Groupe 15 - Rapport du Projet Mini-Internet

Juliette Bluem, Antoine Laguette & Guillaume Tisserand

17 décembre 2022

Table des matières

Partie I : Introduction

A l'issue de ces 2 années d'études IA2R nous avons développé les connaissances et les compétences nécessaires à la réalisation d'un projet à grande échelle comme la réalisation de notre propre réseau internet.

Ce projet va nous permettre de mettre en application et surtout corréler tout ce que nous avons pu apprendre précédemment. Cependant, la principale différence et difficulté résidera dans la syntaxe des commandes de configuration saisies. En effet, le projet mini internet est trop gros pour être modélisé physiquement avec de vrais équipements. De ce fait, tout est virtualisé via FRRouting et OpenVSwitch, deux solutions de virtualisation réseau.

Plusieurs étapes sont nécessaires pour mener à bien ce projet, la première consiste à réaliser la configuration de base de notre mini internet. C'est à dire créer un réseau local, le connecter à un AS (Autonomous System) puis de faire en sorte que tous les hôtes du réseau configuré puissent se contacter. Cette configuration de base va passer par la mise en place de l'adressage, de VLAN, routage OSPF mais également de l'analyse de lien ou traffic-engineering.

Les autres groupes doivent effectuer la même configuration mais avec un adressage différent puis on passera à la seconde étape, l'élaboration d'une configuration BGP Globale avec iBGP et eBGP.

Enfin nous verrons comment mettre en place une police de management pour administrer et gérer notre mini internet.

Partie II : Connectivité Intra-domaine

Comme énoncé dans l'introduction, cette première partie a pour vocation la configuration de base de notre réseau afin que les autres équipes puissent faire de même pour inter-connecter tous les réseaux configurés. Cette partie n'est pas négligeable dans le sens où si nous manquons la moindre configuration d'une adresse IP ou d'un protocole cela peut avoir un impact extrêmement important sur le bon fonctionnement de notre mini internet.

1 Connectivité LAN

On commence par configurer le réseau LAN. C'est une étape que nous maîtrisons car nous avons toujours débuté par cette étape en TP et dans n'importe quelle autre configuration de réseau. On adresse les interfaces des hôtes staff et student respectivement au plan d'adressage donné :

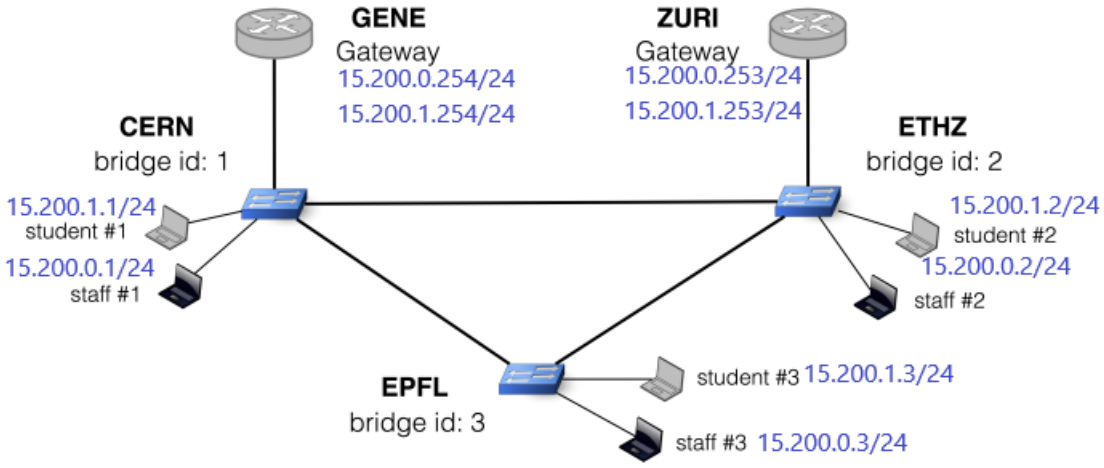

FIGURE 1 – Plan d'adressage

Une fois ce plan respecté nous mettons en place des VLAN pour que les étudiants ne puissent pas contacter le staff et inversement.
En nous intéressant plus aux routeurs, on adresse les interfaces puis on applique un routage dynamique OSPF en l'activant et en déclarant les réseaux voisins de chaque routeur

Après cela nous mettons en place le protocole STP qui nous permettra de conserver un lien de secours automatique si un des liens venait à se couper entre les commutateurs. Plus tard nous pourrions nous intéresser au protocole SNMP dans le cadre de la management policy.

Afin de nous assurer que tout le monde monde puisse maintenant communiquer et que nos règles de séparation Staff et Student soient bien en place, on effectue de simples ping.
Pour avoir plus de détails sur le trajet qu'empruntent nos données, nous effectuons des traceroute pour observer que toutes les données ne transitent pas que par un seul et unique trajet ou équipement.
De notre côté, tout fonctionne (évidemment). Par exemple si on réalise un traceroute entre Staff#2 et Student#1 on peut observer que les données transitent jusqu'au routeur puis redescendent vers les destinations car student et staff ne sont pas dans le même vlan donc ne peuvent pas communiquer via les commutateurs mais les routes réseaux sont configurées donc ils peuvent communiquer par réseau interposé grâce à l'OSPF configuré.

2 Connectivité AS

De la même manière que pour le LAN nous configurons toutes les interfaces de tous les routeurs de notre AS. C'est une partie qui demande beaucoup de minutie car ils y a un gros volume d'adressage à faire. Ainsi, nous respectons le schéma suivant :

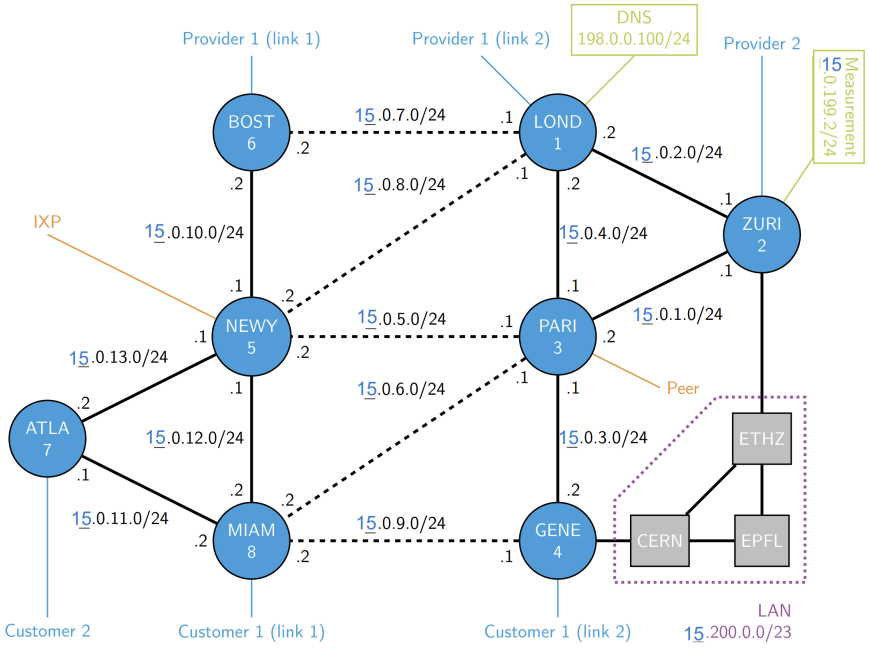

FIGURE 2 – Plan d'adressage

Nous avons configuré notre OSPF afin d'avoir un routage dynamique. Sa configuration est exactement la même que pour que notre LAN, c'est à dire en déclarant nos réseaux voisins sur chaque routeur. Ainsi, chaque hôte va pouvoir communiquer avec les autres.

Également, il ne faut pas oublier d'adresser les hôtes des routeurs de notre AS et les interfaces de loopback. Les interfaces de loopback vont nous servir par la suite pour la configuration de l'iBGP car iBGP établit des connexions TCP/IP grâce à ces interfaces. de même que pour le réseau LAN nous effectuons des ping et traceroute pour observer par où transitent nos données comme ci-dessous avec un traceroute entre l'hôte de Paris et l'hôte d'Atlanta

```
root@staff_2:~# traceroute 15.103.0.1
traceroute to 15.103.0.1 (15.103.0.1), 30 hops max, 60 byte packets
 1  15.200.0.253 (15.200.0.253)  2.714 ms  2.682 ms  2.667 ms
 2  PARI-GENE.group15 (15.0.3.1)  3.757 ms  3.654 ms PARI-ZURI.group15 (15.0.1.2)
2.675 ms
 3  host-PARI.group15 (15.103.0.1)  3.713 ms  3.622 ms  3.601 ms
```

FIGURE 3 – Traceroute hôte Paris vers hôte Atlanta

3 Traffic-engineering

En tant qu'opérateur réseau, notre but est de fournir des performances optimales à nos clients.
Pour cela, nous allons devoir définir plus finement notre configuration OSPF.

Les liens continentaux disposent tous d'une bande passante de 25 Mb/s. Les liens du LAN supportent 10 Mb/s. Concernant les liens transatlantiques, notre AS peut présenter une configuration parmi quatre. Nous devons donc bien sûr la connaître afin d'être certain de ce que nous proposons au client et de l'optimiser le mieux possible.

Afin d'identifier cette topologie, nous effectuons ce que nous appelons du traffic-engineering. Ce principe est basé sur l'étude des équipements et des liens qui composent notre réseau.

Via un outil appelé IPerf nous pouvons réaliser cette identification. En effet IPerf nous permet de faire une état des lieux de notre bande réseau afin d'en déterminer les limites.

Pour trouver la configuration des liens transatlantiques nous allons tester deux liens :
BOST – LOND

```
root@LOND_host:~# iperf3 -c 15.106.0.1 -b 25M
Connecting to host 15.106.0.1, port 5201
[  4] local 15.101.0.1 port 55666 connected to 15.106.0.1 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec  2.82 MBytes  23.7 Mbits/sec    0    105 KBytes
[  4]   1.00-2.00   sec  3.00 MBytes  25.2 Mbits/sec    0    105 KBytes
[  4]   2.00-3.00   sec  3.00 MBytes  25.2 Mbits/sec    0    105 KBytes
[  4]   3.00-4.00   sec  3.00 MBytes  25.2 Mbits/sec    0    105 KBytes
[  4]   4.00-5.00   sec  3.00 MBytes  25.2 Mbits/sec    0    105 KBytes
[  4]   5.00-6.00   sec  3.00 MBytes  25.2 Mbits/sec    0    105 KBytes
[  4]   6.00-7.00   sec  2.88 MBytes  24.1 Mbits/sec    0    105 KBytes
[  4]   7.00-8.00   sec  3.00 MBytes  25.2 Mbits/sec    0    105 KBytes
[  4]   8.00-9.00   sec  3.00 MBytes  25.2 Mbits/sec    0    105 KBytes
[  4]   9.00-10.00  sec  3.00 MBytes  25.2 Mbits/sec    0    105 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-10.00  sec  29.7 MBytes  24.9 Mbits/sec    0             sender
[  4]   0.00-10.00  sec  29.5 MBytes  24.7 Mbits/sec                  receiver
```

FIGURE 4 – Resultat Iperf entre Boston et London

Nous observons que le lien BOST – LOND peut supporter des échanges à 25Mb/s.
Sur nos quatre configurations d'AS possibles, seules deux ont cette capacité sur ce lien.

Nous voulons donc les différencier par le lien MIAM-PARI qui sera soit à 1Mb/s si nous sommes dans la configuration D, soit 25Mb/s dans le cas A.

* PARI – MIAM

```
Connecting to host 15.108.0.1, port 5201
[  4] local 15.103.0.1 port 47292 connected to 15.108.0.1 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec  1.27 MBytes  10.7 Mbits/sec   17   1.41 KBytes
[  4]   1.00-2.00   sec  63.6 KBytes   521 Kbits/sec   16   2.83 KBytes
[  4]   2.00-3.00   sec   191 KBytes  1.56 Mbits/sec   17   1.41 KBytes
[  4]   3.00-4.00   sec  63.6 KBytes   521 Kbits/sec   14   1.41 KBytes
[  4]   4.00-5.00   sec   127 KBytes  1.04 Mbits/sec   14   1.41 KBytes
[  4]   5.00-6.00   sec   127 KBytes  1.04 Mbits/sec   13   1.41 KBytes
[  4]   6.00-7.00   sec   127 KBytes  1.04 Mbits/sec   18   2.83 KBytes
[  4]   7.00-8.00   sec   127 KBytes  1.04 Mbits/sec   14   2.83 KBytes
[  4]   8.00-9.00   sec  63.6 KBytes   521 Kbits/sec   19   2.83 KBytes
[  4]   9.00-10.00  sec   127 KBytes  1.04 Mbits/sec   15   1.41 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-10.00  sec  2.27 MBytes  1.90 Mbits/sec  157             sender
[  4]   0.00-10.00  sec  2.10 MBytes  1.76 Mbits/sec                  receiver

iperf Done.
```

FIGURE 5 – Résultat Iperf entre Miami et Paris

Le lien PARI – MIAM est visiblement limité à 1Mb/s.
Nous éliminons donc la configuration A et pouvons conclure que nous sommes dans la configuration suivante :

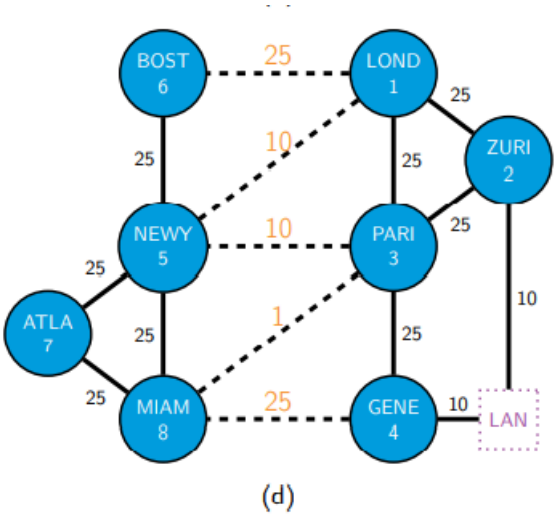

FIGURE 6 – Configuration D

Maintenant que nous avons parfaitement connaissance de la capacité des liens de notre AS, nous allons définir les coûts OSPF afin de privilégier certains liens selon des règles bien précises.

Nous choisissons dans un premier temps d'attribuer tous les liens 25Mb/s à un coût de 1, ceux de 10Mb/s, un coût de 10 et ceux de 1Mb/s, un coût de 100.
De cette façon, nous nous assurons de respecter les règles suivantes :

1. Aucun trafic ne traverse des liens transatlantiques deux fois
2. Les liens de plus forte bande passante sont utilisés en priorité pour les trafic intercontinentaux

Nous voulons maintenant modifier notre règle afin d'assurer une répartition de charge entre plusieurs chemins.

Tout trafic entre MIAM et NEWY doit être réparti sur deux chemins de même coût, le direct et celui passant uniquement par ATLA.
De la même façon, le trafic entre ZURI et LOND sera reparti entre deux chemins, le direct et celui passant par PARI. Pour cela, nous passons simplement le coût des liens directs à 2 au lieu de 1.

Enfin, nous devons nous assurer que le trafic entre ATLA et ZURI est réparti sur les deux liens transatlantiques à forte bande-passante. C'est le cas avec notre architecture actuelle.

Ci-dessous, un schémas récapitulatif de nos choix de coûts sur notre réseau :

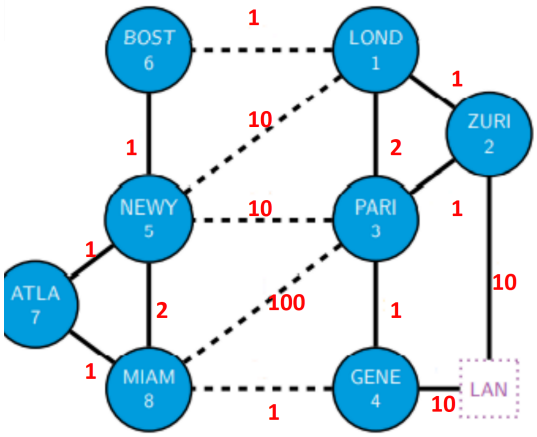

FIGURE 7 – Coûts des liens de notre AS

Nous mettons donc en place sur nos routeurs cette configuration de coût OSPF. Notons qu'ils faut absolument indiquer aux routeurs qu'ils ont le droit d'utiliser plusieurs chemins de même coûts.

Partie III : Configuration BGP globale

Le protocole BGP (Border Gateway Protocol) est un protocole de routage qui permet aux réseaux autonomes (AS, Autonomous System) de communiquer entre eux afin d'échanger des informations de routage sur Internet. BGP est utilisé pour acheminer les données sur Internet en utilisant des chemins les plus efficaces et fiables possibles.

BGP utilise une approche de routage par état de lien pour déterminer le meilleur chemin vers une destination. Les informations de routage sont échangées entre les routeurs BGP d'un réseau autonome à l'autre en utilisant des annonces de routes. Chaque réseau autonome possède une table de routage qui contient des informations sur les destinations atteignables et les chemins les plus efficaces pour y accéder. Lorsqu'un routeur BGP reçoit une annonce de route, il la compare à sa propre table de routage pour déterminer s'il doit mettre à jour sa table ou ignorer l'annonce.

BGP est un protocole essentiel pour le fonctionnement d'Internet, car il permet de connecter les différents réseaux autonomes entre eux et de garantir que les données arrivent à leur destination de manière fiable.

Il se compose de deux variantes : eBGP à l'extérieur de l'AS et iBGP à l'intérieur de l'AS.
Le protocole eBGP est utilisé pour échanger des informations de routage entre différents réseaux autonomes, ce qui permet d'acheminer les données sur Internet à travers les différents réseaux autonomes.
Le protocole iBGP est utilisé pour échanger des informations de routage entre les routeurs d'un même réseau autonome, ce qui permet de garantir que les données sont acheminées de manière efficace au sein du réseau autonome.
Ces deux variantes se configurent de la même façon. Si l'ASN du neighbor est le même que l'ASN local, le routeur saura que c'est du iBGP.

Arbre de décision : Sélection de la route la plus précise (masque le plus grand) puis Arbre de décision interne au protocole BGP. Quand une route a été choisie, elle est envoyée aux autres routeurs (iBGP)

1 iBGP

Dans cette partie nous configurons des sessions BGP sur l'ensemble de nos routeurs et non pas uniquement ceux directement connectés.

Cette étape a pour objectif de créer un réseau dit full-mesh (ou maillé), permettant de garder les connexions fonctionnelles, et ce, même si un ou plusieurs routeurs tombent en panne.

Nous configurons chaque routeur de la manière suivante :
En mode configuration, nous entrons sur la celle du routage de bgp 15. Nous activons l'address family ipv4 unicast. C'est une fonctionnalité de protocole de routage qui permet de configurer des routes IPv4 uniques pour acheminer les données sur un réseau.
On configure enfin l'ensemble des adresses loopback de chaque routeur présent au seins de l'AS 15 grace à ces deux commandes :

```
X_router(config-router)#neighbor 15.X.0.1 remote-as 15
X_router(config-router)#neighbor 15.X.0.1 update-source lo
```

On utilise ces adresses car si nous utilisons une adresse IP d'interface, et que celle-ci tombe en panne, la relation BGP disparaît.
Bien sûr, cela ne fait sens qu'uniquement si notre réseau dispose de liens redondants vers ce voisin.

Pour vérifier que les sessions BGP sont établies nous utilisons la commande suivante.

```
router# show ip bgp summary
```

```
PARI_router# sh ip bgp summary

IPv4 Unicast Summary:
BGP router identifier 15.153.0.1, local AS number 15 vrf-id 0
BGP table version 5
RIB entries 9, using 1656 bytes of memory
Peers 8, using 163 KiB of memory

Neighbor        V         AS MsgRcvd MsgSent   TblVer  InQ OutQ  Up/Down State/PfxRcd
15.151.0.1      4         15     394     399        0    0    0 06:31:18            0
15.152.0.1      4         15     396     401        0    0    0 06:33:09            0
15.154.0.1      4         15     397     402        0    0    0 06:33:37            1
15.155.0.1      4         15     396     399        0    0    0 06:31:06            2
15.156.0.1      4         15     393     398        0    0    0 06:30:13            0
15.157.0.1      4         15     403     409        0    0    0 06:34:15            1
15.158.0.1      4         15     396     400        0    0    0 06:32:40            1
179.0.41.2      4         16     339     341        0    0    0 05:33:55            3

Total number of neighbors 8
```

FIGURE 8 – Session BGP du router PARI

Nous pourrions également utiliser l'outil looking glass pour pouvoir observer la table de routage IGP du routeur PARI par exemple

```
2022-12-05T13:36:32
BGP table version is 67, local router ID is 15.153.0.1, vrf id 0
Default local pref 100, local AS 15
Status codes:  s suppressed, d damped, h history, * valid, > best, = multipath,
               i internal, r RIB-failure, S Stale, R Removed
Nexthop codes: @NNN nexthop's vrf id, < announce-nh-self
Origin codes:  i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*>i1.0.0.0/8        15.152.0.1               0    100      0 14 1 i
*>i2.0.0.0/8        15.155.0.1               0    100      0 2 i
*>i3.0.0.0/8        15.152.0.1               0    100      0 14 1 3 ?
*>i4.0.0.0/8        15.155.0.1               0    100      0 2 4 ?
*>i5.0.0.0/8        15.152.0.1               0    100      0 14 1 4 5 ?
*>i6.0.0.0/8        15.155.0.1               0    100      0 2 4 6 ?
*>i7.0.0.0/8        15.152.0.1               0    100      0 14 7 i
*>i8.0.0.0/8        15.155.0.1               0    100      0 8 i
* i11.0.0.0/8       15.151.0.1               0    100      0 13 11 i
*>i                 15.152.0.1               0    100      0 14 11 i
* i                 15.156.0.1               0    100      0 13 11 i
* i12.0.0.0/8       15.151.0.1               0    100      0 13 12 i
*>i                 15.152.0.1               0    100      0 14 12 i
* i                 15.156.0.1               0    100      0 13 12 i
*>i13.0.0.0/8       15.151.0.1               0    100      0 13 ?
* i                 15.156.0.1               0    100      0 13 ?
*>i14.0.0.0/8       15.152.0.1               0    100      0 14 ?
* i15.0.0.0/8       15.155.0.1               0    100      0 ?
*>                  0.0.0.0                  0         32768 ?
* i                 15.157.0.1               0    100      0 ?
* i                 15.154.0.1               0    100      0 ?
* i                 15.152.0.1               0    100      0 ?
* i                 15.156.0.1               0    100      0 ?
* i                 15.151.0.1               0    100      0 ?
*> 16.0.0.0/8       179.0.41.2               0             0 16 i
*>i17.0.0.0/8       15.158.0.1               0    100      0 17 i
*                   179.0.41.2                             0 16 17 i
*>i18.0.0.0/8       15.157.0.1               0    100      0 18 i
*                   179.0.41.2                             0 16 18 i

Displayed  16 routes and 29 total paths
```

FIGURE 9 – looking glass iBGP

On vérifie ainsi que l'ensemble de notre AS est bien connectée selon le maillage mis en place.

Cela termine la partie de configuration du BGP interne (iBGP)

2 eBGP

La configuration de BGP externe se fait en coordination avec les autres AS.

Dans un premier temps, nous nous sommes organisés avec nos voisins directs afin de déterminer les adresses et réseaux de liaison entre les AS.

AS 15			Neighbor AS			
Router	**Role**	**IP**	**AS**	**Router**	**Role**	**IP**
BOST	Customer	13.208.0.2	13	MIAM	Provider	13.208.0.1
LOND	Customer	13.204.0.2	13	GENE	Provider	13.204.0.1
ZURI	Customer	14.207.0.2	14	ATLA	Provider	14.207.0.1
MIAM	Provider	179.0.38.1	17	ZURI	Customer	179.0.38.2/24
GENE	Provider	179.0.39.1	17	ZURI	Customer	179.0.39.2/24
ATLA	Provider	179.0.40.1	18	ZURI	Customer	179.0.40.2/24
PARI	Peer	179.0.41.1	16	PARI	Peer	179.0.41.2
NEWY	Peer	180.32.0.15	IXP 32		Peer	180.32.0.32

FIGURE 10 – Tableau des voisins

Nous avons donc configuré nos routeurs en suivant ce tableau.

Une fois les groupes arrivés quasiment au même point nous nous coordonnons pour mettre en place le eBGP. De notre coté, voici les commandes utilisez sur chacun de nos routeurs :

```
MIAM_router# conf t
MIAM_router(config)# router bgp 15
MIAM_router(config-router)# address-family ipv4 unicast
MIAM_router(config-router-af)# neighbor 15.151.0.1 next-hop-self
MIAM_router(config-router-af)#neighbor 15.151.0.1 next-hop-self
MIAM_router(config-router-af)#neighbor 15.152.0.1 next-hop-self
MIAM_router(config-router-af)#neighbor 15.153.0.1 next-hop-self
MIAM_router(config-router-af)#neighbor 15.154.0.1 next-hop-self
MIAM_router(config-router-af)#neighbor 15.155.0.1 next-hop-self
MIAM_router(config-router-af)#neighbor 15.156.0.1 next-hop-self
MIAM_router(config-router-af)#neighbor 15.157.0.1 next-hop-self
MIAM_router(config-router-af)#neighbor 15.158.0.1 next-hop-self
```

Comme vous le voyez, nous entrons en mode configuration et, encore une fois, sur celle du routeur bgp 15.
Toutes les commandes comprenant le next-hop-self correspondant aux déclarations des préfixes des routes qui vont êtres annoncées à nos pairs.

Lorsqu'un routeur de bordure apprend une route via eBGP, le next-hop de cette route est l'adresse d'un routeur appartenant à un autre AS. Lorsque cette route est propagée via iBGP au sein de votre AS, le next-hop est toujours une adresse appartenant à un autre réseau IP. Elle n'est donc pas joignable par le plan de routage OSPF.
Grâce à la commande "next-hop-self", le routeur de bordure remplace le next-hop de la route propagée en interne par sa propre adresse, qui est bien joignable au sein de l'AS.

On peut ensuite vérifier la bonne configuration grâce à looking glass.

FIGURE 11 – looking glass eBGP

La table se lit ainsi :
Pour joindre le 11, on passe par le 13
Pour joindre le 12, on passe soit par 13, soit 14
Le lien avec 13 est direct

C'est cohérent avec le schéma de topologie des AS, donc iBGP et eBGP sont bien configurés.

Partie IV : Policy-routing

Grâce a notre configuration précedentes ainsi que la collaboration des autres AS, nous sommes en mesure de recevoir et d'émettre notre préfixe ainsi que tout ceux que l'on apprend. Mais cela pose quelque problème a plus grande échelle. En effet il existe des milliers d'AS sur internet, avec un tel système en plus d'être vulnérable a des attaque d'AS malveillant, le réseaux serait saturé d'annonce BGP. C'est pour cela que certaines règles doivent être mises en places. Nous utiliserons la règles présenté sous forme de schéma ci dessous :

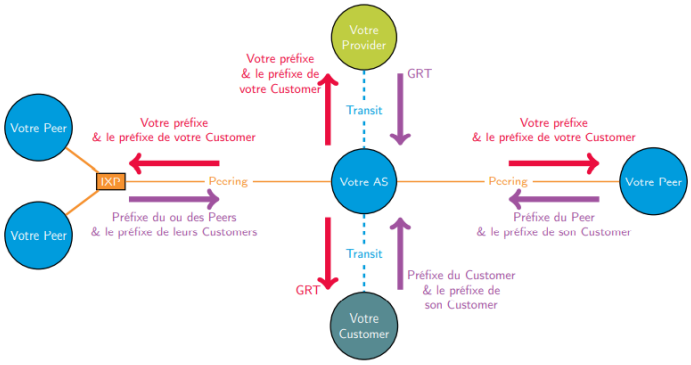

FIGURE 12 – Illustration des relations BGP

Dans les relations customer/provider, le provider exporte toutes les routes à sa connaissance (la GRT) pour fournir un accès Internet à son customer qui lui n'émet que son propre préfixe (et celui de ses propre customers). Ainsi, si le client souhaite accéder à d'autres AS, il passe par son fournisseur d'accès. Dans le cas d'un AS Tier 2 comme le notre, nous devons propagez alors cet accès Internet à nos propres clients l'AS 17 et 18.
Dans une relation peer-to-peer, les deux peers envoient leur propre préfixe (et celui de ses customers). Les peers (et leurs clients) peuvent ainsi s'échanger directement du trafic, sans passer par un fournisseur d'accès. Il s'agit généralement d'arrangement commerciaux entre deux AS qui échangent régulièrement du trafic, et qui souhaitent économiser l'achat de la bande-passante à un fournisseur.
Pour commencer nous avons besoin d'identifier les AS voisins avec qui nous allons échanger des informations.
En reprenant la topologie nous identifions :

- L'AS 14 et 13 comme providers
- L'AS 17 et 18 comme customers
- L'AS 16 comme peer
- L'AS IXP 32 comme liaison IXP

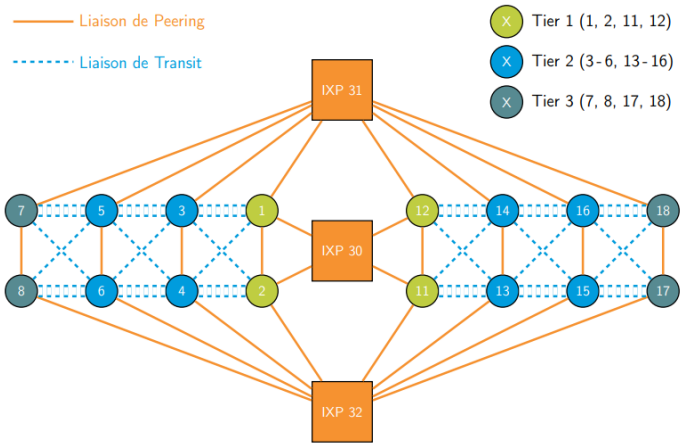

FIGURE 13 – Topologie des AS

Nous pouvons désormais configurer des route-map dans le réseau afin de filtrer les entrées et sorties BGP. Pour cela nous utiliserons des listes blanches,c'est a dire que seul sera autorisé les réseaux inscrit au seins de notre liste, les autres seront automatiquement rejetés.
Nous avons choisis d'utiliser des préfixe-listes spécifique a chacun des routeurs en fonction du trafic et du lien qu'il a avec les autres AS.

Ainsi pour les routeurs en liens avec nos customers (ATLA, MIAM, GENE) :

- Aucune règle de sortie car nous envoyons a nos clients toute notre table GRT
- En revanche nous acceptons de leur part uniquement leur préfixe d'AS ainsi que celui de leur enfants (ici aucun car notre AS est en bout de chaîne)

Pour les routeurs en liens avec nos providers (BOST, LOND, ZURI) :

- Aucune règle d'entrée car nous recevons leur table GRT
- En sortie nous filtrons les préfixes afin de leur renvoyer uniquement le notre et celui de nos enfants (15, 17, 18)

Pour notre routeur en lien avec le peer (PARI)

- Un filtre d'entrée limitant les préfixes arrivant a celui de notre peer ainsi qu'a ses enfants (16, 17, 18)
- Un filtre de sortie limitant notre préfixe et celui de nos enfants

Le routeur en lien avec l'IXP (NEWY) à les mêmes règles que pour la liaison avec un peer.

Nous appliquons donc ces règles au différents routeur puis nous procédons a une phase de test.

On regarde deja sur la Looking Glass que nos clients recoivent la GRT de notre part. Sur l'image si dessous on remarque que le routeur possède bien toute les routes du réseau et passe bien par notre AS, il a donc bien recu la table GRT de notre part.

```
   Network          Next Hop            Metric LocPrf Weight Path
*> 1.0.0.0/8        180.32.0.2                    50      0 2 4 1 i
*                   179.0.39.1                    20      0 15 14 1 i
*                   179.0.38.1                    20      0 15 14 1 i
*                   179.0.44.1                    20      0 16 1 i
*  2.0.0.0/8        179.0.44.1                    20      0 16 15 2 i
*                   179.0.38.1                    20      0 15 2 i
*                   179.0.39.1                    20      0 15 2 i
*>                  180.32.0.2               0    50      0 2 i
*> 3.0.0.0/8        180.32.0.2                    50      0 2 3 ?
*                   179.0.38.1                    20      0 15 14 1 3 ?
*                   179.0.39.1                    20      0 15 14 1 3 ?
*                   179.0.44.1                    20      0 16 1 3 ?
*> 4.0.0.0/8        180.32.0.2                    50      0 2 4 ?
*                   179.0.44.1                    20      0 16 1 4 ?
*                   179.0.38.1                    20      0 15 2 4 ?
*                   179.0.39.1                    20      0 15 2 4 ?
*  5.0.0.0/8        179.0.39.1                    20      0 15 14 1 4 5 ?
*                   179.0.38.1                    20      0 15 14 1 4 5 ?
*>                  180.32.0.2                    50      0 2 4 5 ?
*                   179.0.44.1                    20      0 16 1 4 5 ?
*> 6.0.0.0/8        180.32.0.2                    50      0 2 4 6 ?
*                   179.0.38.1                    20      0 15 2 4 6 ?
*                   179.0.44.1                    20      0 16 1 4 6 ?
*                   179.0.39.1                    20      0 15 2 4 6 ?
*> 7.0.0.0/8        180.32.0.2                    50      0 2 4 6 7 i
*                   179.0.38.1                    20      0 15 14 7 i
*                   179.0.39.1                    20      0 15 14 7 i
*                   179.0.44.1                    20      0 16 7 i
*  8.0.0.0/8        179.0.44.1                    20      0 16 15 8 i
*                   179.0.38.1                    20      0 15 8 i
*                   179.0.39.1                    20      0 15 8 i
*>                  180.32.0.8               0    50      0 8 i
*> 11.0.0.0/8       180.32.0.2                    50      0 2 4 11 i
*                   179.0.38.1                    20      0 15 14 11 i
*                   179.0.39.1                    20      0 15 14 11 i
*                   179.0.44.1                    20      0 16 1 4 11 i
*= 12.0.0.0/8       179.0.38.1                    20      0 15 14 12 i
*>                  179.0.39.1                    20      0 15 14 12 i
*                   179.0.44.1                    20      0 16 15 14 12 i
*  13.0.0.0/8       179.0.44.1                    20      0 16 15 13 ?
*>                  180.32.0.2                    50      0 2 4 11 13 ?
*                   179.0.38.1                    20      0 15 13 ?
*                   179.0.39.1                    20      0 15 13 ?
*  14.0.0.0/8       179.0.44.1                    20      0 16 15 14 ?
*>                  180.32.0.2                    50      0 2 4 11 14 ?
*                   179.0.38.1                    20      0 15 14 ?
*                   179.0.39.1                    20      0 15 14 ?
*  15.0.0.0/8       179.0.44.1                    20      0 16 15 ?
*>                  180.32.0.2                    50      0 2 4 11 13 15 ?
*                   179.0.38.1                    20      0 15 ?
*                   179.0.39.1               0    20      0 15 ?
*> 16.0.0.0/8       180.32.0.2                    50      0 2 4 11 14 16 i
*                   179.0.38.1                    20      0 15 16 i
*                   179.0.39.1                    20      0 15 16 i
*                   179.0.44.1               0    20      0 16 i
* i17.0.0.0/8       17.151.0.1               0   100      0 i
*>                  0.0.0.0                  0         32768 i
*  18.0.0.0/8       180.32.0.2                    50      0 2 4 1 18 i
*                   179.0.38.1                    20      0 15 18 i
*                   179.0.39.1                    20      0 15 18 i
*                   179.0.44.1                    20      0 16 18 i
*>                  179.0.46.2               0    50      0 18 i
```

FIGURE 14 – Table BGP du routeur ZURI du groupe 17

On vérifie aussi notre liason peer to peer en effectuant un traceroute entre un staff de notre LAN et l'hote de PARI de notre AS peer (16). On remarque bien qu'il ne passe pas par notre provider ce qui est une bonne chose.

```
root@staff_3:~# traceroute 16.103.0.1
traceroute to 16.103.0.1 (16.103.0.1), 30 hops max, 60 byte packets
 1  15.200.0.253 (15.200.0.253)  7.037 ms  6.973 ms  6.968 ms
 2  PARI-ZURI.group15 (15.0.1.2)  7.037 ms  7.028 ms  7.013 ms
 3  179.0.41.2 (179.0.41.2)  8.049 ms  8.043 ms  8.029 ms
 4  host-PARI.group16 (16.103.0.1)  9.116 ms  9.059 ms  9.091 ms
root@staff_3:~#
```

FIGURE 15 – Traceroute peer to peer

Nous pouvons donc conclure sur notre policy-routing. Les règles que nous avons mis en place fonctionnent et filtrent correctement le trafic afin d'assurer un internet plus propre et sécurisé.

Partie V : Conclusion

Pour résumer ce semestre, dans le cadre de ce projet, nous avons mis en place différents protocole sur une topologie bien définie.
Nous avons fait de la configuration de LAN, d'AS ainsi que de la mise en commun avec d'autres AS. Nous avons également mis en place des règles de sécurités sur notre réseau.

Ce projet nous a permis d'en apprendre d'avantage sur, d'une part les différentes technologies et protocoles configurés au sein d'un réseau type mini-internet et d'autre-part sur le travail en équipe. en effet nous avons pu collaborer avec les autres équipes sur certains aspects du projet. Cela nous a permit de renforcer nos compétences de futur collaborateurs au sein d'une entreprise.

Également, nous avons pu mettre en application l'intégralité des compétences et des connaissances que nous avons pu acquérir durant ces deux premières années d'école. Ce projet a donc toute son importance dans un cursus tel que le nôtre car il nous a permit de revoir des notions apprises au tout début de la première année.

Partie VI : Summary

To review the work done this semester, we built and operated our own Internet by establishing end-to-end connectivity through AS. We simulated to be a network operator

To do this, we relied on a virtual infrastructure consisting of Open vSwitch Layer 2 switches and routers using the FRRouting software suite. These virtual devices were carried by Docker containers, and already connected to each other. We had to configure them in command line.

At the beginning of this project, we had technical and theoretical knowledge about competing devices. So we had to adapt this knowledge to other technologies.

The work was divided into three main parts : intra-domain connectivity, BGP configuration and policy-routing. Concerning the Intra-domain part, we allowed the users of our LAN to connect to each other through their gateways. Within our AS, we have activated and configured OSPF routing. To finish this part, we discovered the basics of traffic enginnering, that is to say the optimization of the network performance for our customers.
We then moved on to the BGP part. This last one was divided in two aspects : the intra-domain with the iBGP and the inter-domain with the eBGP. The eBGP configuration required coordination with the other groups managing the other AS. But it was not finished : we still had to agree to announce our prefixes.
Finally, we started the policy-routing part by creating business relationships. Indeed, we matched prefix-lists on route-map according to a defined plan.

To finish the exercice, westill have to define the exit and entry of the traffic thanks to local preference and pre-pending, a form of traffic-engineering.

Thanks to this situation, we first learned how to work on an unknown brand of equipment. It was very destabilizing, especially during the first hours of the course. By following the practical course, we first reused the knowledge we had acquired in previous years. Thus we gained in efficiency. Finally, we discovered new protocols and principles of networks. We thus increased our skills and widened our field of action.

It was an enriching experience that made us take a more thoughtful approach to network configuration. Indeed, previously, we were just applying what we were asked. In this situation, we had to anticipate much more and especially bring a part of reflection in the typed commands.