Institut Universitaire de Technologie,
Aix-Marseille Université

RAPPORT DE STAGE
Diplôme Universitaire de Technologie
Spécialité Réseaux et Télécommunications

Mise en place d'un réseau IPv6 pur à destination des utilisateurs

Benjamin GOYON

Aix-Marseille Université

Responsable entreprise : Pascal Mouret

Responsable académique : Arnaud Fevrier

2018

Remerciements

Je tiens tout d'abord à remercier sincèrement toute l'équipe de la DOSICALU pour leur chaleureux accueil, pour leur aide et pour la très bonne ambiance qui m'ont permis de m'intégrer dès le premier jour à leur formidable équipe de travail.

Je tiens tout particulièrement à remercier Pascal pour son aide et pour sa confiance ainsi que pour tout ce que j'ai pu apprendre sur les systèmes et réseaux informatiques dans le milieu professionnel.

Grâce à vous j'ai pu avoir ma première véritable expérience professionnelle et je n'oublierai pas tout ce que vous avez pu faire pour m'aider à réaliser ce stage dans les meilleures conditions.

Table des matières

1 - Introduction

Pour conclure le DUT réseaux et télécommunications de L'IUT de Luminy, j'ai réalisé un stage de 10 semaines à Aix-Marseille Université au sein du service de la DOSICALU avec Pascal Mouret comme tuteur de stage et Arnaud Fevrier comme tuteur académique.
Je vais donc vous présenter mon stage de fin d'études qui s'est déroulé du 9 avril au 15 juin 2018.
L'intitulé de ma mission était la suivante : Mise en place d'un réseau IPv6 pur à destination des utilisateurs.
Je présenterai tout d'abord l'entreprise qui m'a accueilli pour effectuer ce stage, je parlerai ensuite du contexte, du travail réalisé au cours de ce stage et des résultats obtenus, et enfin je conclurai en vous expliquant tout ce que ce stage a pu m'apporter en termes d'expérience et de connaissance.

2 - Présentation de l'entreprise

2.1 Aix-Marseille Université

Aix-Marseille Université est un établissement public qui propose des formations et de la recherche dans tous les champs disciplinaires.
C'est aussi l'une des plus jeune car elle est le résultat de la fusion des trois précédentes universités d'Aix-Marseille en 2012.
Elle est également la plus grande université de France avec plus de 75 000 étudiants dont 10 000 internationaux et plus de 8 000 personnels.
L'université actuelle est composée de 58 sites dont cinq campus : Saint Jérôme, Marseille-Centre, Timone, Aix-en-Provence et bien évidemment Luminy.
L'organisation est divisée en plusieurs services qui permettent de mieux gérer l'université.
J'ai donc fait partie de la Direction Opérationnelle du Système d'information du Campus de Luminy (DOSICALU) qui s'occupe de gérer tout le système d'information de Luminy mais également du site de La Ciotat.

2.2 Organisation de la DOSI

La DOSI de l'université d'Aix-Marseille est divisée en plusieurs antennes qui vont gérer chacun soit des sites soit un campus.
Il y a les antennes transverses qui vont gérer des sites, et les antennes de campus qui gèrent les campus ainsi qu'un ou plusieurs sites.
Dans ces antennes, il existe plusieurs équipes qui ont chacune, une mission au sein du campus.
Il y a généralement trois principales équipes au sein d'une antenne qui sont : la gestion du parc informatique, réseau et systèmes.
Dans notre antenne de campus à Luminy, il existe également une équipe qui gère l'audiovisuel sur le campus.
Les équipes réseaux et systèmes travaillent souvent ensemble dans les différentes antennes de campus.
Pour ma part, j'ai fait partie de l'équipe réseau et système du campus de Luminy car c'est ce qui convenait le mieux pour pouvoir réaliser dans les meilleures conditions mon stage de fin d'étude.
Voici un organigramme hiérarchique permettant de mieux définir ma position au sein de la DOSI (Figure 1).

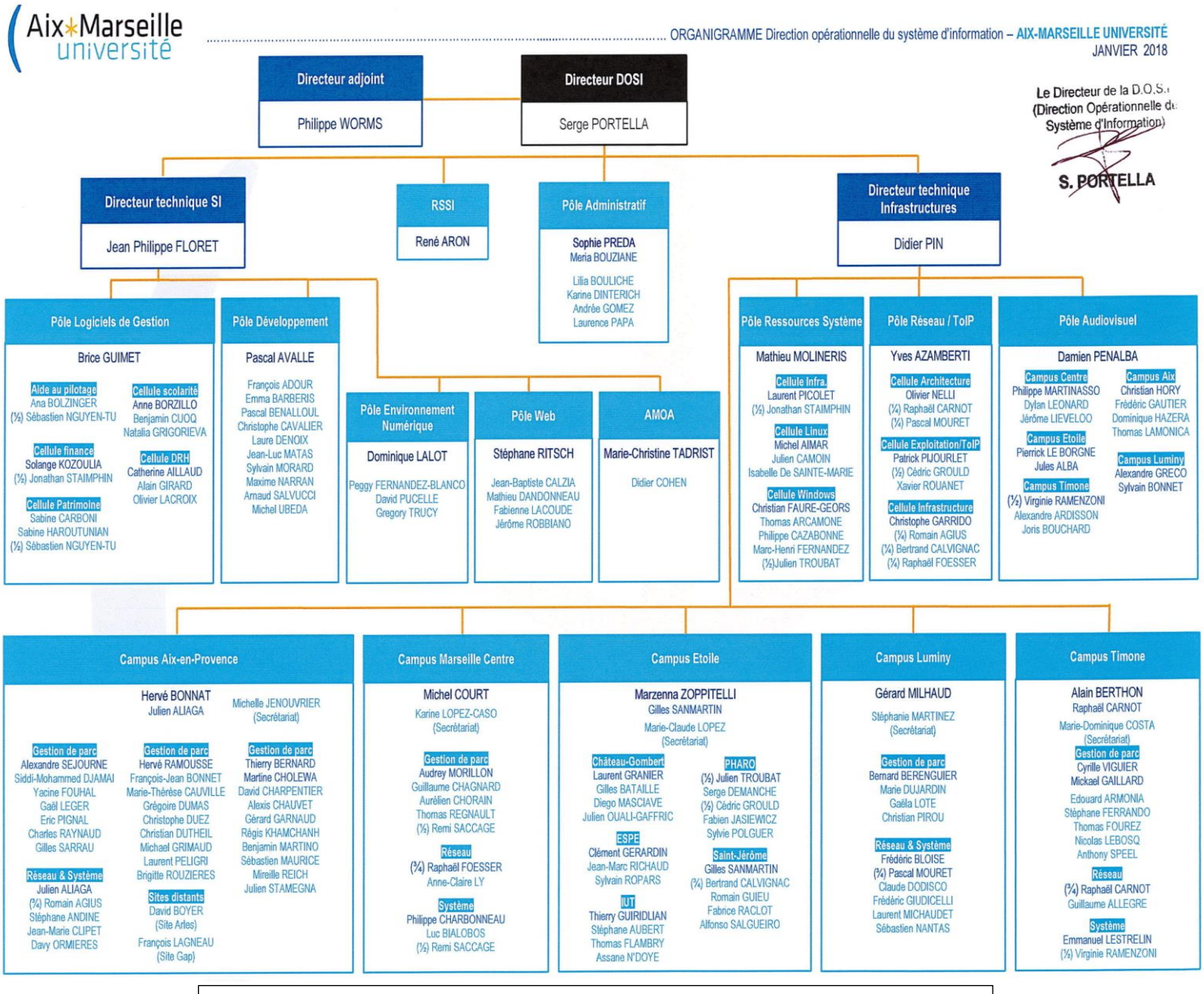

Figure 1 : Organigramme DOSI – Aix-Marseille Université – Janvier 2018

3 - Cadre technique du sujet

3.1 Contexte et objectifs du stage

De nos jours, de plus en plus appareils connectés sont utilisés au quotidien pour communiquer, automatiser, surveiller….
Ces appareils ont donc besoin d'un accès réseau et d'un moyen d'identification pour pouvoir les différencier et les reconnaitre sur le réseau.
C'est donc là qu'intervient le protocole IP.
Aujourd'hui encore, le protocole IP le plus utilisé reste IPv4 mais des problèmes de pénuries d'adresses dues notamment aux nombres d'appareils connectés qui augmentent de manière exponentielle au fil des années (Figure 2) vont empêcher l'utilisation de ce protocole.

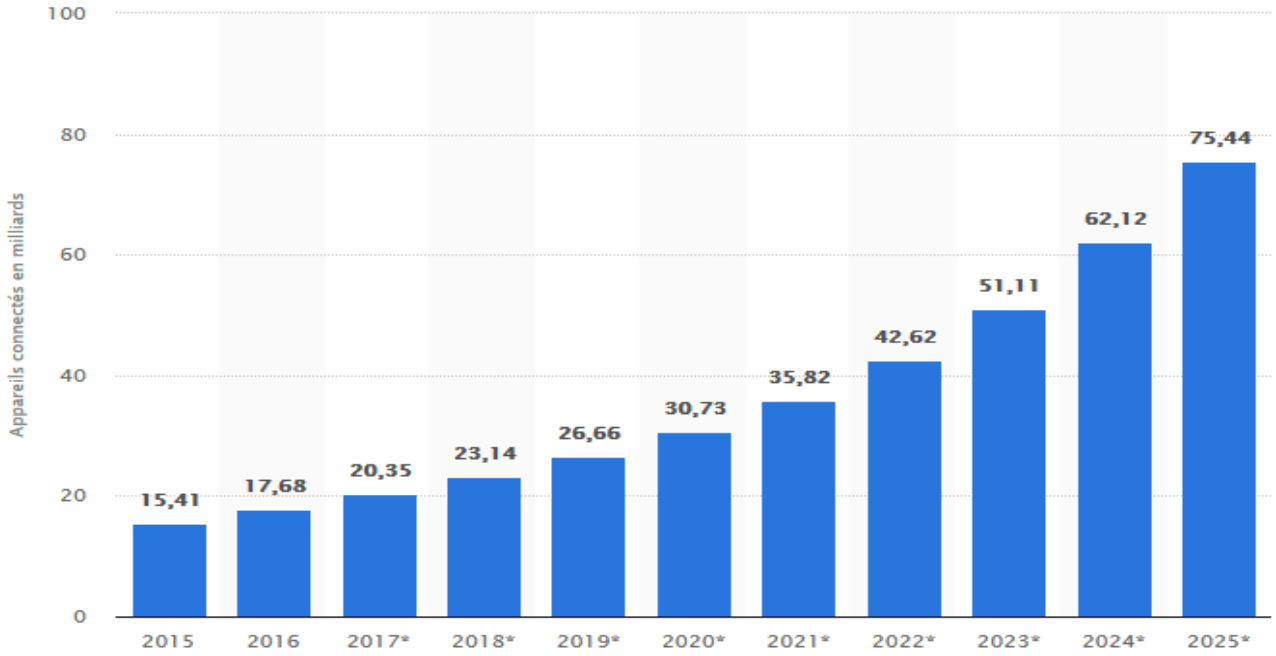

Figure 2 : Estimation de l'évolution du nombre d'appareils connectés en milliard en fonction des années

Pour résoudre ce problème, le protocole IPv6 a été conçu dans les années 90. Ce protocole permet en outre, une meilleure agrégation des routes dans la table de routage, ce qui permet de ne plus avoir de système de traduction d'adresse entre le réseau privé où se trouve la machine et le réseau public pour communiquer avec le reste du monde et donc de ne pas privatiser des réseau qui ne pouvait plus être utilisés sur internet.
On utilise donc directement une adresse dite « global unicast » pour communiquer sur internet.

Le problème majeur d'IPv6, c'est qu'il n'est pas compatible avec IPv4.
Cela force notamment les entreprises à devoir renouveler le matériel informatique pour qu'il prenne en charge IPv6 et à devoir former les ingénieurs réseau au protocole ipv6 qui comporte des différences avec ipv4.
C'est pour cela qu'aujourd'hui encore, IPv6 reste très peu utilisé dans le monde sauf dans les endroits où la pénurie totale d'adresses IPv4 force le passage à IPv6 (l'Asie notamment).

J'ai donc réalisé mon stage d'une durée de 10 semaines au sein de la Direction Opérationnelle du Système d'Information de l'Université d'Aix-Marseille à Luminy (DOSICALU).
Actuellement, l'université dispose d'un réseau IPv6 mais qui quasiment pas utilisé.

De plus, l'Environnement Numérique de Travail (ENT), depuis lequel sont accessibles toutes les ressources en ligne des comptes Aix-Marseille Université, existe uniquement en IPv4.
De ce fait, c'est aussi un problème pour le passage à IPv6 car ce site qui est très important pour tous les membres de l'université ne sera plus accessible ou alors il faudra aussi faire passer le site sur un réseau IPv6.
Ma mission était donc plus précisément de mettre en place un réseau IPv6 pur (c'est-à-dire sans IPv4) qui permettrait d'atteindre des ressources IPv6 ainsi que des ressources IPv4.
L'intérêt majeur d'un réseau ipv6 pur, c'est qu'on sera sûr du protocole IP qui sera utilisé.
Si on avait un réseau en dual stack, c'est-à-dire un réseau ou les machines utilisateurs utilisent à la fois IPv4 et IPv6, on aurait eu des problèmes avec les applications qui ne savent pas gérer correctement les deux protocoles à la fois.
Si par exemple on arrête le réseau ipv6 temporairement et qu'une application était en train de se servir de ce réseau, au moment de passer à l'autre protocole IP, il va y avoir un temps d'attente pour qu'il bascule sur l'autre protocole IP.
Ou bien on fait une requête pour une ressource en IPv4 mais l'application utilise IPv6. En fonction du protocole qu'elle a décidé d'utiliser l'application va donc passer d'un protocole à l'autre ce qui va prendre du temps pendant lequel les fonctionnalités en ligne de l'application ne seront plus utilisables.

3.2 Cahier des charges

Le cahier des charges n'était pas très strict.
En effet les seules obligations que j'avais, c'était d'utiliser un mécanisme de traduction IPv6 vers IPv4 bien précis que l'on appelle le NAT64/DNS64, un adressage des machines contrôlé et la mise en place d'un deuxième serveur de secours en cas de problème avec le premier.
Sinon le reste n'était pas imposé. Par exemple j'avais le choix pour le système d'exploitation des serveurs ainsi que pour la gestion de ceux-ci.

4 - Présentation du travail réalisé

4.1 Mise en place du réseau IPv6 de test

Le réseau IPv6 était déjà existant mais pas utilisé.
Il m'a juste suffi de rediriger le VLAN contenant le réseau de test, ici le VLAN 601, vers mon bureau pour que je puisse y connecter deux machines physiques de test : la première sous Windows et la seconde sous Ubuntu.
J'utilise deux systèmes d'exploitation différents pour m'assurer du bon fonctionnement sous les systèmes Linux et Windows et que le réseau soit compatible avec ces deux systèmes d'exploitation qui sont les plus utilisés au sein du campus.
J'ai donc modifié la destination du VLAN 601 pour qu'il arrive jusqu'à la prise Ethernet de mon bureau.
J'ai ensuite branché un switch HP A3100 à la prise murale, puis j'ai branché les deux machines de test sur le switch et enfin, j'ai redirigé le VLAN 601 vers les deux ports où j'ai branché les machines de test.
Pour pouvoir réaliser les configurations sur les switches HP, j'ai dû me documenter car je n'ai appris à configurer que du matériel Cisco.
J'ai pu remarquer que le principe reste le même mais que la syntaxe des commandes diffère en fonction du constructeur.
Après cela, le réseau IPv6 était accessible mais je ne pouvais pas atteindre les ressources IPv4.

4.2 Mise en place des serveurs sous Debian

Avant de mettre en place le système de traduction IPv6 vers IPv4, il faut préparer les serveurs qui accueilleront la solution de traduction.
Etant donné que Debian est un système d'exploitation très utilisé à la DOSICALU et que j'ai beaucoup travaillé dessus lord de mes TP à l'IUT, j'ai donc décidé de l'utiliser.
Le système d'exploitation sera sur une machine virtuelle pour les tests mais sera ensuite dans un conteneur pour le réseau de production.
L'avantage du conteneur par rapport à la machine virtuelle, c'est que c'est plus facile à copier et à modifier car le but du conteneur est d'avoir un hôte sur lequel va se trouver le noyau du système d'exploitation qui va être identique à tous les conteneurs.
De ce fait, lorsqu'on va copier le conteneur, on ne copiera que les programmes que l'on aura installés sur celui-ci.
J'ai tout de même utilisé des machines virtuelles pour le réseau de test car cela est plus facile à mettre en place et à paramétrer que le conteneur.
Maintenant que les serveurs sont prêts, on va pouvoir s'occuper du système de traduction IPv6 vers IPv4.

4.3 Mise en place du système de traduction NAT64/DNS64

Pour accéder aux ressources IPv4, il a donc fallu que je mette en place un système de traduction NAT64/DNS64 en traduisant donc une adresse IPv4 en adresse IPv6 (RFC 6052).
Le principe du système NAT64/DNS64 est le suivant :
Lorsque vous voulez accédez à un site, vous envoyez une requête au serveur DNS qui vous renvoi une adresse IP correspondant au nom de la requête.
Dans notre cas, on envoie une requête dite AAAA (Requête IPv6) pour obtenir une adresse IPv6 puisque notre réseau ne comprend pas IPv4.
C'est là que le DNS64 intervient car, si le serveur ne renvoie pas une adresse ipv6, alors il fera une requête dite A (Requête IPv4) pour obtenir une adresse IPv4.
Etant donné que notre réseau est en IPv6 seulement, il ne pourra pas accéder au site internet.
Le DNS64 va donc écrire l'adresse ipv4 dans une adresse IPv6 (RFC 6147).

Cette adresse IPv6 ne sera pas celle de notre réseau où seront présentes les machines des clients, ce sera un réseau spécialement utilisé pour la traduction.
Ce réseau, ou préfixe, aura un masque en /96 car une adresse IPv4 contient 32 bits et une adresse IPv6 en contient 128.
On fait donc : 128-32 = 96.
Et dans les 32 derniers bits de l'adresse IPv6, on va donc coder chaque nombre de l'adresse IPv4 avec deux chiffres hexadécimaux pour qu'elle soit ensuite lisible par le NAT64.
Sachant qu'il y a 4 nombres en décimal pointé sur une adresse IPv4 il nous faudra donc 8 chiffres hexadécimaux pour coder notre adresse IPv4.

Par exemple, mon préfixe de traduction est : 64:FF9B::/96 que l'on appelle aussi « Well-Known Prefix » c'est-à-dire un préfixe réservé à cet usage.
Si je fais une requête pour accéder à un site dont l'adresse est 10.10.10.11.
Sachant que 10 vaut 0A et 11 vaut 0B en hexadécimal, le serveur DNS64 va me traduire cette adresse IPv4 comme ceci : 0A0A:0A0B ce qui donne A0A:A0B en retirant les zéros inutiles pour simplifier l'écriture.
Et pour finir, je rajoute ce que je viens de calculer à la fin du préfixe comme ceci : 64:FF9B::A0A:A0B/96.
Nous avons donc ici une adresse IPv6 qui contient l'adresse ipv4 du site que l'on souhaite atteindre.

Maintenant que nous avons réussi à créer notre adresse avec le DNS64, le NAT64 va récupérer cette adresse et la traduire en IPv4 en ne reprenant que les 32 derniers bits de l'adresse IPv6 (RFC 6146).
Il va donc retrouver l'adresse 10.10.10.11 et va faire la demande d'accès à la ressource dans le réseau IPv4 car le serveur sera forcément à la fois sur le réseau ipv6 et ipv4 pour accéder d'un côté aux ressources IPv4 et de l'autre au réseau IPv6 des machines.
Pour cela, on va devoir mapper l'adresse IPv6 d'une machine dans notre réseau en une adresse IPv4.
On va donc avoir besoin d'un pool d'adresses IPv4 pour pouvoir différencier plusieurs machines qui souhaiteraient accéder à des ressources IPv4.
Une fois l'adresse IPv6 mappée, elle passe par l'interface tunnel du NAT64 pour démarrer le processus de translation et devenir à la sortie une IPv4 choisie aléatoirement dans le pool d'adresse IPv4.

Une fois la ressource trouvée, la réponse va donc passer dans l'autre sens dans le NAT64, c'est-à-dire que l'adresse IPv4 va être traduite en adresse IPv6 qui lui correspond et ainsi, la machine accèdera à la ressource.
On peut donc constater que notre serveur joue en quelque sorte le rôle d'un routeur
J'ai mis un schéma permettant d'illustrer le fonctionnement du système NAT64/DNS64 (Figure 3)

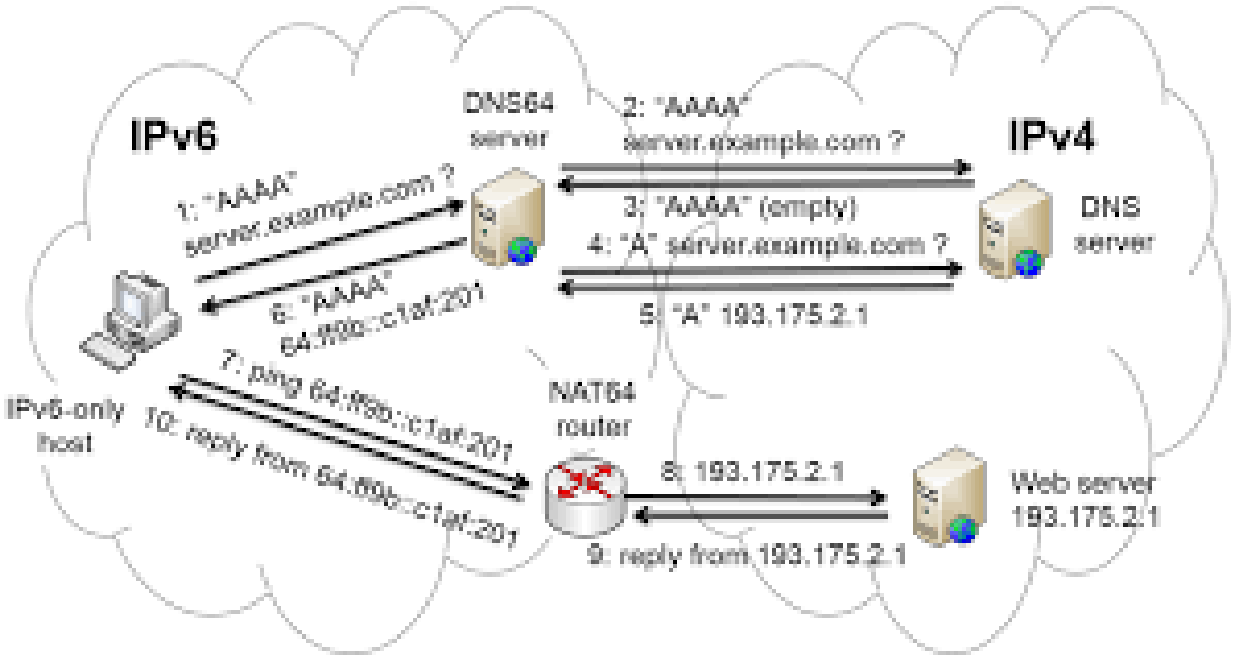

Figure 3 : Fonctionnement du système NAT64/DNS64

Pour pouvoir mettre en place ce système de traduction, il faut tout d'abord choisir un système d'exploitation.
Etant donné que Debian Linux est un système d'exploitation très utilisé à la DOSICALU pour la mise en place de serveur et ayant de l'expérience sur ce système que j'ai acquis pendant mes deux ans d'IUT, j'en ai conclu que c'était une bonne chose pour moi comme pour eux car le système d'exploitation étant connu par le service système et réseau, il sera plus facile pour eux d'apporter des modifications ou des changements par la suite.

Par la suite, j'ai fait des recherches pour trouver un paquet compatible avec Debian pour permettre la traduction DNS64.

J'ai donc utilisé bind9 qui est stable et qui inclus une option configurable très facilement pour ajouter la fonction DNS64
La configuration était très simple puisqu'il suffisait de dire dans le fichier de configuration que l'on souhaitait activer la traduction DNS64 pour tous les utilisateurs qui font une requête au DNS. C'est donc bind9 qui va s'occuper de traduire l'adresse IPv4 en adresse IPv6 du préfixe de traduction contenant l'adresse IPv4.

Pour finir il me fallait un deuxième paquet pour pouvoir faire la traduction NAT64.
J'ai donc utilisé Tayga comme NAT64 car c'est le seul paquet stable pour faire de la traduction NAT64.
Ici en revanche, la configuration est plus compliquée.
En effet j'ai eu beaucoup de problèmes pour configurer correctement Tayga, notamment pour la création de l'interface tunnel qui se fait manuellement en ligne de commande ainsi que les routes statiques pour indiquer que les trames ayant une adresse du préfixe IPv6 et IPv4 comme adresse de destination doivent se rendre vers l'interface tunnel pour que la traduction ait lieu dans les deux sens.
Ces étapes doivent être réalisées à chaque redémarrage du serveur et donc si le serveur redémarrait, le NAT64 n'était plus fonctionnel.
Une fois que j'ai manuellement réussi à faire fonctionner Tayga, j'ai donc créé un script qui a permis d'automatiser ces taches pour les réaliser au lancement du serveur s'il a été éteint ou redémarré et cela a résolu le problème.

J’ai ensuite réalisé la même manipulation en modifiant juste le préfixe IPv6 et le pool de traduction IPv4 qui doivent être différents du premier serveur pour que l’on puisse savoir facilement par quel serveur vont passer les trames et surtout quel serveur va faire la traduction.

Mais maintenant un autre problème se pose. Si Tayga s’arrête mais que bind9 continue de fonctionner, bind9 va continuer de traduire les adresses ipv4 avec le préfixe IPv6 du serveur qui ne fonctionnera plus.
De ce fait le réseau ne fonctionnera plus puisque l’interface tunnel ne sera plus géré par Tayga et l’interface tunnel du deuxième serveur ne sera pas capable de faire la traduction puisqu’il ne traduit pas le même préfixe que le premier serveur.

Il faut donc trouver une solution pour arrêter aussi bind9 si Tayga ne fonctionne plus.

4.4 Mise en place de Mon

Après plusieurs jours de recherches pour trouver une solution stable pour couper bind9, nous avons trouvé une solution avec mon maitre de stage qui puisse répondre à nos besoins.
Mon est un programme qui permet de surveiller tout ce que l’on souhaite sur un système linux, et d’effectuer automatiquement une action lorsqu’un problème est détecté.
Le principe de « Mon » réside dans le fait qu’il créé un utilisateur appelé « mon » sur le serveur qui va réaliser les actions une fois les problèmes détectés
Il répond donc exactement à nos besoins pour permettre de rétablir la situation en cas de problèmes avec Tayga.
Dans notre cas, on veut que bind9 soit arrêté si Mon détecte que Tayga ne fonctionne plus.
Il a donc fallu créer deux scripts, le premier doit vérifier si Tayga continue de fonctionner, le second doit effectuer l’action d’arrêter bind9 si le premier script détecte que Tayga ne fonctionne plus.
Mais un autre problème est apparu, c’est que mon ne faisait la vérification qu’une fois toute les 2 minutes.
Le problème, c’est que pendant deux minutes, le réseau ne fonctionnera plus et il faudra donc attendre que mon détecte la panne.
Heureusement, j’ai pu trouver un paramètre permettant d’établir la vérification une fois toute les secondes ce qui sera quasiment invisible pour l’utilisateur.
De plus, malgré la vérification qui a lieu toutes les secondes, cela n’utilise pas trop les ressources en calcul processeur.
La solution avec Mon a donc aussi été validé par mon maitre de stage.

4.5 Mise en place du système d'adressage des machines

Jusqu'à présent, l'adressage des machines sur le réseau devait se faire de manière statique, c'est-à-dire que je devais rentrer une adresse IP mais aussi fournir la passerelle pour sortir du réseau et les serveurs DNS pour la traduction NAT64/DNS64.
Il a donc fallu trouver un moyen pour que les machines soient paramétrées automatiquement mais que l'on choisisse tout de même les machines que l'on veut ou pas dans le réseau.
Sous IPv6 les paramètres de configuration automatiques sont envoyés par des Router Advertissement aussi appelés RA.
Ils vont indiquer à la machine qu'elle peut se configurer automatiquement à l'aide de la méthode SLAAC (Stateless Address Autoconfiguration), c'est-à-dire que la machine va se baser sur sa propre adresse MAC pour créer une adresse IPv6 unicast globale qu'elle va s'attribuer pour s'identifier sur le réseau et pouvoir accéder à internet.
Mais ce n'est pas tout, les RA peuvent aussi envoyer les annonces des DNS que les machines devront utiliser.
Dans notre cas, c'est indispensable car si on ne dit pas aux machines d'utiliser notre DNS64, nous n'accèderons pas au réseau IPv4.
J'ai donc cherché un logiciel sous Debian capable d'envoyer des RA depuis un serveur et j'ai trouvé RADVD qui est le meilleur logiciel dans son domaine car il est stable et complet.
Il implémente les annonces lien-local des adresses de routeurs IPv6 et des préfixes de réseau IPv6 au moyen du protocole Neighbor Discovery Protocol (NDP) conformément à la RFC 2461.
Après installation de celui-ci, j'ai pu constater que cela fonctionne.
Mais il y a encore un problème.
Dans notre cas actuel, n'importe quelle machine peut se connecter au réseau.
Or ce n'est pas ce que nous voulons avec mon maitre de stage puisque nous voulons ne donner l'accès qu'a certaines machines.
Pour résoudre le problème, j'ai fait des recherches sur les configurations de RADVD et j'ai trouvé un moyen de n'envoyer des RA qu'aux machines dont leurs adresses Link-local sont stockées dans le fichier de configuration de RADVD.
Cela règle donc le problème d'adressage.
Le seul défaut de cette méthode est qu'il faudra que le serveur sur lequel se trouve RADVD soit dans le réseau des machines client car les RA sont envoyés en lien local et ne peuvent donc pas sortir de ce lien.
Sachant que les serveurs seront sur un autre réseau que celui où se trouve les machines client il faudra donc un troisième serveur pour permettre l'envoi des RA au machines clients.

4.6 Optimisation du service de redondance

Après avoir adressé les machines, le projet était prêt à être mis en production mais il y avait une amélioration majeure à apporter.
Le problème ici, c'est que RADVD envoie les adresses des serveurs DNS64 de manière aléatoire.
De ce fait, une machine qui démarre peut avoir son DNS primaire de manière aléatoire et donc les deux serveurs vont être utilisés en même temps alors que le deuxième serveur ne devrait être utilisé qu'en cas de problème sur le premier.
Et c'est ici que nous est venu une idée avec Pascal de mettre en place un logiciel sur les deux serveurs qui permettrait d'avoir deux serveurs à la configuration identique, mais d'avoir un serveur primaire sur lequel la configuration serait opérationnelle et un second en attente qui n'activerait ses services seulement si le premier serveur ne fonctionne plus.
Pour ce faire, les deux serveurs auraient un lien spécial pour s'envoyer des messages qui indiquent qu'ils sont en fonction.

Si le serveur primaire n'envoie plus ces messages, alors le deuxième serveur le déclare comme mort et prend la main en activant ses services.
On aurait donc plus besoin d'avoir deux préfixes de traduction en IPv4 comme en IPv6 puisque seulement un des deux serveurs serait réellement en fonction (Figure 4).

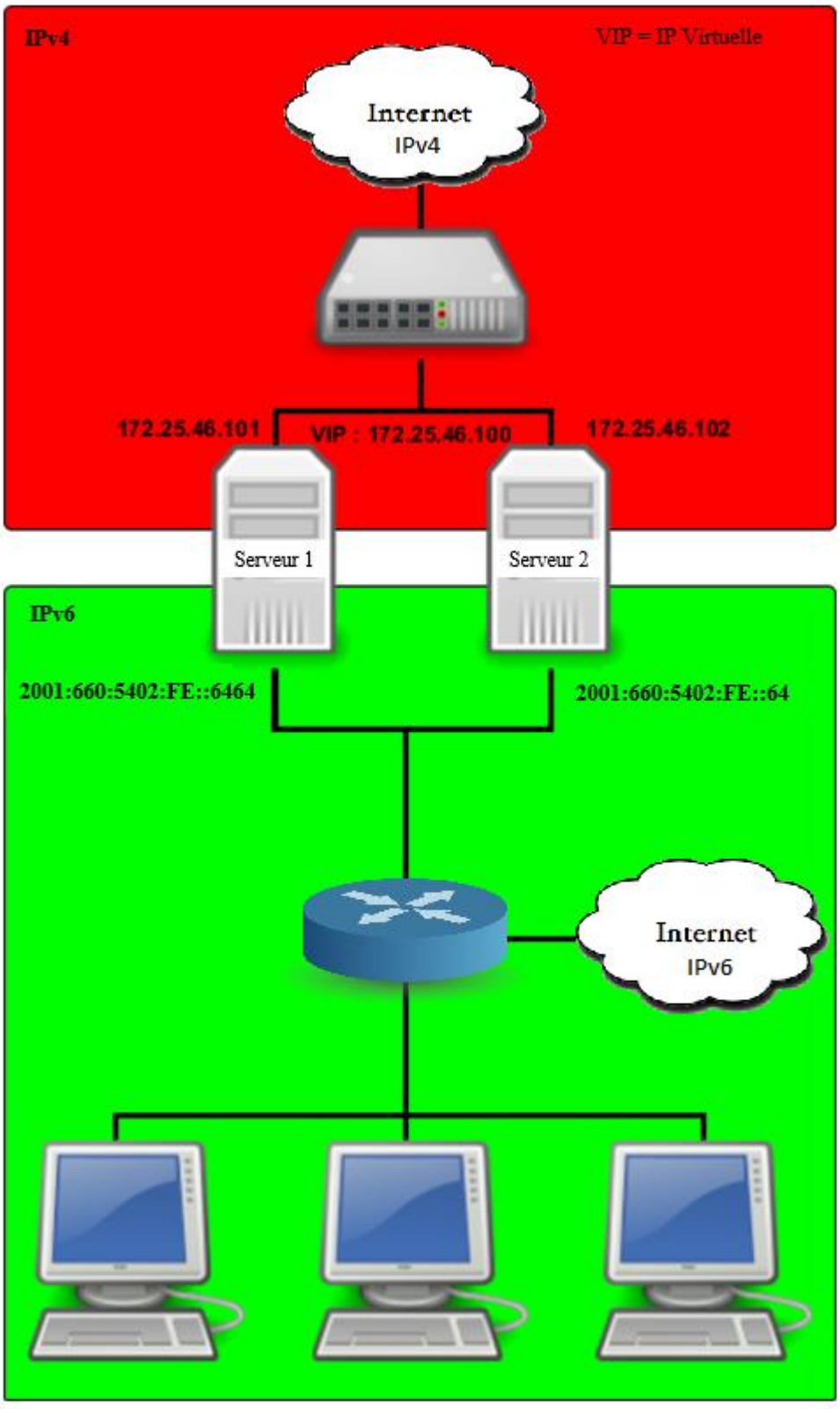

Figure 4 : IP virtuelle sur le réseau IPv4

On libèrera donc un réseau IPv4 ce qui est important au vu du nombre limité de réseaux IPv4 par rapport à IPv6.
Nous avons donc discuté de cela avec Fréderic qui est responsable de l'équipe système.
Il nous a parlé d'un logiciel qui s'appelle Heartbeat et qui correspond exactement à la solution dont j'ai besoin.
De plus, Fréderic connait très bien ce logiciel car il s'en sert déjà pour un autre service sur d'autres serveurs sous IPv4.
De ce fait il sera facile pour eux de pouvoir reconfigurer Heartbeat en cas de changements.
Mais plusieurs problèmes me sont venus à l'esprit.
En premier lieu, Heartbeat n'a pas toutes les fonctionnalités sous IPv6 que l'on peut avoir en IPv4.
En deuxième lieu, le script qu'il va falloir créer pour lancer tous les services est extrêmement similaire au script de démarrage que j'ai précédemment créé.
J'ai donc décidé de déplacer le script de démarrage pour directement faire démarrer les services avec Heartbeat.
Après vérification, Heartbeat se lance correctement et fait bien le travail attendu.
J'ai donc réussi à améliorer considérablement le service de redondance en réduisant le nombre de réseaux utilisés, notamment les réseau IPv4, mais en plus, cela a permis d'unifier la configuration et donc de pouvoir mettre en place plus rapidement la solution que précédemment puisqu'il suffit de dupliquer les machines et de modifier la configuration.
Cela a aussi permis de savoir quel serveur est réellement actif ou non et de ne pas utiliser les deux à la fois.
On peut remarquer sur la figure 4 qu'il existe un réseau IPv6 exclusif aux serveurs.
A l'origine, nous avions un réseau unique dans lequel on avait les machines client ainsi que les serveurs car cela était plus facile pour les tests.
Mais, le problème majeur de cette solution est que l'on devrait créer deux serveurs pour chaque réseaux IPv6 sur lesquels on voulait faire du NAT64/DNS64.
Or ce n'est pas concevable à l'échelle d'un réseau de production il est préférable d'exporter les serveurs dans un réseau à part pour pouvoir les utiliser pour différents réseaux à la fois.

4.7 Autres missions

J'ai également réalisé d'autres missions au sein de la DOSICALU.
J'ai par exemple changé un switch à l'IUT Réseaux et Télécommunications qui ne fonctionnait plus et avait rendu notamment 2 bornes WI-FI inactives.
J'ai réalisé cette opération avec l'aide de Laurent qui fait partie de l'équipe système et réseau.
De plus, l'ai réalisé diverses interventions pendant lesquelles j'ai changé des commutateurs des salles de cours du campus de Luminy car les anciens commutateurs avaient une vitesse de transfert de 100 Mbits/s.
Le but était donc de les changer pour des commutateurs au Gigabit pour augmenter le débit dans les salles de cours.
J'ai également aidé à sortir de l'IUT une dizaine de PC et moniteurs qui ont été donnés à une association qui aide les enfants en difficulté à l'école.
Enfin, j'ai pu visiter le Datacenter du campus pour constater l'ampleur du réseau du campus de Luminy.

5 - Conclusion

A la fin de mon stage, j'ai réussi entièrement ma mission car mon projet est passé en production et peut aujourd'hui être utilisé au sein du campus de Luminy.
Je suis assez fier de cela car le projet avait déjà été tenté par d'anciens stagiaires qui n'avaient pas réussi à rendre un travail qui puisse être mis en production sur le campus.
Il y a évidemment eu des problèmes mais que j'ai pu surmonter et cela m'as permis de renforcer mon travail en autonomie et d'approfondir mes connaissances notamment sur IPv6 et sur le système d'exploitation Debian Linux.
J'ai aussi pu remarquer que les connaissances que j'ai assimilées pendant ces deux années à l'IUT Réseaux et Télécommunications m'ont beaucoup aidé pour la réalisation de mon stage.
Il y a évidemment des points d'amélioration que l'on pourrait apporter à ce projet notamment de mettre en place un système de déploiement de systèmes d'exploitation car la plupart des machines du campus fonctionnent actuellement sur ce système.
A l'heure actuelle, ce n'est pas encore possible notamment à cause des ordinateurs du campus qui ne sont pas assez récents pour la plupart pour pouvoir mettre en place ce système.
Il existe une seconde amélioration qui serait de trouver un moyen de déployer les RA depuis le routeur et qu'il puisse envoyer les DNS aux machines client ce qui n'est pas encore possible sur les routeurs HP.
Mais ce sera un sujet qui sera probablement donné aux stagiaires qui viendront lorsque la situation sur ce domaine aura évolué.
J'ai également pu enrichir mes connaissances sur le monde de l'entreprise et à m'adapter rapidement dans une équipe de travail professionnelle.
Pour conclure, ce stage était très intéressant et m'a beaucoup apporté en termes de connaissances professionnelles et techniques.

6 – Glossaire

DUT, Diplôme Universitaire de Technologie

RFC : Série numérotée de documents officiels décrivant les aspects techniques d'Internet, ou de différents matériels informatiques.

NAT : Network Address Translation que l'on peut traduire en « traduction d'adresses réseau ».

DNS : Domain Name System que l'on peut traduire en « système de noms de domaine »

FTP : File Transfert Protocol que l'on peut traduire en « protocole de transfert de fichier »

VIP : Virtual IP que l'on peut traduire par IP virtuelle