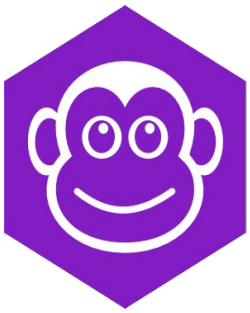

MOOC

Objectif IPv6 !

vers l'internet nouvelle génération

Travaux Pratiques

Séquence 4

Faire inter-opérer des applications IPv6 et IPv4

Licence Creative Nommons CC BY-SA 4.0 International

Attribution - Partage dans les Mêmes Conditions 4.0 International (CC BY-SA 4.0)

Avertissement Ce résumé n'indique que certaines des dispositions clé de la licence. Ce n'est pas une licence, il n'a pas de valeur juridique. Vous devez lire attentivement tous les termes et conditions de la licence avant d'utiliser le matériel licencié.

Creative Commons n'est pas un cabinet d'avocat et n'est pas un service de conseil juridique. Distribuer, afficher et faire un lien vers le résumé ou la licence ne constitue pas une relation client-avocat ou tout autre type de relation entre vous et Creative Commons.

Clause C'est un résumé (et non pas un substitut) de la licence.

http://creativecommons.org/licenses/by-sa/4.0/legalcode

Vous êtes autorisé à :

- **Partager** — copier, distribuer et communiquer le matériel par tous moyens et sous tous formats
- **Adapter** — remixer, transformer et créer à partir du matériel
- pour toute utilisation, y compris commerciale.

L'Offrant ne peut retirer les autorisations concédées par la licence tant que vous appliquez les termes de cette licence.

Selon les conditions suivantes :

Attribution — You must give **appropriate credit**, provide a link to the license, and **indicate if changes were made**. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.

Partage dans les Mêmes Conditions — Dans le cas où vous effectuez un remix, que vous transformez, ou créez à partir du matériel composant l'Oeuvre originale, vous devez diffuser l'Oeuvre modifiée dans les même conditions, c'est à dire avec **la même licence** avec laquelle l'Oeuvre originale a été diffusée.

No additional restrictions — Vous n'êtes pas autorisé à appliquer des conditions légales ou des **mesures techniques** qui restreindraient légalement autrui à utiliser l'Oeuvre dans les conditions décrites par la licence.

Notes: Vous n'êtes pas dans l'obligation de respecter la licence pour les éléments ou matériel appartenant au domaine public ou dans le cas où l'utilisation que vous souhaitez faire est couverte par une **exception.**

Aucune garantie n'est donnée. Il se peut que la licence ne vous donne pas toutes les permissions nécessaires pour votre utilisation. Par exemple, certains droits comme l**es droits moraux, le droit des données personnelles et le droit à l'image** sont susceptibles de limiter votre utilisation.

Les informations détaillées sont disponibles aux URL suivantes :

- http://creativecommons.org/licenses/by-sa/4.0/deed.fr
- http://fr.wikipedia.org/wiki/Creative_Commons

Les auteurs

Bruno Stévant

Bruno STEVANT est enseignant chercheur à l'IMT Atlantique. Il intervient dans l'enseignement et sur les projets de recherche autour d'IPv6 depuis plus de 10 ans. Il est secrétaire et responsable des activités de formation de l'association G6, association pour la promotion et le déploiement d'IPv6 en France.

Jacques Landru

Enseignant chercheur au département Informatique et Réseaux à l'IMT Lille Douai, Jacques est responsable de l'UV de spécialisation ARES (Architecture des RESeaux) à la fois dans le mode traditionnel présentiel que dans sa forme à distance dans le cadre du cursus diplômant TutTelNet.

Jean-Pierre Rioual

Ingénieur Conseil Réseaux – EURÊKOM. Fort de 30 années d'expérience dans le domaine des réseaux, il intervient auprès des entreprises pour des missions d'expertise sur leurs réseaux de transmission de données (intégration, mesures, optimisation, administration), conçoit et anime des actions de formation "réseaux".

Pascal Anelli

Pascal ANELLI est enseignant-chercheur à l'Université de la Réunion. Il enseigne les réseaux depuis plus 20 ans. Il est membre du G6 depuis sa création. A ce titre, il est un des contributeurs du livre IPv6. En 1996, il a participé au développement d'une version de la pile IPv6 pour Linux.

Joël Grouffaud

Joël GROUFFAUD est professeur agrégé de mathématiques. Il est chef du département Réseaux et Télécommunications de l'IUT de la Réunion, une composante de l'université de La Réunion. Au sein du département, il enseigne les réseaux et IPv6. Il anime l'académie Cisco (formations CCNA) de La Réunion.

Pierre Ugo TOURNOUX

Pierre Ugo TOURNOUX est enseignant chercheur à l'Université de la Réunion. Il est responsable des enseignements d'administration réseau, de routage et des réseaux sans fil dans lesquels il intègre IPv6 depuis de nombreuses années.

Remerciements à :

- Vincent Lerouvillois, pour son travail de relecture attentive ;
- Bruno Di Gennaro (Association G6) ;
- Bruno Joachim (Association G6) pour sa contribution à l'activité « Contrôler la configuration réseau par DHCPv6 » ;
- Richard Lorion (Université de la Réunion) pour sa contribution à l'activité « Etablir la connectivité IPv6 tunnels pour IPv6 ».

Tables des activités

Activité 46: Faites interopérer des applications IPv6 et IPv4

Dans l'état d'avancement de la migration vers IPv6, des clients IPv6 vont apparaitre dans l'Internet. En vertu de la continuité du service et de l'unité de l'Internet, ces hôtes doivent pouvoir accéder aux contenus disponibles sur l'Internet v4. À ce stade du déploiement, nous avons 2 Internets :

- un Internet en IPv4 encore prépondérant du fait de ses services ;
- un Internet en IPv6 en croissance mais néanmoins, pour le moment, moins développé en terme de services car il connecte aujourd'hui essentiellement des clients.

L'objectif de cette activité est de présenter les solutions d'interopération d'applications distribuées qui ne fonctionnent pas au-dessus de la même version du protocole IP. Comme nous l'avons dit, le plan de migration originel en double pile n'est plus applicable à cause du manque d'adresses IPv4 disponibles de nos jours.

Les différentes étapes de cette activité vont représenter une évolution temporelle de l'Internet. Pour chacune de ces évolutions, nous montrerons quelle technique de transition appliquer et comment l'appliquer.

La plateforme mise en oeuvre dans ce TP est représentée par la figure 1. Elle comporte :

- un client 'IPv6 uniquement' disposant seulement d'une adresse IPv6. Ce client, localisé sur le noeud PC-1, représente les nouvelles machines qui apparaissent dans l'Internet pour former ce que l'on appellera par la suite l'Internet en IPv6 ;
- un routeur R1 en double pile : c'est un noeud qui a une connectivité avec l'Internet en IPv6 et en IPv4 ;
- un routeur R2 en IPv4 : c'est un noeud représentant l'Internet en IPv4 ;
- un client 'IPv4 uniquement' localisé sur PC-2, représente les machines héritées de l'internet de la génération précédente ;
- un serveur web IPv4. Ce service sera hébergé sur le noeud SRV-3 et représentera les contenus disponibles sur l'Internet IPv4. Ce noeud hébergera également un serveur DNS.

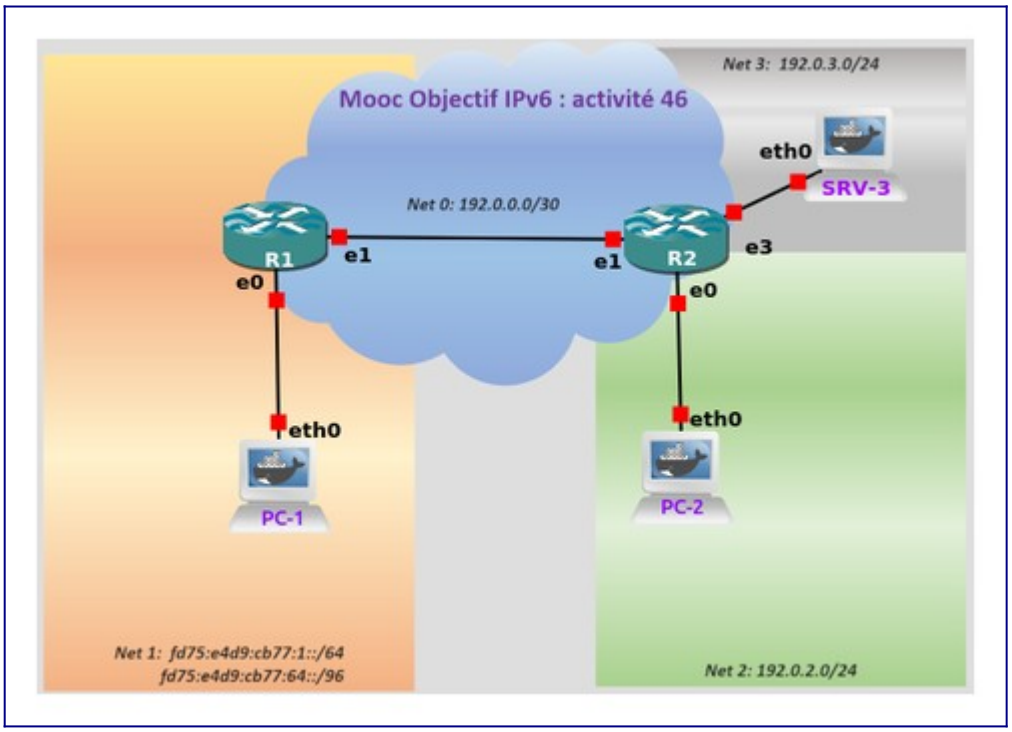

Figure 1 : Plateforme de l'activité.

Etape 0 : Situation initiale

Dans la situation initiale, nous nous situons avec un Internet majoritairement en IPv4 et des nouveaux réseaux en IPv6 hébergeant des clients. La plateforme de la figure 1 illustre cette situation. Le réseau IPv6 symbolise ces nouveaux réseaux de clients. Le coeur de réseau et les services fonctionnent en IPv4. Pour cette phase initiale, l'infrastructure de communication de la plateforme est opérationnelle. Sur SRV-3 les services applicatifs de nommage DNS (`/usr/sbin/named -u bind`) et web (`nginx`) ont été automatiquement lancés au démarrage de la machine.

Note : pour vous loguer sur les stations PC-1, PC-2 et SRV-3, l'identifiant est **root** et il n'y a pas de mot de passe (tapez sur 'retour chariot' (*return* ou Entrée) quand le mot de passe est demandé).

Pour tester la configuration, il faut d'abord valider le DNS en interrogeant le serveur de noms pour un nom de la zone. Il existe des outils sous Linux permettant de faire explicitement ces requêtes, tels que `host` ou `dig`.

***Nota :** Dans le cadre de ce TP, les machines PC-1, PC-2, R1, R2 et SRV-3 ont respectivement été enregistrées pc1, pc2, r1, r2 et srv dans le domaine DNS .tp hebergé sur la machine SRV-3.*

La commande `dig` est disponible sur PC-2. L'adresse du serveur de noms à utiliser dans la commande `dig` est l'adresse de SRV-3, soit `192.0.3.3`. :

```
root@PC-2:~# dig @192.0.3.3 -t A srv.tp
```

Pour un affichage plus concis vous pouvez ajouter le paramètre `+short`

```
root@PC-2:~# dig @192.0.3.3 -t A srv.tp +short
```

Vous pouvez également vérifier que la résolution de noms fonctionne depuis R1 (en mode utilisateur) à l'aide de la commande `host`. Pour vous connecter en mode utilisateur sur R1 :

Nota : *Pour vous loguer sur les routeurs R1 et R2, les identifiant/mot de passe sont* ***vyos/vyos****. (Aucun affichage de caractère n'est produit lorsqu'on entre le mot de passe).*

```
r1 login: vyos
Password:
Linux r1 4.19.28-amd64-vyos #1 SMP Mon Mar 11 16:03:26 CET 2019 x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
vyos@r1:~$
```

L'adresse du serveur de noms à utiliser dans la commande `host` est l'adresse de SRV-3, soit `192.0.3.3`. La syntaxe de la commande devient alors, sur R1 :

```
vyos@r1:~$ host srv.tp 192.0.3.3
```

Vérifiez ensuite le bon fonctionnement du service web de SRV-3 depuis R1, à l'aide de la commande `curl` :

```
vyos@r1:~$ curl http://srv.tp
Un contenu brut HTML doit s'afficher
```

Si vous le souhaitez, vous pouvez recommencer la vérification depuis R2.

Pour cela il vous faut d'abord faire pointer le client DNS de R2 vers SRV-3 en renseignant le fichier de configuration `/etc/resolv.conf`. Passez d'abord en mode *root*, puis créer le fichier fichier du client DNS, à l'aide de la commande `echo` et en redirigeant la sortie standard vers le fichier à créer.

```
r2 login: vyos
Password:
Linux r2 4.19.28-amd64-vyos #1 SMP Mon Mar 11 16:03:26 CET 2019 x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
vyos@r2:~$ sudo su
root@r2:/home/vyos# echo "nameserver 192.0.3.3" > /etc/resolv.conf
root@r2:/home/vyos# exit
exit
```

Vous pouvez ensuite procéder aux tests de la même manière que sur R1.

```
vyos@r2:~$ host srv.tp
```

```
vyos@r2:~$ curl http://srv.tp
Un contenu brut HTML doit s'afficher
```

La problématique qu'il reste à résoudre se pose dans les termes suivants : comment PC1, qui est en *IPv6-only*, peut-il accéder au service web `srv.tp` qui est en *IPv4-only* ?

Etape 1 : Configuration de NAT64/DNS64

Dans cette étape, nous installons la proposition de l'IETF NAT64/DNS64 qui répond à la problématique de l'interopérabilité de systèmes utilisant une version du protocole IP différente. Le relais auxiliaire DNS64 et le NAT64 seront placés sur le noeud en double pile R1.

Le déploiement du NAT64 se situe dans le scénario 1 indiqué par le RFC 6144 dans lequel les clients d'un réseau IPv6 d'une organisation accèdent aux serveurs IPv4 de l'Internet. Dans ce scénario, la solution NAT64 peut être 'avec état' ou 'sans état'.

Le NAT64 que nous allons déployer sur notre plateforme est un traducteur 'sans état'. Avant d'étudier sa mise en oeuvre effective, nous allons revoir le principe de fonctionnement de cette solution de traduction. Le traducteur effectue la traduction de l'adresse source et de l'adresse de destination d'un paquet par la méthode 'sans état'. Ce mode de traduction de l'adresse implique que l'adresse IPv4 est imbriquée dans l'adresse IPv6 comme indiquée par le RFC 6052. Ainsi, lorsqu'un client IPv6 envoie une requête à un serveur IPv4, l'adresse IPv4 du serveur doit être transformée en une adresse IPv6 pour le client. C'est cette adresse qui sera utilisée comme adresse de destination par le client. Et quand le serveur IPv4 renvoie une réponse au client IPv6, il doit utiliser une adresse IPv4 comme adresse de destination, adresse qu'il aura apprise en recevant la requête. Comme la traduction d'adresse s'effectue 'sans état', ceci implique que l'adresse IPv4 du client a été fournie par le client lui-même. En d'autres termes, le client IPv6 dispose, parmi ses adresses unicast, d'une adresse IPv6 qui imbrique son adresse IPv4. Donc, d'après la terminologie indiquée par le RFC 6144, il s'agit d'allouer au client IPv6 une adresse IPv6 *IPv4-translatable* en plus de son adresse IPv6 unicast routable.

Reste le problème de la transformation de l'adresse IPv4 du serveur en une adresse IPv6 pour qu'elle soit utilisable par le client pour envoyer ses requêtes. Cette transformation est indispensable pour rendre IPv4 transparent aux protocoles applicatifs et aux utilisateurs IPv6. C'est ici que nous avons besoin des services d'un relais DNS auxiliaire (*DNS Application Layer Gateway*), communément appelé DNS64. Celui-ci traduira, à la volée, les adresses IPv4 en adresses IPv6 avec un préfixe particulier qui sera routé vers le relais réseau NAT64. L'adresse IPv6 ainsi créée à partir de l'adresse IPv4 du serveur est qualifiée *IPv4-converted*. Du point de vue du client, DNS64 se comporte comme n'importe quel relai DNS de proximité. Il accepte les requêtes et les transfère au serveur DNS de rattachement, s'il ne dispose pas déjà de l'information dans son cache local. Lorsque le client IPv6 formule une requête AAAA, le relais DNS la transfère au serveur DNS. Si la réponse est une réponse de type A uniquement, il ajoute un préfixe particulier, conforme au RFC 6052, aux 32 bits de l'adresse IPv4. Les paquets du client ayant une adresse destination avec ce préfixe seront routés par le réseau vers le relais réseau (NAT64). Le préfixe habituellement réservé pour cet usage par le RFC 6052 est le

préfixe bien connu WKP (*Well Known Prefix*) (`64:ff9b::/96`). Toutefois, celui-ci ne doit pas être utilisé pour traduire des adresses privées définies par le RFC 1918. On peut également, alors, employer un préfixe spécifique NSP (*Network Spécifique Prefix*) non utilisé et réservé à cet usage du plan d'adressage du site en respectant le format du RFC 6052. Dans la figure 2, le préfixe utilisé noté `pref64::` indique un préfixe IPv6 réservé à l'usage de la traduction.

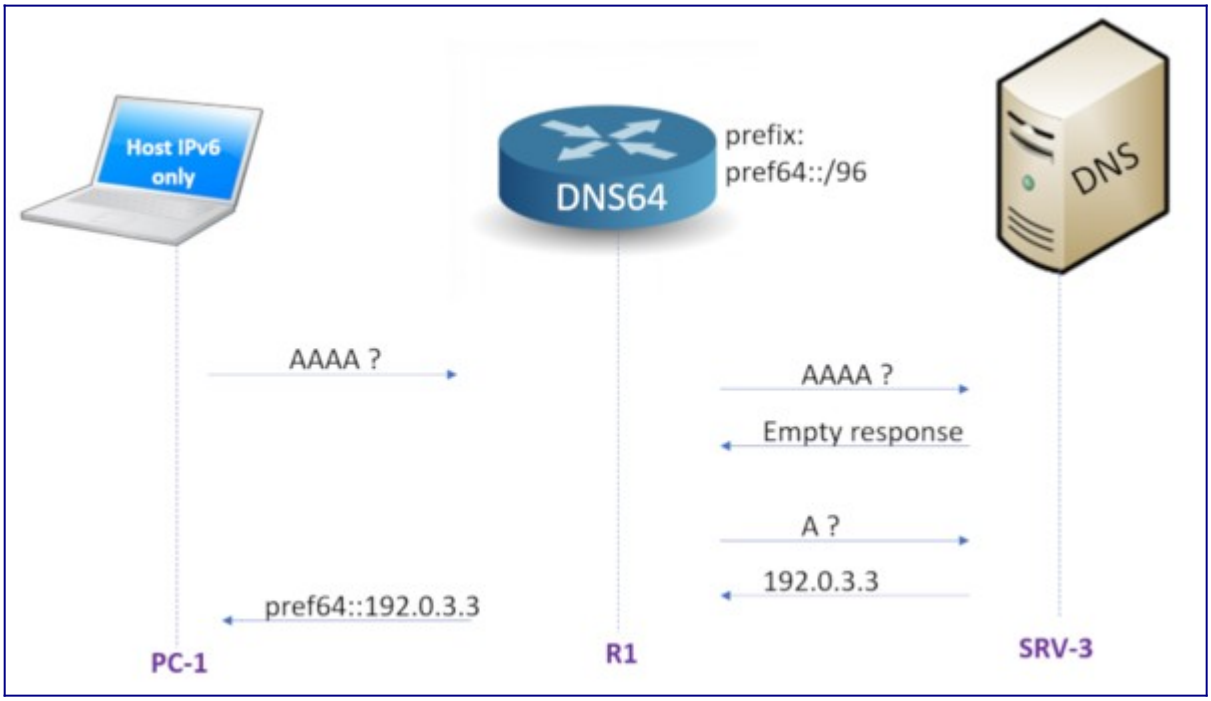

Figure 2 : Opérations du DNS64.

Allocation d'adresses

Le préfixe IPv6 alloué à l'organisation en charge du réseau IPv6 est `fd75:e4d9:cb77::/48`. Elle réserve le SID (*Subnet ID*) 64 pour constituer un préfixe IPv6 NSP (*Network-Specific Prefix*). Le NSP utilisé pour la traduction est donc `fd75:e4d9:cb77:64::/96` (nous choisissons un préfixe de 96 bits pour embarquer l'adresse IPv4 sur les 32 bits de poids faible de l'adresse *IPv4-converted* comme indiqué par le RFC 6052). Le réseau IPv6, quant à lui, est identifié par le SID de valeur 1 pour former le préfixe `fd75:e4d9:cb77:1::/64`. Cette organisation dispose aussi du préfixe IPv4 `192.0.1.0/24`, qu'elle réserve pour les noeuds IPv6 utilisant le service NAT64. La figure 3 montre la répartition des identifiants d'interface, des SID et des préfixes IPv4. Les identifiants d'hôte, au niveau du réseau IPv4 central, sont 1 pour R1 et 2 pour R2. Les identifiants d'hote pour R2 sur les réseaux Net2 (192.0.2.0/24) et Net3 (192.0.3.0/24) sont fixés à 254.

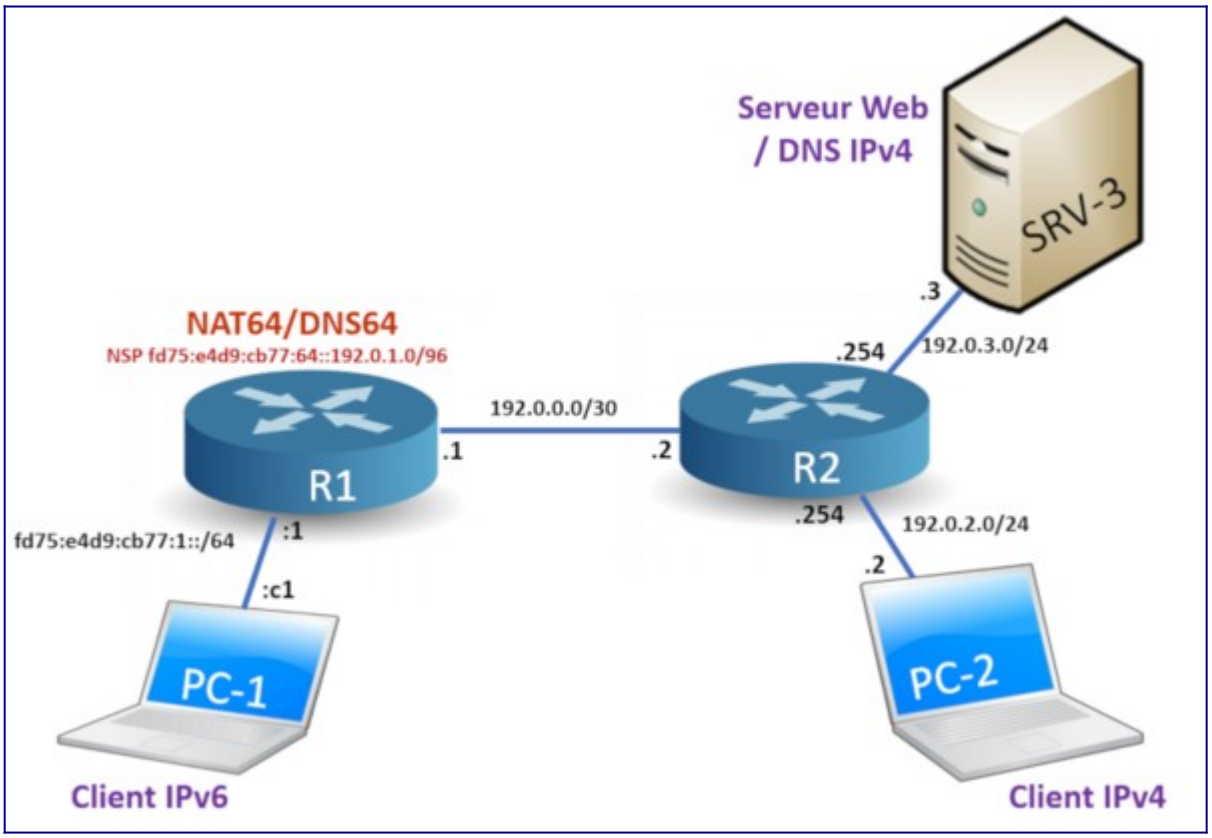

Figure 3 : Allocation des préfixes et identifiants.

L'allocation des adresses pour chaque noeud est donc faite selon le tableau 1.

Noeuds	Interface	@ IPv4	@ IPv6
PC-1	eth0		**fd75:e4d9:cb77:1::c1**
	eth0	*192.0.1.1*	**fd75:e4d9:cb77:64::192.0.1.1**
R1	eth0		**fd75:e4d9:cb77:1::1**
	eth0	*192.0.1.254/24*	**fd75:e4d9:cb77:64::192.0.1.254**
	eth1	**192.0.0.1/30**	
R2	eth1	**192.0.0.2/30**	
	eth0	**192.0.2.254/24**	
	eth3	**192.0.3.254/24**	
PC-2	eth0	**192.0.2.2/24**	
SRV-3	eth0	**192.0.3.3/24**	

Tableau 1 : Allocation des adresses.

Il est à noter que bien que PC-1 soit un noeud IPv6 uniquement, il possède une adresse IPv4. Cette adresse n'est pas allouée à l'interface. Elle est imbriquée dans une adresse IPv6 (*IPv4-translatable*) qui sera attribuée à l'interface réseau de PC1. De ce fait, PC-1 aura bien 2 adresses unicast IPv6 routables : une utilisée pour les communications avec des noeuds IPv6 (SID positionné à 1) et une autre utilisée pour les communications avec des noeuds IPv4 (SID positionné à 64). L'algorithme du choix de l'adresse source, lorsqu'il a plusieurs adresses IPv6, est défini par le RFC 6724. Lorsque PC-1 envoie une requête à SRV-3, il utilise, sur le réseau IPv6, comme adresse source : `fd75:e4d9:cb77:64::192.0.1.1` (soit en hexadécimal `fd75:e4d9:cb77:64::c000:101`), et comme adresse de destination : `fd75:e4d9:cb77:64::192.0.3.3` (soit en hexadécimal `fd75:e4d9:cb77:64::c000:303`). Une fois le NAT64 passé, les adresses deviennent respectivement sur les réseaux IPv4 : source `192.0.1.1` et destination `192.0.3.3`.

Maintenant que nous avons posé le principe de fonctionnement de la technique DNS64/NAT64 pour cette plateforme, nous allons configurer ces différents éléments.

Configuration de PC1

Vérifiez que l'adresse IPv6 de l'interface eth0 de PC-1 a bien son IID fixé à la valeur **`c1`**, conformément au tableau 1.

```
root@PC-1~# ifconfig eth0
```

Puis, ajoutez sur PC-1 l'adresse IPv6 traduisible en IPv4. La longueur du préfixe est ici fixée à 128 bits car le préfixe n'identifie pas un lien.

```
root@PC-1:~# ip -6 addr add fd75:e4d9:cb77:64::192.0.1.1/128 dev eth0
```

Vérifiez votre saisie en affichant de nouveau les adresses de l'interface eth0.

```
root@PC-1~# ip addr show eth0
```

Nota : *La notation canonique de la nouvelle adresse que vous venez d'attribuer à l'interface affiche l'adresse IPv4 en hexadécimal avec l'IID `::c000:101`*

Ensuite, il faudrait théoriquement indiquer une route au préfixe NSP réservé à la traduction. Cette route doit faire converger le trafic vers le traducteur NAT64. Ceci serait surtout vrai s'il y avait des routeurs intermédiaires entre PC-1 et le traducteur NAT64, ou si le traducteur était hébergé sur un équipement distinct du routeur. Elle pourrait être positionnée par la commande :

```
route -A inet6 fd75:e4d9:cb77:64::/64 gw fd75:e4d9:cb77:1::1
```

Dans notre cas, PC-1 n'a pas besoin de route spécifique pour le préfixe NSP IPv6 de traduction puisque la passerelle de traduction NAT64 est elle-même hébergée sur le routeur par défaut pour PC-1.

Enfin, renseignez le serveur de noms qui sera utilisé au niveau du réseau IPv6 : il s'agit du relais DNS qui sera configuré par la suite sur R1. L'adresse de R1 est indiquée dans le fichier `/`

`etc/resolv.conf` qui ne contiendra qu'une seule ligne. Pour éviter que la seule ligne puisse être découpée (par l'insertion d'un retour à la ligne) par l'éditeur, l'éditeur de texte `nano` sera appelé avec l'option `-w` (*no wrap*). L'appel de l'éditeur sur PC-1 est donc le suivant :

```
root@PC-1:~# nano -w /etc/resolv.conf
```

et saisissez la ligne ci-dessous avant de sauvegarder le fichier à l'aide de la commande , puis de sortir de l'éditeur par :

```
nameserver fd75:e4d9:cb77:1::1
```

Vérifiez vote saisie en affichant le contenu du fichier à l'aide de la commande `cat`

```
root@PC-1:~# cat /etc/resolv.conf
```

Mise en oeuvre du DNS64 sur SRV-3

Les versions récentes du logiciel serveur DNS, BIND/named, peuvent assurer le rôle DNS64, tel qu'il est illustré à la figure 2. Sur notre plateforme, c'est le serveur SRV-3 qui supporte à la fois le service DNS et la fonction de DNS64 auxilliaire. Nous allons passer en revue sa configuration avant de tester son fonctionnement. Observons le contenu de la base de données du serveur en consultant le fichier */etc/bind/db.tp à l'aide de la commande `cat`*

```
root@SRV-3:/# cat /etc/bind/db.tp
```

On constate que certaines ressources sont référencées à la fois en IPv6 et en IPv4 avec des RR (*Resource Record*) de type AAAA et A et d'autres uniquement en IPv4 sous forme de RR de type A.

Les options de démarrage du serveurs sont fixées dans le fichier */etc/bind/named.conf.options*

```
root@SRV-3:/# cat /etc/bind/named.conf.options
```

Nota *Dans ce fichier, les commentaires sont préfixés par `//` (*double caractères diviseur*).*

A la fin du fichier vous trouverez les clauses de configuration de la fonction DNS64. Quel est le préfixe choisi pour la translation ? Pour quels clients cette fonction est elle activée ?

Avant de procéder aux tests, nous devons activer la fonction de relais récursif DNS de proximité de R1 pour que les requêtes des clients sur le sous réseau Net1 soient relayées vers le serveur DNS SRV-3, en enchaînant les commandes suivantes

```
vyos@r1$ configure
[edit]
vyos@r1# set service dns forwarding cache-size 0
vyos@r1# set service dns forwarding listen-address fd75:e4d9:cb77:1::1
vyos@r1# set service dns forwarding nameserver 192.0.3.3
vyos@r1# commit;save
vyos@r1#
```

Pour tester le bon fonctionnement du DNS64, nous allons faire une résolution de nom du web depuis PC-1. Sur PC-1, demandez l'adresse IPv4 de `srv.tp` de la manière suivante :

```
root@PC-1:~# dig srv.tp +short
```

puis l'adresse IPv6 :

```
root@PC-1:~# dig -t AAAA srv.tp +short
```

Observez le préfixe IPv6 qui a été ajouté à l'adresse IPv4 (notée en hexadécimal `c000:303`). L'adresse IPv4 occupe les 32 bits de poids faible (à droite) de l'adresse IPv6 affichée.

Vous pouvez analyser les échanges en relançant ces commandes, après avoir démarré une capture du trafic sur l'interface `eth0` et sur l'interface `eth1` de R1 afin d'illustrer le fonctionnement du DNS64 comme le fait la figure 2. Cette capture est à effectuer lors d'une résolution de `srv.tp` en une adresse IPv6 initiée par PC-1.

Mise en oeuvre de NAT64 sur R1

TAYGA[1] (*Simple, no fuss NAT64 for Linux*) est une mise en oeuvre libre sous Linux du RFC 6145, la version 'sans état' du NAT64. Ce NAT64 fonctionne sur une machine double pile et marque la frontière entre le réseau IPv6 et le réseau IPv4. Il nécessite un service de résolution de noms reposant sur un DNS64 auxiliaire que nous venons de voir. Les messages des requêtes des clients IPv6 ont une adresse de destination dont le préfixe correspond au préfixe ajouté par le DNS64. Le rôle du relais NAT64 est de prendre en charge les flux pour ces adresses et de les traduire pour accéder aux services disponibles dans le monde IPv4.

TAYGA est installé sur un routeur en double pile, à savoir R1. Nous allons vérifier en premier lieu que R1 satisfait bien cette condition. La configuration de NAT64 sur R1 s'effectue en mode *root*. Le mode *root* va nous servir à entrer des commandes Unix. Si une session est ouverte dans le mode non *root* sur le routeur, il faut la quitter par la commande `exit`. Pour cela, entrez la commande suivante :

```
vyos@r1:~# exit
exit
vyos@r1:~$ exit
logout

Welcome to VyOS - r1 ttyS0
```

Pour vous connecter en mode *VyOS* sur R1 :

```
r1 login: vyos
Password:
Linux r1 4.19.28-amd64-vyos #1 SMP Mon Mar 11 16:03:26 CET 2019 x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
```

```
permitted by applicable law.
vyos@r1:~$
```

L'invite de commande dans ce mode est :

```
vyos@r1:~$
```

Pour passer en mode *root* sur R1 :

```
vyos@r1:~$ sudo su
root@r1:/home/vyos# sh interfaces
```

Vous pouvez observer que l'interface `eth0` est bien configurée avec une adresse IPv6 routable et que l'interface `eth1` est configurée avec une adresse IPv4. Ce noeud a bien la capacité de communiquer avec les 2 versions du protocole IP. C'est donc bien un noeud en double pile.

La configuration de **TAYGA** commence par l'édition de son fichier de configuration`/etc/tayga.conf`. Le fichier ayant été pré-configuré, nous allons nous contenter de vérifier que les paramètres correspondent bien à notre architecture en affichant son contenu à l'aide de la commande `cat`. Aidez vous de votre souris et de l'ascenceur dans la partie droite de la fenêtre pour la navigation. :

```
root@r1:/home/vyos# cat /etc/tayga.conf
...
tun-device nat64                # device name for nat64
...
ipv4-addr 192.0.1.254           # IPv4 address that TAYGA will  use
...
prefix fd75:e4d9:cb77:64::/96   # NSP
...
```

Nota : les commentaires préfixés par le caractère '#' ne sont pas à saisir. Il servent uniquement à expliciter les lignes à mettre dans le fichier.

Configurez l'interface interne logique `nat64` et les routes pour TAYGA sur R1 :

```
root@r1:/home/vyos# tayga --mktun
# create nat64 device
root@r1:/home/vyos# ip link set nat64 up
# activate nat64 device
root@r1:/home/vyos# ip addr add 192.0.0.1 dev nat64
# IPv4 router address
root@r1:/home/vyos# ip -6 addr add fd75:e4d9:cb77:1::1 dev nat64
# IPv6 router address
root@r1:/home/vyos# ip route add 192.0.1.0/24 dev nat64
# route to IPv6 nodes (nodes with IPv4-tranlsatable address)
root@r1:/home/vyos# ip -6 route add fd75:e4d9:cb77:64::/96 dev nat64
#  global route to IPv4 nodes (nodes with IPv4-converted address)
root@r1:/home/vyos# ip -6 route add fd75:e4d9:cb77:64::c000:101/120 dev
eth0 # route to PC-1 and other nodes with IPv4 converted adress on NET1
```

Nota 1 : *les commentaires préfixés par le caractère '#' ne sont pas à saisir. Ils servent uniquement à expliciter les lignes à entrer.*

Nota 2 : *Pour les deux commandes d'affectation d'une adresse IPv4 et d'une adresse IPv6 à l'interface interne logique nat64 : `ip addr add 192.0.0.1 dev nat64` et `ip -6 addr add fd75:e4d9:cb77:1::1 dev nat64` **il ne faut pas indiquer de longueur de préfixe !** sinon l'interface nat64 prendra la priorité sur l'interface eth1 et l'interface eth0 dans la table de routage pour ces préfixes respectifs et la fonction NAT64 ne fonctionnera pas.*

Sur R2, ajoutez la route vers R1 pour le préfixe IPv4 utilisé pour représenter les noeuds IPv6 dans le réseau IPv4. Pour cela, vous allez vous connecter en mode utilisateur et passer en mode Quagga :

```
vyos@r2:~$ vtysh

Hello, this is FRRouting (version 7.0).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

r2# configure terminal
r2(config)# ip route 192.0.1.0/24 192.0.0.1
r2(config)# end
r2#
```

192.0.1.0/24 représente le réseau IPv4 fictif qui est inclus dans l'infrastructure IPv6.

Démarrez **TAYGA** sur **R1** :

```
root@r1:/home/vyos# tayga
```

Le service est maintenant opérationnel. Pour le valider, faites un simple test de connectivité ping depuis PC-1 :

```
root@PC-1:~# ping6 -c 5 srv.tp
```

Maintenant, testez que le client IPv6 (sur PC-1) accède bien au service web (sur SRV-3) :

```
root@PC-1:~# curl http://srv.tp
Un contenu brut HTML doit s'afficher
```

Relancez ces deux dernières commandes en observant parallèlement avec wireshark les captures des flux de part et d'autre du traducteur **TAYGA** *(lancez parallèlement les captures wireshark sur les liens des interfaces `eth0` et `eth1` de R1)*. Quelles sont les versions du protocoles IP, quelles sont les adresses source et destination des paquets ?

Etape 2 : Connectivité IPv6 par tunnel

Au fil du temps, l'Internet en IPv6 croît. L'organisation en charge du serveur web obtient une connectivité en IPv6. Sur notre plateforme, cette connectivité IPv6 va être étendue jusqu'au réseau supoortant notre service web (srv.tp) sur SRV-3 au moyen d'un tunnel. Le tunnel va servir à traverser les équipements d'infrastructure qui ne sont pas encore compatibles avec IPv6. Dans notre cas, le tunnel reliera le réseau IPv6 (de R1) au routeur de bordure du réseau du serveur web (R2) comme indiqué par la figure 4. Le tunnel aura un préfixe extrait du préfixe

`fd75:e4d9:cb77::/48` et complété avec le SID à la valeur `fff` pour former un préfixe de 64 bits *(cf figure 4)*.

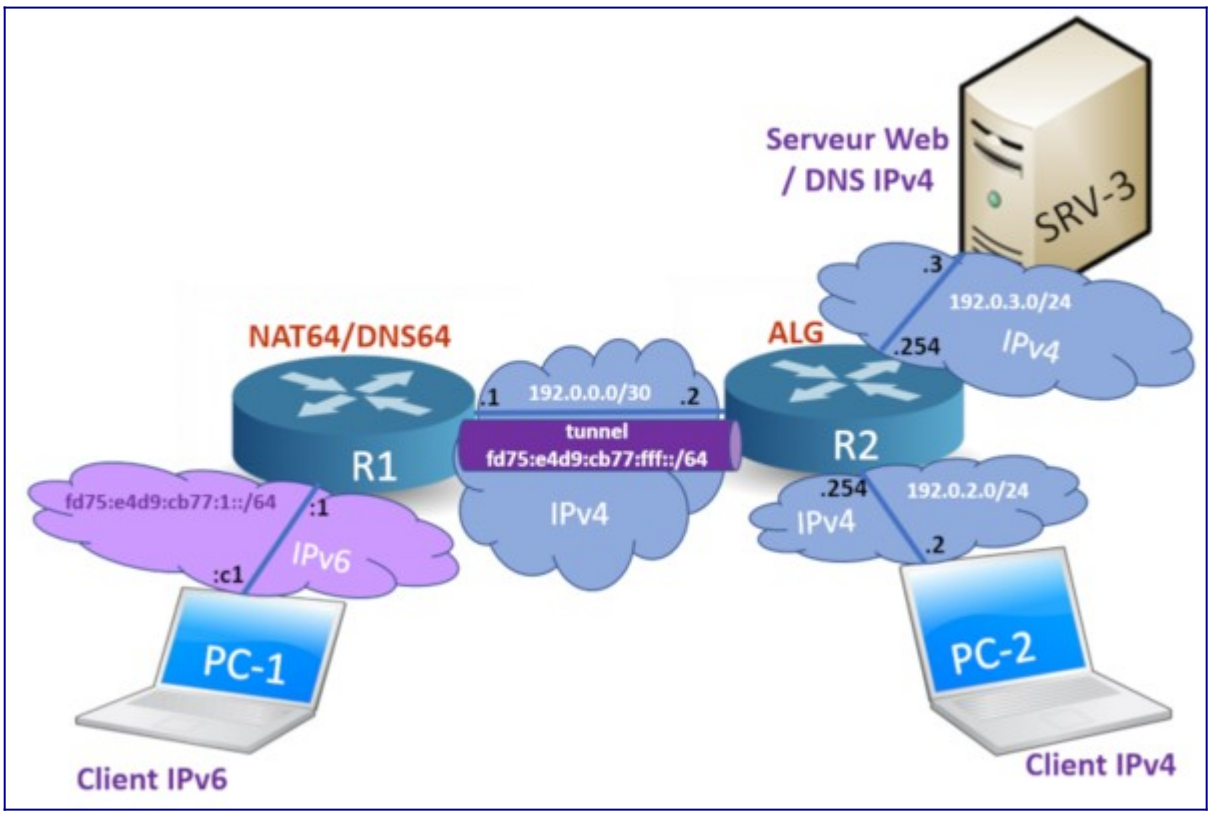

Figure 4 : Connectivité par tunnel.

Avant de passer à la configuration du tunnel, revenez en mode utilisateur sur les routeurs. Sur R1 :

```
root@r1:/home/vyos# exit
exit
vyos@r1:~$
```

sur R2:

```
r2# exit
vyos@r2:~$
```

Pour rappel, l'invite de commande dans ce mode est :

```
vyos@r1:~$
```

Sur le routeur R1, l'extrémité du tunnel (coté réseau IPv6) sera créée et configurée en enchaînant les commandes suivantes :

- La première commande consiste à passer en mode administrateur par `configure`.
- La commande `set interfaces tunnel tun0 address 'fd75:e4d9:cb77:fff::1/64'` crée une interface de tunnel configuré, dénommée `tun0`. Cette interface est identifiée par une adresse IP. Le tunnel est un lien et, en tant que tel, il possède un préfixe. Ce préfixe est ajouté à la table de routage de R1.
- Le mode d'encapsulation pour le tunnel est spécifié par la commande `set interfaces`

`tunnel tun0 encapsulation 'sit'` . Le mode SIT (*Simple Internet Transition*) indique une méthode d'encapsulation d'IPv6 dans IPv4.

- La commande `set interfaces tunnel tun0 local-ip '192.0.0.1'` précise l'adresse IPv4 des paquets IPv4 qui encapsuleront les paquets IPv6. Il s'agit ici de l'adresse source pour les paquets émis et l'adresse de destination pour les paquets reçus.
- La commande `set interfaces tunnel tun0 remote-ip '192.0.0.2'` précise l'adresse IPv4 de l'autre extrémité du tunnel. Cette adresse est utilisée comme adresse de destination pour les paquets IPv4 émis. Ainsi, un paquet IPv6 émis dans ce tunnel sera encapsulé dans un paquet IPv4 dont les adresses source et destination sont connues.
- La commande `set interfaces tunnel tun0 mtu '1480'` précise la taille de la MTU (*Maximum Transmission Unit*) appliquée sur ce tunnel. Par défaut, tous les liens utilisés par IPv6 doivent pouvoir acheminer des paquets de taille de 1280 octets sans demander de fragmentation [RFC 2460]. Dans le RFC 4213, il est précisé qu'un tunnel statique a une MTU par défaut de 1280 octets. Dans notre cas, l'interface physique sur laquelle repose le tunnel est Ethernet. La taille de la MTU est de 1500 octets. Autrement dit, un paquet IPv4 aura une longueur maximum sans segmentation de 1500 octets (en-tête compris). Si un tel paquet IPv4 encapsule un paquet IPv6, il reste 1480 octets (1500 octets moins les 20 octets pour l'en-tête IPv4) pour le paquet IPv6. Donc, pour améliorer la performance du tunnel, nous pouvons indiquer une MTU plus importante que 1280 octets. Ainsi, moins de paquets seront nécessaires pour transporter la même quantité de données pour les transferts de fichiers.
- Enfin, la commande `commit` valide la séquence des commandes en mode administrateur pour revenir en mode utilisateur.

Dans leur ensemble, les commandes à exécuter sont les suivantes :

```
vyos@r1:~$ configure
[edit]
vyos@r1# set interfaces tunnel tun0 address fd75:e4d9:cb77:fff::1/64
[edit]
vyos@r1# set interfaces tunnel tun0 encapsulation sit
[edit]
vyos@r1# set interfaces tunnel tun0 local-ip 192.0.0.1
[edit]
vyos@r1# set interfaces tunnel tun0 remote-ip 192.0.0.2
[edit]
vyos@r1# set interfaces tunnel tun0 mtu 1480
[edit]
vyos@r1# commit
[edit]
vyos@r1#
```

Vérifiez votre saisie en affichant la configuration des interfaces (en mode Quagga) :

```
vyos@r1# vtysh

Hello, this is FRRouting (version 7.0).
Copyright 1996-2005 Kunihiro Ishiguro, et al.
```

```
r1# show interface
```

Vous devez retrouver sur l'interface `tun0` l'adresse IPv6 que vous venez d'attribuer. Vérifiez la configuration des routes en affichant le contenu de la table de routage; une entrée indexée par le préfixe du tunnel et pointant sur tun0 doit avoir été ajoutée :

```
r1# show ipv6 route
```

Recommencez la même série de commandes, en ajustant les paramètres, pour l'autre extrémité du tunnel ; à savoir, sur le routeur R2.

```
vyos@r2:~$ configure
[edit]
vyos@r2# set interfaces tunnel tun0 address fd75:e4d9:cb77:fff::2/64
[edit]
vyos@r2# set interfaces tunnel tun0 encapsulation sit
[edit]
vyos@r2# set interfaces tunnel tun0 local-ip 192.0.0.2
[edit]
vyos@r2# set interfaces tunnel tun0 remote-ip 192.0.0.1
[edit]
vyos@r2# set interfaces tunnel tun0 mtu 1480
[edit]
vyos@r2# commit
[edit]
vyos@r2#
```

Vérifiez votre saisie en affichant la configuration des interfaces (en mode Quagga) :

```
vyos@r2# vtysh

Hello, this is FRRouting (version 7.0).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

r2# show interface
```

Vous devez retrouver sur l'interface `tun0` l'adresse IPv6 que vous venez d'attribuer. Vérifiez la configuration des routes en affichant le contenu de la table de routage; une entrée indexée par le préfixe du tunnel et pointant sur tun0 doit avoir été ajoutée :

```
r2# show ipv6 route
```

Depuis R1, toujours en mode Quagga, vérifiez que le tunnel est actif et fonctionne en effectuant un test de connectivité avec l'autre extrémité du tunnel :

```
r1# ping ipv6 fd75:e4d9:cb77:fff::2
```

Nota : l'arrêt de la commande `ping` s'effectue par un 'Ctrl + C' (appui simultané sur les touches Ctrl et C).

Sur une des interfaces Ethernet entre R1 et R2, lancez une capture wireshark pour observer les paquets qui vont transiter par le tunnel.

Recommencez le test de connectivité depuis PC-1.

```
root@PC1:~# ping6 -c 5 fd75:e4d9:cb77:fff::2
```

Cela fonctionne, la connectivité IPv6 a bien été étendue jusqu'au routeur R2.

Si vous avez lancé la capture de trafic sur une des interfaces Ethernet entre R1 ou R2, vous pouvez voir les messages ICMPv6 encapsulés dans des paquets IPv6, eux-mêmes encapsulés dans des paquets IPv4. Observez les différentes encapsulations : `[Ethernet[IPv4[IPv6[ICMPv6]]]]`. Vous pouvez voir comment IPv6 est encapsulé dans un paquet IPv4. D'ailleurs, quel est le numéro de protocole utilisé dans l'en-tête IPv4 pour identifier un contenu IPv6 ?

On constate sur la capture, que PC-1 a utilisé son adresse traduisible `fd75:e4d9:cb77:64::c000:101` comme adresse source. Cela fonctionne-t-il également avec l'autre adresse routable, `fd75:e4d9:cb77:1::c1` de son interface `eth0` ?

Pour forcer l'adresse source utilisez le paramètre `-I` (*Interface*) de la commande Ping6 en forçant sa valeur à l'adresse source souhaitée.

```
root@PC1:~# ping6 -c 5 -I fd75:e4d9:cb77:1::c1 fd75:e4d9:cb77:fff::2
```

Cela fonctionne également. Observez le changement d'adresse source des paquets émis par PC-1 sur la capture.

Au cours de cette étape, nous avons utilisé un tunnel configuré (encore appelé statique). La connectivité IPv6 a pu ainsi s'étendre au-dessus d'équipements qui devaient rester en IPv4. Ceci montre qu'il n'est pas nécessaire de changer ses équipements réseaux pour déployer IPv6 et que cela se fait bien progressivement.

Etape 3 : Configurer un reverse proxy web sur R2

Dans cette dernière étape, la base des clients en IPv6 est devenue conséquente. Une connectivité IPv6 arrive jusqu'au réseau du serveur web. Celui-ci fonctionne malgré tout toujours en IPv4. Cependant, l'hébergeur du serveur ne souhaite pas passer le service web en IPv6. Il n'est pas certain que l'applicatif web soit compatible avec IPv6, surtout s'il n'a pas les codes sources. Dans le doute, il préfère donc laisser son service en IPv4 mais il souhaite cependant répondre, nativement en IPv6, aux requêtes des clients en IPv6.

Nous avons vu, dans l'étape 1, qu'un client IPv6 peut accéder au serveur à l'aide d'un NAT64. Mais nous avons vu aussi qu'il faut faire des modifications de routes, qu'il faut gérer un plan d'adressage en IPv6 et en IPv4 pour la traduction. Tout ceci n'est pas toujours possible, notamment si les droits pour la configuration du réseau ne sont pas disponibles aux personnes en charge des serveurs. Une autre solution existe dans ce cas. Cette solution repose sur le niveau application de l'architecture de réseau et n'implique plus la couche de réseau. Il s'agit maintenant de déployer une passerelle applicative dite ALG (*Application Layer Gateway*).

Dans cette étape, nous allons mettre en oeuvre un *reverse proxy* web. Il sera en charge de recevoir les requêtes IPv6 pour le serveur et de les relayer au serveur en IPv4.

Sur le routeur R2, passez en mode *root*, en terminant la session en cours par la commande `exit`, ensuite `sudo su`. L'affichage des interfaces, permet de s'assurer de nouveau de la connectivité IPv6 de R2 par son interface `tun0` :

```
r2# exit
vyos@r2:~$
vyos@r2:~$ sudo su
root@r2:/home/vyos# sh interfaces
```

Vérifiez la configuration "reverse proxy IPv6" de R2 en consultant le fichier `/etc/nginx/sites-available/default` à l'aide de la commande `less`.

Rappel : *La commande `less` vous permet d'afficher, sans l'éditer, le contenu d'un fichier texte en naviguant simplement avec les touches de direction du curseur. Pour quitter l'affichage du fichier, appuyez simplement sur la touche 'q' qui signifie "quit".*

```
root@r2:/home/vyos# less /etc/nginx/sites-available/default
```

Verifiez d'abord que le serveur reverse-proxy sera bien en écoute sur ses interfaces ipv6 en consultant la clause **listen**

```
...
server {
       listen [::]:80 default_server;
...
```

Vérifiez ensuite sa configration en mode "reverse-proxy" vers l'adresse du serveur SRV-3 en consultant la clause **location**

```
...
location / {
        proxy_pass http://192.0.3.3/;
}
...
```

Redémarrez le serveur nginx :

```
root@r2:/home/vyos# service nginx restart
```

Vérifiez l'état du service

```
root@r2:/home/vyos# service nginx status
```

Le service de DNS doit maintenant identifier le *reverse proxy* comme le serveur web en IPv6. Pour cela, sur **SRV-3**, modifiez la configuration du DNS pour enregistrer l'adresse IPv6 du proxy en ajoutant un RR AAAA à la valeur `fd75:e4d9:cb77:fff::2:`

```
root@SRV-3:/# nano -w /etc/bind/db.tp
...
srv             IN              A               192.0.3.3
```

```
                IN              AAAA            fd75:e4d9:cb77:fff::2
;
```

Relancez le serveur named. Pour cela, validez d'abord votre nouvelle configuration à l'aide de la commande `named-checkconf -z`. Si tout est correct, relancez le service à l'aide de la commande `service bind9 restart`:

```
root@SRV-3:/# named-checkconf -z
...
root@SRV-3:/# service bind9 restart
...
[ ok ] Starting domain name service...: bind9.
```

Verifiez l'état du service

```
root@SRV-3:/# service bind9 status
```

Sur le relais DNS de R1, afin de réinitialiser le cache DNS, rédemarrez le service `pdns-recursor`. Sinon les réponses des reqêtes de PC-1 concernant SRV-3 pourraient continuer à être basée sur l'adresse traduisible `fd75:e4d9:cb77:64:c000:303` de SRV-3 mémorisée dans le cache DNS du relais R1. Sur R1 en mode *root*

```
r1#  exit
[edit]
vyos@r1# sudo su
root@r1:/home/vyos# service pdns-recursor restart
```

Sur le client IPv6 PC-1, vérifiez que la résolution de nom du serveur web donne bien l'adresse IPv6 du proxy `fd75:e4d9:cb77:fff::2`.

```
root@PC-1:~# dig -t AAAA srv.tp +short
```

Vérifiez que l'accès au web est maintenant possible à travers le proxy par la commande `curl`. Mais, avant cela, vous allez activer une capture du trafic entre R1 et R2, et entre R2 et SRV-3.

```
root@PC-1:~# curl -6 http://srv.tp
Un contenu brut HTML doit s'afficher
```

Dans les fenêtres de capture, vous pouvez vérifier que :

- les paquets d'établissement d'une connexion TCP entre PC1 et R2 sur IPv6 circulent entre R1 et R2 ;
- l'adresse de destination IPv6 de la connexion (`fd75:e4d9:cb77:fff::2`) est l'adresse IPv6 du *reverse proxy*, qui est aussi celle de l'interface du tunnel du noeud R2 ;
- après R2, les paquets sont au format IPv4. Quelles sont les adresses de source et de destination ?

On constate, de nouveau sur la capture, que PC-1 a utilisé son adresse traduisible `fd75:e4d9:cb77:64::c000:101` comme adresse source. Cela fonctionne-t-il également avec l'autre adresse routable, `fd75:e4d9:cb77:1::c1` de son interface `eth0` ?

Pour forcer l'adresse source utilisez le paramètre `--interface` de la commande `curl` en forçant sa valeur à l'adresse source souhaitée.

```
root@PC-1:~# curl -6 --interface fd75:e4d9:cb77:1::c1 http://srv.tp
Un contenu brut HTML doit s'afficher
```

Conclusion

Au cours de cette activité, nous avons pu déployer 2 solutions d'interopérabilité entre un client 'IPv6 uniquement' et un serveur resté en IPv4. Nous avons aussi pu expérimenter comment étendre la connectivité au-dessus d'équipements qui ne sont pas en IPv6, au moyen d'un tunnel. Au-delà des techniques, ce que nous avons voulu montrer, c'est que le déploiement d'IPv6 s'effectue bien progressivement, sans arrêter l'existant ou sans avoir à acheter de nouveaux équipements ou, pire, remplacer ceux déjà en place. De plus, cela montre également que le déploiement de nouveaux réseaux peut (devrait !?) se faire nativement (et uniquement !?) en IPv6 puisque l'accès aux ressources restées dans l'ancien monde IPv4 peut se faire de manière transparente du point de vue des clients 'uniquement v6'.

Certes, la coexistence avec IPv4 complique le déploiement d'IPv6. Mais la réutilisation de l'existant est la condition nécessaire à l'adoption d'IPv6 dans l'Internet. Cette activité a montré que des solutions fonctionnelles existent pour utiliser IPv6 dans un réseau IPv4.

Pour aller plus loin

1. ↑ http://www.litech.org/tayga/