

Name 1:  
Name 2:

---

# COM-407: TCP/IP NETWORKING

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

---

### 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 three subsequent sections, we will let you do case study on basic but important BGP operations and properties. In Section 7 we deal with policy routing. In particular, we first consider the design of filters that allow us to accept or deny a route based on its properties. Then, we introduce several advanced policy routing concepts, which permit you to modify the values of path attributes.

### 1.2 LAB REPORT

In the report, you are expected to answer the questions that accompany different parts of the lab. When answering the questions, fill in your answers directly in the spaces provided in this document. You should hand in one report per group. Don't forget to write your names on the first page of the report.

The deadline is **December 18, 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 python simulation file and 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 but all the paths in the scripts and config files need to be changed accordingly. For 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. Log files will not be cleaned after exit from `mininet`. All the log, pid, and api files will be checked and removed (if present) each you run the script.

Before continue, make sure that you know how to

1. Check routing tables, 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 7 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 /media/sf_share/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>).

On Moodle, the *bgpd* configuration files for the routers R1, R3 and R4, are already created for you. By looking at the above example as well as the provided files, you are expected to write the *bgpd* configuration files for the routers R2 and R5. Also, you are expected to create missing *zebra* configuration files for the routers R1 and R4.

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`.

Use `show bgp ipv4` and `show bgp ipv6` to visualize the BGP database at the router. 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 R1 and R4 (you can inspire from given *zebra* configuration files for other routers). Once you have finished creating the configuration files, run 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 enable zebra processes for all routers in python script `Lab6_Network.py`.)**

Now, verify that ip configurations of routers R1 and R4 match with the topology in Figure 1. You can use `ip addr show` on routers R1 and R4 to verify the proper configurations of their network interfaces. If 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 R2 and R5. Each router should redistribute the directly connected subnets into BGP. Once you have finished creating the configuration files, run 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 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/ Show the BGP database entries of R5, and look for the subnets 192.10.10.0/24, 192.12.12.0/24 and 192.20.20.0/24. How did R5 learn about these subnets? What is the next hop for 192.10.10.0/24 and why? For the network 192.12.12.0/24, what hops are listed as next hops and which hop is finally selected as the next hop? Please justify why this particular hop is selected as next hop. Can you somehow experimentally verify it? If yes, please explain what you did.

[A1]

Start Wireshark on R1 and observe the BGP packets exchanged between R1 and its BGP peers.



Q2/ What is the role of KEEPALIVE messages? Can you find the period of them? Explain briefly how to find.

[A2]

## 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.



Q3/ Can all routers reach all subnets? You may try pinging between hosts.

[A3]



Q4/ Observe the BGP database of R1 and R2. What are the entries for subnet 192.20.20.0/24 in R1 and for subnet 192.10.10.0/24 in R2? What are the AS paths for these entries? Furthermore, check also the corresponding subnets and AS paths in IPv6. Is there anything essentially different from IPv4?

[A4]

## 5 RUNNING BGP ONLY ON EDGE ROUTERS

In the previous section, 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 changes:

- Undo the changes you made above to *bgpd* configuration files of R1 and R2 so that R1 and R2 are now BGP peers.
- Redistribute BGP into OSPF in R3 and R4.

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).



Q5/ Observe the routing tables (via *zebra*) at R4. How many entries are there for the prefix 192.10.10.0/24? Which entry is chosen as the best route to this destination? Why? **You may need to wait around one minute before routing protocols stabilize.**

[A5]



Q6/ Can you ping between all hosts? Is there missing information at any router?

[A6]

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).



Q7/ By looking at the routing tables of the routers, can all routers reach all subnets in the network? How many entries are there in the BGP database of R1 for the subnet 192.50.50.0/24? Which entry is chosen as the best one? Why?

[A7]

**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.

## 6 BROKEN LINK

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.
- **DON'T** redistribute OSPF into BGP in R3 and R4.
- Run `bgpd` on R1, R2, R3 and R4.
- Run `ospfd` and `ospf6d` on all routers in AS 65345.

### 6.1 EXTERNAL BROKEN LINK

In this experiment, we will simulate a broken link between R3 and SW13 by shutting down the interface `eth1` at R3. Utilize the commands you have already seen in Lab 4 to shut down the interface. Namely:

```
telnet localhost zebra
enable
configure terminal
interface r3-eth1
shutdown
```


Though the BGP connection between AS 65100 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.



Q8/ In the log file of `bgpd` at R4, there are UPDATE messages received from R3 as a result of the link failure. What do they mean?




[A8]

 Q9/ How does R3 route packets to 192.10.10.0/24 subnet?

[A9]

## 6.2 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 R3 that you shut down in the previous experiment (the command to do this is `no shutdown`). Then, simulate a broken link between R3 and SW345 by shutting down the interface *eth2* at R3. (Again apply the same commands you used previously.)

 Q10/ Look at the BGP database of R1. How many entries are there for the subnet 192.50.50.0/24? Which entry is chosen as the best one? Can you ping from R1 to this subnet? Explain briefly the reason.

[A10]



11/ Will R3 learn the subnet 192.34.34.0/24 by BGP? Why or why not?

[A11]

## 7 POLICY ROUTING

### 7.1 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 that 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.

## 7.2 MODIFYING PATH ATTRIBUTES (ROUTE MAP)

Filtering is an important tool in the process of route selection. However, they only allow you to accept or reject a route. In order to take more sophisticated actions (*e.g.*, give different preferences to different routes), you have to use a *route map*.

A route map is a more powerful tool than a filter. In addition to accepting/rejecting a route, a route map allows you to modify its attributes. A route map is typically applied to a network interface in the *incoming* or *outgoing* direction. The syntax to do this is the following:

```
neighbor <ip_address> route-map <rm_name> in/out
```

- <ip\_address>: is an IPv4 or IPv6 address of the BGP peer,
- <rm\_name>: is the route map name (any sequence of characters and/or numbers),
- in/out: specifies whether the route map is applied in the incoming or in the outgoing direction.

The definition of a route map has the following form. It consists of *filter-action constructs*.

```
route-map <rm_name> permit <order>  
match <filter_type> <filter_name>  
(action)
```

- <rm\_name> is the name of the route map this *filter-action construct* belongs to,
- <order> is an integer that determines in which order the constructs are processed. *Filter-action constructs* with lower order value are processed first.
- <filter\_type> is the type of the filter used for matching BGP updates (*i.e.*, as-path if matching by AS path or ip(v6) address if matching by prefix);
- <filter\_name> is the name of the filter, which has to be defined in the same configuration file,
- (action) is the action performed with the updates that match the filter.

Here is an example of a route map (two filters in the `match` commands are also provided):

```
ip as-path access-list Gamma_1 permit 65201 65432  
access-list Delta_2 permit 192.54.15.0/24  
!  
route-map ccw permit 20  
match as-path Gamma_1  
set weight 200  
!  
route-map ccw permit 10  
match ip address Delta_2  
set weight 500
```

The route map in the example above has two *filter-action constructs*. For each update that passes through the interface to which the route map is applied (in the indicated direction), the list of *filter-action constructs*

is scanned in the increasing order. The construct with the lowest *order* value is processed first. Unlike the filtering rules, the order in which the constructs appear in a route map does not matter. Thus, in the example above, the construct with the order value 10 is checked first.

If the filter part of the construct matches the arriving update, the construct action is performed. The remaining constructs are not processed. If the filter part of the construct does not match, the next construct is processed and so on. **If none of the constructs matches, the update is dropped.** Thus, always bear in mind that you might need a default permit construct at the end of the route map.

A number of possible actions can be performed with a route. Here are some of the path attributes that can be modified using the `set` command:

- the *weight* associated to a route: by default, all routes have weight 0. The exception are locally connected subnets that have weight 32768. The router selects the route with the highest weight. The weight can be set to any positive integer. The syntax is:

```
set weight 500
```

- the *local preference* associated to a route: by default, it is equal to 100. Higher local preference is preferred. The difference between weight and local preference is that weight is local to a router, whereas local preference is announced to all BGP routers in the same AS. The syntax is:

```
set local-preference 50
```

- the *AS path sequence* of the advertised route. It can be modified using the `set as-path` command. An AS sequence can be prepended to the advertised AS path, as shown in the following example:

```
set as-path prepend 65289 65453 65789
```

The command inserts the sequence 65289 65453 65789 at the beginning of the advertised AS path.

**Note:** `route-map` is used in a similar way to `distribute-list`, please make sure that all the declarations and definitions are correctly positioned.

## 7.3 POLICY ROUTING PLAYGROUND

### 7.3.1 AVOID TRANSIT TRAFFIC

Recover the configuration in Section 4.2 (recall that this means `r1` and `r2` are not bgp peers and that `zebra` and `bgp` run on all routers; also, please do not forget to remove the `network` command you used to manually advertise networks in bgp config files in section 5) and imagine that AS 65345 in Figure 1 represents an enterprise network of the Foot Clan, which is connected to the Internet via two ISPs—AS 65100 and AS 65200. You are the administrator of this enterprise network and you do not want your autonomous system AS 65345 to become a transit AS, *i.e.*, you do not want the traffic that neither originates nor terminates in AS 65345 to be routed through this AS. At the same time other autonomous systems have to know about all the prefixes in AS 65345.



12/ Give the most important lines of the modified `bgpd` configuration file at R3 and R4. (Hint: use `distribute-list` and `filtering by prefix`)

[A12]

If the modifications are correct, you should be able to ping between H1, H5 (and similarly between H2, H5), but not between H1, H2.

### 7.3.2 POLICY WAR (OPTIONAL, BONUS COUNTED AS RESEARCH EXERCISE)

Recover the configuration in Section 4.1 and imagine that the autonomous systems AS 65100, AS 65200, and AS 65345 represent domains of three internet service providers (ISPs), as indicated in Figure 1. The relations among these ISPs or ASs are such that AS 65100 and AS 65200 pay a fee to AS 65345 to have it carry their traffic (*i.e.*, AS 65345 is a service provider for the other two ISPs). AS 65100 and AS 65200 have a peering agreement, *i.e.*, they exchange traffic without paying fees to each other.

Now, let us consider the subnets 192.50.50.0/24 and 2001:1:0:5050::/64 that belong to AS 65345. AS 65100 carries a lot of traffic to these subnets and it would consequently prefer the traffic destined to these subnets (and *only* the traffic destined to these subnets, other traffic should flow normally!) to go via AS 65200, if possible. This would be a cheaper solution for AS 65100, but of course unfair to AS 65200.



**13/** Write down the most significant lines of the modified configuration files at R1. (*Hint: use route-map, weights, and filtering by prefix. Remember to have a default-case access-list.*)

[A13]

AS 65200 notices the changes and took this as the beginning of a small “war”. To avoid carrying traffic between AS 65100 and AS 65345’s subnets 192.50.50.0/24 and 2001:1:0:5050::/64, AS 65200 decides to change the configuration at the router R2 (again, other traffic should not be affected by these modifications).

The first plan for AS 65200 is to add an output filter towards R1 that discards all updates for the subnets 192.50.50.0/24 and 2001:1:0:5050::/64.



**Q14/** Write down the most significant lines of the modified configuration files at R2. (*Hint: use distribute-list and filtering by prefix.*)

[A14]



**Q15/** Can you check that your solution works by inspecting the BGP database at R1?

[A15]

The second plan for AS 65200 is to do the same as AS 65100. Namely, AS 65200 would prefer the traffic destined to 192.50.50.0/24 and 2001:1:0:5050::/64 to go via AS 65100.



**Q16/** What will happen?

[A16]

## 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 to your exams.**





## A LIST OF FIGURES

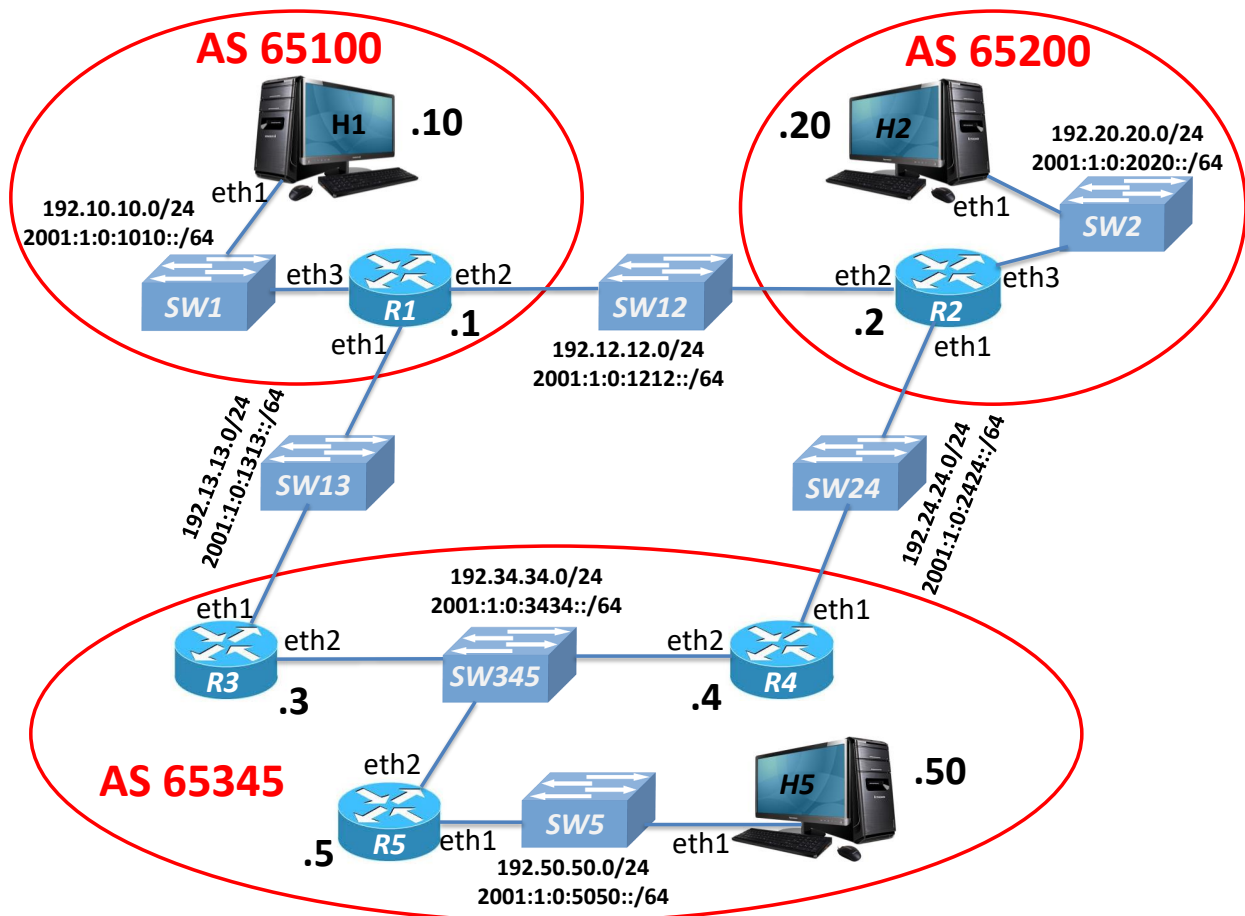


Figure 1: The network topology for this lab. It consists of 5 routers that belong to 3 autonomous systems.