Modélisation d'un lien WAN avec PPPoE

Philippe Latu
philippe.latu(at)inetdoc.net

http://www.inetdoc.net

Résumé

Cet article est un exemple de modélisation d'un lien de réseau étendu (WAN) utilisant le protocole PPP. La modélisation s'appuie sur deux instances de systèmes virtualisés qui ne disposent que d'interfaces Ethernet. On présente donc une implémentation particulière du protocole PPP sur Ethernet : PPPoE.

Table des matières

1. Copyright et Licence

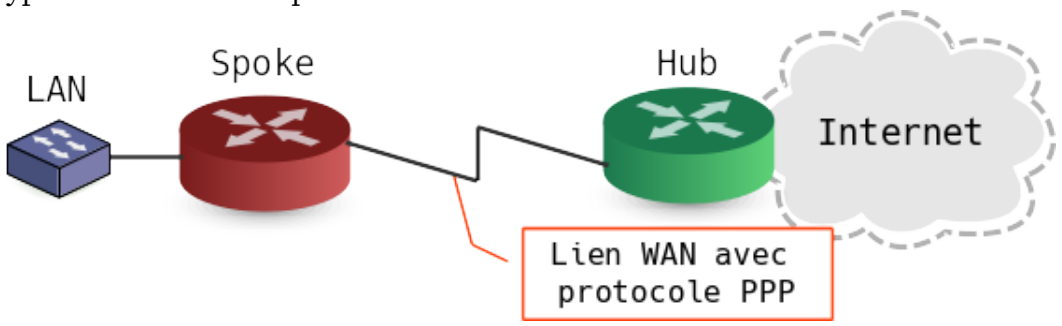
1.1. Méta-information

Cet article est écrit avec DocBook[1] XML sur un système Debian GNU/Linux[2]. Il est disponible en version imprimable au format PDF : `pppoe.pdf`[3].

[1] http://www.docbook.org
[2] http://www.debian.org
[3] http://www.inetdoc.net/pdf/pppoe.pdf

2. Le contexte

Dans la série de travaux pratiques sur le thème de l'Interconnexion de réseaux locaux et étendus[4], plusieurs supports utilisent des liaisons de réseaux étendus (WAN) avec le protocole PPP. Ces sujets de travaux pratiques méritent d'être revus plusieurs fois et les équipements matériels dédiés ne sont pas toujours disponibles. L'objectif de cet article est donc de proposer une modélisation de topologie type mettant œuvre le protocole PPP.



Topologie WAN avec PPP[5]

La virtualisation de la topologie proposée ci-dessus sur un unique système hôte peut poser quelques soucis. Fort heureusement, il existe une parade ! On peut très bien utiliser le protocole Modélisation d'un lien WAN avec PPPoE[6] sur un domaine de diffusion un peu particulier pour reproduire les caractéristiques d'une liaison WAN. Relativement aux connexions physiques réelles, cette parade présente des avantages et des inconvénients que l'on peut énoncer comme suit.

Les avantages.
Toutes les caractéristiques du protocole PPP[7] sont effectivement exploitées. On illustre le volet négociation des paramètres d'une connexion et l'authentification entre deux extrémités avec les fonctions de la sous-couche Link Control Protocol (LCP). Une fois le «lien» établi, ce sont les fonctions de la sous-couche Network Control Protocol (NCP) qui permettent l'échange des adresses IP entre l'extrémité qui joue le rôle Hub (celle qui a la maîtrise du plan d'adressage) et celle qui joue le rôle Spoke (celle qui s'authentifie auprès du site central).

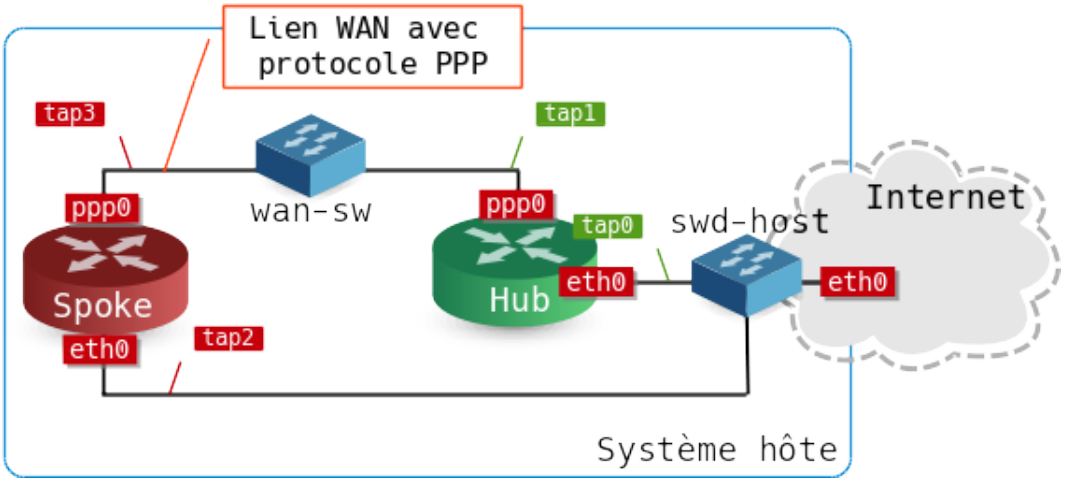
Les inconvénients.
Le recours au format de trame Ethernet est certainement le problème pédagogique le plus sensible. Si l'étude des liaisons WAN a encore un sens aujourd'hui, c'est justement de montrer qu'un autre format de trame existe : le format HDLC[8]. Souvent, les étudiants qui débutent dans le domaine des réseaux, n'envisagent pas qu'un réseau puisse s'appuyer sur un autre format de trame qu'Ethernet. Dans leur environnement immédiat, les réseaux filaires des campus universitaires sont tout Ethernet, les réseaux Wifi publics ou domestiques utilisent aussi des trames Ethernet, etc. L'utilisation du protocole PPPoE présente donc un inconvénient puisque l'on retombe dans l'utilisation du fameux format de trame Ethernet dont tous les champs sont prédéfinis et non paramétrables.

L'autre contrainte est liée à la solution de virtualisation. Dans le but d'illustrer un maximum de possibilités d'interconnexion réseau, on s'appuie sur le guide Virtualisation système et enseignement[9]. Dans la section suivante, on présente donc la mise en œuvre de la topologie avec l'hyperviseur KVM associé au commutateur Open vSwitch[10].

[4] http://www.inetdoc.net/formations/interco-lan-wan/
[5] images/pppoe-context.png
[6] http://www.inetdoc.net/articles/pppoe/
[7] http://en.wikipedia.org/wiki/Point-to-Point_Protocol
[8] http://en.wikipedia.org/wiki/HDLC
[9] http://www.inetdoc.net/guides/vm/
[10] http://openvswitch.org

3. La topologie virtualisée

Suite à la définition de topologie type donnée dans la section précédente, voici un schéma représentant la virtualisation de cette topologie. Il fait apparaître deux instances de machines virtuelles nommées Hub et Spoke ainsi que les commutateurs virtuels utilisés pour l'interconnexion entre réseaux.



Topologie de virtualisation d'un lien WAN[11]

Le routeur Hub joue le rôle de concentrateur des demandes d'accès émises par les routeurs distants : les Spokes. Dans la maquette présentée ci-dessus, on n'utilise qu'un seul routeur Spoke pour illustrer le dialogue PPP sur un seul lien WAN. En ajoutant, d'autres instances de machines virtuelles, il serait possible de multiplier les routeurs Spokes.

> **Note**
>
> Dans l'architecture des connexions xDSL[12] le routeur Hub est aussi appelé Broadband Remote Access Server ou BRAS.

Paquets et droits d'accès

Avant de passer à la configuration de l'interconnexion, il faut préparer le système hôte : installer les paquets nécessaires à la virtualisation système et ouvrir les permissions d'accès.

- On commence par l'installation des paquets utiles après avoir vérifié que le processeur du système hôte dispose bien des fonctions de virtualisation.

```
# egrep '(vmx|svm)' /proc/cpuinfo
```

```
# aptitude install qemu-system-x86 openvswitch-switch
```

- On s'assure ensuite que l'utilisateur normal est bien membre des groupes système associés à la virtualisation. Dans notre cas, on se contente des groupes `sudo` et `kvm`. L'ajout d'un utilisateur à un groupe système se fait à l'aide de la commande suivante.

```
# adduser etu kvm
```

Commutateurs et cordons de brassage

Sur le schéma ci-dessus, on compte 4 cordons de brassage. À chacun de ces cordons correspond une interface de type `tap`. Voici une copie du script de création de ces cordons de brassage virtuels.

[11] images/pppoe-virtualized.png
[12] https://upload.wikimedia.org/wikipedia/commons/d/d5/XDSL_Connectivity_Diagram_en.svg

```
#!/bin/bash

for intf in tap0 tap1 tap2 tap3
do
        if [ -z "$(ip addr ls | grep $intf)" ]; then
                echo setting $intf up
                sudo ip tuntap add mode tap dev $intf group kvm
                sudo ip link set dev $intf up
        else
                echo $intf is already there!
        fi
done
```

Voici la syntaxe de création du commutateur `wan-sw`. La création du commutateur `swd-host` utiliserait la même syntaxe. Ce dernier commutateur fait partie de la configuration permanente du système hôte puisqu'il est utilisé pour raccorder celui-ci à l'Internet.

```
$ sudo ovs-vsctl add-br wan-sw
```

On passe maintenant au brassage des cordons sur les commutateurs.

- Commutateur `swd-host`

```
$ sudo ovs-vsctl add-port swd-host tap0 -- set port tap0 vlan_mode=access
$ sudo ovs-vsctl add-port swd-host tap2 -- set port tap2 vlan_mode=access
```

- Commutateur `wan-sw`

```
$ sudo ovs-vsctl add-port wan-sw tap1 -- set port tap1 vlan_mode=access
$ sudo ovs-vsctl add-port wan-sw tap3 -- set port tap3 vlan_mode=access
```

Instances de systèmes virtuels

Le guide Virtualisation système et enseignement[13] fournit le script de lancement d'une machine virtuelle raccordée à un commutateur Open vSwitch[14] qui sert au lancement des instances de systèmes virtuels. Ce script ne décrit qu'une interface réseau pour une instance de système virtuel. Il est donc nécessaire de le compléter avec la description des interfaces supplémentaires.

```
#!/bin/bash

../scripts/ovs-startup.sh hub.qed 1024 0 \
        -device virtio-net,netdev=net2,mac=de:ad:be:ef:00:01 \
        -netdev tap,ifname=tap1,id=net2,script=no

../scripts/ovs-startup.sh spoke.qed 1024 2 \
        -device virtio-net,netdev=net2,mac=de:ad:be:ef:00:03 \
        -netdev tap,ifname=tap3,id=net2,script=no
```

Il faut noter que sur la schéma proposé ci-dessus, l'interface `eth0` du routeur Spoke, raccordée via le cordon `tap2` devient inutile dès que la configuration du lien WAN est en place. Cette interface est cependant nécessaire pour accéder à la console du routeur Spoke et installer les paquets manquants.

[13] http://www.inetdoc.net/guides/vm/
[14] http://www.inetdoc.net/guides/vm/vm.appendix-ovs-network.html

4. La configuration PPPoE

Les deux protocoles PPP[15] et PPPoe se distinguent par le fait qu'une session PPPoe doit être établie avant que les échanges PPP ne débutent.

Le caractère asymétrique du dialogue PPPoE a une influence importante sur la configuration des deux (systèmes|routeurs) virtuels. Le routeur `Hub` doit exécuter un processus spécifique `pppoe-server` qui lui permet d'attendre les requêtes du routeur `Spoke`. La configuration de ce routeur `Spoke` est très voisine de celle de n'importe quel client PPP. Il suffit juste de spécifier l'utilisation du greffon `pppoe` dans la configuration.

4.1. Le routeur Hub

Par définition, un système Broadband Remote Access Server assure les fonctions suivantes :

- Agréger le trafic en provenance des équipements Digital subscriber line access multiplexer[16] (DSLAM). Sur la maquette, nous utilisons le commutateur `wan-sw` en lieu et place du DSLAM.
- Fournir une connectivité de niveau 2 à travers des réseaux Ethernet ou ATM sur lesquels les sessions PPP sont établies. Cette fonction est bien assurée par l'interface `eth1` du routeur `Hub`.
- Faire respecter les politiques de qualité de service (QoS). Cette fonction n'est pas traitée ici.
- Fournir une connectivité de niveau 3 et un routage des paquets IP vers le fournisseur d'accès (et|ou) l'Internet. Cette fonction est aussi assurée par le routeur `Hub`.

Comme cette liste correspond bien au rôle assigné au routeur `Hub` on peut passer à la configuration de la nouvelle interface WAN. On commence par s'intéresser à la liste des paquets Debian utiles. On détaille ensuite les fichiers de configuration correspondants.

```
$ aptitude search ~ippp
i A ppp          - Point-to-Point Protocol (PPP) - daemon
i   pppoe - PPP over Ethernet driver
```

Le paquet `pppoe` contient l'outil pppoe-server qui permet de gérer le dialogue de gestion de session du protocole. Les outils fournis avec ce paquet sont une implémentation complète du protocole PPPoE dans l'espace utilisateur.

```
$ dpkg -S `which pppoe-server`
pppoe: /usr/sbin/pppoe-server
```

Il est tout à fait possible d'utiliser cet outil directement à partir de la ligne de commande. Dans le cas présent, on souhaite retrouver la configuration d'une exécution à l'autre des instances de machines virtuelles. On essaie donc de paramétrer le serveur PPPoE dans les fichiers de configuration usuels de la distribution. Voici une liste des fichiers de configuration qui permettent de lancer le service.

- Le fichier `/etc/network/interfaces` :

[15] http://en.wikipedia.org/wiki/Point-to-Point_Protocol
[16] http://en.wikipedia.org/wiki/Digital_Subscriber_Line_Access_Multiplexer

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
 address 192.0.2.62/26
 gateway 192.0.2.1

iface eth0 inet6 static
 address 2a01:240:fe00:8175:b8ad:ff:feca:fe00/64
 gateway fe80::9083:f2ff:febf:2ef0

auto eth1
iface eth1 inet manual
 up ip link set dev $IFACE up
 up pppoe-server -I $IFACE -C BRAS -L 203.0.113.1 -p /etc/ppp/ip-range
 down killall pppoe-server
 down ip link set dev $IFACE up
```

La méthode de configuration de l'interface `eth1` est dite manuelle. Avec cette méthode aucun paramètre n'est requis. On donne deux instructions. On active l'interface au niveau liaison (couche 2) et on lance l'outil pppoe-server avec cette même interface comme paramètre.

- Le fichier `/etc/ppp/pppoe-server-options` :

```
debug
login
require-chap
nodefaultroute
ms-dns 192.0.2.1
+ipv6
```

La présence de ce fichier est obligatoire pour le fonctionnement de l'outil pppoe-server. Dans le cas présent, on se utilise le jeu d'options suivant extrait des pages de manuels du démon `pppd`.

- `debug` : affichage des étapes d'établissement de session dans les journaux système.
- `login` : référence de compte utilisateur à utiliser lors de l'authentification du routeur `spoke`. Cette option implique que le compte utilisateur existe sur le système et qu'il soit présent dans le fichier `/etc/ppp/chap-secrets`.
- `require-chap` : impose au routeur `spoke` une authentification via le protocole CHAP (Challenge Handshake Authentication Protocol).
- `nodefaultroute` : préserve la route par défaut (accès Internet) du routeur `hub`.
- `ms-dns` : publie l'adresse IPv4 du serveur DNS à utiliser.
- `+ipv6` : active les protocoles IPv6CP et IPv6.

- Le fichier `/etc/ppp/chap-secrets` :

```
# Secrets for authentication using CHAP
# client        server  secret                  IP addresses
"hub"           *       "5up3r53cr3t"           *
"spoke"         *       "5up3r53cr3t"           *
```

Comme l'option `login` impose l'existence d'un compte utilisateur pour référencer la ligne du fichier `chap-secrets`, le moyen le plus simple est de créer un compte local sur le routeur `hub`.

```
# adduser --disabled-login spoke
```

- Le fichier `/etc/ppp/ip-range` :

```
203.0.113.10-100
```

Ce fichier définit la liste des adresses IPv4 que le démon peut délivrer en fonction des demandes de connexion PPP.

Une fois la session PPPoE établie, le seul travail effectué par l'outil pppoe-server consiste à lancer le démon pppd qui se charge de gérer ses propres transactions aux deux niveaux LCP et NCP.

4.2. Le routeur d'accès Spoke

La configuration côté routeur d'accès ne présente qu'une seule spécificité : l'appel au greffon `pppoe`. Comme dans le cas précédent, on commence par le contrôle du paquet `ppp` et on poursuit avec les fichiers de configuration.

```
$ aptitude search ~ippp
i   ppp        - protocole point à point (PPP) - démon
```

Le paquet `ppp`, en plus du code nécessaire à la gestion du dialogue PPP, contient un greffon `pppoe`. Ce greffon permet l'utilisation du code PPPoE fourni avec le noyau Linux.

Comme avec l'autre routeur, il est possible d'exécuter le démon PPP avec l'ensemble de ses paramètres directement à la ligne de commande. On suit ici le même principe que pour le routeur `Hub` : une configuration sauvegardée dans les fichiers de configuration du système.

- Le fichier `/etc/network/interfaces` :

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
allow-hotplug eth0
iface eth0 inet dhcp

auto eth1
iface eth1 inet manual
        up ip link set dev $IFACE up
        down ip link set dev $IFACE down

auto dsl-provider
iface dsl-provider inet ppp
        pre-up ifup eth1
        provider dsl-provider
```

La définition de l'interface `ppp0` utilise la méthode `ppp`. Avec cette méthode, on peut utiliser la directive `provider` qui permet d'automatiser le lancement du démon PPP avec les paramètres spécifiés dans le fichier `dsl-provider`. Comme avec l'autre routeur, on fait précéder le lancement du démon par l'activation de l'interface au niveau liaison (couche 2).

Le fichier présenté est directement inspiré de la page du wiki Debian PPPoE[17]

- Le fichier `/etc/ppp/peers/dsl-provider` :

[17] https://wiki.debian.org/fr/PPPoE

```
# There should be a matching entry with the password in /etc/ppp/chap-secrets.
user "spoke"

# Load the PPPoE plugin.
plugin rp-pppoe.so

# Ethernet interface to which the modem is connected.
eth1

# Assumes that your IP address is allocated dynamically by the ISP.
noipdefault
# Try to get the name server addresses from the ISP.
usepeerdns
# Use this connection as the default route.
defaultroute

# Makes pppd "dial again" when the connection is lost.
persist

# Do not ask the remote to authenticate.
noauth

debug
+ipv6
```

C'est à ce niveau que l'on trouve l'appel au greffon PPPoE. Sa fonction est essentiellement l'utilisation de l'interface Ethernet `eth1` en lieu et place d'une interface «série» classique. Le fichier présenté est directement inspiré du modèle fourni avec la documentation du paquet `ppp` : `/usr/share/doc/ppp/examples/peers-pppoe`.

Les options ajoutées : `debug` et `+ipv6`, sont relatives au niveau des détails présents dans les journaux systèmes et l'activation des protocoles associés à IPv6.

5. La séquence de tests

Une fois que les configurations sont en place à chaque extrémité, il ne reste plus qu'à tester cette nouvelle liaison WAN virtuelle. Comme toute séquence de tests qui se respecte, on remonte les couches de la modélisation une à une.

Au niveau commutation de trame

On affiche la table des adresses MAC connues du commutateur `wan-sw` à partir de la console du système hôte.

```
$ sudo ovs-appctl fdb/show wan-sw
 port  VLAN  MAC                Age
    1     0  de:ad:be:ef:00:01    9
    2     0  de:ad:be:ef:00:03    9
```

On retrouve bien les adresses MAC «fantaisistes» attribuées aux deux routeurs lors du lancement des machines virtuelles. Voir la section intitulée « Instances de systèmes virtuels ».

Au niveau du routeur Hub

- Les journaux système révèlent l'établissement de la session PPPoE suivie du lancement du démon PPP et des transactions LCP et NCP pour les deux protocoles de couche réseau IPv4 et IPv6.

```
hub pppoe-server[106]: Session 1 created for client de:ad:be:ef:00:03 (203.0.113.10) on eth1 using
hub pppd[106]: pppd 2.4.6 started by etu, uid 0
hub pppd[106]: using channel 4
hub pppd[106]: Using interface ppp0
hub pppd[106]: Connect: ppp0 <--> /dev/pts/1
hub pppd[106]: sent [LCP ConfReq id=0x1 <auth chap MD5> <magic 0xcd537d21>]
hub pppd[106]: rcvd [LCP ConfAck id=0x1 <auth chap MD5> <magic 0xcd537d21>]
hub pppd[106]: rcvd [LCP ConfReq id=0x1 <mru 1492> <magic 0xf9d3d6a1>]
hub pppd[106]: sent [LCP ConfAck id=0x1 <mru 1492> <magic 0xf9d3d6a1>]
hub pppd[106]: sent [LCP EchoReq id=0x0 magic=0xcd537d21]
hub pppd[106]: sent [CHAP Challenge id=0xad <d3ea7672130fd1d22b91cfbe6e1355da>, name = "hub"]
```

```
hub pppd[106]: rcvd [LCP EchoReq id=0x0 magic=0xf9d3d6a1]
hub pppd[106]: sent [LCP EchoRep id=0x0 magic=0xcd537d21]
hub pppd[106]: rcvd [LCP EchoRep id=0x0 magic=0xf9d3d6a1]
hub pppd[106]: rcvd [CHAP Response id=0xad <d66b72abdc872fdba648a68009be6514>, name = "spoke"]
hub pppd[106]: sent [CHAP Success id=0xad "Access granted"]
hub pppd[106]: Initializing PAM (2) for user spoke
hub pppd[106]: ---> PAM INIT Result = 0
hub pppd[106]: Attempting PAM account checks
hub pppd[106]: PAM Account OK for spoke
hub pppd[106]: PAM Session opened for user spoke
hub pppd[106]: user spoke logged in on tty  intf ppp0
hub pppd[106]: local  LL address fe80::5c3a:77a1:aad1:a619
hub pppd[106]: remote LL address fe80::b474:5671:fc2c:ddd8
hub pppd[106]: local  IP address 203.0.113.1
hub pppd[106]: remote IP address 203.0.113.10
```

- Une entrée de la table de routage a bien été définie en fonction des éléments donnés ci-dessus.

```
# ip route ls
default via 192.0.2.1 dev eth0
192.0.2.0/26 dev eth0  proto kernel  scope link  src 192.0.2.10
203.0.113.10 dev ppp0  proto kernel  scope link  src 203.0.113.1
```

  Côté IPv6, l'activation de l'interface `ppp0` a bien été prise en compte.

```
# ip -6 route ls dev ppp0
fe80::/10  metric 1  pref medium
fe80::/10  proto kernel  metric 256  pref medium
```

- Pour identifier la liste des processus associés au démon `pppoe-server`, on peut utiliser une commande du type suivant.

```
# pstree -p $(pidof pppoe-server)
pppoe-server(10628)───pppd(10692)───sh(10693)───pppoe(10695)
```

  Un autre affichage donne la liste des arguments associés aux démons. Le lancement du démon `pppd` via `pppoe-server` impose une liste de paramètres autres que ceux définis dans le fichier `/etc/ppp/pppoe-server-options`.

```
# ps -fwwp $(pgrep -d, ppp)
UID        PID  PPID  C STIME TTY          TIME CMD
root     10628     1  0 oct.10 ?       00:00:00 pppoe-server -I eth1 -C BRAS -L 203.0.113.1 -p /et
root     10692 10628  0 oct.10 ?       00:00:00 pppd pty /usr/sbin/pppoe -n -I eth1 -e 1:de:ad:be:
 file /etc/ppp/pppoe-server-options \
 203.0.113.1:203.0.113.10 \
 nodetach \
 noaccomp \
 nobsdcomp \
 nodeflate \
 nopcomp \
 novj \
 novjccomp \
 default-asyncmap
nobody   10695 10693  0 oct.10 ?       00:00:00 /usr/sbin/pppoe -n -I eth1 -e 1:de:ad:be:ef:00:03
```

Au niveau du routeur Spoke

- Comme avec l'autre extrémité du lien point à point, les journaux système montrent les phases de l'établissement de la session PPPoE avant le dialogue PPP. En revanche, c'est le code fourni par le noyau Linux (kernel) qui est utilisé cette fois-ci.

```
spoke pppd[2211]: Plugin rp-pppoe.so loaded.
spoke pppd[2218]: pppd 2.4.6 started by etu, uid 0
spoke pppd[2218]: Failed to disconnect PPPoE socket: 114 Operation already in progress
spoke pppd[2218]: Send PPPOE Discovery V1T1 PADI session 0x0 length 12
spoke pppd[2218]:  dst ff:ff:ff:ff:ff:ff  src de:ad:be:ef:00:03
spoke pppd[2218]:  [service-name] [host-uniq  aa 08 00 00]
spoke pppd[2218]: Recv PPPOE Discovery V1T1 PADO session 0x0 length 44
spoke pppd[2218]:  dst de:ad:be:ef:00:03  src de:ad:be:ef:00:01
spoke pppd[2218]:  [AC-name BRAS] [service-name] [AC-cookie  4a f2 52 4d 60 60 27 c6 00 63 97 78 3
spoke pppd[2218]: Send PPPOE Discovery V1T1 PADR session 0x0 length 36
spoke pppd[2218]:  dst de:ad:be:ef:00:01  src de:ad:be:ef:00:03
```

```
spoke pppd[2218]:  [service-name] [host-uniq  aa 08 00 00] [AC-cookie  4a f2 52 4d 60 60 27 c6 00
spoke pppd[2218]: Recv PPPOE Discovery V1T1 PADS session 0x4 length 12
spoke pppd[2218]:  dst de:ad:be:ef:00:03  src de:ad:be:ef:00:01
spoke pppd[2218]:  [service-name] [host-uniq  aa 08 00 00]
spoke pppd[2218]: PADS: Service-Name: ''
spoke pppd[2218]: PPP session is 4
spoke pppd[2218]: Connected to de:ad:be:ef:00:01 via interface eth1
spoke pppd[2218]: using channel 4
spoke pppd[2218]: Using interface ppp0
spoke pppd[2218]: Connect: ppp0 <--> eth1
spoke pppd[2218]: sent [LCP ConfReq id=0x4 <mru 1492> <magic 0xf4de2406>]
spoke pppd[5046]: rcvd [LCP ConfAck id=0x1 <mru 1492> <magic 0xf9d3d6a1>]
spoke pppd[5046]: sent [LCP EchoReq id=0x0 magic=0xf9d3d6a1]
spoke pppd[5046]: rcvd [LCP EchoReq id=0x0 magic=0xcd537d21]
spoke pppd[5046]: sent [LCP EchoRep id=0x0 magic=0xf9d3d6a1]
spoke pppd[5046]: rcvd [CHAP Challenge id=0xad <d3ea7672130fd1d22b91cfbe6e1355da>, name = "hub"]
spoke pppd[5046]: sent [CHAP Response id=0xad <d66b72abdc872fdba648a68009be6514>, name = "spoke"]
spoke pppd[5046]: rcvd [LCP EchoRep id=0x0 magic=0xcd537d21]
spoke pppd[5046]: rcvd [CHAP Success id=0xad "Access granted"]
spoke pppd[5046]: CHAP authentication succeeded: Access granted
spoke pppd[5046]: CHAP authentication succeeded
spoke pppd[5046]: peer from calling number DE:AD:BE:EF:00:01 authorized
spoke pppd[5046]: sent [IPCP ConfReq id=0x1 <addr 0.0.0.0> <ms-dns1 0.0.0.0> <ms-dns2 0.0.0.0>]
spoke pppd[5046]: sent [IPV6CP ConfReq id=0x1 <addr fe80::b474:5671:fc2c:ddd8>]
spoke pppd[5046]: rcvd [IPCP ConfReq id=0x1 <addr 203.0.113.1>]
spoke pppd[5046]: sent [IPCP ConfAck id=0x1 <addr 203.0.113.1>]
spoke pppd[5046]: rcvd [IPV6CP ConfReq id=0x1 <addr fe80::5c3a:77a1:aad1:a619>]
spoke pppd[5046]: sent [IPV6CP ConfAck id=0x1 <addr fe80::5c3a:77a1:aad1:a619>]
spoke pppd[5046]: rcvd [IPCP ConfRej id=0x1 <ms-dns1 0.0.0.0> <ms-dns2 0.0.0.0>]
spoke pppd[5046]: sent [IPCP ConfReq id=0x2 <addr 0.0.0.0>]
spoke pppd[5046]: rcvd [IPV6CP ConfAck id=0x1 <addr fe80::b474:5671:fc2c:ddd8>]
spoke pppd[5046]: local  LL address fe80::b474:5671:fc2c:ddd8
spoke pppd[5046]: remote LL address fe80::5c3a:77a1:aad1:a619
spoke pppd[5046]: Script /etc/ppp/ipv6-up started (pid 5112)
spoke pppd[5046]: rcvd [IPCP ConfNak id=0x2 <addr 203.0.113.10>]
spoke pppd[5046]: sent [IPCP ConfReq id=0x3 <addr 203.0.113.10>]
spoke pppd[5046]: rcvd [IPCP ConfAck id=0x3 <addr 203.0.113.10>]
spoke pppd[5046]: not replacing default route to eth0 [192.0.2.1]
spoke pppd[5046]: local  IP address 203.0.113.10
spoke pppd[5046]: remote IP address 203.0.113.1
spoke pppd[5046]: Script /etc/ppp/ip-up started (pid 5114)
spoke pppd[5046]: Script /etc/ppp/ipv6-up finished (pid 5112), status = 0x0
spoke pppd[5046]: Script /etc/ppp/ip-up finished (pid 5114), status = 0x0
```

- On retrouve une entrée symétrique dans la table de routage. Il faut cependant noter que la route par défaut IPv4 n'a pas été remplacée par l'activation du lien WAN. C'est un problème sur lequel il faut revenir par la suite.

```
# ip route ls
default via 192.0.2.1 dev eth0
192.0.2.0/26 dev eth0  proto kernel  scope link  src 192.0.2.11
203.0.113.1 dev ppp0  proto kernel  scope link  src 203.0.113.10
```

  Côté IPv6, l'activation de l'interface `ppp0` a bien été prise en compte.

```
# ip -6 route ls dev ppp0
fe80::/10  metric 1  pref medium
fe80::/10  proto kernel  metric 256  pref medium
```

- Comme le noyau fournit l'essentiel du code, un seul processus est présent en espace utilisateur.

```
# ps -fwwp $(pgrep -d, ppp)
UID        PID  PPID  C STIME TTY          TIME CMD
root      5046     1  0 oct.10 ?       00:00:00 /usr/sbin/pppd call dsl-provider
```

Au niveau réseau
: Les tests traditionnels ICMP et ICMP6 confirment que les deux routeurs peuvent échanger des paquets. Depuis le routeur `spoke`, on lance les commandes suivantes.

```
$ ping -c2 203.0.113.1
PING 203.0.113.1 (203.0.113.1) 56(84) bytes of data.
64 bytes from 203.0.113.1: icmp_seq=1 ttl=64 time=1.12 ms
64 bytes from 203.0.113.1: icmp_seq=2 ttl=64 time=0.845 ms

--- 203.0.113.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.845/0.987/1.129/0.142 ms
```

```
$ ping6 -c2 fe80::5c3a:77a1:aad1:a619%ppp0
PING fe80::5c3a:77a1:aad1:a619%ppp0(fe80::5c3a:77a1:aad1:a619) 56 data bytes
64 bytes from fe80::5c3a:77a1:aad1:a619: icmp_seq=1 ttl=64 time=1.28 ms
64 bytes from fe80::5c3a:77a1:aad1:a619: icmp_seq=2 ttl=64 time=0.808 ms

--- fe80::5c3a:77a1:aad1:a619%ppp0 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.808/1.044/1.281/0.238 ms
```

6. Routage via le lien WAN

Après avoir testé les échanges de paquets sur le lien WAN dans la section précédente, on se préoccupe maintenant du routage depuis et vers le routeur `spoke` via ce même lien.

Désactivation de la configuration de l'interface LAN du routeur Spoke

Dans les sections précédentes, l'interface `eth0` du routeur `spoke` a servi à communiquer avec l'Internet. On doit donc désactiver la configuration automatique de cette interface aussi bien en IPv4 qu'en IPv6.

Voici un extrait du fichier `/etc/network/interfaces` du routeur `spoke` pour l'interface `eth0` sans configuration automatique.

```
allow-hotplug eth0
iface eth0 inet manual
 up echo 0 > /proc/sys/net/ipv6/conf/$IFACE/forwarding
 up echo 0 > /proc/sys/net/ipv6/conf/$IFACE/accept_ra
 up ip link set dev $IFACE up
 down ip link set dev $IFACE down
```

Cette solution permet de conserver un accès depuis le système hôte via l'adresse IPv6 de lien local.

Pour forcer l'application de la route par défaut via le protocole PPP, on utilise les scripts système de manipulation des interfaces.

```
# ifdown dsl-provider
# ifup dsl-provider
```

La table de routage IPv4 devient :

```
# ip route ls
default dev ppp0  scope link
203.0.113.1 dev ppp0  proto kernel  scope link  src 203.0.113.11
```

La table de routage IPv6 reste inchangée tant que le routeur `hub` ne dispose pas d'outil d'annonce SLAAC.

Activation du routage IPv4 & IPv6 sur le routeur Hub

Si on respecte la règle de configuration adoptée dans les sections précédentes, on utilise les fichiers système pour configurer le routage. Ici, c'est le fichier `/etc/sysctl.conf` qui nous intéresse. Voici la liste des options de configuration utilisées. Dans la copie d'écran ci-dessous, on a supprimé les commentaires et les lignes vides.

```
# egrep -v '(^#|^$)' /etc/sysctl.conf
net.ipv4.conf.default.rp_filter=1
net.ipv4.conf.all.rp_filter=1
net.ipv4.ip_forward=1
net.ipv6.conf.all.forwarding=1
net.ipv4.conf.all.accept_redirects = 0
net.ipv6.conf.all.accept_redirects = 0
net.ipv4.conf.all.log_martians = 1
```

Ces options sont appliquées à l'aide de la commande `# sysctl -p`.

Traduction d'adresses IPv4 sur le routeur Hub

Le démon `pppd` du routeur `hub` délivre des adresses IPv4 appartenant au préfixe `203.0.113.0/24`. Ce préfixe de la famille TESTNET est non routable sur l'Internet. On a donc recours à la traduction d'adresses sources pour que le trafic du routeur `spoke` soit correctement routé.

On utilise le paquet ipatbles-persistent pour préserver les règles de traduction d'adresses d'un redémarrage système à l'autre. Le fichier de sauvegarde est : `/etc/iptables/rules.v4`.

```
*nat
:PREROUTING ACCEPT [0:0]
:INPUT ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
-A POSTROUTING -o eth0 -p tcp -m tcp --syn -m tcpmss --mss 1400:1536 -j TCPMSS --clamp-mss-to-pmtu
-A POSTROUTING -o eth0 -j MASQUERADE
COMMIT
```

Il suffit de consulter les compteurs associés aux règles de filtrage pour valider leur fonctionnement.

```
# iptables -t nat -vL  POSTROUTING
Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source     destination
   30  1800 TCPMSS     tcp  --  any    eth0    anywhere   anywhere  tcp flags:FIN,SYN,RST,ACK/SYN tcpms
  256 26578 MASQUERADE all  --  any    eth0    anywhere   anywhere
```

Note

L'utilisation du module `TCPMSS` joue un rôle important dans la gestion de l'encapsulation entre les couches transport, réseau et liaison. En effet, le dialogue PPPoE consomme 8 octets de chaque trame. La quantité maximale d'octets qu'une trame peut encapsuler passe de 1500 à 1492. Voir la page Maximum transmission unit[18].

Le protocole TCP de la couche transport négocie, lors de l'établissement d'une connexion, la quantité maximale d'octets que peut contenir un segment. Comme la couche transport ne sait rien de l'encapsulation en couche liaison, la règle de filtrage qui fait appel au module `TCPMSS` est là pour forcer l'adaptation au contexte PPPoE du paramètre Maximum segment size[19] lors de l'établissement des connexions TCP.

Adresses IPv6 ULA

Avec IPv6, les adresses dites Unique local address[20] sont l'équivalent des adresses privées IPv4. Ce document constitue une bonne occasion d'illustrer leur utilisation.

Par définition, le préfixe réseau à utiliser est le : `fd00::/8`. À ce préfixe il faut adjoindre 40 bits aléatoires pour obtenir un préfixe `/48` qui constitue la base du plan d'adressage.

On commence donc par générer une chaine de 40 bits aléatoires notée en hexadécimal.

```
$ openssl rand -hex 5
271684e79b
```

Le préfixe réseau obtenu est le : `fd27:1684:e79b::/48`. Pour adresser les différents routeurs `spoke` potentiels, l'espace d'adressage est ainsi de 2^16 liens WAN possibles. Les 16 bits réseau sont obtenus en faisant la différence entre 64 (le nombre de bits de la partie hôte) et 48 (le nombre de bits de la partie réseau). Puisque la maquette utilisée ne comprend qu'un seul lien WAN, on utilise le premier préfixe réseau disponible pour la suite des manipulations : `fd27:1684:e79b::/64`.

Une fois le plan d'adressage défini, on passe à la configuration du routeur `hub`.

- Affectation d'une adresse IPv6 à l'interface `ppp0`.

[18] https://en.wikipedia.org/wiki/Maximum_transmission_unit
[19] https://en.wikipedia.org/wiki/Maximum_segment_size
[20] https://en.wikipedia.org/wiki/Unique_local_address

```
# ip -6 addr add fd27:1684:e79b::1/64 dev ppp0
```

- Configuration du démon radvd pour annoncer le préfixe réseau ainsi que le resolver DNS en direction du routeur `spoke`.

```
interface ppp0
{
   AdvSendAdvert on;
   prefix fd27:1684:e79b::/64
   {
       AdvOnLink on;
       AdvAutonomous on;
       AdvRouterAddr on;
   };

   RDNSS 2001:db8:fe00:8175::1
   {
   };
};
```

Ces deux étapes de configuration doivent normalement être réalisées lors de l'établissement du lien WAN via le démon pppd. On peut donc placer deux scripts dans le répertoire `/etc/ppp/ipv6-up.d/`. Ainsi, dès que l'interface `pppX` est activée, on effectue automatiquement les opérations de configuration spécifiques à IPv6.

- Ajout automatique de l'adresse IPv6 de la famille Unique local address[21] à l'interface PPP. Voici le script `/etc/ppp/ipv6-up.d/ipv6-ula-address`

```
#!/bin/sh

# ULA network prefix
prefix=fd27:1684:e79b

# extract $IFACE number as network nibble
netnum=$(echo $IFACE | grep -o [0-9]*)

# compute ULA interface address
ula=$prefix:$netnum::1/64

ip -6 addr add $ula dev $IFACE
```

- Le démon radvd est conçu pour réagir dynamiquement aux activations/désactivations d'interfaces et au modifications de son fichier de configuration : `/etc/radvd.conf`. Voici le script qui permet de compléter dynamiquement la configuration en fonction de l'activation d'une interface PPP : `/etc/ppp/ipv6-up.d/radvd-conf-update`.

```
#!/bin/sh

# radvd main configuration file
radvd_conf="/etc/radvd.conf"

if [ -z "$(grep $IFACE $radvd_conf)" ]
then

# ULA network prefix
prefix=fd27:1684:e79b

# DNS resolver IPv6 address
ipv6_resolver=2001:db8:fe00:8175::1

# extract $IFACE number as network nibble
netnum=$(echo $IFACE | grep -o [0-9]*)

cat << EOF >> $radvd_conf

interface $IFACE
{
 AdvSendAdvert on;
```

[21] https://en.wikipedia.org/wiki/Unique_local_address

```
 prefix $prefix:$netnum::/64
 {
  AdvOnLink on;
  AdvAutonomous on;
  AdvRouterAddr on;
 };

 RDNSS $ipv6_resolver
 {
 };
};
EOF

systemctl restart radvd

fi
```

Traduction d'adresses IPv6 sur le routeur Hub

On reprend ici la même démarche que pour le protocole IPv4 avec l'utilisation du paquet ipatbles-persistent et la sauvegarde des règles de filtrage dans le fichier `/etc/iptables/rules.v6`.

```
*nat
:PREROUTING ACCEPT [15:2078]
:INPUT ACCEPT [9:1586]
:OUTPUT ACCEPT [13:1536]
:POSTROUTING ACCEPT [9:1120]
-A POSTROUTING -o eth0 -p tcp -m tcp --syn -m tcpmss --mss 1400:1536 -j TCPMSS --clamp-mss-to-pmtu
-A POSTROUTING -s fd00::/8 -o eth0 -j SNAT --to 2001:db8:fe00:8175:b8ad:ff:feca:fe00
COMMIT
```

Comme avec le routage des paquets IPv4, les compteurs servent à qualifier l'utilisation des règles de filtrage.

- Sur le routeur `spoke`, on lance le chargement d'une page web.

```
$ wget -q -6 -O /dev/null http://inetdoc.net
```

- Sur le routeur `hub`, on relève les compteurs des règles de la table `nat`.

```
# ip6tables -t nat -vnL POSTROUTING
Chain POSTROUTING (policy ACCEPT 13 packets, 2149 bytes)
 pkts bytes target     prot opt in     out     source    destination
    8   640 TCPMSS     tcp      *      eth0    ::/0      ::/0     tcp flags:0x17/0x02 tcpmss match 14
   14  1312 SNAT       all      *      eth0    fd00::/8  ::/0     to:2001:db8:fe00:8175:b8ad:ff:feca:
```

> **Note**
>
> Avec le protocole IPv6, l'emploi du module `TCPMSS` est aussi utile. Voir la Note sur les règles de filtrage IPv4.

7. Pour conclure

Voilà ! On dispose maintenant d'un modèle de liaison WAN point à point. Il ne reste plus qu'à l'exploiter avec les supports de travaux pratiques tels que la mise en œuvre de la Topologie Hub & Spoke avec le protocole PPP[22], le Filtrage réseau avec netfilter/iptables[23] ou l'Étude de cas de synthèse sur l'interconnexion LAN/WAN[24].

[22] http://www.inetdoc.net/travaux_pratiques/interco.ppp.q/
[23] http://www.inetdoc.net/travaux_pratiques/interco.netfilter.q/
[24] http://www.inetdoc.net/travaux_pratiques/interco.cs/