Université Catholique de Louvain

Samir Marini Mohamed- 01271500
SALEY ABDOU Djafarou - 24031600
Pierre-Rodéric Roose - 60071500

***LINGI2142** : Computer networks: configuration and management*
***Project** : Frrouting configuration*

2019-2020
Group 7

Table of Content

Introduction

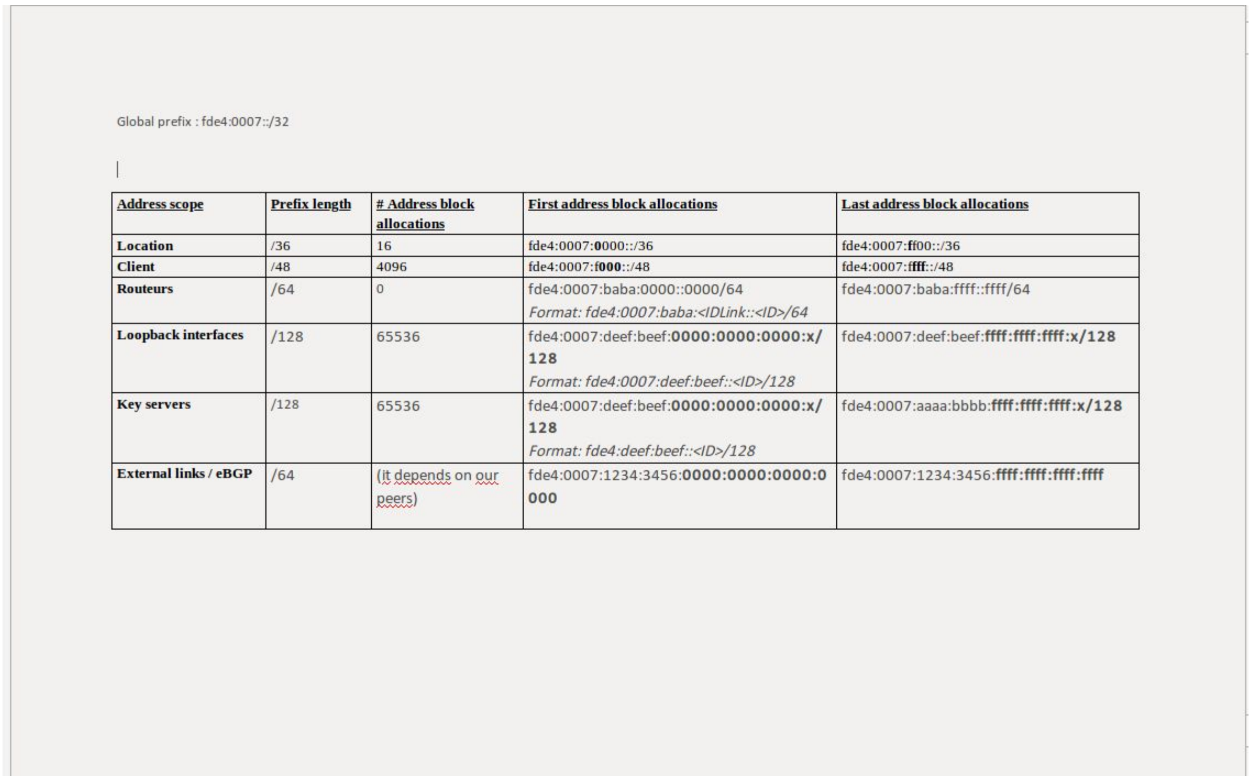
During the last 6 weeks, we have played the role of a network engineer in a large Internet provider company. This report is therefore the result of the work done throughout this period to design the best possible network for this supplier in terms of speed, robustness and automation. Our report will be divided into 3 main parts in parallel with the 3 main parts that have formed our work.

The first part of this report will discuss our choices of addressing plan and topology.
The second part of this report will discuss our design choices regarding the eBGP, iBGP protocol and how we have established the various connections with other major Internet providers and the third part of the report will discuss our choices in terms of implementing automation scripts, our BGP policies and the security aspect to reduce attacks against our network

We worked on the project with a virtual machine on which we used the Frrouting library to set up all the configuration files.

Part 1 : *Addressing plan, first network topology, OSPF and Configuration file script*

1.1 Addressing plan



Global prefix : fde4:0007::/32

Address scope	**Prefix length**	**# Address block allocations**	**First address block allocations**	**Last address block allocations**
Location	/36	16	fde4:0007:**0**000::/36	fde4:0007:**f**f00::/36
Client	/48	4096	fde4:0007:f**000**::/48	fde4:0007:**ffff**::/48
Routeurs	/64	0	fde4:0007:baba:0000::0000/64 *Format: fde4:0007:baba:<IDLink::<ID>/64*	fde4:0007:baba:ffff::ffff/64
Loopback interfaces	/128	65536	fde4:0007:deef:beef:**0000:0000:0000:x/128** *Format: fde4:0007:deef:beef::<ID>/128*	fde4:0007:deef:beef:**ffff:ffff:ffff:x/128**
Key servers	/128	65536	fde4:0007:deef:beef:**0000:0000:0000:x/128** *Format: fde4:deef:beef::<ID>/128*	fde4:0007:aaaa:bbbb:**ffff:ffff:ffff:x/128**
External links / eBGP	/64	(it depends on our peers)	fde4:0007:1234:3456:**0000:0000:0000:0000**	fde4:0007:1234:3456:**ffff:ffff:ffff:ffff**

Location :
We have decided to configure four bits to identify the location where the object will be located. Using four bits gives the possibility to have 2^4 possible locations, for a total of 16 possible locations. We considered in the framework of this project that we were a large ISP serving Western European countries (France, Belgium, Netherlands, Great Britain, Luxemburg), this address is more than enough. Despite the reservation by our team of the prefix baba, this reservation respects the bit of place set to "b" because our isp is based in Belgium.

Below you will find our correspondence in terms of the location of our customers :

- Clients in *Belgium*:
  fde4:7:bxxx::/36

- *Clients in France:*
  fde4:7:fxxx::/36

- *Clients in Luxembourg:*
  fde4:7:axxx::/36

- *Clients in Dutchland:*
  fde4:7:cxxx::/36

- *Clients in Great Britain:*
  fde4:7:dxxx::/36

Clients *:*

We have decided to assign use 3 bits to identify our customers. We will then advertise them a /48 address, be careful that in this /48 we have reserved the fde4:0007:baba::/48 for our own infrastructures which leaves 2^12 -1 possibilities of identification of customers per location which still represents 4095 different customers which is largely enough because we have as customer the large companies of Northern Europe which can be counted on the fingers of one hand.

Routers :

We could use addresses /128 for our routers, but after reading many articles, they recommended not to use prefixes larger than /64 for security reasons, so we decided to use /64.

Loopback interfaces :

We decided to configure /128 loopback addresses following this format => fde4:7:deef:beef::<Router id>/128. We could have used the prefix /48 that we reserved for ourselves (fde4:7:baba) but we wanted to clearly identify these addresses as loopback by giving them their own format.

External links/ eBGP :

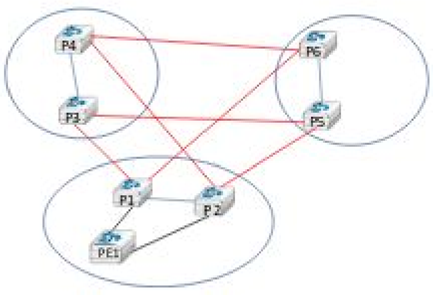
We decided to select a /64 for our external link addresses for the simple reason that it was observed around the other ISPs who worked with us that the /64 came out most often in eBGP link addresses and so we decided to do a "standardization".

1.2 Network topology

First network topology :
Before showing you our final topology, let us explain what the first one we did looked like.

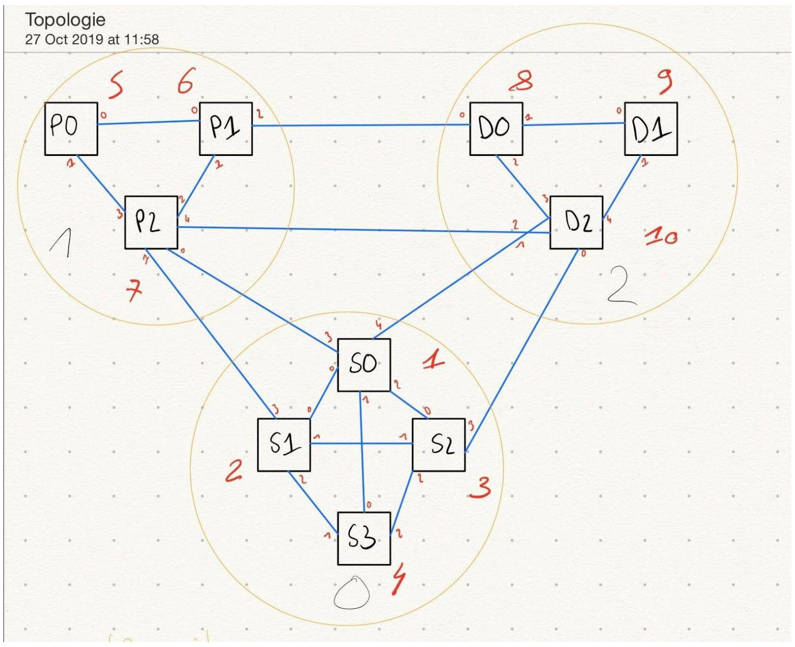
Based on the article written by Mr. Bonaventure (http://informatique.umons.ac.be/networks/igen/downloads/itc21-igen-bqu.pdf?fbclid=IwAR3b5W4HnUiWOTUVgbU9wO4aAzRyyVr5bLyHv4_A41AEwga2vq0EVqFf3Ds) we had decided to group a total of 3 routers per pop except for the backbone pop that contains 4 of them.

We were offered a different set of layouts at the beginning of the project and decided to choose a "EVEN-ODD" topology as you can see in the picture below. Our final topology will include 4 routers at the backbone POP level with an odd number of routers in the other POPs (in our case, it will be 3). The advantage of this is that adding routers by empowerment will be simplified since we know that we will always have the same number of routers per POP.



(a) "Even-odd" connection

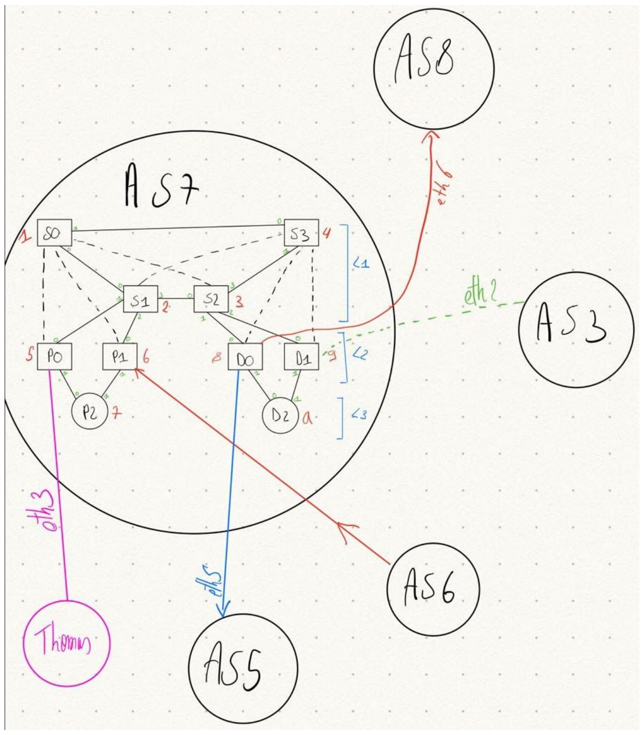
Now let me show you to the topology we had chosen at that time :



This topology respects the rule on page 4 of the article as "robust to failure", there is at least one router per pop except the backbone pop that is connected by two links to 2 different backbone routers

Definitive topology :

After considering the addition of reflector roads, hierarchy levels and other levels, we have finally arrived at this topology, which we will study in detail later in this report



1.3 OSPF

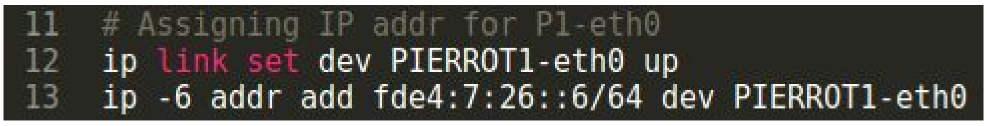
We will now study all the modifications made to our files in order to run OSPF within our network

Our ospf file reflects the topology at the level of physical links linking each router, a router in our case contains a maximum of 7 interfaces. Each interface will therefore be linked via a physical link to the interface of another router of our choice.

Below is the structure of the configuration files of the PIERROT1 router used as an example. We will study the additions to the file that allowed this router to be part of this exchange between routers that is OSPF.

```
<nomDeNotreFile>/grp7_cfg/
├── PIERROT1_start    << we gave modifications to this file
├── PIERROT_boot  << we did not add modifications in this file
└── PIERROT1/
    ├── PIERROT1_ospf.conf   << We gave modifications to this file
    ├── PIERROT1_zebra.conf << We gave modifications to this file
    └── ld.so.conf  << we did not add modifications in this file
```

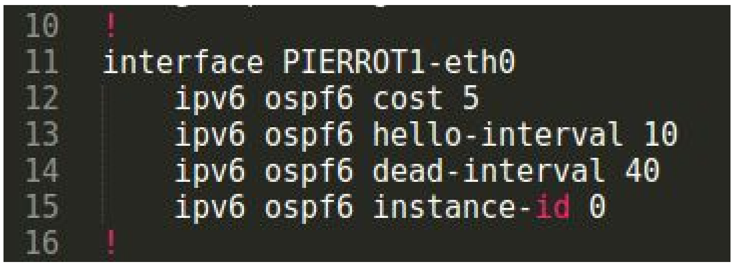
PIERROT1_start :



```
# Assigning IP addr for P1-eth0
ip link set dev PIERROT1-eth0 up
ip -6 addr add fde4:7:26::6/64 dev PIERROT1-eth0
```

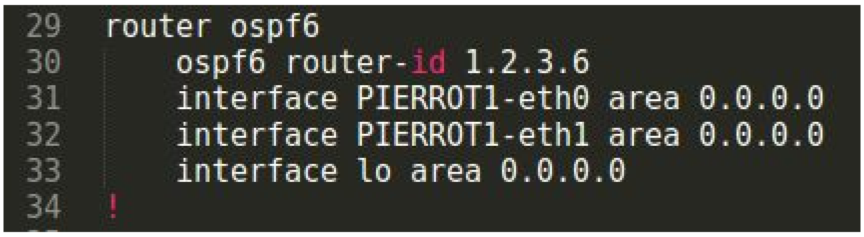
You can see in the start file of PIERROT1 it is 2 lines that will describe the link address between the SAMIR1 router and PIERROT1. The system is as follows, in the "26", the "2" corresponds to the router id of SAMIR2 and the "6" to the router ID of PIERROT1 and the "2" comes before the "6" because 2 is numerically smaller than 6 and the "6" at the end of the address corresponds to the router ID of PIERROT1 which means that we have this address in the PIERROT1 start file. In this example, we have awakened the eth0 interface of PIERROT1, as a reminder, a router can have up to 7 interfaces.

PIERROT1_ospf.conf :



```
!
interface PIERROT1-eth0
    ipv6 ospf6 cost 5
    ipv6 ospf6 hello-interval 10
    ipv6 ospf6 dead-interval 40
    ipv6 ospf6 instance-id 0
!
```

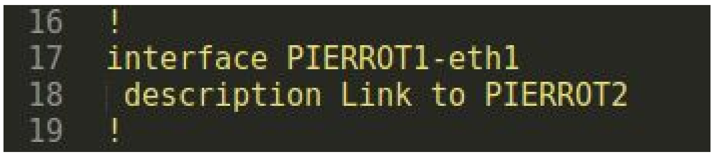
In the router's OSPF file, the first step is to initiate the different values required by OSPF such as the frequency at which it will send its hello messages, the cost associated with the route from the interface etc. In this screenshot, I showed the example of the initialization of the parameters related to the eth0 interface of PIERROT1. You will notice that the id instance is always set to 0, indeed, the id instance allows to differentiate OSPF communities but in our case, we only want to make BGP communities, so we will keep this set parameter at 0 (ref : https://learningnetwork.cisco.com/thread/32497)



```
router ospf6
    ospf6 router-id 1.2.3.6
    interface PIERROT1-eth0 area 0.0.0.0
    interface PIERROT1-eth1 area 0.0.0.0
    interface lo area 0.0.0.0
!
```

Still in the ospf file, after having initiated the values for the different interfaces, it is important not to forget to specify the router id, we have decided for the router id to have a prefix which is 1.2.3 and the last digit corresponds to the router ID on the topology scheme we proposed above. In must also specify to which area each interface of the router points. It should be noted that in the context of the areas, it is the vector routing distance that is applied and the area system being quite complicated to set up and obsolete, we have decided to keep all our routers in the same area.

PIERROT1_zebra.conf :



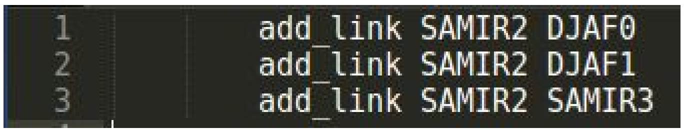
```
!
interface PIERROT1-eth1
 description Link to PIERROT2
!
```

The zebra file always follows the same structure, we bind each interface of the corresponding router to it’s neighbour connected via a physical link

We study below a particular file which is the configuration file of our topology. We also made mummifications to this file in order to create physical links between the routers of our topology.

<nomDeNotreFile>/grp7_cfg/
├── PIERROT1_start   << we gave modifications

grp7_topo_conf :



```
        add_link SAMIR2 DJAF0
        add_link SAMIR2 DJAF1
        add_link SAMIR2 SAMIR3
```

Here are the commands we have added here in our configuration file to create the physical links necessary for OSPF to work properly. The add_link function will therefore create a physical link between the 2 routers given in the parameter.

1.4 Configuration file script

You will find in the folder scripts/ 3 mako files and 6 json files. There is a mako file for the template of each bgp, start and ospf file. To recreate the files automatically, you must type these commands :

these first commands will be used to automatically recreate the bgp files of the routers, the samir correspond to the routers labeled S in our topology, the djaf to the D and the pierrot to the P

```
./make_bgp.py -i samir_bgp.json -t bgp.mako -m
./make_bgp.py -i djaf_bgp.json -t bgp.mako -m
/make_bgp.py -i pierrot_bgp.json -t bgp.mako -m
```

these first commands will be used to automatically recreate the start files of the routers, the samir correspond to the routers labeled S in our topology, the djaf to the D and the pierrot to the P

```
./make_start.py -i samir_start.json -t start.mako -m
./make_start.py -i djaf_start.json -t start.mako -m
./make_start.py -i pierrot_start.json -t start.mako -m
```

these first commands will be used to automatically recreate the routers' ospf files, the samir correspond to the labeled S routers in our topology, the djaf to the D and the pierrot to the P

```
./make_ospf.py -i samir_ospf.json -t ospf.mako -m
./make_ospf.py -i djaf_ospf.json -t ospf.mako -m
```

```
./make_ospf.py -i pierrot_ospf.json -t ospf.mako -m
```

Part 2 : *eBGP, iBGP, sharing cost link, client/provider link, Route Reflector*

The second part of the project was the set up of our Border Gateway Protocol (BGP), itself being separated into 2 sections : external (eBGP) and internal (iBGP) . This setup needs the creation of a new file for each router (pierrot1_bgp.conf, for example) as well as a new line in the *xxxx_start* file indicating the path to the BGP service with the new file path. The part about the route reflectors (2.1 Route reflector) will be explained before the actual BGP links configuration although a route reflector system is based on BGP because it was the reason for the new topology of our network (which will be explained in that very part).

2.1 Route reflector

First of all, to avoid a full mesh between our routers (each router is connected to every other routers, which would be cumbersome and would add a lot of work if we would want to put more routers in our network in the future), we inserted route reflectors in our topology, allowing us to reduce the total amount of physical connections between our routers. These route reflectors will have some knowledge of the topology and will decide which router a packet has to be sent to. We had to build a 2-levels hierarchy or route reflectors. Its implementation requires us to redesign our original topology. Here's a rapid presentation of the new one : the first layer (L1) contains 4 routers mapped as Tier 1 route reflectors, connected in full mesh. The second layer L2 presents the Tier 2 route reflectors, disposed into 2 sets of 2 routers. The third and last layer L3 is composed of standard-type routers.
Why different tier ? The purpose of this project is to simulate the configuration of a large ISP. In that sense, having multiple levels of route reflector serves multiple purposes :

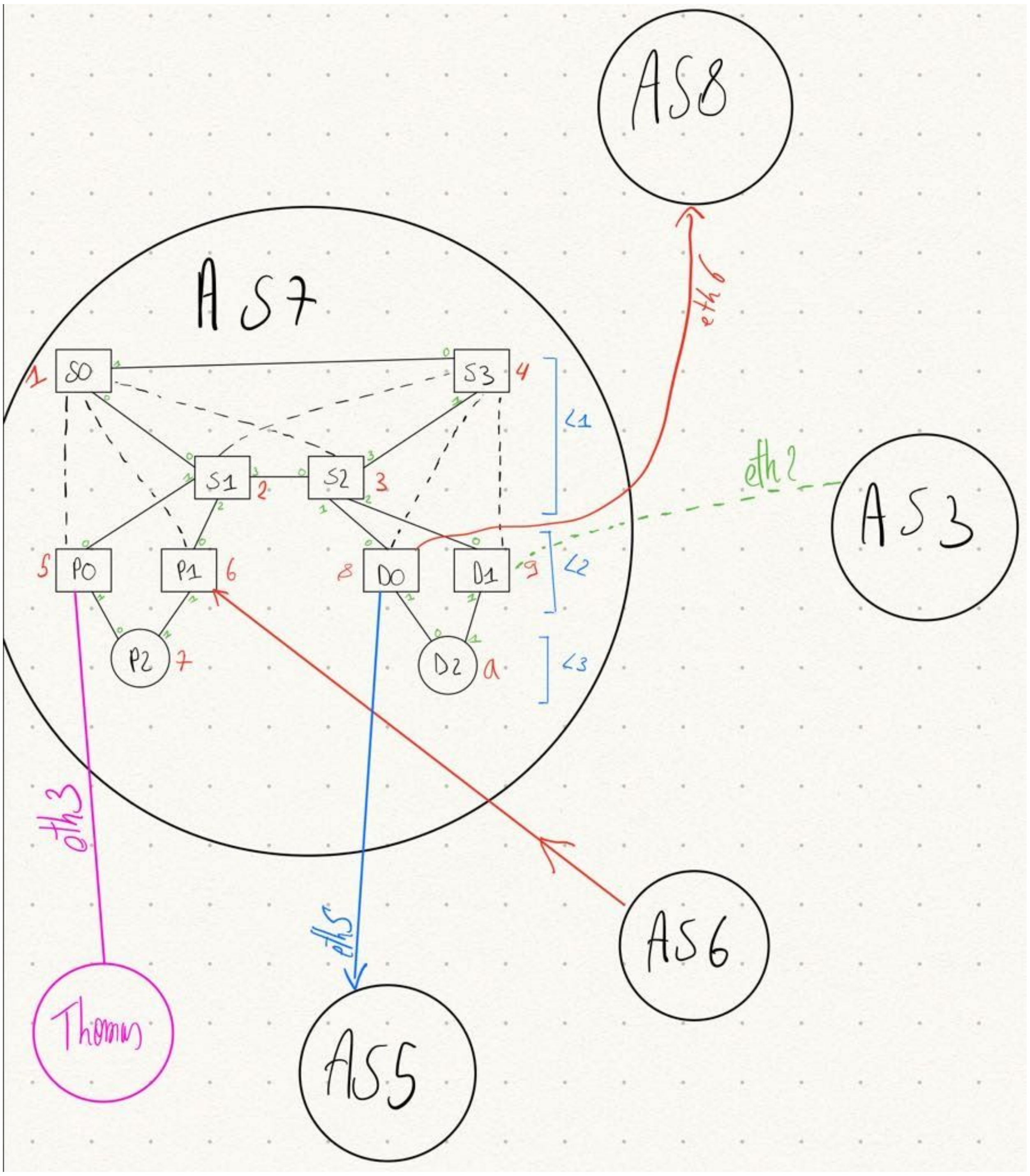
- A reduction of the amount of needed connection : only the top tier of route reflector has to be implemented in full mesh. If the top tier is actually the only level of route reflectors, a lot of connections would have to be added (unless you decide to only have a few of them, but they might be quickly overloaded). In case of multiple levels, the top tier can send packets to a "general direction", by relegating some work to the lower tier, while keeping a relatively small size, hence limiting the amount of connections.
- Easier scaling and flexibility : the top tier serves as a "central unit" to which lower tiers can be attached. These lower tiers could be used to manage different part of the network, and an easy connection with the "central unit" would keep this part connected to the rest

of the network. The top tier route reflectors are in charge of managing these different parts.

<u>Deeper view into our route reflectors</u>

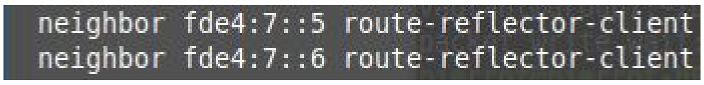
Starting with the first layer, they're connected in full mesh, as required by the project's instructions. This allow the data to be dispatched to the correct router/sub-layer route reflectors more easily and rapidly, considering more direct links. Note that this full-mesh topology is only true for the BGP (internal, more precisely) connections. Indeed, after deciding the destination for the data packet, the BGP protocol will make use of the OSPF protocol to get to that destination through the fastest route. (In our schema below, a BGP connection is to be considered for each physical link). Our 4 top tier route-reflector routers in this layer (S0, S3, S2 and S1) are thus all connected to each other by BGP, but only physically connected in a circle. Putting 4 route reflectors in this first tier might be much, but we wanted to have a strong "core" to be able to allow multiple routers to be added in the future, as an ISP usually have a lot more routers.
Following the first layer comes the second tier route reflectors : P0 and P1 on one side and D0 and D1 on the other. You'll notice that no link whatsoever has been created between routers in any group (group P0-P1 and group D0-D1) as we wouldn't need it, because it would only need two hops for a paquet to get from one router to the other, and it might cause deflection when more routers are added to this layer or the lower one. The 2 groups in this layer are not directly connected to each other, but can still communicate through the first layer.
Finally comes the 3rd and last layer of our topology, composed of the regular routers P2 and D2, without any "special" functionality.

Accordingly to the project guidelines, all routers are connected to 2 routers from the above layer for redundancy (except for the top layer, obviously, which is in a full-mesh configuration). This double connection allows for a more secure topology : if one (physical) link happens to break, both of the router affected by this failure still can communicate to the rest of the network. It was not required to anticipate a router break, so we didn't apply any policies for it, and a failure in S1 or S2 would break down our network, but a solution might be to add physical links between P0-S0, P1-S0, D0-S3 and D1-S3.



Implementation
Here are the lines of code you can find in order to initialize our routers as tier 2 and tier 1 route reflectors.

samir1_bgp.conf



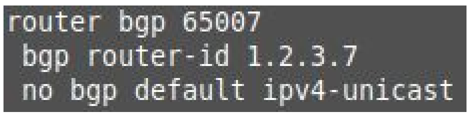
```
  neighbor fde4:7::5 route-reflector-client
  neighbor fde4:7::6 route-reflector-client
```

Here's the code for the top tier route reflector (S1) configuration. These lines advertise the neighbor routers fde4:7::5 (P0) and fde4:7::6 (P1) as its clients (the same lines can be found in the *samir0_bgp.conf*, which has the same route reflector clients). This will allow the S1 router to manage the information received by P0 and P1 and communicate it with the other router of the same layer. A variant of these lines is contained in the S3, S2, P0, P1, D0 and D1 BGP files with their according clients.

2.2 iBGP

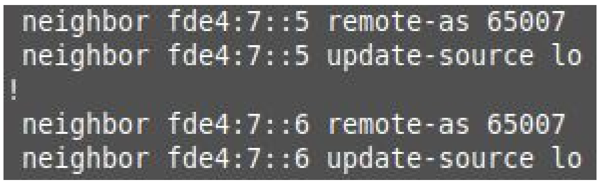
A lot has already been explained about our new topology, but here are more information about iBGP.
The internal BGP serves the purpose of communicating information such as the different neighbors, the different known addresses, … It is used with the route reflector system for the reasons explained in the previous section. As for the route reflectors, the configuration has to be implemented through the BGP files, and the code lines will now be explained.

pierrot2_bgp.conf



```
router bgp 65007
 bgp router-id 1.2.3.7
 no bgp default ipv4-unicast
```

These first lines can be associated to the identification of the router in the BGP. "*router bgp 65007"* indicates a new BGP router in the autonomous system (AS) number 65007, which is actually our AS number. Each of our router BGP file must have that exact line. Next comes the actual identification number of this specific router : *1.2.3.7* (the last number corresponding to the router number, this screenshot is taken from the P2's BGP file), which identifies him between the different routers present in the AS 65007. We chose to use the same ID number that the one contained in the OSPF file.
Lastly, a bit less important is the indication that we will not be using IPv4 (we only use IPv6 in our network).



```
 neighbor fde4:7::5 remote-as 65007
 neighbor fde4:7::5 update-source lo
!
 neighbor fde4:7::6 remote-as 65007
 neighbor fde4:7::6 update-source lo
```

Here, you can find the declaration of the router neighbors (P0 and P1 are indeed P2's BGP neighbors in our topology). You get 2 commands for each neighbor : the first one points out the AS appartenance of this neighbor. A *"remote-as"* followed by the number of our AS shows a router of our system.
The second command activates the source address or interface to use for the BGP session to this neighbor (here the loopback address).

```
address-family ipv6 unicast
 neighbor fde4:7::5 activate
 neighbor fde4:7::6 activate
exit-address-family
```

This last part is needed to activate the IPv6 interface of the Frrouting operation : it usually works with IPv4, so we need to manually activate the IPv6 interface for our neighbors.

2.3 eBGP

The external BGP is used to connect our network (AS) to another one, an **external** one. After creating and modifying our BGP files for each router for the iBGP, we can now add commands to set up our eBGP.

pierrot0_bgp.conf

```
neighbor fde4::1 remote-as 64512
neighbor fde4::1 interface thomas
```

Similarly to the iBGP part, the eBGP make use of the neighbor command, but with the difference that the AS number is not ours. Indeed, 64512 correspond to Thomas' (the course assistant) router. It will allow our program to discern this neighbor as a foreign neighbor and treat it accordingly.
This connection with Thomas' network is quite special, as a specific interface has been initialized for us to connect to the router. It's the only network that gives us access to the Internet (without a correct connection with it, "ping6 google.com" would fail, for example).
We normally don't have to connect directly to Thomas router (not stated in the multi-AS topology from the project), however, it was the first fully functional router to which we could have a connection, and was a great way to test our first eBGP session.

Different types of links exist in the eBGP. We will now see the client-provider link as well as the shared-cost link.

Client-provider link

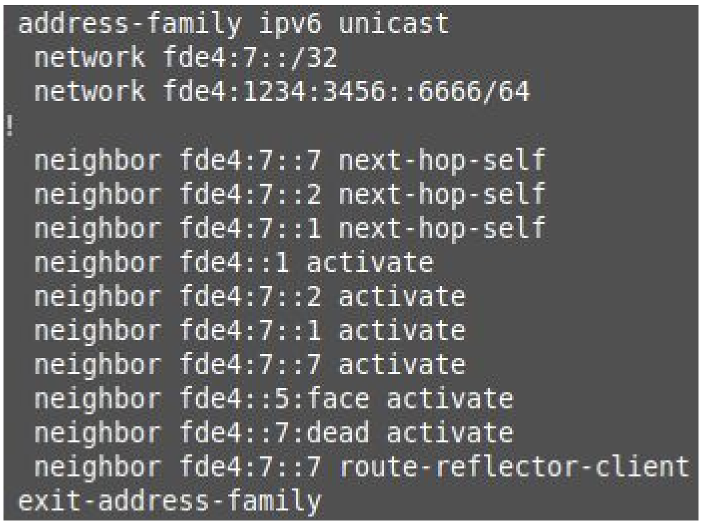
This link is made between an ISP and its client (an ISP can itself be a client of another one). Of course, this link is implemented differently depending on which side of the link the router is.

```
neighbor fde4::7:dead remote-as 65006
neighbor fde4::7:dead update-source grp6
```

The AS 65006 correspond to our client's AS. In order to establish a connection with them, we had to agree on a common address to share. The chosen address was thus fde4::7:dead

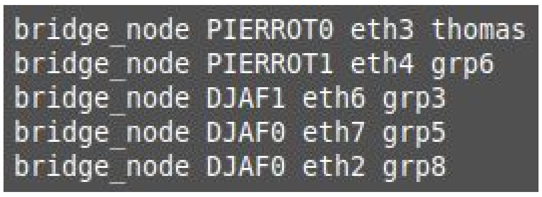
```
neighbor fde4::5:face remote-as 65005
neighbor fde4::5:face update-source grp5
```

```
neighbor peer-group upstream
```



```
address-family ipv6 unicast
 network fde4:7::/32
 network fde4:1234:3456::6666/64
!
 neighbor fde4:7::7 next-hop-self
 neighbor fde4:7::2 next-hop-self
 neighbor fde4:7::1 next-hop-self
 neighbor fde4::1 activate
 neighbor fde4:7::2 activate
 neighbor fde4:7::1 activate
 neighbor fde4:7::7 activate
 neighbor fde4::5:face activate
 neighbor fde4::7:dead activate
 neighbor fde4:7::7 route-reflector-client
exit-address-family
```

grp7_topo_conf



```
bridge_node PIERROT0 eth3 thomas
bridge_node PIERROT1 eth4 grp6
bridge_node DJAF1 eth6 grp3
bridge_node DJAF0 eth7 grp5
bridge_node DJAF0 eth2 grp8
```

Beware, all the previous modification added to our BGP files will not work as is, we still have to add these lines into the *grp7_topo_conf.* This *"bridge_node"* modification will define the interfaces eth3 to eth7 of our virtual machine running the network to the interface of our eBGP-enabled routers. For example, the *eth3* interface of our virtual machine is mapped to the *thomas* interface of the router *PIERROT0* (P0). Without these precisions, the routers will not be able to communicate to the "outside world", as they would be in a closed simulation.

2.3 Shared cost link

We were asked to configure a shared cost connection from the BGP protocol point of view with group 3. The latter was done via router D1 (DJAF1). Through this link, we will only announce the prefix of our client who is the group6. You will see in our BGP configuration file at the DJAF1 router that this announcement was not made using the network command for the simple reason that following an address redesign on our side and on our customer's side, we no longer had time to reconfigure the link between us a second time, which means that we no longer momentarily announce their prefix because the connection between our 2 groups is still not made

2.4 Client provider link

It should be noted that in our current configuration, we have 2 ISP suppliers. The first is our "official" supplier and is Group 8 with which we are connected via the D0 router (DJAF0). You will also notice that we have tried to share the load of bgp connections between the different routers to avoid any overload. This group 8 provider will normally be expected to be able to provide us with the route to Thomas who gives us the routes to join the internet. In our acutels files, we know directly a route to Thomas but it was only a question of knowing it to test our network configuration. Our second supplier is Group 8, but the latter is only a24 backup supplier. It is there if and only if the link to group 8 breaks.

Part 3 : *Automated script, BGP communities, Security concern*

3.1 Automated script [SALEY ABDOU Djafarou]

for the moment I only have 2 scripts, one called initiate_network.sh which will clean up and then launch the network. I put a 1min20 timer in order to let the network converge completely so that the following tests are not affected by a lack of waiting. I have another script called test-interface.sh, you have to connect to a router, launch it and the router will test all interfaces (loopback and link addresses) of all other routers in the network and redirect the result to a file called pingResult-output.txt. If the file pingResult-output.txt already exists when you open the folder and it is full of data, you can delete it via the command => sudo rm pingResult-output.txt. The script once restarted will regenerate one, if you don't, the new data will concatenate at the end of the existing file and this could cause you to waste some time having to go down to the end of the file

the commands to follow are as follows. Suppose we try to reach all interfaces (loopback address and link address included) of a router x.

1) connect to the router x in question via the command :
sudo ./connect_to.sh grp_cfg/ x

2) run the interface test script via this command (don't forget to give the name of the router you are connected to as a parameter of this script) (the execution of this script takes a few seconds so please wait a little while so that all ping can be done):
./test_interface.sh x

3) You can exit the router with the command :
exit

4) You will find the result of the set in the output file => pingResult-output.txt

5) You can repeat the experience with a router y and the output of the latter will concatenate in a structured way to the output of the previous router in the file pingResult-output.txt

3.2 BGP policies [Roose Pierre-Rodéric]

Fitlers :
Some rules are set to manage the addresses announced between customer, provider and shared-cost peers. We use filters to announce or receive the correct prefixes from our peers. Here is the rules we set

- with providers : we accept all the prefixes they want to announce but give them only our prefixe and the prefixes known by our customers (we thus filter out the prefixes from our shared-cost peers).
- with the customers : we accept only their (and normally, their client's) prefix. We need to be careful as the topology of all the ASes for the project is a circle (we might learn our provider prefix from our provider **and** our client, if it forwards it all the way to us).
  We advertise them all our known prefixes.
- with the shared-cost peers : we accept all their prefixes and we announce them only our client's prefix and ours.

3.3 Security concern [Samir Marini]

OSPF protocol protection :

With simple authentication, the password goes in clear-text over the network,in order to exchange routing update information in a secure manner we decided to enable OSPF authentication. Based on the recommendations of the instructions, we wanted to use MD5 passwords for authentication.

However, after multiple searches, multiple unsuccessful attempts, and after asking Thomas. We concluded that FRRouting had not yet developed authentication for OSPFv3, which is the version we use

eBGP and iBGP protection :
To prevent interference with our routing tables, we decided to authenticate our iBGP and eBGP sessions.

To do this, we have enabled MD5 authentication for all our iBGP sessions using the command:
neighbor ip_address password

For our eBGP sessions, we were able to agree on a password with only one of the groups (the 8), so our eBGP connections are protected only with that group.

Firewall :
We set up a firewall in each router. We set it in the router's _boot file. We perform several filters with this firewall, including :

- OSPF packets coming from outside the network are dropped
- We have banned a set of global addresses, on the recommendation of this article: https://www.iana.org/assignments/ipv6-unicast-address-assignments/ipv6-unicast-address-assignments.xhtml
- Following the recommendations of the BCP38, we filter the addresses of hosts we do not know. In particular, if we receive our own fde4:7 prefix from the outside, we drop it.
- We allow ICMPv6
- We allow our locals addresses
- After some measures, it was decided to prevent DDOS by blocking ASs that send more than 15 packets per second.

The firewall is automated thanks to the scripts boot.mako, boot.json and make_boot found in scripts/

Conclusion

Through this project we have established the configuration of the 10 ruters within a large ISP. This experience taught us to establish the configuration of the OSPF and BGP protocol within a virtual network. In fact, it would be necessary to take into account several parameters such as the possible overload of routers, CPU capacity, RAM, the price of cable connections in our network and many other factors. Throughout the project we tried to think in the most visionary way possible by thinking about what this or that implementation choice would bring in the case where we would actually have to implement it on a large scale. This is why we found ourselves with 2 hierarchies of RRs to try to limit as many physical links as possible.

Bibliography

ospf :

https://learningnetwork.cisco.com/thread/32497

https://github.com/FRRouting/frr/blob/master/bgpd/bgpd.conf.sample?fbclid=IwAR3qkZTGxdU04J-Ox7zqrucyGsBlyO1uH2xp9Kek6RttVkIWfSSg9Ax8_1U

http://docs.frrouting.org/en/latest/bgp.html?fbclid=IwAR2qnBn5d8LlEgFcJV5X8PPFLbqxjYF6AVzpboNZ2My63Hy3NMHL6P1lI2s#route-reflector

Addressing plan :

https://community.cisco.com/t5/networking-documents/ipv6-subnetting-overview-and-case-study/ta-p/3125702?fbclid=IwAR0aAD2K_JwYmwgdsxL6Dp3OB8F2TXcC3GpSSYox6SiwFZcI-3LzE_Hm_do

Topology :

http://informatique.umons.ac.be/networks/igen/downloads/itc21-igen-bqu.pdf?fbclid=IwAR3b5W4HnUiWOTUVgbU9wO4aAzRyyVr5bLyHv4_A41AEwga2vq0EVqFf3Ds

About eBGP and IBGP :

https://www.cisco.com/c/en/us/support/docs/ip/border-gateway-protocol-bgp/13751-23.html?fbclid=IwAR3byshzWtfvNC19wvHK7beLHaoYG88sMs3hzvO0rn59olWCed_gOFRZTwo

http://www.mustbegeek.com/configure-ibgp-in-cisco-ios-router/?fbclid=IwAR1O3bZ4TkSvVt8VwKOk2SlLhafPS9pF-N3bbeEu6MOFhMAybYP1fFAD6jk#.XdbbjNHjLCJ

https://www.editions-eni.fr/open/mediabook.aspx?idR=335c365f0774ade9c1e42ff17cfb1bb4&exp=bgp&fbclid=IwAR17XQ4BQdRxLt3lEo11fkIyqzTcX62et0qYoBuTr7lU-58xCAQqs_Y2Fbc

https://books.google.be/books?id=cevlBQAAQBAJ&pg=PA445&lpg=PA445&dq=how%20to%20configure%20iBGP%20cisco%20router&source=bl&ots=H8ry3TKCuC&sig=ACfU3U0isiAPAFQp2yaFcItLb_jZjHt4Jg&hl=fr&sa=X&ved=2ahUKEwj08Kqv59blAhW1uXEKHYJ8BzM4ZBDoATAFegQICRAB&fbclid=IwAR1lWC4gMAu2Xm_4WxVUu05g5MbckbNYrVcjRMIfi6vGo3Azk1zXJ9y6y00#v=onepage&q=how%20to%20configure%20iBGP%20cisco%20router&f=false

https://books.google.be/books?id=cevlBQAAQBAJ&pg=PA445&lpg=PA445&dq=how+to+configure+iBGP+cisco+router&source=bl&ots=H8ry3TKCuC&sig=ACfU3U0isiAPAFQp2yaFcItLb_jZjHt4Jg&hl=fr&sa=X&ved=2ahUKEwj08Kqv59blAhW1uXEKHYJ8BzM4ZBDoATAFegQICRAB#v=onepage&q=how%20to%20configure%20iBGP%20cisco%20router&f=false

http://brianrlarson.blogspot.com/2014/07/bgp-lab-route-reflectors-and-clustering.html

About route reflector :

https://community.cisco.com/t5/routing/bgp-route-reflector-question-regarding-prefixes-not-being-seen/td-p/3725957?fbclid=IwAR2Op9YwTCqYGWmeasjz9GnuvQBMdUFP5zjiQo5u208Zz3xL_vPtB6si2Yk

https://nsrc.org/workshops/2017/apricot2017/bgp/bgp/preso/09-BGP-Communities.pdf

https://community.cisco.com/t5/routing/route-reflector-bgp/td-p/1602252

https://support.huawei.com/enterprise/en/doc/EDOC1100055018/a2762e21/configuring-a-bgp-route-reflector http://brianrlarson.blogspot.com/2014/07/bgp-lab-route-reflectors-and-clustering.html

https://networkdirection.net/articles/routingandswitching/routereflectordesign/

https://brianrlarson.blogspot.com/2014/07/bgp-lab-route-reflectors-and-clustering.html?fbclid=IwAR17XQ4BQdRxLt3lEo11fkIyqzTcX62et0qYoBuTr7lU-58xCAQqs_Y2Fbc