0.2 newy:1.0.0.1 0 0 0
2.0.0.2 1 Full/Backup 2.0.0.2 newy:2.0.0.1 0 0 0

We see that the router has established two OSPF session with two neighbors. The first one is connected via the interface 1.0.0.1 and its IP is 1.0.0.2. The second one is connected via the interface 2.0.0.1 and its IP is 2.0.0.2. Since you are now connected to two other routers through OSPF, they can send you information about the topology of the network. Let’s take a look at the routes received by OSPF using the following command.

router# show ip route ospf

O 1.0.0.0/24 [110/10] is directly connected, newy, 07:09:33
O 2.0.0.0/24 [110/10] is directly connected, atla, 06:14:24
O>* 10.104.0.0/24 [110/20] via 2.0.0.2, atla, 00:00:10

You can see that our router has learned how to reach the subnet 10.104.0.0/24. The O at the beginning of each line indicates that the router has learned this subnet from OSPF. To reach it, it must send the packets to its neighbor router 2.0.0.2. If you want to have more information about the routers of this OSPF area, you can use **show ip ospf database**.

2.4 Configuring BGP

While OSPF is only used to provide IP connectivity within an AS, BGP can also be used to advertise prefixes between different ASes. Unlike OSPF, BGP routers will not automatically establish sessions. Each session needs to be configured individually. The following commands show you how to start a BGP process and establish two BGP sessions with neighboring routers. The integer following router bgp indicates your (local) AS-number. Here, the local AS-number is 2.

router_name# conf t
router_name(config)# router bgp 2
router_name(config-router)# neighbor 150.0.0.1 remote-as 15
router_name(config-router)# neighbor 2.0.0.2 remote-as 2

By default, whenever the remote-as is different from the local number (here, 2), the BGP session is configured as an external one (i.e., an eBGP session is established). In contrast, when the remote-as is equivalent to the local one, the BGP session is configured as an internal one. Here, the first session is an eBGP session, established with a router in another AS (150.0.0.1), while the second one is internal session (iBGP), established with a router within your AS (2.0.0.2). You can check the state of your BGP sessions using the following command.

router_name# show ip bgp summary

Neighbor V AS MsgRcvd MsgSent TblVer InQ OutQ Up/Down State/PfxRcd
2.0.0.2 4 2 3 6 0 0 0 00:01:16 0
150.0.0.1 4 15 2009 1979 0 0 0 01:31:42 1

You can see that our two BGP sessions are up. Notice that if the State/PfxRcd column shows “Active” or “Idle” instead of a number, that indicates that the BGP session is not established.

You might find it useful to manage several BGP neighbors as a group. By doing so, you can apply configurations to the whole group rather than typing similar commands for every

neighbor, which could be tedious. Check **peer-group** command in the Quagga documentation for more information.

After BGP sessions are established, you can advertise prefixes using **network** command. For example, the following command advertise a prefix 10.104.0.0/24.

router_name(config-router)# network 10.104.0.0/24

Notice that you will only use this command to start advertising prefixes that are directly connected to your network. For prefixes that can only be reached indirectly through other networks (e.g., prefixes advertised by your neighbor), you simply propagate the existing advertisements.

After you do that, your neighbor 150.0.0.1 will receive this advertisement and know that it can forward you all the packets with a destination IP in the subnet 10.104.0.0/24.

You can check the routes learned from other BGP neighbors using the following command:

router_name# show ip route bgp
B>* 2.101.0.0/24 [20/0] via 2.0.0.2, interface used, 00:03:17

In this example, we can see that neighbor 2.0.0.2 has advertised one prefix: 2.101.0.0/24. The B at the beginning of the line indicates that the router has learned this prefix from BGP.

2.5 Configuring BGP policies

After you have BGP sessions running successfully, you might want to configure some BGP policies. For example, you may want to prefer one provider because it is cheaper than another provider. For another example, you may want to avoid sending traffic to one AS for a particular prefix for security reasons. BGP offers several ways to configure such policies. The LOCAL-PREF attribute can help you influence outbound routing, while the MED attribute, AS-Path prepending, or selective advertisements can help you influence the inbound routing. In Quagga, you can use **route-maps** to implement these policies.

Route-map

Route-maps allows you to take actions on BGP advertisements immediately after an advertisement has been received, or right before being sent to a neighbor. A route-map is composed of route-map entries, and each entry contains three parts: **matching policy**, **match, and set**.

The matching policy can either be “permit” or “deny”. It specifies whether to permit or deny the routes that match the conditions in the match part. Intuitively, only permitted routes will go

through the actions in the set part, while denied routes won't be considered further. The match part is a boolean predicate that decides on which route to apply the actions to, and the set part defines the actual actions to take if a route matches.

Let's take a look on what you can match on:

```
router_name# conf t
router_name(config)# route-map <MY_ROUTE_MAP> permit 10
router_name(config-route-map)# match ?
        as-path             Match BGP AS path list
        community           Match BGP community list
        interface           Match first hop interface of route
        ip                  IP information
        metric              Match metric of route
        origin              BGP origin code
        peer                Match peer address
        ...                 ...
```

As you can see, you can match pretty much any attribute contained in a BGP UPDATE including: AS-PATH, community value, IP addresses or subnets, etc.

A BGP community can be seen as a label or a tag that can be attached to any route. You can use this attribute in both the match part and the set part of a route-map. As a convention, a community is often expressed as two separate integers, with the first one identifying the AS number that defined the community (either to use in routes internally within the AS, on routes the AS passes to neighbors, or for neighbors to attach to routes passed to the AS). More than one community can be attached to a route.

Now let's take a look on what you can do with the matched routes using the set part.

```
router_name# conf t
router_name(config)# route-map <MY_ROUTE_MAP> permit 10
router_name(config-route-map)# set ?

        as-path             Transform BGP AS-path attribute
        community           BGP community attribute
        ip                  IP information
        metric              Metric value for destination routing protocol
        local-pref          BGP local preference path attribute
        origin              BGP origin code
        ...                 ...
```

This set part enables you to modify any route attributes. Among others, you may find the following attributes most useful in this project: local-pref, community attributes, AS-path (to perform AS-path prepending.) Notice that these operations will only be applied on routes that match the condition specified in match part, if you use the permit clause.

A route-map can contain multiple entries. Entries are processed in a chain. The order in which entries are processed is given by a sequence number (in the previous example, 10). The entry with the lowest sequence number is executed first. For example, if there are two entries for a route-map, with sequence number 10 and 20 respectively, a route will be first examined using the entry with sequence number 10, and then using the entry with sequence number 20. Notice that by default, if a route matches an entry, it won't be considered for the following entries with higher sequence numbers. Please check the official Quagga documentation to learn more about the default behaviors and how to change that if needed.

Important. Whenever you use a route-map, an entry that denies everything is implicitly added with the largest sequence number. That is to say, any routes that do not match in the previous entries would by default be denied. You need to add an additional entry to permit any routes to change this behavior.

Applying route-maps to BGP sessions

Once you have defined a route-map, you can apply it to a BGP session. A route-map can either be applied on incoming routes or on outgoing routes. For example, you can use the following commands to apply a route-map to the outbound advertisements to a BGP neighbor 150.0.0.1.

```
router_name# conf t
router_name(config)# router bgp 2
router_name(config-router)# neighbor 150.0.0.1 route-map <MY_ROUTE_MAP> out
```

The keyword ***out*** at the end of the last line means that this route-map is applied to the routes your router advertises. The keyword ***in*** would have applied the route-map to the routes the neighbor is advertising to your router.

Example

Let's assume multiple routers in your AS (AS 15) have an eBGP session with one of your providers (AS 2). However, you strongly prefer your provider to send traffic to you through one of them. You have agreed with your provider to use a BGP community value equal to 15:100 (15 is your AS number) to indicate the path you prefer.

The following commands show you how you can configure the route-map and apply it to the preferred router to tag any outgoing routes with such community.

router_name# conf t
router_name(config)# route-map <COMMUNITY_VALUE> permit 10
router_name(config-route-map)# set community 15:100
router_name(config-route-map)# exit
router_name(config)# router bgp 15
router_name(config-router)# neighbor 2.0.0.2 route-map <COMMUNITY_VALUE> out

Your provider can check that your prefixes have been advertised with the BGP community value with this command: **show ip bgp community 15:100.** The prefixes listed are the prefixes with the community value **15:100**.

Tip. Sometimes it takes time for BGP to converge, so you might not see the results corresponding to your changes immediately. You can use **clear ip bgp *** to force it to converge faster.

Now your provider can match on this tag to set the local-preference attribute accordingly and force traffic to exit via the chosen exit point. The following commands show you how your provider can do that.

router_name# conf t
router_name(config)# ip community-list standard <TAG> permit 15:100
router_name(config)# route-map <LOCAL_PREF> permit 10
router_name(config-route-map)# match community <TAG>
router_name(config-route-map)# set local-preference 1000
router_name(config-route-map)# exit
router_name(config)# router bgp 2
router_name(config-router)# neighbor 15.0.0.2 route-map <LOCAL_PREF> in

You can check that the local-preference has been correctly set with the following command:

router# show ip bgp
Network Next Hop Metric LocPrf Weight Path
***> 10.104.0.0/24 1.0.0.1 0 100 0 2 i**
***> 15.1.0.0/24 150.0.0.1 0 1000 0 15 i**

The prefix advertised from AS 15 (15.1.0.0/24) with the community value 15:100 now has a local-preference value equals to 1000. In contrast, another prefix (10.104.0.2/24) without the community value have the default local-preference value 100.

This section only sketched the basics for route-maps. In this project, one of the learning goals is to get yourself more familiar with route-maps in order to implement the correct BGP policies.

3. General project setup

3.1 Accessing Your VM

You will manage your own AS, just like a normal network operator does. The routers in your AS are already running in a VM that is assigned to you. All the VMs are running on our server, where we have set up the proper connection between these VMs to simulate the connections between actual ASes.

To access your VM, SSH into your VM as root, using ports starting from 3000. We will notify you individually about your AS number and the password to access the VM. Your assigned port number is equal to 3000 + your AS number. For example, if your AS number is 1, here is how you can access the VM with ssh:

> ssh -p 3001 root@zeus.winet.dcc.ufmg.br

Passwords are unchanged. If you want to simplify the access to your VM, you can use SSH key-based authentication, but *do not* change the password. If you want to download an entire directory (e.g., the configs directory) from your VM to the current directory of your own machine, you can use scp:

> scp -r -P 3001 root@zeus.winet.dcc.ufmg.br:~/<remote-dir> <local-dir>

Important. If you use tmux, a terminal multiplexer, *do not* use the session mininext. The virtual network is running in this session. If you need to reboot your VM, please contact the TAs directly.

3.2 Accessing routers and hosts

When you log into your VM, you should first check that the message “mininext is running” does appear. Report to TAs immediately if it does not. You can then use the script *bash-in.sh* to switch to a router or a host. For example, with the following command, you will access the router LOSA.

$ ./bash-in.sh LOSA

To access the router CLI use:

LOSA$ vtysh

Type **exit** to leave the CLI or the router.

From your VM, you can also go to a host. For example, if you want to go to the host which connects to router SEAT, you can use the following command.

> ./bash-in.sh SEAT-host

When you are in a host, you can use **ifconfig** to see the interface which connects to the router. In this case, the name of the interface is seat.

3.3 Configuring Quagga router

Important. When you want to configure your router, *do not* edit the config files directly. Instead, always use the Quagga CLI.

Important. The current running configurations are not automatically synchronized with (i.e., written to) the configuration files you can find in the directory configs. *You must synchronize them manually.* To write your current configuration to the configuration files, you can use the following command. Also, you will have to do that for each router.

router_name# write file

We recommend you to regularly save your configuration (the directory configs) to your own machine, since all your configuration will be reset if we reboot your VM. In case of a reboot, you can quickly put back your configurations in the routers by copy/pasting your configurations into the Quagga CLI.

3.4 Network topology

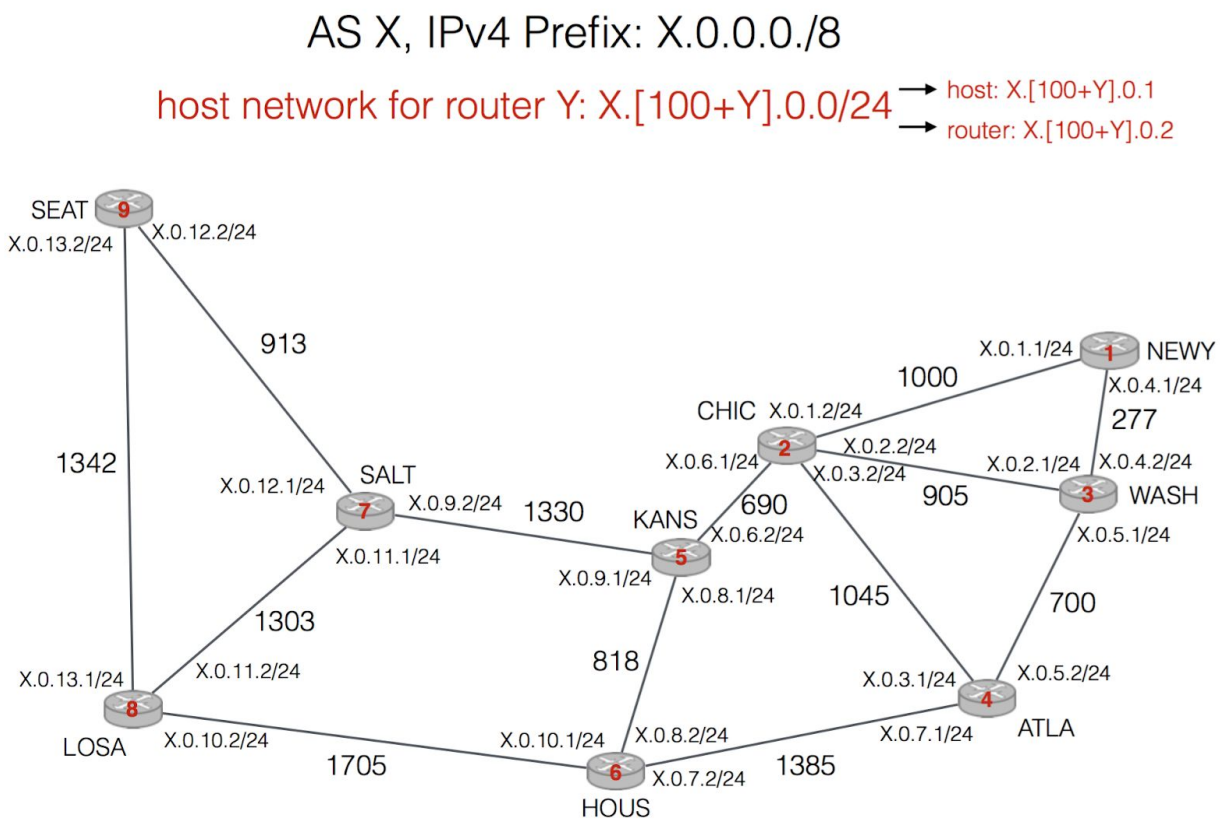

Figure 1: The network topology of your AS. Your AS is composed of 9 routers and is similar to the Internet2 topology. A /8 prefix has been assigned to you. You can use it to configure your local networks. The subnets you must use for each of your local networks are indicated on each interface. The OSPF weights are also indicated on each link.

You will be assigned a unique AS number. Routers and hosts composing your AS are already running in your VM. The topology of your network is similar to the Internet2 topology (the main not-for-profit research and education Internet provider in the US). Figure 1 shows what your network looks like. You will be assigned a /8 prefix that you can use. If you are AS X, then the prefix X.0.0.0/8 is yours. For example, AS 5 owns the prefix 5.0.0.0/8. When you assign IP addresses to your routers, you will have to use a different subnet (but all belonging to your /8 prefix) between each pair of routers. The subnets you must use are also indicated in Figure 1. For example, between SALT and KANS, you *must* use the subnet X.0.9.0/24. The interface in KANS that is connected to SALT must have the IP address X.0.9.1 and the interface in SALT that is connected to KANS must have the IP address X.0.9.2 (with X being your AS number).

One normal host is also connected to each router. Between each router and its host, you will have to use the subnet X.[100+Y].0.0/24, where X is your AS number, and Y is the ID of the

router (IDs are shown on each router, for example, the ID of LOSA is 8). Then, the host will have the IP address X.[100+Y].0.1 and the interface of the router that is connected to this host will have the IP address X.[100+Y].0.2. As an example, if you are AS 5 and you want to configure the host connected to ATLA. This host will have the IP address 5.104.0.1/24 and the interface of ATLA connected to this host will have the IP address 5.104.0.2/24.

On each link, the OSPF weight is also indicated. For example, between WASH and NEWY, you will have to configure an OSPF weight equal to 277. The traffic between NEWY and SEAT should use the shortest path (based on the OSPF weights), which is NEWY-CHIC-KANS-SALT- SEAT.

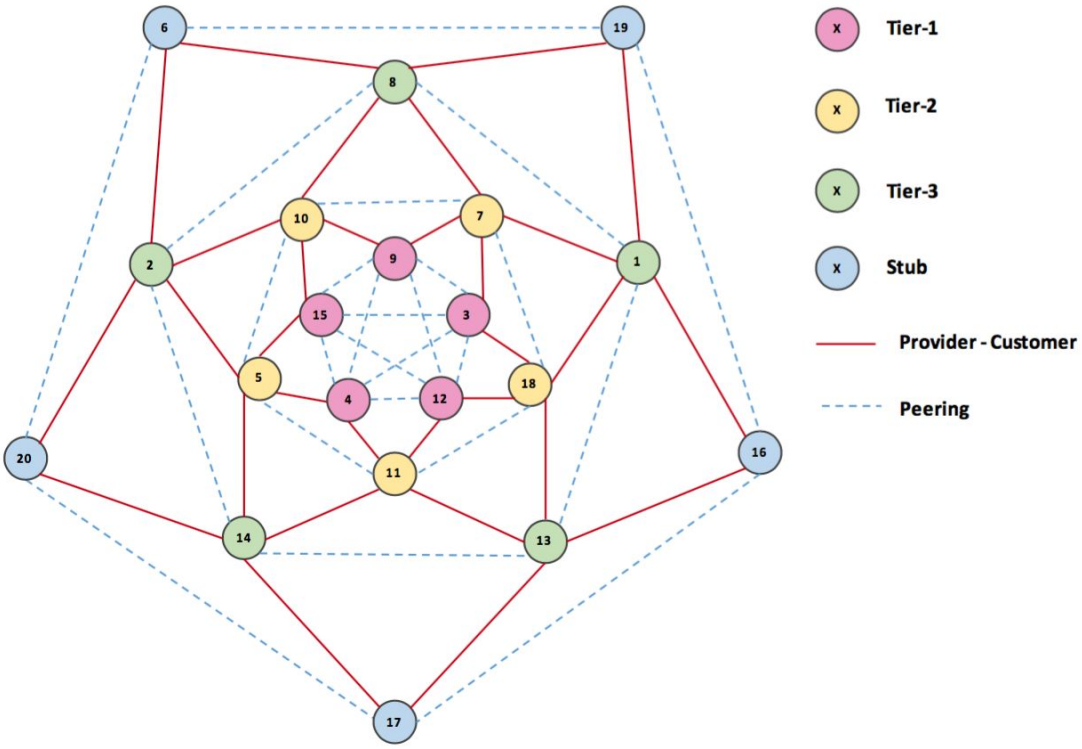

Figure 2: AS-level network topology

Different ASes are connected with each other as shown in Figure 2. There are 3 types of networks: tier-1, tier-X, and stub networks. For example, ASes 1-5 are tier-1 networks. For each tier-1 AS, it has 4 tier-1 peers and 2 tier-2 customers. For each tier-X AS, it has 2 tier-(X-1) providers, 2 tier-X peers, and 2 tier-(X+1) customers. For each stub AS, it has 2 providers and 2 peers.

3.5 Management VM

We will set up a management VM which allows you to launch pings and traceroutes from any AS (not necessarily your AS), towards any destination in the mini-Internet. It will help you to

debug your configuration in part B and part C. To do so, you must first set up a connection with this VM via the interface **mgt** of the router **HOUS**. You must configure the IP address for this interface using X.0.199.1/24, where X is your AS number. For example, AS 5 must set the IP address to 5.0.199.1/24.

You can access the management VM with the following command.

> ssh -p 3099 root@zeus.winet.dcc.ufmg.br

The password will be announced during the class hackathon.

Inside the management VM, there are virtual interfaces that connects to other VMs. These interfaces are named gX, with X the AS number they are connected to. For example, the virtual interface connected to the AS 11 is named g11. You can see all interfaces using **ifconfig**.

With this management VM, you are able to launch pings and traceroutes from any group to any other group using **nping**. Nping is a very powerful networking tool as it gives you the ability to craft almost any field in packet headers. In your case, when you will launch a ping or a traceroute, you will have to specify the source IP address, the destination IP address, the destination MAC address, the interface to use in the management VM, as well as few additional parameters to customize your measurements. We strongly recommend you to look at the nping manual page using man nping to understand all the possible options.

For example, it you want to launch a ping, from AS10, towards the host connected to NEWY in the AS20, you can use the following command.

> nping --dest-mac <42:4c:04:09:97:97> --interface <g10> --source-ip <10.0.199.2> --dest-ip <20.101.0.1> -v0

Please always use **X.0.199.2** (IP of interface gX) as the source IP, with X the AS number you wish to issue pings/traceroutes from. For the destination MAC address, please use the MAC address of the interface mgt of the router HOUS in AS X. You can find the corresponding MAC address for each AS in ***mac_addresses.txt***, which is available under the home directory of the management VM.

To launch traceroutes, just add the parameter --traceroute (or --tr). For example, the following command launches a traceroute from AS10 towards the host connected to SALT in AS20.

> nping --dest-mac <42:4c:04:09:97:97> -- interface <g10> --source-ip <10.0.199.2> --dest-ip 20.107.0.1 -v0 --tr

If you want, you can also set the initial TTL value by yourself with the option **--ttl**.

4. Questions

This project is composed of four parts, including a class-wide “Internet Hackathon” being the second one. The first part involves setting up your OSPF and iBGP configuration, and must be finished before the Internet Hackathon. The second part (Internet Hackathon) involves bringing up your eBGP sessions with your neighboring ASes and advertising your prefixes to your neighbors. The third part will be done after the Hackathon, and involves implementing BGP policies according to the business relationships that you have with your neighbors. The fourth part builds on the third to answer questions and carry out experiments.

Part A (60 points)

Question 1 - (20 points)

Configure OSPF to enable end-to-end connectivity between all the hosts inside your AS. Before configuring OSPF, you will have to configure all the IP addresses for each interface of your routers and hosts (also remember to configure the interface connected to the management VM).

For routers, there are several interfaces we already created for you, so you only need to configure the correct IP address to corresponding interfaces using the commands in section 2.2. You can view all the existing router interfaces using the command **show interface** in the Quagga CLI. For hosts, you will also have to configure a default gateway. For example, if you are AS 1 and you want to configure the IP address and the default gateway for the host connected to NEWY router, you can use the following commands:

> sudo ifconfig newy 1.101.0.1/24
> sudo route add default gw 1.101.0.2

Make sure that each router and host can ping its directly connected router before you start the OSPF configuration. After you have configured OSPF for each router, test host-to-host connectivity with pings. Do not move on to other parts of the assignment before you verify that every host can ping every other hosts.

In the report, please paste the screenshot of ping results from NEWY-host to SEAT-host. You can use the command “**ping <ip_address> -c 10”** to conduct 10 pings to the IP address specified). Note that you will also be graded based on connectivities other than the one between these two hosts.

Question 2 - (8 points)

Assign the OSPF weights based on Figure 1. Use traceroute to verify that the paths with smallest costs are used. In the report, please paste traceroutes results from NEWY-host to SEAT-host, and explain how the result suggests that you had configured OSPF weights properly. Note that you may also be graded based on results other than the one between these two hosts.

Question 3 - (12 points)

Assume that a huge amount of traffic is continuously sent from SEAT to NEWY, producing congestion in some of the links between these two routers. With OSPF, the shortest-path is used to reach a destination. Yet, if multiple shortest-paths exist, routers will automatically split traffic across all of them. This technique, which is called Equal Cost Multi-Path (ECMP), is often used by operators in practice.

Please compute new OSPF weights and configure them on the links such that ECMP is used between NEWY and SEAT in order to avoid the congestions. Write in the report about what changes you have made (including what specific weights you configured and why), as well as the evidence (with explanation) that suggest your modification effectively split the traffic on different paths. (Hint: consider issuing several traceroutes from SEAT-host to NEWY-host.)

Question 4 - (20 points)

Configure internal BGP sessions (iBGP) between all pairs of routers (full-mesh). Please use the interface that connects to the host when specifying BGP neighbors. You might need to specify using this particular interface when sending announcements to other BGP routers. (Hint: there is a command to do so.) Verify that each of your routers does have an iBGP session with all the other routers with the command **show ip bgp summary**.

For each iBGP neighbor, please also add this command and think about why it is essential for iBGP neighbors. We will discuss this more in part B.

router_name(config-router)# neighbor <ip_address> next-hop-self

In your report, please paste the screenshot of your **show ip bgp summary** results from NEWY router. Also, describe what commands you used to specify a particular router interface to send BGP announcements.

Part B (40 points)

Question 5 - (10 points)

Attend the in-class Hackathon.

Question 6 - (10 points)

A file *as_connections.txt* (under the home directory of your VM) shows all the connections that should be implemented between ASes. Each connection record shows the type of the connection, which router to use to connect to a neighboring AS, and the subnet that should be used for this connection.

AS_type	connection_type	local_AS	remote_AS	peering_location	subnet
NotTier1	Prov1	10	15	NEWY	179.24.38.0/24
NotTier1	Prov2	10	9	SEAT	179.24.39.0/24
NotTier1	Peer1	10	5	WASH	179.24.35.0/24
NotTier1	Peer2	10	7	SALT	179.24.22.0/24
NotTier1	Cust1	10	2	KANS	179.24.40.0/24
NotTier1	Cust2	10	8	LOSA	179.24.41.0/24
NotTier1	Mgnt	10	100	HOUS-MGT	10.0.199.1/24

Table 1: An example of what you can find in *as_connections.txt*

Table 1 shows some connection records related to AS 10 in *as_connections.txt*. The records suggest that AS 10 is a tier-2 network, and has two providers (AS 15 and AS 9), two peers (AS 5 and AS 7), and two customers (AS 2 and AS 8). AS 10 is connected to AS 15 via its router NEWY. Between AS 10 and AS 15, the subnet 179.24.38.0/24 must be used.

You must negotiate with your neighboring ASes to decide who takes which IP address in the subnet used to connect your AS and your neighboring AS. After that, you have to configure the interface ***ebgp_peer*** with the IP address you took. For example, AS 10 can configure the *ebgp_peer* interface at NEWY using 179.24.38.1, and AS 15 can use 179.24.38.2 after negotiation.

Ping the address taken by your neighboring AS from your peering router to test the connectivity after both ASes have configured the *ebgp_peer* interface.

Question 7 - (5 points)

Configure the external BGP sessions (eBGP) with your neighboring ASes. Use the IP address of *ebgp_peer* interface when specifying a BGP neighbor.

Verify that eBGP sessions are established successfully using command **show ip bgp summary**.

Question 8 - (15 points)

Once the eBGP sessions are up, advertise your prefix to your peers. You must *only* originate the /8 that has been assigned to you.

In the meantime, your peers should advertise to you their /8 prefix, as well as all the /8 prefixes they have learned from other ASes (since there are no BGP policies, yet). Verify that you indeed received those advertisements with **show ip route bgp** or **show ip bgp**, and that those advertisements have been correctly propagated through your iBGP sessions.

Test your connectivity from your hosts towards hosts in other ASes using ping and traceroute.

Important. Double check that you have added **next-hop-self** command when configuring iBGP sessions if you can reach destinations in another AS from a peering router, but not from a non-peering router. The advertisements from external networks will have the next-hop attribute set to the IP address belonging to the subnet connecting your AS and the other AS, and only one of your routers knows how to reach that address. Thus, this router needs to use **next-hop-self** to modify the next-hop attribute, so that other routers know that they can send traffic to this router if they want to reach the destinations in the external advertisement.

Part C (70 points)

Question 9 - (30 points)

Configure your local-preference as well as the exportation rules to implement your customer/provider and peer/peer business relationships with neighboring ASes. You should implement (1) no-valley routing and (2) prefer-customer routing. No-valley routing specifies that you should not advertise routes learned from one provider/peer to another provider/peer, and prefer-customer routing specifies that if you can reach a destination through multiple AS-paths, you should always prefer the paths that go through your customers first, then peers, then providers. If you have paths through multiple neighbors of the same class, prefer-customer does not specify which to choose.

Hint. To implement no-valley routing, we advise you to use BGP communities to keep track of where the routes have been learned, and set up exportation rules based on BGP communities. You can use other BGP attributes as well if you like. However, it is a requirement that all best routes that would not cause a valley should be announced. Similarly, all the routes being announced shouldn’t result in a valley. Also, your solution should not depend on the actual prefixes announced by your neighbors.

To test that your BGP policies work correctly, you can use the management VM to launch traceroute from another AS. For example, you can launch a traceroute from one of your peers towards another peer, and verify that your AS does not provide transit service between these two peers.

In the report, describe (1) how you use BGP attributes to implement no-valley routing and prefer-customer routing, (2) the actual router configuration you made, (3) evidences to support that you have implemented these two policies correctly (4) explanation about how to interpret the evidence.

Question 10 - (20 points)

For this question, you will configure BGP policies in order to conduct traffic engineering on the outbound traffic. To optimize your resources, you want to split your outbound traffic between at least two of your peers (or providers).

For tier-1 networks: For some prefixes that are reachable through more than one peers, force your routers to send part of the traffic to one of your peers, and another part of the traffic to another peer. Regardless of which router you use, traffic to the same destination IP address should go through the same peer.

For other networks: For some prefixes that are reachable through more than one provider, force your routers to send part of the traffic to one of your providers, and another part of the traffic to another provider. Regardless of which router you use, traffic to the same destination IP address should go through the same provider.

Hint. AS-path length can change outside the control of your AS, so the way you split your traffic between the two peers (or providers) should not be based on the AS-path length--it should work regardless of changes other ASes make.

Launch traceroutes from your AS towards different destinations (reachable from both peer/providers) and verify that your configuration works well.

In your report, please describe (1) the prefixes you choose to conduct traffic engineering on, (2) how you plan to split traffic to these prefixes across peers/providers, (3) how you used BGP attributes to implement such outbound traffic engineering, (4) the actual router configuration you made, (5) evidences to support that you have implemented these two policies correctly, and (6) explanation about how to interpret the evidence. You should only use one single method to conduct outbound traffic engineering, even if you apply that method consistently to multiple prefixes.

Question 11 - (20 points)

For this question, you will configure BGP policies in order to conduct traffic engineering on the inbound traffic destined to your own prefix.

For tier-1 networks: Configure BGP policies such that the inbound traffic destined to your own network through peering links to come in priority from NEWY, then SEAT, then SALT or WASH.

For other networks: Configure BGP policies such that the inbound traffic destined to your own network through provider links uses the provider connected to NEWY in priority.

You can use BGP attributes to express your route preference to other networks. However, do not setup any exportation rules to hide links. For example, only advertising paths in NEWY is not a proper solution.

You can test your configuration by launching traceroutes using the management VM. Consider launching traceroutes from a AS which is reachable only via your peers (if you are a tier-1) or your providers (if you are not a Tier1) towards an IP belonging to your prefix, and see if the traceroute does use the correct peer or provider. For example, if you are AS 10, one way to test the configuration is by issuing traceroutes from AS 4, since it connects to both providers of AS 10, but not to AS 10 directly.

Hint. It is possible that other ASes implement outbound traffic engineering as required in question 10. In these cases, traceroutes from them may also be influenced by their outbound traffic engineering in addition to your inbound traffic engineering. However, keep in mind that other ASes will not conduct traffic engineering to their customers. Also, the ASes that are controlled by TAs (AS 4, AS 10, AS 18, AS 19, AS 20) will not implement any traffic engineering. You should always be able to find at least one AS to issue traceroutes from without worrying about its outbound traffic engineering policy.

In your report, please describe (1) how you used BGP attributes to implement such inbound traffic engineering, (2) the actual router configuration you made, (3) evidences to support that you have implemented these two policies correctly (4) explanation about how to interpret the evidence. For (3), if you use traceroutes from different ASes to test your configurations, please make sure to include which ASes you choose, why did you choose them, and the traceroute results.

Part D (30 points)

Note: The routing policies of different ASes can impact results for this section. Therefore:

- We will make an announcement once we have verified that all students have implemented their routing policies (Part C) sufficiently to answer Question 12 correctly.

- We will make a second announcement once we verified that all policies are implemented correctly to answer Question 13 consistently. We will also make an announcement if any ASes change their policies as a result of us discovering errors.
- If you make a change to routing policies in your AS that could impact the routes that classmates may observe, you are required to announce the change on the discussion forum. Please do not make changes unless required for correctness.

Question 12 - (15 points)

Other than the ASes shown in Figure 2, there is a hidden network (AS 21) that is also connected to some of the ASes. You can use traceroutes from different ASes toward this hidden AS to infer the relationship between it and other ASes. You can also examine the BGP announcements you received as additional information. AS 21 AS and all other TA ASes have implemented no-valley and prefer-customer routing policies.

In your report, please describe (1) what are the ASes that connect to the hidden network, (2) what are the business relationship between the hidden AS and its neighbors, (3) what leads you to your answers in (1) and (2). To get full credit, you must indicate all the ASes that are connected to the hidden network. Also, notice that this hidden AS may not fall perfectly into tier1/tier2/tier3/stub categories.

Question 13 - (15 points)

For this question, you will study the impact of prefix hijacking. You are allowed to hijack one or more prefixes within the subnets 4.X.0.0/16, 10.X.0.0/16, 18.X.0.0/16, 19.X.0.0/16, 20.X.0.0/16, with X being your AS number. AS 4/10/18/19/20 will also advertise the corresponding /16 prefixes at the same time; they are considered to be the valid owners of the prefixes.

Important. *Do not* hijack any prefixes that is not listed above. You will receive penalty on the project if you do that.

Use traceroutes from different ASes to the destinations that you hijacked to verify that you can intercept some traffic successfully.

In your report, please describe (1) what are the prefixes that you hijacked and where do you make these hijacking announcements, (2) the evidences to support that you have successfully hijacked some traffic, (3) explanation about how to interpret the evidence. Also, discuss (4) whether your hijacking attack captures the traffic to the hijacked prefix(es) from all ASes. If yes, why is that? If no, what are the cases you fail (illustrate with real traceroute examples)? What are the cases you succeed? Be sure to list every AS as either failed or successful. And what is the reason for such difference?

5. Submission and other information

This assignment 1 is worth 50 points (Part A: 15 points, Part B: 5 points, Part C: 20 points, Part D: 10 points).

Part A:

You must include the following files in a compressed file called **project1a_Lastname_Firstname.zip**. The files are:

1. Written report, with filename **report.pdf**
2. The entire **configs** directory on your VM under home directory.

Part B:

No submission required. Please attend the in-class Hackathon.

Part C:

You must include the following files in a compressed file called **project1c_Lastname_Firstname.zip**. The files are:

1. Written report, with filename **report.pdf**
2. The entire **configs** directory on your VM under home directory.

Part D:

Please submit the written report with filename **project1d_report_Lastname_Firstname**.pdf.

Questions and office hours

Please ask any general questions related to this assignment on Moodle, and only contact the TAs through mail for personal questions (e.g., unable to access the VM, issues about grading).

For assignment-related office-hours, please see the teacher either directly after class, Monday/Wednesdays 13:00-13:45, or Tuesday/Thursdays 9:00-9:45.

Academic integrity: Zero tolerance on plagiarism

If you have any questions about academic integrity for the course, please ask the professor before submitting the project, with enough time to resolve the issue before the deadline.

Please note that this class requires closely obeying the policy on academic integrity, and has zero tolerance on plagiarism for all assignments, including both programming assignments and paper responses. By zero tolerance, we mean that the minimum punishment for plagiarism is an "F" for this class. For programming assignments, in particular, you must write all the code you hand in, except for code that we give you as part of the assignments. You are not allowed to look at anyone else's solution (including solutions on the Internet, if there are any), and you are not allowed to look at code from previous years. You may discuss the assignments with other students at the conceptual level, but you may not write pseudocode together, or look at or copy each other's code. Please do not publish your code or make it available to future students -- for example, please do not make your code visible on Github. You may look at documentation from the tools' websites. However, you may not use external libraries unless granted explicit permission by the professor or TA.

For each programming assignment, we will use software to check for plagiarized code. **So, be really careful and do not copy-and-paste code or text.**