Name 1: ______________________________ Group number: _________
Name 2: ______________________________

Computer Networking
Lab Exercises (TP) 3
Internal and External Routing

November 12, 2009

Abstract

In this TP you will work with the Internet's Inter-Domain Routing Protocol (BGP). You will learn about (and implement) peering mechanisms, advertisement filtering and routing policies. You will also learn how to run and manage routers running RIP. We will use the equipment in the lab room to model a few Internet Service Providers (ISPs) and their customer networks.

The TP is organized into two main parts. The first part is a tutorial where you will learn how to configure RIP and BGP on a router. In the second part, you will put into practice what you learned in the tutorial. You will be asked to build a BGP network, which can be seen as a small scale model (highly simplified, of course) of the real Internet. Further, you will solve some configuration problems.

1 Organization of the TP

A group of two students works at one workspace. There are two computers per workspace. The workspace number is written on a sticker on the side of each computer. In order to perform the experiments, you will have to collaborate with (at least) two other groups. We advise you to choose the groups (workspaces) you intend to work with before starting with the experiments. Like this you can synchronize with the other groups allowing you to work more efficiently.

1.1 TP Report

When answering questions, fill in your answers directly in the spaces provided in this document. That will be your TP report. At the end of the third session (at the latest) you have to hand it in (in principle one per group, but it can be one per student if you worked alone). Don't forget to write both your names and group number on the first page of the report (Not more than two names per report!).

The deadline is **November 26, before the lecture.**

2 Tutorial: How to Talk to a Router

After having followed the course so far, you should know the main ideas on which static and dynamic routing are based and how RIP and BGP work. The only thing that you are missing to become a routing expert, is how to put your knowledge into practice. We will start this lab nice and easy with a tutorial that will teach you exactly this.

Throughout this lab, you will be confronted with practical networking problems, in which you will need to configure a set of routers and implement a (hopefully working) small network of Autonomous Systems (AS). How do you do this? How do you ”talk to a router” to make the network behave the way you want? This tutorial will provide you with a basic introduction to the main tools and commands used to configure routers in order to implement a given routing behavior. During the tutorial you are not asked to implement anything on the machines in the lab room. The examples given are merely for illustration. Still, you are of course free to try them out if you want.

2.1 What is Zebra?

Zebra (now maintained as Quagga Software Routing Suite) is a suite of applications that we will use to run routing processes and manage them. These tools include several routings algorithms, such as RIP, OSPF and BGP for IPv4 and IPv6. In addition to its simplicity, another advantage of zebra is that it has a configuration syntax very close to the language used by Cisco routers.

Zebra is composed of several processes, and each process can be launched and run independently from the others:

- *zebra*: manages the network interfaces of the machine. It allows you to configure them, to monitor their states, and to monitor the routing table with a more detailed view than the `netstat` command. If you want, it is in a way a replacement for the networking commands (`ifconfig`, `route`, `netstat`) you saw in the two previous labs.
- *ripd*: handles RIP routing.
- *bgpd*: handles BGP routing.

2.1.1 Running a Process of the Zebra Family

In order to run any process of the set of zebra processes, the syntax is the following:

```
<process name> --user quagga --group quagga -d -f <configuration file>
```

where:

- `<process name>` is the name of the process you want to run (zebra, ripd or bgpd);
- `--user root --group root` sets user permissions for running that process;
- `-d` allows to run the process in daemon mode;
- `-f` allows to specify the configuration file name and its location
- `<configuration file>` is the name of the configuration file. Sample configuration files (extension `.conf.sample`) are present in `/usr/share/doc/quagga/examples/`).

Note: in this tutorial we teach you how to configure a zebra process by editing the corresponding configuration file and then running the process with this edited configuration file. In principle, a processes can also be configured ”on the fly” (while running) via telnet. We will however prefer the first method, as it is more efficient and allows you to have a clear view of the active configuration of your router. We will mainly use telnet to monitor the state of the process.

2.1.2 Connecting to a Process via Telnet

In order to monitor the activity of a running zebra process, you can connect to the processes via telnet. Generally, you telnet to the process on the same machine on which the process is running, but you can also do it from other machines. The port number associated with each process is defined in `/etc/services`. As an example, in order to telnet to the bgpd process, you type

```
telnet localhost bgpd
```

- The passwords to be used are in the configuration file of the process you want to run. The default one is “zebra”;
- We advise you to telnet to the process mainly to check routing tables, BGP and RIP routing databases, but to change the configuration by editing the configuration file;
- if you remain inactive while logged in to a process via telnet, you will be disconnected after a certain time, forcing you to login again.

2.2 Configuring “zebra”

Instead of using the usual `ifconfig` command, you can use the `zebra` process to configure network interfaces on any PC on which you will have to run a routing process (`ripd`, `bgpd`).

An interesting feature of this process is the possibility to assign more than one IP address (and the corresponding subnets) to the same physical interface, which will prove to be useful in what follows. `zebra` also gives you a more detail view of the routing tables.

Before running `zebra`, you must set up its configuration file. Here is a simple example of such a configuration file for `zebra`:

```
!
!  zebra sample configuration file
!
hostname Router
password zebra
enable password zebra
!
interface lo
!
interface eth0
ip address 192.168.34.2/24
ip address 102.33.129.7/22
!
interface eth1
ip address 192.168.100.2/24
!
!
```

```
line vty
!
```

Some explanations:

- you can give any name you want to the configuration file;
- `!`: lines starting with `!` are comments, they are ignored;
- `hostname`: the name you choose will appear before the cursor when you log in to the process through telnet
- `password`: the password required to telnet to the process
- `enable password`: the password required to pass to privileged mode. Some commands require you to be in privileged mode in order to be executed (similar to root on UNIX). In order to change to privileged mode you have to type `ena`. When the command ”enable password” is present in the configuration file you will then be prompted for this password.
- for each interface, you find the IP addresses (there can be more than one per physical interface) and the associated subnets. The first IP address in the list is the one that will be assigned to the physical interface, and that will appear when checking interfaces with the ifconfig command.
- `line vty`: enables telnet login.

An important command you can use in a zebra configuration file is the one that defines a static route in the local routing table. The syntax is

```
ip route (ip address) (netmask) (next hop)
```

Example:

```
!
ip route 192.168.50.100 255.255.0.0 192.168.100.100
!
```

2.2.1 Some Useful Commands

When logging in to the zebra process via telnet, the most important commands are:

- `show ip ro`: visualizes the routing table.
- `show running-config` (or `sh run`, you can almost always abbreviate zebra commands in a similar way): allows you to check the active configuration (types the main entries of the conf file). Note this command requires privileged mode, otherwise it will not work.;
- `show interface`: visualizes data about all the interfaces at the machine, configured through zebra;

2.2.2 Some Advice

- We strongly advise you to either configure interfaces by `zebra` (preferred), or by the `ifconfig` command. Do not use both ways of configuring interfaces on the same machine, as interaction between the two is not always clear.

- We observed that once you kill the `zebra` process, the interface configuration keeps on being active on the machine. It can thus happen that an interface to which you assigned more than one IP address through `zebra` (and that `ifconfig` always "sees" as having only one IP address), still replies to pings to all the IP addresses configured through zebra, even after killing the zebra process.
  Therefore, always put your machine to a clean state before running any process of the zebra family:click twice on the Network Workspace Manager icon that you can find on the desktop of each machine. From the menu that opens select "Load from Disk" and choose "lab-1-default: IEW default configuration for lab1".
- never start the same process on the same machine if it is already running. You can check whether a `zebra` process exists by observing the processes list using the command

```
    ps -ef | grep zebra
```

- if more than one IP address is assigned to one interface you need to enable IP forwarding on this machine (even it is a machine with only one interface), otherwise packets received from the network corresponding to one of the IP addresses will not be forwarded to networks corresponding to any of the other IP addresses (similar to what happens on a router with two interfaces when IP forwarding is not enabled).

In order to streamline this process, you should use the following script. Type it using a text editor (e.g. gedit), and save it with an extension `.sh` (the script assumes that your configuration files for `zebra`, `ripd` and `bgpd` are, respectively, `zebra.conf`, `ripd.conf` and `bgpd.conf` and are present in the same directory as the present script):

```
#!/bin/sh

killall zebra ripd bgpd
zebra --user quagga --group quagga -d -f zebra.conf
#uncomment whatever lines you need
#ripd --user quagga --group quagga -d -f ripd.conf
#bgpd --user quagga --group quagga -d -f bgpd.conf
sysctl -w net.ipv4.ip_forward=1
```

Whenever you want to comment a line, precede it with the character #.
Then type `chmod 755 (script name file)` in order to change permissions for that file such that it becomes executable. You can execute it by typing the script name preceded by "`./`".

2.3 Configuring "ripd"

As you can deduce from the course theory, the main configuration steps that must be performed when running RIP on a router are:

- enabling RIP, and choosing its version;
- choosing which networks should be advertised through RIP;
- enabling those interfaces that have to take part in the exchange of routing informations.

Similarly to the `zebra` process, the `ripd` process can be configured both via telnet and through preliminary editing of its configuration file.

Here is an example of a `ripd` configuration file:

```
!
!  ripd sample configuration file
!
hostname ripd
password zebra
!
router rip
version 2
network 192.44.31.0/24
redistribute connected
!
line vty
!
```

The main commands to be used in configuring a RIP process (in the configuration file) are:

- `router rip`: enables the RIP process;
- `version`: allows to specify which RIP version the process will run. Example:

  ```
  version 1
  ```

  enables RIP version 1.
- `network`: is one of the two ways in which you can allow a prefix to be exchanged through RIP. All the interfaces at the router whose IP address belongs to the prefix declared with the command `network` will take part in the RIP protocol data exchange and the prefix that follows the command will be advertised through RIP. The syntax is:

  ```
  (no) network (network prefix)
  ```

  Add the word `no` at the beginning if you want a particular prefix not to be exchanged over RIP.
- `redistribute`: it enables the announcement through RIP of all those prefixes which have been learned from a given source. The syntax is:

  ```
  redistribute protocol
  ```

  where `protocol` specifies the source from which the prefixes have been learned. This can be `bgp` or `ospf` (when you want to redistribute routes learned through those protocols into RIP), `static`, when you want to redistribute statically configured networks (added through `ip route` in the `zebra` process), or `connected`. In this last case it refers to those networks that the router learns from the configuration of its interfaces (redistribution of directly connected networks). **Note: you always have to use this command in conjunction with the command `network`, as you must tell the router on which interfaces to enable RIP messages exchange.**

In the above example, we advertise through RIP all directly connected networks, but we allow only those router interfaces in the subnetwork 192.44.31.0/24 to send and receive RIP advertisements.

2.3.1 Some Useful Commands

In order to inspect the RIP database, the only way is to connect to the process via telnet. The most important commands you will use are:

- `show ip rip` visualizes the RIP database at the router.
- `show running-config`: allows you to check the active configuration (types the main entries of the conf file). Again, run `ena` first;

Note that if an interface is not on one of the advertised networks, no RIP advertisement will be sent (nor will it be accepted) through it (you can see this through ethereal).

2.4 CONFIGURING "BGPD"

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

- enabling BGP;
- declaring peers;
- choosing which prefixes to redistribute into BGP;
- aggregating advertised paths.
- configure policy routing, by setting up filtering rules and changing path attributes.

Let's now have a closer look at each of those configuration tasks. In what follows you will be explained how to configure all these tasks through editing the configuration file. The commands shown are therefore to be applied on the configuration files (although many of them have the same syntax when used to configure the router through telnet).

2.4.1 BASIC BGP CONFIGURATION

Here is an example of a simple configuration file for the `bgpd` process:

```
!
!  Bgpd configuration file example
!
hostname bgpd
password zebra
!
router bgp 65101
bgp router-id 200.44.0.14
network 192.45.88.0/24
neighbor 200.44.0.13 remote-as 65131
neighbor 200.44.0.12 remote-as 65121
!
line vty
!
```

- `router bgp AS`: enables bgp, and defines the AS number of the AS the router belongs to (in the example above, 65101).
- `bgp router-id`: defines the router id. By definition this is the highest IP address on the router, or the highest IP address of its loopback interface. If you configured the interfaces via `zebra`, it should learn it automatically and you should not have to specify the id and can thus omit this command.
- `network 192.45.88.0/24`: enables redistribution over BGP of the prefix `192.45.88.0/24`. The command `network`, which has the same syntax as for ripd, is used to redistribute selectively into BGP only some of the directly connected networks. In general, if we want all directly connected

subnets to be advertised over BGP, we rather use the command `redistribute` (which has the same syntax as for ripd), and set the parameter `protocol` to `connected`.
When we want to redistribute routes learned through rip, ospf, or statically configured routes, `protocol` must respectively be set to `rip`, `ospf` , and `static`.

- `neighbor`: allows to declare all neighboring BGP routers (inside or outside the AS of the router we are configuring): each of the BGP routers declared with these commands will be considered as a directly connected BGP neighbor, and a TCP connection to exchange BGP routing data will be established between the local router and the one declared. The syntax is:

```
neighbor IP_address remote-as as_number
```

  where

  - `IP_address` is the IP address of the interface of the neighboring router to which all routing data will be sent;
  - `remote-as as_number` specifies the AS number of the AS the neighboring router belongs to.

Note that no distinction between I-BGP and E-BGP is made. This information is learned automatically from the AS numbers.

2.4.2 AGGREGATION OF ADVERTISED PATHS

Path aggregation can be accomplished through the command `aggregate-address`. Its syntax is the following:

```
aggregate-address _._._._/_ [summary-only]
```

The command is always followed by a prefix (in the form IP address/netmask), which represents the aggregation of a set of prefixes. The optional command `summary-only` enables advertising of only the aggregate prefix: if it is not used, the BGP router advertises the aggregated prefix *plus* all the prefixes it would have advertised without the command `aggregate-address`.

2.4.3 FILTERING OF ADVERTISEMENTS

Advertisement filtering is the process of selecting path advertisements at a router. By filtering, the router can decide to accept certain advertisements and reject others. You can apply filters at the input of a router (i.e. to all advertisements *coming from* a given neighbor, *before* they are put into the BGP database) and at the output of a router (that is, to the advertisements *sent to* a given neighbor, *before* that they are actually sent).
Moreover, filtering can be based on *prefix* or on the *as-path* (that is, by selecting all path advertisements depending on the advertised network prefixes or by selecting all path advertisements containing a given AS number or a given sequence of AS numbers).

Filters must be applied to advertisements coming from or sent to a given neighbor. In case of filtering *by prefix*, this is accomplished through the command `distribute-list`. The syntax is:

```
neighbor IP_address distribute-list filter_name in/out
```

where `IP_address` is the IP address at which a given neighbor can be reached, `filter_name` is the name of the filter, and `in/out` specifies whether the filter is applied at the input or output of the router.

In case of filtering *by as-path*, applying a filter is accomplished through the command `filter-list`. The syntax is otherwise the same as above:

```
neighbor IP_address filter-list filter_name in/out
```

In order to be able to apply the filter to advertisements of a certain neighbor, you need to define the filter. Every filter consists of a set of rules. Defining a filter equals defining its set of rules.

The syntax to define a rule *for a filter by prefix* is the following:

```
access-list filter_name permit/deny prefix
```

- `filter_name`: the name of the filter: it can be composed by letters and/or numbers.
- `permit/deny`: specifies whether the advertisements that the filter will select are those for a prefix matching a given prefix, or all those that do not match it;
- `prefix`: the prefix we match against. It can be substituted by `any`, in which case it applies to any advertised prefix.

An example is the following:

```
access-list Alpha_1 permit 192.168.33.0/24
access-list Alpha_1 permit 133.33.23.0/24
```

Here the filter is called `Alpha_1` and it accepts advertisements advertising prefixes 192.168.33.0/24 and 133.33.23.0/24.

If you apply a filter to a neighbor, the router will for all advertisements coming from or going to this neighbor (depending on whether the filter is in or out) scan the list of rules that constitute this filter. It scans them in the order of appearance in the configuration file. If a matching rule is found, the advertisement ”passes” the filter. Subsequent rules are not checked anymore. If no rule matches, the advertisement doesn’t pass the filter and is consequently dropped.

The syntax to define a rule *for a filter 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 by letters and/or numbers.
- `permit/deny` specifies whether the paths that the filter will select are those matching a given AS number sequence, or all those that do not match it;
- `as_number_sequence` is a sequence of one or more AS numbers. It can be substituted by `.*`, in which case it applies to any AS path.

An example is the following:

```
ip as-path access-list Beta_2 permit 65121
ip as-path access-list Beta_2 permit 65111 65131
```

In this example, the filter with name Beta_2 will select all advertisements containing AS 65121, and those that contain the sequence of AS 65111 65131 in this order, in whatever position inside the advertised AS path.

Finally, let's put everything together in the following example:

```
!
!  Bgpd configuration file - example on filtering
!
hostname bgpd
password zebra
!
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 Filter1 in
neighbor 200.44.0.12 remote-as 65121
neighbor 200.44.0.12 filter-list Filter2 out
!
access-list Filter_1 permit 192.168.33.0/24
access-list Filter_1 permit 133.33.23.0/24
!
ip as-path access-list Filter_2 permit 65111
!
line vty
!
```

Question 1: Explain the result of applying filters Filter1 and Filter2.

Note: Order in the configuration file matters to some extent when it comes to filters

- all "neighbor" statements have to occur before any filter declaration
- all "aggregate" statements have to occur before any filter declaration

2.4.4 Modifications of Path Attributes

As you have seen, filters are an important tool in order to change the path selection process at a given router, and to influence the one at neighboring routers. You might have noticed, however, that filters alone cannot be the whole answer when it comes to implementing policy routing. They only allow you to accept or reject certain route. In the course you saw some more advanced things, like giving different preferences to different routes.

That is where *route map* come into play. A route map is a more general and more powerful tool than a filter. Like a filter it allows you to select certain advertisements. Unlike a filter it then also lets you perform certain actions (basically, modifications of path attributes) on the selected advertisements. The principle on which it works is similar to the one of a filter: applied to all the advertisements coming from (or sent to) a given neighbor, it selects those advertisements with given characteristics, and it modifies their attributes in order to influence the path selection process at the router itself, or on one of its neighbors.

As it was the case for filters, a route map must be applied to advertisements coming from or sent to a given neighbor. This is accomplished through the command `route-map` which is similar to the commands `distribute-list` and `filter-list` you saw before:

```
neighbor IP_address route-map rm_name in/out
```

where `rm_name` is the route map name (any sequence of characters and/or numbers), and `in/out` specifies whether the route map should be applied to the advertisements coming from the specified neighbor, or to those sent to it.

In order to be able to apply a route map to advertisements of a certain neighbor, you need to define the route map. Every definition of a route map consists of a set of *filter-action constructs* of the following form:

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

where:

- `rm_name` is the name of the route map this *filter-action construct* belongs to (it can be any sequence of characters and/or numbers);
- `order` is an integer that determines in which order the constructs are processed. *Filter-action constructs* with a lower order value are processed first.
- `filter_type` is the type of filter used for selecting advertisements: it can be `as-path` (for a filter by as path) or `ip address` (for a filter by prefix);
- `filter_name` is the name of the filter (of course the corresponding filter needs to be defined somewhere in the same configuration file);
- `(action)` is the action (modification of some path attributes) that will be performed to all path advertisements that match the filter.

An example of a route map definition is the following:

```
!
route-map ccw permit 20
match as-path Beta_2
set weight 200
!
route-map ccw permit 10
match as-path Beta_1
set weight 500
!
```

For each advertisement coming from (or sent to) the neighbor to which the route map is applied, the list of *filter-action constructs* is scanned. The constructs are scanned in increasing order, the construct with the lowest associated order value is processed first (unlike filter rules, the order in which the constructs appear in the configuration file does not matter!). In the example above, the construct with order value 10 will thus be checked first. If the filter part of the construct matches, the corresponding action is applied and the advertisement is put into the local BGP database (or sent to the neighbor). The remaining constructs are not processed. If the filter part of the construct doesn't match, the next construct is processed and so on. If none of the filters match, the advertisement is dropped.

There are a number of possible actions (path attributes that can be modified), and all of them use the command `set`. Here are some that we might use:

- the *weight* associated to a given path: by default, all paths have weight 0, and the router selects those advertised paths with the largest weight. A valid weight value is any positive integer. In the previous example, we associated a weight of 500 to some paths:

```
set weight 500
```

- the *local preference* value associated to a given path: by default, it is equal to 100. The router selects those advertised paths with the largest local preference value. Example:

```
set local preference 50
```

  The difference between weight and local preference is that weights are only local to the router, whereas local preferences are announced to all other BGP routers in the same AS.
- the *sequence of AS path* that constitutes the advertised route, through the command `set as-path`. In particular we can add, at the beginning of any path to which this action applies, any sequence of AS paths. As an example, the command

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

  will add the sequence of AS paths 65289 65453 65789 to the beginning of the advertised path.

Example: Consider the network of four BGP routers in Figure 2.4.4. The configuration file for the BGP router in AS 65100 contains the following lines:

```
!
router bgp 65100
bgp router-id 200.44.0.12
neighbor 200.44.0.13 remote-as 65099
neighbor 200.44.0.14 remote-as 65101
neighbor 200.44.0.14 route_map myRouteMap1 in
!
ip as-path access-list Beta_1 permit 65099
!
ip as-path access-list Beta_2 permit 65102
!
route-map myRouteMap1 permit 10
match as-path Beta_2
```

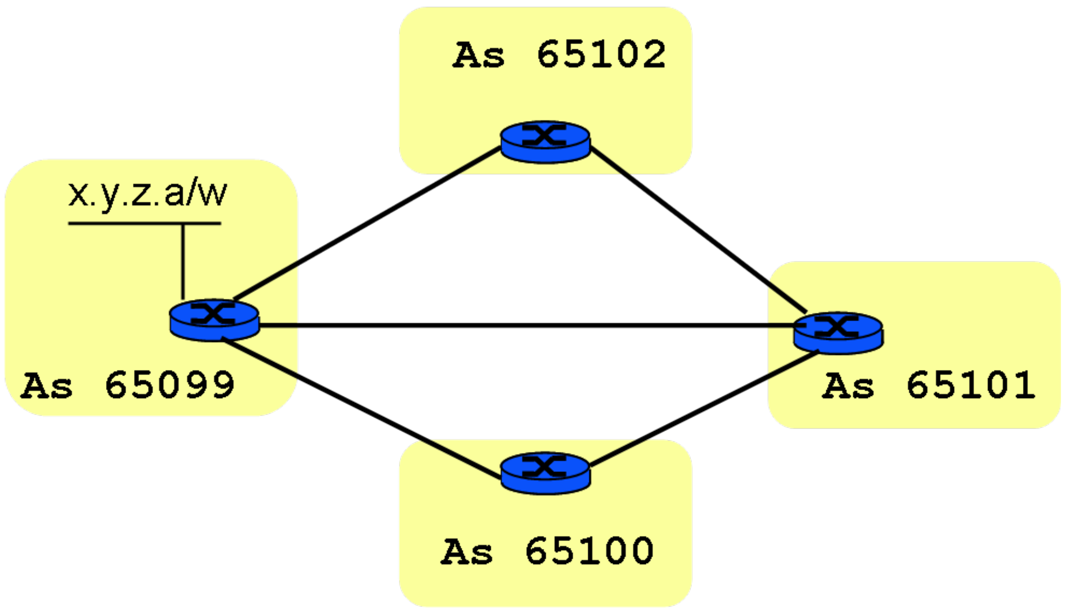

Figure 1: The Network of Autonomous Systems (ASs) in the exercise.

```
set local preference 200
!
route-map myRouteMap1 permit 20
match as-path Beta_1
set local preference 5
!
```

We assume that the other routers do not apply any policies, and choose the routes with the shortest AS paths.

Question 2: Let's consider the BGP database at the router in the AS 65100. Which advertised paths will the BGP database contain, for the prefix x.y.z.a/w? Which one will be selected as the best?

What is the answer to the previous questions, if for some reason the TCP connection between AS 65099 and AS 65101 goes down (e.g., in case of link failure)?

Question 3: Assume now that there are no policies applied at any of the routers. Routers in all four autonomous systems choose the routes with the shortest AS paths. However, router in AS 65099 would like to route its traffic for AS 65101 through AS 65102 instead of using the direct link that exists between AS 65099 and AS 65101. How can this be implemented? Hint: use route map(s).

Question 4:The administrators of AS 65102 now decided to apply an access on the BGP router in AS 65102, in order to block the transit traffic through its network (and only the transit traffic). Write down the BGP.conf file at router in AS 65102 that contains this access list.

2.4.5 Other Configuration Options

Route flap dampening is off by default.

- `bgp dampening`: turns on route flap dampening;
- `no bgp dampening`: turns off route flap dampening;
- `bgp dampening` *half-life-time*: changes the half-life time (time after which the penalty attributed to a route halves).

2.4.6 Some Useful Commands

When connecting via telnet to the `bgpd` process, the most important commands you will use are:

- `show running-config` allows you to check the active configuration of the BGP process (again, use `ena` first;
- `show ip bgp` visualizes the BGP database at the router.
- `clear IP BGP *` clears the BGP database at the router, restarting the process of BGP advertisements exchange.

2.5 Additional Resources

In configuring routers, the set of commands you are using is a (not so small) subset of the commands used to configure commercially available Cisco routers. Documentation can be found on the Zebra website (*www.zebra.org*), but as the syntax is close to the one employed by Cisco routers, you can also find good references on the Cisco website.

3 Experiments: Small-scale Internet - Building a BGP Network

In the tutorial, you learned how to configure a router to perform a series of different tasks. Now you are ready to put your obtained knowledge into practice: working together with your colleagues, you will build a whole BGP network.

Even if in what follows you work with other groups, and although reciprocal help and discussions are more than encouraged, **you must try to solve each problem by yourself alone**. Please do so before comparing and discussing with the other groups. Also answer the questions in the TP according to **your own** understanding. Remember: if well done, this TP is a unique opportunity to understand (by touching it with your hands) one of the most tricky subjects of the course. If done mechanically (or worse, if delegated to your colleagues), it will lose all its benefit and become only a boring waste of time...

Note: you need up to 6 PCs to realize the network configurations in the experiments. This means that three groups need to collaborate. Organize yourself by contiguous workspaces (or use hubs and long cables if you are collaborating with groups that are far away...). To make your life easier, **choose from the very beginning whom to work with**.

We advise you to bring some pieces of paper (or post-it) in order to label PCs and to have less trouble in identifying them. **Do not write on the machines!** If you didn't bring anything, ask one of the teaching assistants and he will give you post-its.

Another good idea is to bring along a USB stick to store the configuration files so you still have them the week after. Finally, please don't forget to erase all your configuration files and reboot before leaving the room.

3.1 Problem 1: Basic BGP configuration, Policy routing, Interaction between EGP and IGP

3.1.1 The Scenario

In this first problem you are asked to build a small BGP network and to implement policy routing. The focus is also on the options a BGP router has for informing all other routers in the same AS, about a set of network prefixes learned from other BGP routers.

In this experiment, the scenario you have to implement is depicted in Figure 2. It includes three ASs. AS 65432 is composed of two BGP speakers (R3 and R4) and two other routers, R5 and R6. AS 65101 and AS 65201 are composed of one single BGP speaker each.

Arrows in the scheme do not indicate physical links but relations between autonomous systems. Physical links (cables) are indicated with dashed lines. Prefixes close to the arrows denote the network prefixes of physical networks between the endpoints of the arrows (e.g. R1 and R2 are BGP peers as indicated by the arrow. They also have a physical connection through a hub and are on the same network with prefix 192.44.21.0/24).

3.1.2 Setting Up A Working Network

Configure the IP addresses of the interfaces of the six routers according to the indications in the figure (note that R3 and R4 must necessarily be two router PCs, as they need two network interfaces, one towards the inter-AS network, and one in the intra-AS network). The last 8 bits of all the IP addresses always correspond to the number of the machine. Example: R3 will have IP address 192.44.31.3 on its interface on subnet 192.44.31.0/24. You can either write the zebra files yourself (provided you stick to the indicated

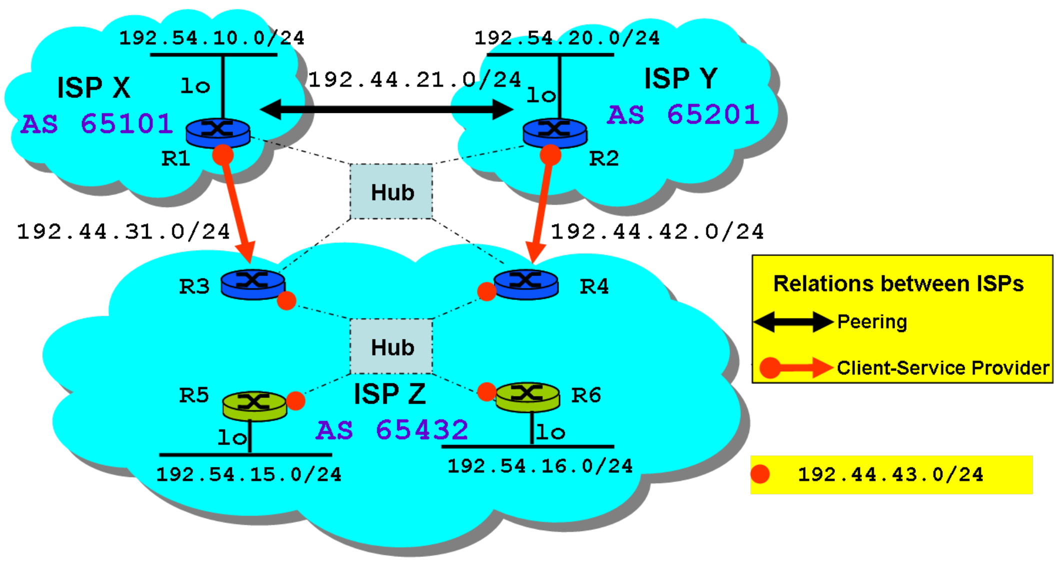

Figure 2: The Network of Autonomous Systems (ASs) used in Problem 2.

addressing scheme) or use the ones that are provided on the course webpage.

Now start the BGP routing process on **all** routers. Remember that R3, R4, R5 and R6 must have the same AS number. On each BGP router, redistribute directly connected networks into BGP.

Also, run RIP on all routers in AS 65432. At this stage, do not configure the interaction between RIP and BGP (as if BGP were not running on those routers).

Further, enable IP forwarding where it is required (you can put it as an additional line into your script that starts the zebra processes).

Have a look at the routing tables at R5:

Do you think all routers in this network have all the sufficient information to forward a packet between any two subnetworks? Justify your answer.

Write down the most significant lines of the configuration files at router R3 for ripd and bgpd, relative to the configuration you implemented.

Have a look at the routing table of the router R1 using *zebra*:

What is the meaning of the ”B” flag at the beginning of the line?

Start Ethereal on that router and have a look at BGP packets exchanged between BGP routers.

What is the role of KEEPALIVE messages? How often are they sent?

Now imagine that AS 65432 is composed of 100 routers, but that the whole inter-AS connection is still the one in the figure (i.e. two BGP speakers for AS65432, one for each neighboring AS).

Assume that in order to route packets originating from AS 65432 towards other ASs, you decide to run BGP on each of the 100 routers. Thinking about how BGP works, what do you think are the drawbacks of such a solution? Would specific BGP features such as policy routing still be fully exploited in such a scenario? Explain why.

Another solution to solve the routing problem is to run an interior gateway protocol inside each AS, and to run BGP only at the border gateways.

We are now going to look at this second solution. Run RIP at all routers in AS 65432. Run BGP only on R1, R2, R3 and R4.

Reminder: in order to configure RIP, you have to:

- enable RIP version 2;
- announce directly connected networks;
- tell RIP to redistribute directly connected networks.

Look at the routing table at R5: do you think it is sufficient to route a packet to any destination? What is missing?

Same question for R1. Can you route to any destination? If not, what is missing? why?

How can you manage to make RIP and BGP exchange routing data?

- BGP speakers in AS 65432 need to know which prefixes from AS 65432 they have to advertise to other AS;
- At the same time, they have to inform all routers in their AS about how to reach networks which are located outside their AS.

How can you change the configuration at routers R3 and R4 in order to achieve the first part (advertise prefixes of AS 65432 to other ASs)?

How can you change the configuration at routers R3 and R4 in order to achieve the second point (inform routers in AS 65432 of how to reach other networks) *in the most efficient and scalable way*, that is, by adding the least possible number of entries into the routing tables at routers R5 and R6?

Now implement your solution, by modifying the configuration files at R3 and R4 accordingly. Check that your solution works, by checking routing tables at R5 or R6, and BGP databases at R4.

Write down all the significant lines of the configuration files for ripd, zebra, and bgpd at router R4 that are needed for your solution to work.

What happens if router R3 or R4 fails? Implement your solution in a way that is resilient to failure of one of the two routers (R3 or R4). Test it by doing the following: ping 192.54.10.0 from R6. Check the routing table at R6 and determine which of the two routers R3 or R4 is the next hop towards 192.54.10.0. Simulate a failure of this router by disconnecting both of its cables. What do you observe at R6? Explain how your solution recovers from this failure. How long does it take to recover?

Reconnect the router. Have a look at the BGP database on R1. Certain prefixes are associated with more than one path.

How many paths exist for the prefix $192.54.15.0/24$? How do you explain that?

In the present network configuration, what is the main criterion for your router to choose the best advertisement (the one preceded by the character “>” in the listing)?

Imagine now that AS 65432 in Figure 2 represents an enterprize network that is connected to the Internet via two ISPs (ISP X and ISP Y). You are the administrator of this enterprize network and you don’t want your autonomous system (AS 65432) to become a transit AS, i.e. you don’t want the traffic that neither originates nor terminates in the AS 65432 to be routed through this AS. At the some time the other networks have to know about the prefixes in your network so these have to be advertised.

How can this be achieved? Propose a solution that satisfies both requirements.

Give the most important lines of the R3 and R4 BGP configuration files.

Check if the configuration changes you introduced in BGP routers R3 and R4 actually achieve the desired result, by checking the BGP database at R1. If everything looks fine you passed your test as an enterprize network administrator.

Now imagine that the ASs 65101, 65201 and 65432 represent networks of three internet service providers (ISPs), as indicated in Figure 2. Pay attention to the *nature* of the relations among these ISPs indicated in Figure 2. ISP Z is a service provider for the other two ISP: they pay it a fee for carrying their traffic. ISPs X and Y have a peering agreement: they exchange traffic without paying anything to each other.

Now let's consider the subnet 192.54.15.0 that belongs to ISP Z. ISP X carries a lot of traffic to this subnet and would consequently prefer traffic destined to this subnet(and *only* the traffic destined to this subnet, all other traffic should flow normally!) to pass through ISP Y, if possible. This would be a cheaper solution for ISP X. However it is of course unfair for ISP Y.

How can ISP X implement this? Find the modifications to the configuration of R1 that achieve this and write down the most significant lines of the modified configuration files.

Check that your solution works by doing a traceroute from R1 to 192.54.15.0. Make sure that the routes the packets choose correspond to the policy above. If this is not the case, you have to refine your solution.

Now, ISP Y took this as the beginning of a little "war": in order to avoid to carry traffic between ISP X and ISP Z's network 192.54.15.0, ISP Y decides to change the configuration of router R2 (again all other traffic should flow normally).

How can ISP Y implement this policy? Find the modifications to the router configuration that achieves this and write down the most significant lines of the modified configuration files.

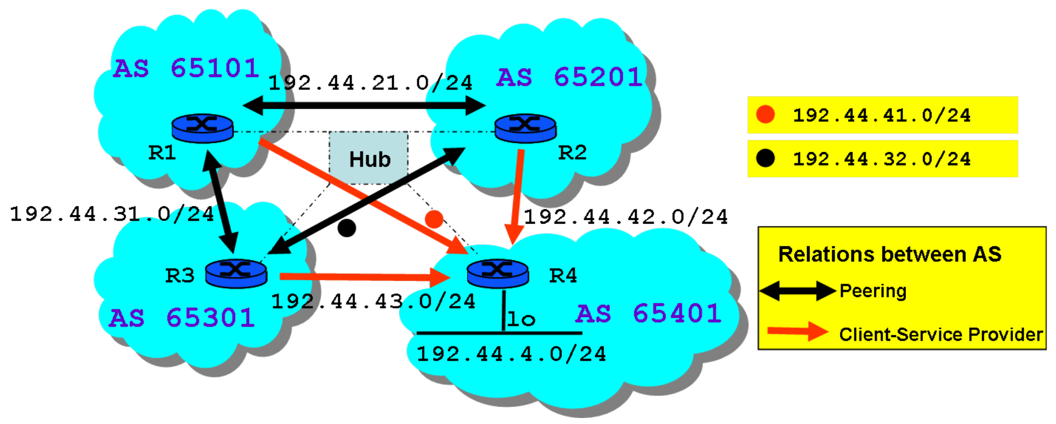

Figure 3: The Network of Autonomous Systems (ASs) used for the Bad Gadget problem.

Can you check that your solution works by looking at the BGP database at R1?

Also check that it works by doing a traceroute from R1 to 192.168.15.0. Make sure that the routes the packets choose correspond to the policies implemented at R1 and R2. If this is not the case, you have to refine your solution.

3.2 Problem 2: The Bad Gadget

Policy routing, which is basically a result of commercial agreements, sometimes results in a set of path selection criteria in each BGP router which are different than simple shortest path selection.

In this last experiment, you will study this on the network topology given in Figure 3.

Start the zebra process on all routers and configure the interfaces as shown in the figure. Again, the last 8 bits of all the IP addresses always correspond to the number of the machine. You can either write the zebra files yourself (provided you stick to the indicated addressing scheme) or use the ones that are provided on the course webpage.

All four routers run BGP. Therefore, start the BGP process at all routers and configure it according to the AS scheme indicated. Now focus on the paths that routers R1, R2, and R3 select in order to reach network 192.44.4.0/24.

Which is the path to that network at the three routers? By which criteria it is chosen?

Look at Figure 3. Suppose that the preferred paths to reach network 192.44.4.0/24, stored in Loc-RIB, are:

- 65401 at router R3
- 65301-65401 at router R2

Now suppose that for some reason, R3 changes its preferred path to that network to 65101-65401. Consequently, it will advertise this new preferred path to all the neighboring ASs.

Do you think that R2 will keep its old preferred path, after receiving the update from R3? If so, why? If not, why?.

Now alter the path selection criteria at routers R1, R2 and R3, for the considered network $192.44.4.0/24$, according to the following policies:

- R1 must prefer the path R1-R2-R4 to the direct path R1-R4. Moreover, R1 prefers the direct path R1-R4 to any path containing the AS of R3.
- R2 must prefer the path R2-R3-R4 to the direct path R2-R4. Moreover, R2 prefers the direct path R2-R4 to any path containing the AS of R1.
- R3 must prefer the path R3-R1-R4 to the direct path R3-R4. Moreover, R3 prefers the direct path R3-R4 to any path containing the AS of R2.

How does the bgpd configuration file look like at R1, after implementing the above path selection criteria? Write down all the significant lines of the bgpd configuration file at router R1.

Now look at the BGP path database at routers R1, R2 and R3.

At R1, what is the preferred path to network 192.44.4.0/24?

And after 2 min? After 10 min?

What do you think is the reason for what you see?

If you observed a problem in the previous question, can you come up with a solution for this problem?