COM-407: TCP/IP NETWORKING

LAB EXERCISES (TP) 6
INTER-DOMAIN ROUTING: BGP-4
With Solutions

December 15th, 2022
Deadline: January 4th, 2023 at 23h55

Abstract

This lab covers BGP-4, which is the Inter-Domain Routing Protocol of the Internet. You will be asked to configure peering between dual-stack edge routers, to create filters for the advertisements exchanged by the peers and to set up some simple routing policies.

Similar to the previous labs, you will use the virtual environment to configure a network setup that represents a simplified model of the Internet. This model is constructed by interconnecting the networks of three small Internet Service Providers (ISPs).

1 INTRODUCTION

1.1 BRIEF ORGANIZATION

For this lab, it is assumed that you have mastered the configuration of network interfaces with *zebra* and the configuration of OSPF with *ospfd* and *ospf6d* daemons.

In this document, we immediately start with the basics of BGP configuration using FRR's *bgpd* daemon. In Section 3, we cover the fundamental concepts and commands used to configure the exchange of IPv4 and IPv6 routes between different autonomous systems (AS's). In the subsequent section, we will let you do a case study on basic but important BGP operations and properties. In Section 5, we deal with policy routing. In particular, we consider the design of filters that allow us to accept or deny a route based on its properties. In the two last sections, we give you some other basic case studies on BGP operations and properties.

1.2 SUBMISSION AND DEADLINE

Just like in the previous labs, you are expected to answer the questions directly in Moodle.

This lab is designed to be done in one week. As a consequence, it will count for half a lab in the labs grading at the end (idem for the bonus research exercise). However, the deadline is already extended until January of next year, although we do recommend starting as soon as possible so you benefit from the exercise session on Friday.

The deadline is **January 4th 2023, 23:55.**

2 NETWORK SETUP

For this lab, we will construct a network of 3 hosts, 5 routers and 7 ethernet switches, as given in Figure 1. **The figure is placed in the end of this document. You will need to consult this figure/topology throughout the lab, therefore, it is wise to keep it open separately on the computer or print it on a paper.**

Note that the python simulation file and the FRR configuration files are already provided in the package on Moodle. Please download it and unzip it **in the Desktop of your virtual machine**. You can choose to unzip it at some other location on your virtual machine (not in the shared folder!!) but all the paths in the scripts and config files need to be changed accordingly. For a hassle-free start, we recommend you to extract the zip on the Desktop of your VM (as you did in lab 4).

3 BASIC BGP CONFIGURATION

As shown in Figure 1 (present on the last page of the lab), the network consists of three autonomous systems (AS 65345, AS 65100 and AS 65200). Specifically, AS 65345 contains two edge routers (R3 and R4) and an internal router (R5). And each of AS 65100 and AS 65200 has a single router. Please note the numbering of autonomous systems (ASs): number 65 in the beginning of all AS numbers correspond to private ASs, "345" in AS 65345 tells that routers R3, R4, and R5 are present in this AS, router R1 is present in AS 65100, and router R2 is present in AS 65200. We hope this numbering style makes it easy for you to remember the routers in a particular AS.

3.1 WARMING-UP REMINDERS

By reading all the provided files, you will notice that:

1. The passwords for *zebra*, *ospfd*, *ospf6d*, *bgpd* processes are set in corresponding configuration files;
2. All the above processes (if enabled in `Lab6_Network.py` script) will automatically start whenever you run mininet;
3. When exiting mininet, the log files are not automatically erased. In order to avoid issues, the provided python script first removes all present log, pid and api files before creating the topology.

Before you continue, make sure that you know how to

1. Check routing tables, and link state databases;
2. Bring up and shut down interfaces on-the-fly.

3.2 FRR'S BGPD PROCESS

Just as *ospfd* and *ospf6d* processes allow you to configure OSPF routing, the FRR's *bgpd* process allows you to configure BGP routing. Again, the preferred way of configuring BGP is to use the configuration files. "On-the-fly" configuration is also possible, but not advised, as you are prone to make mistakes when using this method.

The key difference between the *bgpd* process and the other two processes (*i.e., ospfd* and *ospf6d*) is that *bgpd* is dual-stack. This means that you only need one instance of *bgpd* to support both IPv4 and IPv6 routing. You do not need to start two separate processes (daemons).

However, although a single instance of *bgpd* is sufficient to support both IPv4 and IPv6 routing, we need two TCP connections between our dual-stack peers in order to exchange IPv4 and IPv6 updates (prefixes). In the next section we explain how this dual-stack peering should be configured.

3.3 BGPD CONFIGURATION FILE EXAMPLE

The main configuration steps that must be performed in order to run BGP on a router are:

- enabling BGP routing;
- declaring peers;
- choosing certain prefixes to redistribute into BGP;
- configuring policy routing (by setting up filtering rules and/or changing the path attributes).

Let's have a closer look at the first three bullets. The fourth bullet is covered in Section 5 of this lab. An example of the configuration file for a *bgpd* process is shown as following.

```
!bgpd configuration file example
!
hostname r1
password bgpd
enable password bgpd
!
log file /home/lca2/Desktop/lab6/logs/bgpd_r1.log
debug bgp updates
!debug bgp keepalives
!debug bgp events
!
router bgp 65100
network 192.10.10.0/24
neighbor 192.13.13.3 remote-as 65345
no bgp default ipv4-unicast
neighbor 2001:1:0:1313::3 remote-as 65345
!
address-family ipv6
network 2001:1:0:1010::/64
neighbor 2001:1:0:1313::3 activate
exit-address-family
```

Let's examine the commands used in the example above:

- `router bgp <as-number>`: enables BGP routing, and specifies the AS number of the domain that the router belongs to (in the example above, the AS number is 65100).
- `network 192.10.10.0/24`: instructs the *bgpd* process on the local router to announce the prefix `192.10.10.0/24` to all BGP peers. The command has the same syntax for both IPv4 and IPv6 prefixes (see the line `network 2001:1:0:1010::/64`). If instead of a particular subnet (prefix), we want to announce all directly connected subnets via BGP, then we can use the command `redistribute connected`. If we want to redistribute into BGP the routes learned via another routing protocol, or statically configured routes, we use `redistribute <protocol>`, where `<protocol>` can be set to `rip`, `ospf` , or `static`, depending on the source of the routes.

- `neighbor <ip address> remote-as <as-number>`: is used to declare a BGP peer (that belongs to the same or a different AS). A TCP connection is established between the local *bgpd* process and the peer. This connection is used to exchange keepalives and BGP routing updates. The command `neighbor <ipv6 address> activate` activates the exchange with the declared IPv6 peer. Note that no distinction between I-BGP and E-BGP is made. This information is learned automatically from the AS numbers.
- `no bgp default ipv4-unicast`: By default the IPv4 prefixes (also called the IPv4 family prefixes) are exchanged, both between the IPv4 peers and between the IPv6 peers. When this command is used before the peer declaration (*i.e.,* before the `neighbor` command), it prevents the exchange of IPv4 prefixes via the TCP connection established between the BGP IPv6 peers.
- `address-family ipv6` and `exit-address-family`: these two commands enclose the block of commands that configure the IPv6 communication with the declared IPv6 peer.
- `debug bgp updates`, `debug bgp keepalives` and `debug bgp events`: debugging of the data exchanged between BGP peers.

The set of commands used in the configuration file above is just a subset of the BGP configuration commands offered by the FRRouting suite. For a complete reference, check the documentation on the FRR website (http://docs.frrouting.org/en/latest).

The *bgpd* configuration files for the routers R2, R3 and R5, are already provided to you in the lab folder you can download from Moodle. By looking at the above example as well as the provided files, you will be asked to write the *bgpd* configuration files for the routers R1 and R4. Also, you will be asked to create the missing *zebra* configuration files for the routers R2 and R3.

Please note that, depending on the exercise, the *bgpd* configuration files may require certain modifications. In some of the exercises, the *bgpd* processes will not be running on all routers. Also, some of the exercises require you to implement policies. **When creating or editing a configuration file, please do it in the virtual machine using a text editor, *e.g.,* leafpad.**

3.4 BGP Monitoring Commands

As you may already know, in order to monitor the activity of a running FRR process, you can enter the FRR configuration mode by typing `telnet localhost bgpd` with password `bgpd`.

The commands you will be using to visualize the BGP database at the router are `show bgp ipv4` and `show bgp ipv6`. Note that the BGP database is the same as BGP RIB-In with the best route to each destination marked by ">".

4 Standard BGP mode of operation

4.1 BGP Running on All Routers

Create the *zebra* configuration files for the routers R2 and R3 (you can get inspiration from the given zebra configuration files for the other routers). Once you have finished creating the configuration files, run the python script `Lab6_Network.py` with python3 in order to start the network simulation with *zebra* processes on all routers. (**Recall that you need to ensure that the zebra processes for all routers are enabled in the python script** `Lab6_Network.py`.)

Now, verify that the ip configurations of routers R2 and R3 match with the topology in Figure 1. You can use `ip addr show` on routers R2 and R3 to verify the proper configurations of their network interfaces.

If the network configurations do not match with the ones shown in the topology/figure, please check if your zebra configuration files are properly written.

Once network interfaces are properly configured, please stop the mininet simulation by typing `exit` in the mininet prompt and proceed with creating the *bgpd* configuration files for the routers R1 and R4. Each router should redistribute the directly connected subnets into BGP. Once you have finished creating the configuration files, run the python script `Lab6_Network.py` with python3 in order to start the network simulation with both *zebra* and *bgpd* processes running on all routers. (**Recall that you need to enable zebra and bgpd processes for all routers in the python script** `Lab6_Network.py`.)

Please note that you might need to wait around one minute until convergence of a routing protocol. Afterwards, you can check the connectivity of the network.

As all hosts and routers are properly configured, you should now be able to reach every host and router in the network. Please verify that using `pingall` command in the mininet prompt. If you cannot reach all hosts now, there is some problem with the configuration of bgpd config files. Please try to correct that yourself and in case of problems, please contact a TA.

Q1/ Answer **Lab6 Part 1 Q1** on Moodle.

The databse st R5 is:

```
   Network          Next Hop        Metric  LocPrf  Weight  Path
*>i192.20.20.0/24   192.24.24.2     0       100     0       65200 ?
```

Solution. *Next hop for 192.20.20.0/24 is 192.24.24.2*

Q2/ Answer **Lab6 Part 1 Q2** on Moodle.

The databse st R5 is:

```
   Network          Next Hop            Metric  LocPrf  Weight  Path
*>i192.12.12.0/24   192.13.13.1              0     100       0  65100 ?
*  i                192.24.24.2              0     100       0  65200 ?
```

Solution. *192.13.13.1 and 192.24.24.2 are listed as next hop for 192.12.12.0/24.*

The selected next hop (symbol ">") is 192.13.13.1. Both lines both have the same AS path, are learned via I-BGP (symbol "i"), both have the same distance to next-hop according to IGP so the remaining criteria to select the preferred next hop is here the lowest BGP identifier.

Start Wireshark on R2 and observe the BGP packets exchanged between R2 and its BGP peers.

Q3/ Answer **Lab6 Part 1 Q3** on Moodle.

Solution. *KEEPALIVE messages are used to (periodically) check if the BGP peers are still reachable and active. By observing the BGP packets exchanged between BGP peers, we know that they are sent every 60 seconds.*

4.2 R1 AND R2 NOT BGP PEERS

For this part of the lab, **we will modify the *bgpd* configuration files of R1 and R2** so that AS 65100 and AS 65200 won't be BGP neighbors *i.e.,* R1 and R2 don't exchange BGP routing updates with each other. After you do the necessary modifications, restart the network.

Q4/ Answer **Lab6 Part 1 Q3** on Moodle.

Solution. *All routers can reach all subnets because the network is not partitionned.*

Q5/ Answer **Question 5 in Lab6 Part 1** on Moodle.

Solution. *In BGP database at R1: 192.20.20.0 (Network) 192.13.13.3 (Next Hop) 0 (Weight) 65345 65200 (Path); In BGP database at R2: 192.10.10.0 (Network) 192.24.24.4 (Next Hop) 0 (Weight) 65345 65100 (Path). To reach 192.20.20.0/24, R1 should go through AS 65345 and AS 65200. To reach 192.10.10.0/24, R2 should go through AS 65345 and AS 65100.*

For IPv6, everything is similar to that of IPv4.

5 Policy Routing

5.1 Filtering BGP Updates (Access List)

Filtering BGP updates allows a router to accept some of them and reject the others. A filter can be applied:

- to the BGP updates coming from a BGP peer, before they are added to the BGP database of the router;
- to the updates sent to a BGP peer, before they are actually sent.

Filtering can be based **on *prefix*** or **on *as-path* attribute** (an AS number or a sequence of AS numbers).

In the case of filtering by *prefix*, the `distribute-list` type of filter is used. The syntax is:

```
neighbor <ip_address> distribute-list <filter_name> in/out
```

- `<ip_address>`: is an IPv4 or IPv6 address of the BGP peer,
- `<filter_name>`: is the name of the filter, and
- `in/out`: specifies whether the filter is applied to the input or to the output of the router.

In the case of *as-path* based filtering, the `filter-list` type of filter is used. The syntax is:

```
neighbor <ip_address> filter-list <filter_name> in/out
```

To apply a filter to the updates sent by a peer, you must define it first. A filter is defined with a list of rules, which is called the `access-list`. The syntax to define an `access-list` rule differs, depending on whether the filtering is performed by *prefix* or by *as-path*.

The syntax to define an `access-list` rule in the case of filtering by *prefix* is the following:

```
(ipv6) access-list <filter_name> permit/deny <prefix>
```

- `ipv6`: this part of the command exists only in case of an IPv6 prefix
- `<filter_name>`: the name of the filter: it can be composed of letters and/or numbers,
- `permit/deny`: specifies whether the interface permits or denies the updates that match the prefix,
- `<prefix>`: the prefix the filter should match. It can be replaced by `any`, in which case the filter applies to any advertised prefix.

The syntax to define an `access-list` rule in the case of filtering by *as-path* is the following:

```
ip as-path access-list <filter_name> permit/deny <as_number_sequence>
```

- `<filter_name>`: is the name of the filter; it can be composed of letters and/or numbers,
- `permit/deny`: specifies whether the interface permits or denies the updates that match the AS path,
- `<as_number_sequence>`: is the sequence of one or more AS numbers. It can be substituted by `.*`, in which case it applies to any AS path.

When a filter is applied to a network interface, each BGP update that passes through the interface (in the indicated direction) is checked against the list of rules that constitute the filter. The rules are checked in the same order in which they appear in the configuration file. If a matching rule is found, the subsequent rules are not checked. **If no matching rule is found, the update is dropped (implicit deny).**

If this sounds a bit complicated, here is an example to help you:

```
router bgp 65101
bgp router-id 192.45.88.3
network 192.45.88.0
neighbor 200.44.0.13 remote-as 65131
neighbor 200.44.0.13 distribute-list Alpha_1 in
neighbor 200.44.0.12 remote-as 65121
neighbor 200.44.0.12 filter-list Beta_2 out
!
access-list Alpha_1 permit 192.168.33.0/24
access-list Alpha_1 permit 133.33.23.0/24
!
ip as-path access-list Beta_2 permit 65121
ip as-path access-list Beta_2 permit 65111 65131
```

In the example above, the filter `Alpha_1` is an input filter. It accepts the updates for prefixes 192.168.33.0/24 and 133.33.23.0/24, from the BGP peer at address 200.44.0.13 in AS 65131.

The filter `Beta_2` is an output filter. It allows the updates that contain AS 65121 or the AS sequence 65111 65131 (in this order and at an arbitrary position inside the AS path) to be sent to the BGP peer at address 200.44.0.12 in AS 65121.

Note:

- A filter (like `Alpha_1`) has to be assigned to an interface (like for bgp `neighbor 200.44.0.13`) before the filter definition (like `access-list`);
- In the case of IPv4, a filter should be assigned to an interface by declaring the filter just after bgp neighbor;
- In the case of IPv6, a filter should be assigned to an interface by declaring the filter just after the interface activation inside the *address-family ipv6* block;
- For IPv4 and IPv6, please give detailed filter definitions after the *address-family ipv6* block.

5.2 Policy Routing Playground

5.2.1 Avoid Transit Traffic

Recover the configuration in Section 4.1 (recall that this means that zebra and bgp run on all routers and that R2 and R1 are BGP peers) and imagine that AS 65200 in Figure 1 represents an enterprise network of the Foot Clan, which is connected to the Internet via two ISPs—AS 65100 and AS 65345. You are the administrator of this enterprise network and you do not want your autonomous system AS 65200 to become a transit AS, *i.e.,* you do not want the traffic that neither originates nor terminates in AS 65200 to be routed through this AS. At the same time other autonomous systems have to know about all the prefixes in AS 65200.

Q6/ Answer **Lab6 Part 2 Q1** on Moodle.

Solution. *For R2:*
neighbor 192.12.12.1 distribute-list advertise-own-ipv4 out
neighbor 192.24.24.4 distribute-list advertise-own-ipv4 out
access-list advertise-own-ipv4 permit 192.20.20.0/24
neighbor 2001:1:0:1212::1 distribute-list advertise-own-ipv6 out
neighbor 2001:1:0:2424::4 distribute-list advertise-own-ipv6 out
ipv6 access-list advertise-own-ipv6 permit 2001:1:0:2020::/64

Q7/ Answer **Lab6 Part 2 Q2** on Moodle.

Solution. *We can ping h5 from h2. We can ping h5 from h1.*

Now shutdown the interface eth1 at R1. Use the commands you have already seen in Lab 4 to shut down the interface. Namely:

```
telnet localhost zebra
enable
configure terminal
interface r1-eth1
shutdown
```

Q8/ Answer **Lab6 Part 2 Q3** on Moodle.

Solution. *We cannot ping h5 from h1 but we can ping h5 from h2. The link between AS65100 and AS65345 is broken, and transiting by AS65200 is forbidden due to filtering rules, therefore there is no route to go from A65100 to AS65345.*

6 RUNNING BGP ONLY ON EDGE ROUTERS (BONUS)

Let us come back to the configuration of Section 4.1 (**don't forget to remove the filters you applied in Section 5**). In Section 4.1, BGP runs on all three routers of AS 65345. We now want to make the choice that BGP should only run on routers that are connected to the outside world. Therefore, we ask our friends Mikey and Don to come up with an alternative solution. Mikey proposes the following solution.

Mikey believes that we can avoid a full-mesh I-BGP by running BGP only on edge routers, run OSPF on all routers inside an AS, and redistribute BGP into OSPF on edge routers running both OSPF and BGP. Hence he wants you to test his solution by simply running BGP on edge routers and redistributing BGP into OSPF on these routers. To test Mikey's initial solution, make the following change:

- Redistribute BGP into OSPF in R3 and R4, by modifying the ospf and ospf6 config files.

After making these changes, restart the network with *zebra* on all routers, *bgpd* on routers R1, R2, R3 and R4, and *ospfd* and *ospf6d* on all routers in AS 65345 (*i.e.,* R3, R4, and R5).

Q9/ Answer **Question 1 in Lab6 Bonus Part 1** on Moodle.

Solution. *B 192.20.20.0/24 [200/0] via 192.24.24.2, 00:00:21*
*O > * 192.20.20.0/24 [110/20] via 192.34.34.4, r3-eth2, 00:00:21*
The entry that is learned through OSPF is chosen as the best one. This is because the administrative distance is 110 for OSPF but 200 for I-BGP.

Q10/ Answer **Question 2 in Lab6 Bonus Part 1** on Moodle.

Solution. *No. Subnets 192.50.50.0/24 and 2001:1:0:5050::/64 are not reachable from R1 and R2.*

From your experiment, you must have observed that if an AS has subnets that are not directly connected to an edge router (BGP speaker), routers in other AS's won't be able to route packets to such "hidden" subnets.

Mikey has also observed this, he therefore asks Don for help. Don decides to use the `network <prefix>` command on the edge router's *bgpd* configuration files to instruct the edge routers to include the prefixes of "hidden" subnets in their BGP routing updates. He considers this as the best solution because all AS's know the list of prefixes in their network and should not be difficult for them to use network commands in the edge routers for all such prefixes.

Don wants you to test his solution and see if it works. To test Don's solution,

- Modify the *bgpd* configuration files of R3 and R4 such that you use `network <prefix>` command to advertise the 192.50.50.0/24 and 2001:1:0:5050::/64 subnet in their BGP routing updates.

Restart the network with *zebra* on all routers, *bgpd* on routers R1, R2, R3 and R4, and *ospfd* and *ospf6d* on all routers in AS 65345 (*i.e.,* R3, R4 and R5).

Q11/ Answer **Question 3 in Lab6 Bonus Part 1** on Moodle.

Solution. *Yes. Two entries with next-hops 192.24.24.4 and 192.12.12.1. Entry with next-hop 192.24.24.4 is chosen because of shortest AS path.*

Comment: Instead of using the `network` command, it is possible to redistribute OSPF into BGP, using `redistribute ospf internal`. The effect is to make this router aware of all subnets that are visible in this node via OSPF and are internal (i.e. in the same AS) without having to specify which are such networks. This is therefore preferable to Don's method; unfortunately, it is not supported by the current version of FRR.

7 BROKEN LINK (BONUS)

In this section we observe how connectivity is affected if a link breaks in the network. Continue working with Don's configuration *i.e.,*

- use `network <prefix>` command in R3 and R4 to advertise the subnet 192.50.50.0/24 and the subnet 2001:1:0:5050::/64 in their BGP routing updates.
- Redistribute BGP into OSPF in R3 and R4.
- Run *bgpd* on R1, R2, R3 and R4.
- Run ospfd and ospf6d on all routers in AS 65345.

7.1 EXTERNAL BROKEN LINK

In this experiment, we will simulate a broken link between R4 and SW24 by shutting down the interface $eth1$ at R4. Recall that the commands to do this are:

```
telnet localhost zebra
enable
configure terminal
interface r4-eth1
shutdown
```

Though the BGP connection between AS 65200 and AS 65345 will be broken as a result of the link failure, the ring topology of the network will let BGP converge after some exchange of messages between the different BGP peers.

For the next question, you might need to inspect the BGP log files generated by *bgpd*. To open the files, you should either do it as root (`sudo leafpad <logfile_name>`) or enable the reading rights for the file (`sudo chmod +r <logfile_name>`).

Q12/ Answer **Question 1 in Lab6 Bonus Part 2** on Moodle.

Solution. *R4 sends UPDATE Messages to R3 when there is link failure. Its purpose is to withdraw routes to the following subnetworks: 1. 192.12.12.0/24; 2. 192.20.20.0/24; 3. 192.24.24.0/24. (Similar for IPv6.)*

Q13/ Answer **Question 2 in Lab6 Bonus Part 2** on Moodle.

Solution. *R4 takes the "Next Hop" as 192.34.34.3 at R3.*

7.2 INTERNAL BROKEN LINK

In the previous experiment, we saw what happens when a link connecting two AS's breaks in the presence of redundancy (*e.g.,* ring topology). In what follows, we will look at a pathological case where a link failure causes a disconnected AS. In order to do this experiment, first bring up interface $eth1$ of R4 that you shut down in the previous experiment (the command to do this is `no shutdown`). Then, simulate a broken link between R4 and SW345 by shutting down the interface $eth2$ at R4. (Again apply the same commands you used previously.)

Q14/ Answer **Question 3 and 4 in Lab6 Bonus Part 2** on Moodle.

Solution. *There are two routes:*
192.50.50.0 (Network) 192.24.24.4 (Next Hop) 0 (Metric) 0 (Weight) 65345 i (Path)
192.12.12.1 (Next Hop) 0 (Weight) 65100 65345 i (Path)
The first one is chosen as the best because of shorter AS path. However, we cannot ping this subnetwork 192.50.50.0/24 from R2. This is because the link R4-eth2 - SW345 is broken. Actually, R4 keeps advertising this route because it was put manually in the bgp configuration file so that R4 can advertise it. This is a serious drawback of manually putting the subnets for advertisement using the network command.

Q15/ Answer **Question 5 in Lab6 Bonus Part 2** on Moodle.

Solution. *No. This is because when R2 advertises to R4 the route to 192.34.34.0/24, R4 finds its own AS in the AS-path. So, in order to avoid a loop, R4 ignores the route.*

8 CONCLUSION

This is your final Lab for the TCP/IP course. We hope you enjoyed the Labs during the semester. **All the TA team members wish you happy holidays and good luck for your exams.**

A List of Figures

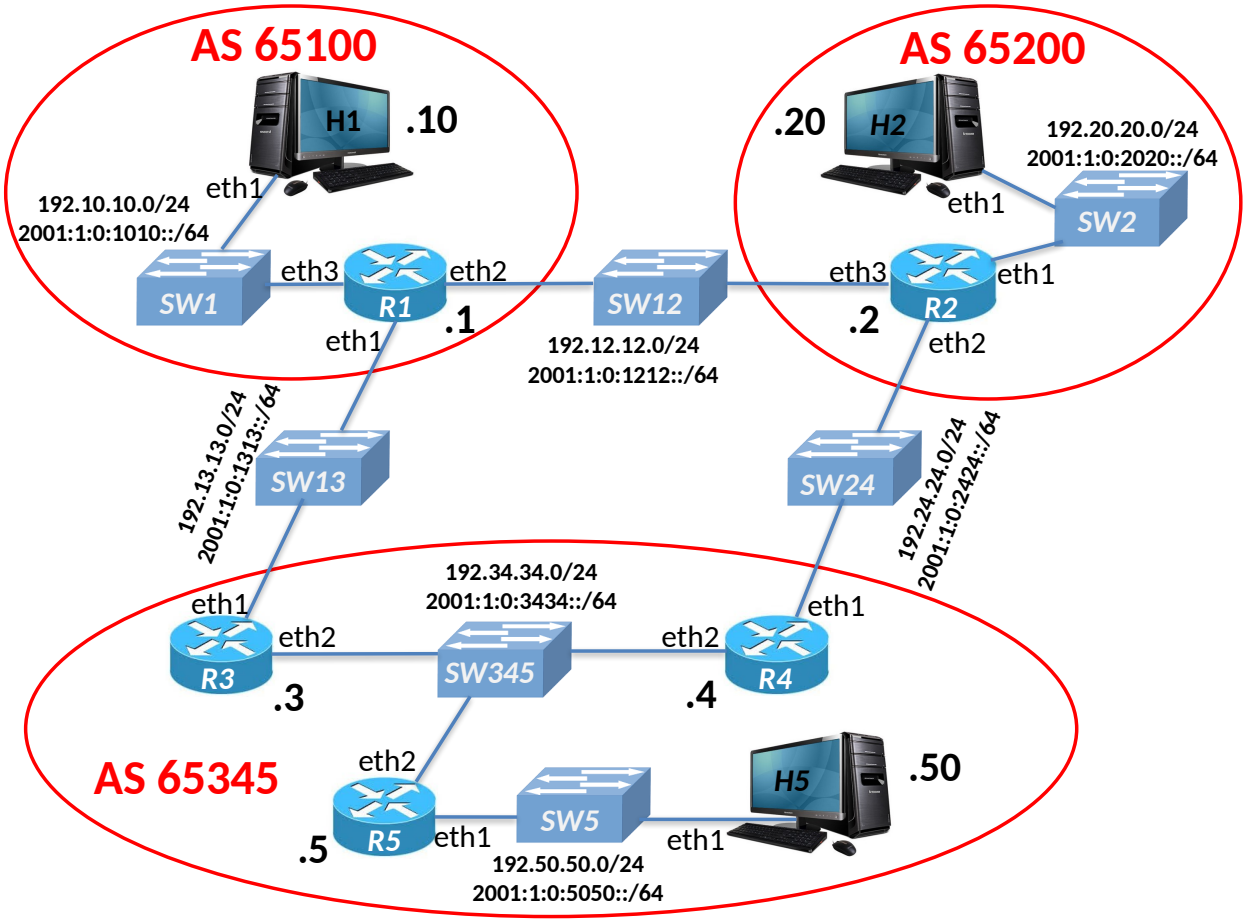

Figure 1: The network topology for this lab. It consists of 5 routers that belong to 3 autonomous systems.