MOOC
Objectif IPv6 !
vers l'internet nouvelle génération

Document Compagnon[1]

Séquence 4

Activité pratique

Faites 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 ».

Sommaire

Les auteurs **5**

Session3-Activité 46: Faites interopérer des applications IPv6 et IPv4 ..... **9**

Etape 0: Situation initiale **9**

Etape 1: Configuration de NAT64/DNS64 ..... **10**
Allocation d'adresses 12
Configuration de PC1 13
Mise en oeuvre du DNS64 sur R1 14
Mise en oeuvre de NAT64 sur R1 15

Etape 2: Connectivité IPv6 par tunnel **17**

Etape 3: Configurer un reverse proxy web sur R2 ..... **20**

Conclusion **22**

Pour aller plus loin **22**

Session3-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 plus marginal et 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 PC1, 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 IPv4 et en IPv6;
- un routeur R2 en IPv4: c'est un noeud représentant l'Internet en IPv4;
- un serveur web IPv4. Ce service sera hébergé sur le noeud PC2 et représentera les contenus disponibles sur l'Internet IPv4. Ce noeud hébergera également un serveur DNS.

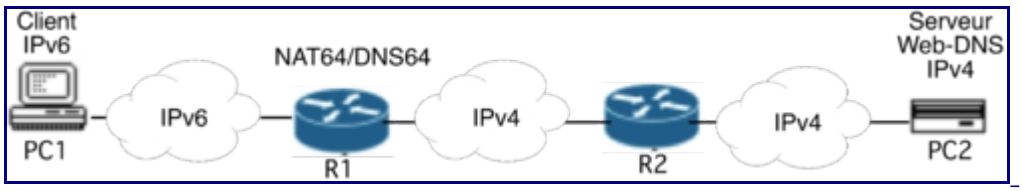

Figure 1: Plateforme de l'activité.

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 PC2 les services applicatifs de nommage DNS (`named -c /usr/local/etc/bind/named.conf`) et web (`nginx`) ont été automatiquement lancés au démarrage de la machine.

Note: pour vous loguer sur les stations PC1 et PC2, l'identifiant est **apprenant** 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` . La commande `dig` est disponible sur PC2:

```
apprenant@MOOCIPv5:~$ dig @192.0.2.1 -t A www.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:

```
login: vyos
Password: vyos
```

L'adresse du serveur de noms à utiliser dans la commande `host` est l'adresse de PC2, soit `192.0.2.1` . La syntaxe de la commande devient alors, sur R1:

```
vyos@vyos:~$ host www.tp 192.0.2.1
```

Vérifiez ensuite le bon fonctionnement du service web de PC2 depuis R1, à l'aide de la commande `curl` :

```
vyos@vyos:~$ curl http://www.tp

Un contenu HTML doit s'afficher
```

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

```
vyos@vyos:~$ curl http://192.0.2.1

Un contenu 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 `www.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 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 serveur DNS de rattachement. 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 employer un préfixe 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.

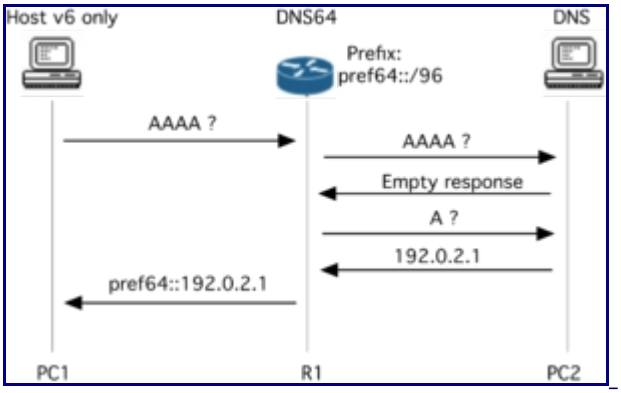

Figure 2: Opérations du DNS64.

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 `192.0.3.0/24` , 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 `129` pour R2 et `130` pour R1.

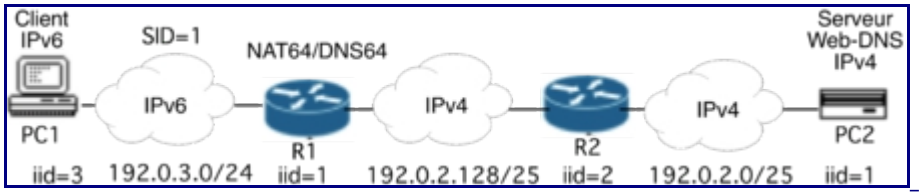

Figure 3: Allocation des préfixes et identifiants.

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

Noeuds	@ IPv4	@ IPv6
PC1		`fd75:e4d9:cb77:1::3`
	`192.0.3.3`	`fd75:e4d9:cb77:64::192.0.3.3`
R1	`192.0.3.1`	`fd75:e4d9:cb77:64::192.0.3.1`
		`fd75:e4d9:cb77:1::1`
	`192.0.2.130`	
R2	`192.0.2.129`	
	`192.0.2.2`	
PC2	`192.0.2.1`	

Tableau 1: Allocation des adresses.

Il est à noter que bien que PC1 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, PC1 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 PC1 envoie une requête à PC2, il utilise, sur le réseau IPv6, comme adresse source: `fd75:e4d9:cb77:64::192.0.3.3` (soit en hexadécimal `fd75:e4d9:cb77:64::c000:303`), et comme adresse de destination: `fd75:e4d9:cb77:64::192.0.2.1` (soit en hexadécimal `fd75:e4d9:cb77:64::c000:201`). Une fois le NAT64 passé, les adresses deviennent respectivement sur les réseaux IPv4: source `192.0.3.3` et destination `192.0.2.1` .

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

Fixez l'adresse IPv6 de l'interface eth0 de PC1, avec l'IID positionné à la valeur `3` , conformément au tableau 1. La commande de configuration d'interface doit s'exécuter avec les droits *root* indiqués par la commande *sudo* :

```
apprenant@MOOCIPv6:~$ sudo ifconfig eth0 add fd75:e4d9:cb77:1::3/64
```

Puis, ajoutez sur PC1 l'adresse IPv6 traduisible en IPv4. La longueur du préfixe est ici fixée à 128 bits car le préfixe n'identifie pas un lien.

```
apprenant@MOOCIPv6:~$ sudo ifconfig eth0 add
fd75:e4d9:cb77:64::192.0.3.3/128
```

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 PC1 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, PC1 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 PC1.

Enfin, renseignez le serveur de noms qui sera utilisé au niveau du réseau IPv6: il s'agit du DNS64 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 PC1 est donc le suivant:

```
apprenant@MOOCIPv6:~$ sudo nano -w /etc/resolv.conf
```

et placez la ligne ci-dessous à la place de l'existant dans le fichier:

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

Mise en oeuvre du DNS64 sur R1

Les versions récentes du logiciel serveur DNS, BIND/named, peuvent assurer le rôle DNS64. Le logiciel **TOTD** (*Trick Or Treat Daemon*) peut également être utilisé pour cet usage. **TOTD** est un petit DNS cache également dénommé DNS auxiliaire dans notre contexte de mise en oeuvre de la traduction NAT64. Son objectif principal est de traduire les adresses IPv4 en IPv6 en ajoutant le préfixe réservé à la traduction à l'adresse IPv4 retournée par le DNS et de répondre au client DNS avec le RR de type AAAA correspondant (voir la figure 2).

Sur notre plateforme, c'est le noeud R1 qui supportera le service DNS auxiliaire et le noeud PC2 qui fera office de serveur DNS général. La configuration de NAT64/DNS64 sur R1 s'effectue en mode *root* . Le mode *root* va nous servir à entrer des commandes Unix. Si une session est ouverte dans un mode non *root* sur le routeur, il faut la quitter par la commande `exit` :

```
vyos@vyos~$ exit
```

Pour vous reconnecter en mode *root* sur R1:

```
login: root
Password: root
```

L'invite de commande dans ce mode est:

```
 root@vyos:~#
```

Avant de démarrer le service DNS auxiliaire, complétez le contenu du fichier de configuration par les informations nécessaires à la traduction. Sur R1, éditez le fichier de configuration de **TOTD** :

```
root@vyos:~# nano -w /etc/totd.conf
```

pour localiser et renseigner les informations suivantes dans le fichier:

```
forwarder 192.0.2.1          # IPv4 address for name server
prefix fd75:e4d9:cb77:64::   # /96 by default
port 53
```

Le *forwarder* correspond à l'adresse IPv4 du serveur DNS général. La ligne *prefix* indique dans notre cas le NSP qui sera utilisé pour transformer une adresse IPv4 en une adresse IPv6 dite (*IPv4-converted*).

Démarrez le logiciel:

```
root@vyos:~# /etc/init.d/totd start
```

Vérifiez que TOTD ne s'est pas arrêté, en consultant la fin du journal système à l'aide de la commande:

```
root@vyos:~# tail /var/log/messages
```

Vous pouvez également vérifier la présence du processus 'daemon totd' à l'aide de la commande:

```
root@vyos:~# ps -edf | grep totd
```

Note: le symbole | (pipe Unix) est obtenu en appuyant simultanément sur les touches 'Alt Gr' et '6' du clavier azerty de votre PC ou Shift + Alt + L (appui simultané sur les 3 touches) de votre Mac.

Pour tester le bon fonctionnement du DNS64, nous allons faire une résolution de nom du web depuis PC1. Sur PC1, demandez l'adresse IPv4 de `www.tp` de la manière suivante:

```
apprenant@MOOCIPv6:~$ dig www.tp +short
```

puis l'adresse IPv6:

```
apprenant@MOOCIPv6:~$ dig -t AAAA www.tp +short
```

Observez le préfixe IPv6 qui a été ajouté à l'adresse IPv4 (notée en hexadécimal `c000:201`). 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 `www.tp` en une adresse IPv6 initiée par PC1.

Ainsi, par l'affichage des messages *DNS standard query* et *DNS standard response* pour les requêtes en amont et en aval de R1, vous verrez que le DNS auxiliaire a bien transformé la réponse de type 'A' en une réponse de type 'AAAA'.

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 DNS auxiliaire tel que **TOTD** 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. Pour cela, entrez la commande suivante:

```
root@vyos~# ifconfig
```

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` :

```
root@vyos~# nano -w /etc/tayga.conf
...
tun-device nat64                # device name for nat64
ipv4-addr 192.0.3.1             # 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 les routes pour TAYGA sur R1:

```
root@vyos~# tayga --mktun              # create nat64 device
root@vyos~# ip link set nat64 up       # activate nat64 device
root@vyos~# ip addr add 192.0.2.130/25 dev nat64                      # IPv4
router address
root@vyos~# ip -6 addr add fd75:e4d9:cb77:1::1/64 dev nat64           # IPv6
router address
root@vyos~# ip route add 192.0.3.0/24 dev nat64                       #
route to IPv6 nodes (nodes with IPv4-tranlsatable address)
root@vyos~# ip -6 route add fd75:e4d9:cb77:64::/96 dev nat64          #
route to IPv4 nodes (nodes with IPv4-converted address)
root@vyos~# ip -6 route add fd75:e4d9:cb77:64::c000:303/120 dev eth0 #
route to pc1
```

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

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:

```
root@vyos:~$ vtysh
vyos# configure terminal
vyos(config)# ip route 192.0.3.0/24 192.0.2.130
vyos(config)# exit
vyos#
```

192.0.3.0/24 représente le réseau IPv4 fictif qui est inclus dans l'infrastructure IPv6.

Démarrez **TAYGA** sur R1:

```
root@vyos~# tayga
```

Le service est maintenant opérationnel. Pour le valider, faites un simple test de connectivité ping depuis PC1:

```
apprenant@MOOCIPv6:~$ ping6 -c 5 www.tp
```

Observez le contenu des trames échangées de part et d'autre du traducteur TAYGA (capture sur les liens des interfaces `eth0` et `eth1` de R1): type de datagrammes, adresses source et destination...

Maintenant, testez que le client IPv6 (sur PC1) accède bien au service web (sur PC2):

```
apprenant@MOOCIPv6:~$ curl http://www.tp
```

Observez les captures de part et d'autre du traducteur **TAYGA** .

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 est établie 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. Les identifiants d'interfaces des extremités du tunnel sont indiqués par la figure 4.

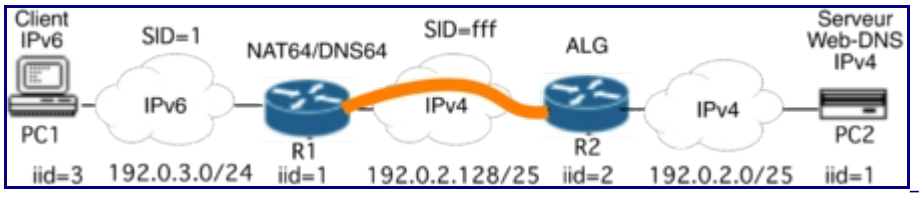

Figure 4: Connectivité par tunnel.

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

```
root@vyos:~# exit
```

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

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

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 '192.0.2.130'` 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 '192.0.2.129'` 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` termine la séquence des commandes en mode administrateur pour revenir en mode utilisateur.

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

```
vyos@vyos:~$ configure
vyos@vyos# set interfaces tunnel tun0 address 'fd75:e4d9:cb77:fff::1/64'
vyos@vyos# set interfaces tunnel tun0 encapsulation 'sit'
vyos@vyos# set interfaces tunnel tun0 local-ip '192.0.2.130'
vyos@vyos# set interfaces tunnel tun0 remote-ip '192.0.2.129'
vyos@vyos# set interfaces tunnel tun0 mtu '1480'
vyos@vyos# commit
```

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

```
vyos@vyos:# vtysh
vyos# 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:

```
vyos# 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@vyos:~$ configure
vyos@vyos# set interfaces tunnel tun0 address 'fd75:e4d9:cb77:fff::2/64'
vyos@vyos# set interfaces tunnel tun0 encapsulation 'sit'
vyos@vyos# set interfaces tunnel tun0 local-ip '192.0.2.129'
vyos@vyos# set interfaces tunnel tun0 remote-ip '192.0.2.130'
vyos@vyos# set interfaces tunnel tun0 mtu '1480'
vyos@vyos# commit
```

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:

```
vyos# 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, effectuez une capture lors du ping entre ces 2 noeuds afin d'observer les paquets échangés. Vous pouvez voir comment IPv6 est encapsulé dans un paquet IPv4. D'ailleurs, quel est le numéro de protocole utilisé par IPv4 pour identifier IPv6?

Recommencez le test de connectivité depuis PC1.

```
apprenant@MOOCIPv6:~$ ping6 -c 5 fd75:e4d9:cb77:fff::2
```

Oups! cela ne marche plus. Quel est le problème? Vous pouvez vérifier que PC1 a bien une route par défaut:

```
apprenant@MOOCIPv6:~$ route -A inet6
```

Qu'en est-il de la route de la réponse à la requête du ping? Regardez la table de routage sur R2 (en mode Quagga):

```
vyos@vyos:~$ vtysh
vyos# show ipv6 route
```

R2 n'a pas de route pour joindre le lien de PC1 qui est identifié par le préfixe `fd75:e4d9:cb77:1::/64` . Il faut donc ajouter une route à R2. En fait, nous n'allons pas ajouter une route particulière mais une route générale, à savoir la route par défaut. En effet, on peut considérer que l'Internet v6 est du coté de R1. Sur R2, cet ajout s'effectue en mode configuration dans Quagga de la manière suivante:

```
vyos# configure terminal
vyos(config)# ipv6 route  ::/0  tun0
vyos(config)# exit
```

Verifiez que l'entrée "route par défaut" a bien été ajoutée à la table de routage IPv6 de R2

```
vyos# show ipv6 route
```

Recommencez le test de connectivité depuis PC1.

```
apprenant@MOOCIPv6:~$ ping6 -c 5 fd75:e4d9:cb77:fff::2
```

Si vous avez laissé 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.

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 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.

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, éditez la configuration du serveur `nginx` pour activer la redirection des requêtes vers le serveur. Pour cela, passez en mode *root* , en terminant la session en cours par la commande `exit` jusqu'à voir s'afficher l'invite de connexion:

```
login: root
password: root
```

Editez le fichier `/etc/nginx/nginx.conf` pour remplacer la directive

```
...
pid /run/nginx.pid
...
```

par

```
...
pid /var/run/nginx.pid
...
```

à l'aide de l'éditeur nano.

```
root@vyos:~# nano -w /etc/nginx/nginx.conf
```

Editez le fichier `/etc/nginx/sites-available/default` pour remplacer la directive

```
...
location / {
       # First attempt to serve request as file, then
       # as directory, then fall back to displaying a 404.
       try_files $uri $uri/ =404;
       # Uncomment to enable naxsi.rules
       # include /etc/ngingx/naxsi.rules
}
...
```

par

```
...
location / {
        proxy_pass http://192.0.2.1/;
}
...
```

Attention! le **;** en fin de ligne est nécessaire. Les **...** ne sont pas à saisir; ils représentent le contenu actuel du fichier.

L'appel de l'éditeur s'effectue par la commande:

```
root@vyos:~# nano -w /etc/nginx/sites-available/default
```

Démarrez le serveur nginx:

```
root@vyos:~# nginx -c /etc/nginx/nginx.conf
```

Le service de DNS doit maintenant identifier le *reverse proxy* comme le serveur web en IPv6. Pour cela, sur PC2, modifiez la configuration du DNS pour enregistrer l'adresse IPv6 du proxy:

```
apprenant@MOOCIPv6:~$ sudo nano -w /usr/local/etc/bind/db.tp
...
www             IN              A               192.0.2.1
                IN              AAAA            fd75:e4d9:cb77:fff::2
```

Relancez le serveur named . Pour cela, arrêtez le processus *named* puis relancez-le:

```
apprenant@MOOCIPv6:~$ sudo killall named
apprenant@MOOCIPv6:~$ sudo named -c /usr/local/etc/bind/named.conf
```

Sur le client IPv6 PC1, vérifiez que la résolution de nom du serveur web donne bien l'adresse IPv6 du proxy.

```
apprenant@MOOCIPv6:~$ dig -t AAAA www.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 PC2.

```
apprenant@MOOCIPv6:~$ curl -6 http://www.tp
```

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?

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 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/