Aire OSPF et passerelles multiples

Philippe Latu
philippe.latu(at)inetdoc.net

https://www.inetdoc.net

Résumé

Cet article est une illustration des fonctions de routage avancées sur les systèmes GNU/Linux. À partir d'une topologie de travaux pratiques comprenant trois routeurs en «triangle» dans une aire OSPF et un quatrième routeur représentant un hypothétique opérateur, il couvre les notions de répartition de trafic (*load balancing*) et de tolérance aux pannes réseau (*failover*).

Table des matières

1. Copyright et Licence

1.1. Méta-information

Cet article est écrit avec *DocBook* XML sur un système *Debian GNU/Linux*. Il est disponible en version imprimable au format PDF : `ospf-triangle-multiple-default.pdf`.

1.2. Conventions typographiques

Tous les exemples d'exécution des commandes sont précédés d'une invite utilisateur ou *prompt* spécifique au niveau des droits utilisateurs nécessaires sur le système.

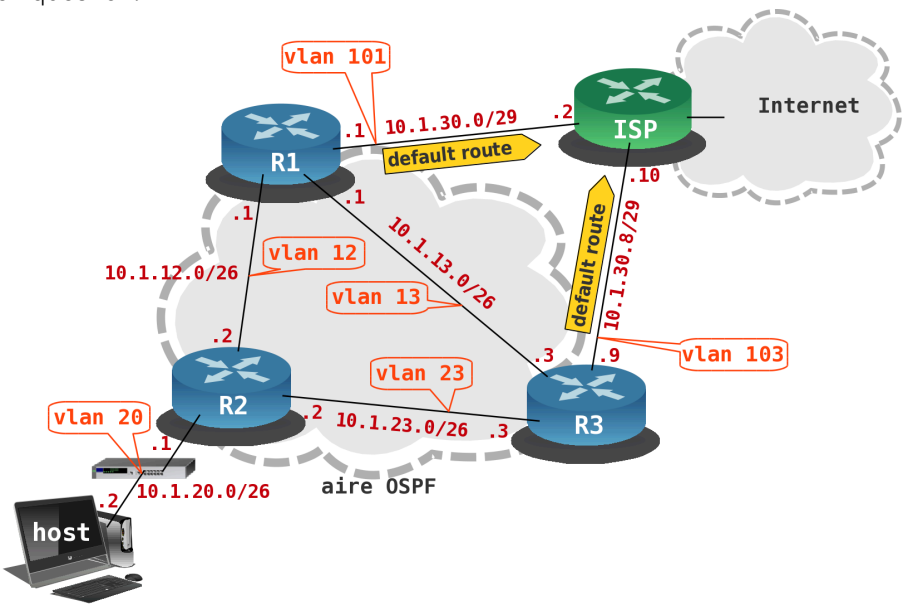
- Toute commande précédée de l'invite `$` ne nécessite aucun privilège particulier et peut être utilisée au niveau utilisateur simple.
- Toute commande précédée de l'invite `#` nécessite les privilèges du super-utilisateur.

2. Présentation de la topologie réseau étudiée

En fait, cet article fait suite à la rédaction du support *Synthèse sur l'interconnexion LAN/WAN* dans lequel on constitue une agrégation de topologies *Hub and Spoke* avec les douze postes d'une salle de travaux pratiques. Chacun des *Hubs* est interconnecté avec deux autres routeurs dans une aire OSPF. Cet ensemble de routeurs dispose de deux liens montant vers l'Internet.

Vu de l'aire OSPF, la diffusion d'une route par défaut via le protocole de routage dynamique se fait à l'aide de l'instruction `default-information originate`. Cette instruction est présentée à la page https://www.inetdoc.net/travaux_pratiques/interco.ospf-frr.qa/interco.ospf-frr.default-information.html du support *Introduction au routage dynamique OSPF avec FRRouting*.

La situation devient un peu plus ***intéressante*** lorsque cette aire OSPF dispose de plusieurs portes de sortie vers les autres réseaux. En vis-à-vis de ces routeurs de bordure le routeur du niveau supérieur doit être capable d'acheminer correctement les flux sortant et entrant de l'aire. Voici une représentation de la topologie logique en question.



Cette topologie reprend très exactement la disposition en triangle présentée dans le support *Introduction au routage dynamique OSPF avec FRRouting*. L'utilisation du routage interVLAN sert à mettre en évidence la dissociation entre une topologie physique en étoile issue des définitions de la technologie Ethernet et une topologie logique en triangle dans laquelle chaque routeur doit apprendre au moins un réseau via ses voisins.

À la topologie initiale de travaux pratiques, on a adjoint un réseau local supplémentaire dans lequel on trouve un hôte baptisé (avec beaucoup d'originalité) ***host*** ainsi que le fameux routeur de niveau supérieur baptisé `ISP` pour *Internet Service Provider*. Dans tous les tests l'hôte ***host*** est la source et la destination du trafic routé. La configuration du routeur `ISP` fait l'objet de toute l'attention de cet article.

Avant d'étudier le routage dans l'aire OSPF et les conditions à satisfaire pour la répartition du trafic entre les deux liens de raccordement de l'aire au routeur `ISP` ainsi que la tolérance aux pannes sur ces mêmes liens, il est important de préciser que toutes les connexions sont homogènes. On considère que toutes les interfaces des routeurs et du poste de travail ***host*** sont connectées sur des ports de commutateur Ethernet gigabit.

3. Routage au sein de l'aire OSPF

Sur `R2`, la table de routage vue du système est donnée ci-dessous. Cette visualisation au «niveau du noyau» est la synthèse de toutes les routes apprises par les différents canaux d'information : routes statiques et protocoles de routage dynamique. Dans le cas du routeur `R2`, les deux canaux d'information sont les connexions directes et le protocole OSPF.

- Toutes les entrées avec l'indication `proto kernel` correspondent aux connexions directes héritées de la configuration des interfaces réseau du routeur.
- Toutes les entrées avec l'indication `proto zebra` correspondent aux routes apprises via un démon actif du paquet `frr`. Ici, le démon `ospfd` est la seule source d'information.

```
$ ip route ls
default  proto zebra  metric 10
        nexthop via 10.1.23.3  dev eth0.23 weight 1
        nexthop via 10.1.12.1  dev eth0.12 weight 1
10.1.12.0/26 dev eth0.12  proto kernel  scope link  src 10.1.12.2
10.1.13.0/26  proto zebra  metric 2
        nexthop via 10.1.12.1  dev eth0.12 weight 1
        nexthop via 10.1.23.3  dev eth0.23 weight 1
10.1.20.0/26 dev eth0  proto kernel  scope link  src 10.1.20.1
10.1.23.0/26 dev eth0.23  proto kernel  scope link  src 10.1.23.2
```

La même table de routage vue du démon `zebra` est donnée ci-dessous. Relativement à la vue synthétique précédente, on obtient davantage d'informations sur les mécanismes d'apprentissage et de décision.

- Toutes les lignes débutant par le caractère C correspondent au connexions directes.
- Toutes les lignes débutant par le caractère O correspondent aux routes apprises à l'aide du démon `ospfd`.
- Toutes les entrées marquées par le caractère * correspondent aux routes retenues pour l'acheminement des paquets. Toutes les routes listées sont présentes dans la ***Routing Information Base*** (RIB) au niveau ***Control Plane*** et celles qui sont marquées sont mémorisées dans la ***Forwarding Information Base*** (FIB) au niveau ***Data Plane***.

  Avant stockage dans la base FIB, les routes retenues sont converties en une table de hachage pour assurer la fonction de commutation de paquets. Cette fonction est plus ou moins optimisée suivant les composants électroniques présents sur le routeur. Dans le cas de cette maquette de travaux pratiques, les machines virtuelles utilisent la mémoire vive pour stocker les tables de hachage.

```
R2-Zebra# sh ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
       I - ISIS, B - BGP, > - selected route, * - FIB route

O>* 0.0.0.0/0 [110/10] via 10.1.23.3, eth0.23, 1d10h31m
  *                    via 10.1.12.1, eth0.12, 1d10h31m
O   10.1.12.0/26 [110/1] is directly connected, eth0.12, 5d00h09m
C>* 10.1.12.0/26 is directly connected, eth0.12
O>* 10.1.13.0/26 [110/2] via 10.1.12.1, eth0.12, 5d00h09m
  *                      via 10.1.23.3, eth0.23, 5d00h09m
O   10.1.20.0/26 [110/1] is directly connected, eth0, 5d00h09m
C>* 10.1.20.0/26 is directly connected, eth0
O   10.1.23.0/26 [110/1] is directly connected, eth0.23, 5d00h09m
C>* 10.1.23.0/26 is directly connected, eth0.23
C>* 127.0.0.0/8 is directly connected, lo
```

La base des routes du démon `ospfd` est donnée ci-dessous. On ne retrouve ici que les routes présentes dans la configuration du démon local et celles apprises via les échanges avec les démons OSPF voisins.

- Les réseaux correspondant aux connexions directes du routeur apparaissent avec la mention `directly attached`.
- Les autres réseaux apparaissent avec l'adresse IP du ou des routeurs voisins. Ainsi, depuis le routeur `R2` le réseau `10.1.13.0/26` est appris via les deux routeurs `R1` (`10.1.12.1`) et `R3` (`10.1.23.3`).

```
R2-ospfd# sh ip ospf route
============ OSPF network routing table ============
N    10.1.12.0/26          [1] area: 0.0.0.0
                           directly attached to eth0.12
N    10.1.13.0/26          [2] area: 0.0.0.0
                           via 10.1.12.1, eth0.12
                           via 10.1.23.3, eth0.23
N    10.1.20.0/26          [1] area: 0.0.0.0
                           directly attached to eth0
N    10.1.23.0/26          [1] area: 0.0.0.0
                           directly attached to eth0.23

============ OSPF router routing table =============
R    0.0.0.1               [1] area: 0.0.0.0, ASBR
                           via 10.1.12.1, eth0.12
R    0.0.0.3               [1] area: 0.0.0.0, ASBR
                           via 10.1.23.3, eth0.23

============ OSPF external routing table ===========
N E2 0.0.0.0/0             [1/10] tag: 0
                           via 10.1.23.3, eth0.23
                           via 10.1.12.1, eth0.12
```

Comme indiqué dans le support de travaux pratiques sur *Introduction au routage dynamique OSPF avec FRRouting*, l'utilisation de l'instruction `default-information originate` sur les routeurs `R1` et `R3` change les rôles de ces deux routeurs. Ils deviennent routeurs de bordure ou ***Autonomous System Boundary Router*** (ASBR). Ils sont situés à la frontière entre l'aire OSPF et un autre système autonome qui peut utiliser un autre type de routage.

Dans notre cas, les routeurs `R1` et `R3` possèdent une route statique définie au niveau système vers `ISP`. L'instruction donnée ci-dessus assure la redistribution des routes statiques vers `R2`. Ces routes apparaissent comme des entrées de type `E2` dans la table de `R2`.

L'indicateur `E2` correspond au type par défaut des routes apprises par le biais de la redistribution. La métrique est un point important à considérer avec les routes de ce type. Elles ne présentent que le coût du chemin allant du routeur ASBR vers le réseau de destination (`10` dans notre exemple) ; ce qui ne correspond pas au coût réel du chemin entre `R2` et `ISP`. Sachant que tous les liens de la topologie étudiée sont identiques et ont le même coût, on se contente d'utiliser ces entrées de type `E2` en l'état.

4. Répartition de trafic - *load balancing*

Dans la configuration de l'aire OSPF présentée dans la section précédente, les routeurs et les hôtes réseau en général disposent de deux liens vers le routeur de niveau supérieur `ISP`. Si les passerelles par défaut des deux routeurs de bordure `R1` et `R3` sont publiées via le protocole de routage dynamique à destination de tous les routeurs de l'aire, le routeur `ISP`, lui ne dispose pas de cette information.

4.1. Route de synthèse multichemins sur le routeur ISP - *multipath static summary route*

Sans configuration particulière, le routage se fait principalement sur la base des adresses IP destination et le trafic sortant peut transiter aussi bien par `R1` que par `R3`. Le trafic retour ou entrant transite exclusivement par `ISP` et il est nécessaire de renseigner les réseaux appartenant à l'aire OSPF dans sa table de routage. La technique usuelle dans le contexte de la topologie étudiée consiste à implanter une route de synthèse qui englobe la totalité des réseaux situés aux niveaux inférieurs.

Pour déterminer l'adresse réseau qui englobe les réseau de l'aire, on dispose de l'outil aggregate fourni avec le paquet du même nom. Voici deux exemples d'exécution sur le routeur `R2` après affichage de la table de routage.

```
# ip route ls
default  proto zebra  metric 10
        nexthop via 10.1.23.3  dev eth0.23 weight 1
        nexthop via 10.1.12.1  dev eth0.12 weight 1
10.1.12.0/26 dev eth0.12  proto kernel  scope link  src 10.1.12.2
10.1.13.0/26  proto zebra  metric 2
        nexthop via 10.1.12.1  dev eth0.12 weight 1
        nexthop via 10.1.23.3  dev eth0.23 weight 1
10.1.20.0/26 dev eth0  proto kernel  scope link  src 10.1.20.1
10.1.23.0/26 dev eth0.23  proto kernel  scope link  src 10.1.23.2
```

Dans le premier exemple ci-dessous, l'adresse `10.1.0.0/20` n'englobe que les réseaux `10.1.12.0/26` et `10.1.13.0/26` marqués avec le signe -. Le masque réseau `/20` est donc insuffisant pour couvrir la liste complète des réseaux de niveau inférieur.

```
# (echo '10.1.0.0/20' ;\
ip route ls | grep -Eo '^([0-9]{1,3}\.){3}[0-9]{1,3}/[0-9]{2}') |\
aggregate -v
aggregate: maximum prefix length permitted will be 32
[    1]   10.1.0.0/20
[    2] - 10.1.12.0/26
[    3] - 10.1.13.0/26
[    4]   10.1.20.0/26
[    5]   10.1.23.0/26
```

Dans le second exemple ci-dessous, l'adresse `10.1.0.0/19` englobe bien tous les réseaux de l'aire OSPF. Le masque réseau `/19` permet de couvrir tous les réseaux de l'aire OSPF en une seule et unique entrée.

```
# (echo '10.1.0.0/19' ;\
ip route ls | grep -Eo '^([0-9]{1,3}\.){3}[0-9]{1,3}/[0-9]{2}') |\
aggregate -v
aggregate: maximum prefix length permitted will be 32
[    1]   10.1.0.0/19
[    2] - 10.1.12.0/26
[    3] - 10.1.13.0/26
[    4] - 10.1.20.0/26
[    5] - 10.1.23.0/26
```

On peut donc implanter sur le routeur `ISP` une route statique de synthèse ou ***static summary route*** désignant les deux passerelles de l'aire OSPF.

```
# ip route add 10.1.0.0/19 nexthop via 10.1.30.1 nexthop via 10.1.30.9
```

Une fois cette commande exécutée, la table de routage du routeur `ISP` est la suivante.

```
# ip route ls
default via 192.0.2.1 dev eth0
10.1.0.0/19
        nexthop via 10.1.30.1  dev eth1.101 weight 1
        nexthop via 10.1.30.9  dev eth1.103 weight 1
10.1.30.0/29 dev eth1.101  proto kernel  scope link  src 10.1.30.2
10.1.30.8/29 dev eth1.103  proto kernel  scope link  src 10.1.30.10
192.0.2.0/27 dev eth0  proto kernel  scope link  src 192.0.2.3
```

- Le mot clé `nexthop` sert à définir plusieurs passerelles pour un même réseau de destination.
- Les interfaces via lesquelles les paquets doivent être transmis sont ajoutées automatiquement en fonction de l'adresse IP de passerelle. Par exemple, la passerelle `10.1.30.1` est joignable via l'interface `eth1.101`.
- Sans indication particulière, les passerelles ont la même pondération : `weight 1`. Comme précisé dans la Section 2, « Présentation de la topologie réseau étudiée », toutes les liaisons sont homogènes et offrent le même débit réseau.

4.2. Analyse réseau sur le routeur ISP avec répartition de trafic

Maintenant que la capacité à exploiter les deux liens est implantée à chaque extrémité, il ne reste qu'à valider le fonctionnement à partir du trafic émis et reçu l'hôte `host` transitant par `ISP`. C'est sur ce routeur que l'on procède à l'analyse du trafic avec l'outil tshark qui permet la capture de trafic en mode console. Le système du routeur `ISP` a en plus été configuré pour pouvoir effectuer les captures au niveau utilisateur normal en appliquant les instructions données dans l'article *Capturer le trafic réseau au niveau utilisateur avec Wireshark*.

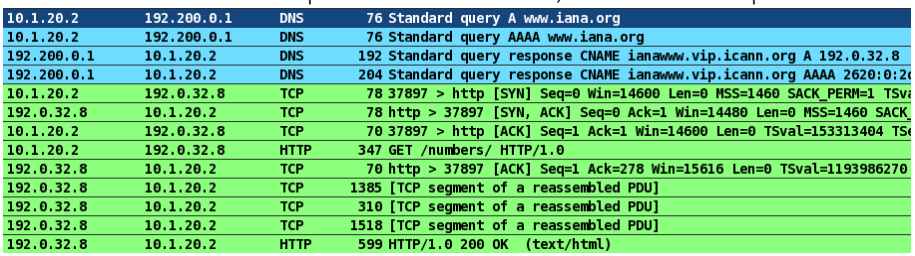
Voici une séquence typique de capture sur le routeur `ISP` pendant le chargement d'une page ***Web*** sur `host`.

```
etu@isp:~$ screen tshark -i eth1 ! port 22  -w /var/tmp/multipath-host.pcap
[detached from 13986.pts-0.isp]
etu@isp:~$ ssh 10.1.20.2
Linux host 3.1.0-1-amd64 #1 SMP Mon Nov 14 08:02:25 UTC 2011 x86_64
<snipped/>
etu@host:~$ lynx www.iana.org
Recherche  'www.iana.org' premier
etu@host:~$ logout
Connection to 10.1.20.2 closed.
etu@isp:~$ screen -r
[screen is terminating]
etu@isp:~$ ls -lh /var/tmp/multipath-host.pcap
-rw------- 1 etu etu 12K déc.  11 23:25 /var/tmp/multipath-host.pcap
```

- La commande screen fournie avec le paquet du même nom, sert à ouvrir un terminal dans lequel on lance la capture des paquets avec tshark.
- L'interface utilisée pour la capture réseau est `eth1`. Comme le «composant» contrôleur d'interface réseau sur une machine virtuelle ne dispose pas de fonctions évoluées, tous les types de trames transitant par l'interface sont capturés. On enregistre ainsi les trames avec les balises IEEE 802.1Q des VLANs `101` et `103`.
- L'option `! port 22` élimine tous les segments utilisant le port SSH comme source ou comme destination. Comme le protocole SSH est utilisé pour ouvrir les sessions sur les routeurs et l'hôte `host`, il est inutile de capturer ce trafic.
- Les paquets capturés sont enregistrés dans le fichier `/var/tmp/multipath-host.pcap`. Comme le ***directory sticky bit*** est positionné sur le répertoire `/var/tmp/`, l'utilisateur normal dispose des droits d'écriture nécessaires à l'enregistrement des paquets capturés.
- Une fois la capture lancée, on ouvre un terminal sur `host` et on lance le navigateur web lynx. L'ouverture de la page `www.iana.org` entraîne les échanges classiques de requêtes DNS puis HTTP. Ce sont ces échanges qui doivent valider l'utilisation des deux liens entre le routeur `ISP` et l'aire OSPF.

- Après avoir consulté une ou deux pages web, on referme la session sur `host` et on arrête la capture réseau. Il ne reste alors qu'à analyser le contenu du fichier de capture.

En ouvrant le fichier de capture avec Wireshark, on retrouve les phases de la consultation de page web.



Cette copie d'écran montre les échanges au niveau réseau ainsi que les protocoles utilisés par les couches supérieures. On retrouve bien les adresses IP des différents protagonistes.

- `10.1.20.2` : hôte `host` à l'initiative du trafic réseau.
- `192.0.2.1` : système hôte hébergeant les instances de machines virtuelles sur lequel on trouve aussi un service de résolution des noms de domaines de type cache.
- `192.0.32.8` : serveur web hébergeant le site `www.iana.org`.

À première vue, la copie d'écran et l'identification des hôtes en communication ne montrent rien d'original. Il faut aller jusqu'à la visualisation des trames au niveau liaison pour caractériser la distribution du trafic sur les différents liens.

Prenons les trois étapes de l'établissement de connexion TCP entre le poste client et le serveur web.

1. La demande d'établissement de connexion ([SYN]) est émise par le poste client. Elle transite par le lien correspondant au VLAN 103.

```
No.  Time        Source                Destination           Protocol Info
  19 21.215999   10.1.20.2             192.0.32.9            TCP      56075 > http [SYN] Seq=0

Frame 19: 78 bytes on wire (624 bits), 78 bytes captured (624 bits)
Ethernet II, Src: RealtekU_12:34:06 (52:54:00:12:34:06), Dst: de:ad:be:ef:01:03 (de:ad:be:ef:01:03)
802.1Q Virtual LAN, PRI: 0, CFI: 0, ID: 103
    000. .... .... .... = Priority: Best Effort (default) (0)
    ...0 .... .... .... = CFI: Canonical (0)
    .... 0000 0110 0111 = ID: 103
    Type: IP (0x0800)
Internet Protocol Version 4, Src: 10.1.20.2 (10.1.20.2), Dst: 192.0.32.9 (192.0.32.9)
Transmission Control Protocol, Src Port: 56075 (56075), Dst Port: http (80), Seq: 0, Len: 0
```

2. L'acquittement de la demande d'établissement de connexion et du numéro initial de séquence ([SYN, ACK]) du poste client est émis par le serveur web. Il transite par le lien correspondant au VLAN 101.

```
No.  Time        Source                Destination           Protocol Info
  20 21.216434   192.0.32.9            10.1.20.2             TCP      http > 56075 [SYN, ACK] Seq=0 Ack=1

Frame 20: 78 bytes on wire (624 bits), 78 bytes captured (624 bits)
Ethernet II, Src: de:ad:be:ef:01:01 (de:ad:be:ef:01:01), Dst: RealtekU_12:34:04 (52:54:00:12:34:04)
802.1Q Virtual LAN, PRI: 0, CFI: 0, ID: 101
    000. .... .... .... = Priority: Best Effort (default) (0)
    ...0 .... .... .... = CFI: Canonical (0)
    .... 0000 0110 0101 = ID: 101
Type: IP (0x0800)
Internet Protocol Version 4, Src: 192.0.32.9 (192.0.32.9), Dst: 10.1.20.2 (10.1.20.2)
Transmission Control Protocol, Src Port: http (80), Dst Port: 56075 (56075), Seq: 0, Ack: 1, Len: 0
```

3. L'acquittement du numéro initial de séquence ([ACK]) du serveur web est émis par le poste client. Il transite par le lien correspondant au VLAN 103.

```
No.  Time        Source                Destination           Protocol Info
  21 21.217229   10.1.20.2             192.0.32.9            TCP      56075 > http [ACK] Seq=1 Ack=1

Frame 21: 70 bytes on wire (560 bits), 70 bytes captured (560 bits)
Ethernet II, Src: RealtekU_12:34:06 (52:54:00:12:34:06), Dst: de:ad:be:ef:01:03 (de:ad:be:ef:01:03)
802.1Q Virtual LAN, PRI: 0, CFI: 0, ID: 103
    000. .... .... .... = Priority: Best Effort (default) (0)
    ...0 .... .... .... = CFI: Canonical (0)
    .... 0000 0110 0111 = ID: 103
Type: IP (0x0800)
Internet Protocol Version 4, Src: 10.1.20.2 (10.1.20.2), Dst: 192.0.32.9 (192.0.32.9)
Transmission Control Protocol, Src Port: 56075 (56075), Dst Port: http (80), Seq: 1, Ack: 1, Len: 0
```

> Note
>
> Le fichier de capture réseau comprenant l'ensemble des échanges est téléchargeable : multipath-host.pcap

Avec ces trois étapes de l'établissement d'une connexion TCP on caractérise bien la répartition du trafic sur les deux liens de transit entre les passerelles de l'aire OSPF et le routeur de niveau supérieur ISP. Cette répartition n'est pas nécessairement équitable dans la mesure où l'alternance de l'utilisation des liens n'intervient que lors des permutations entre les adresses IP source et destination. Ainsi, avec un profil utilisateur de type «globe oculaire passif», le téléchargement de flux vidéo n'utilise qu'un seul lien ; ce qui provoque un déséquilibre en volume de trafic sachant que la requête HTTP n'occupe que quelques centaines d'octets tandis que le fichier vidéo demandé occupe quelques mégaoctets. Si on disposait d'un parc important de postes de travail en lieu et place du seul système host, on pourrait dire que la répartition de trafic est statistiquement équitable.

Si la balance de charge entre plusieurs liens, telle qu'elle est présentée ici est très intéressante, elle peut cependant engendrer de gros problèmes dans le cas où un lien viendrait à être indisponible. En effet, la solution de répartition utilise ***nécessairement*** toutes les passerelles introduites dans les tables de routage. Il serait donc souhaitable de compléter la solution en introduisant la notion de tolérance aux pannes. C'est justement l'objet de la section suivante.

5. Tolérance aux pannes réseau - *multilink failover*

Pour aborder la tolérance aux pannes réseau sur les deux liens qui relient l'aire OSPF au routeur de niveau supérieur ISP, on reprend très exactement la configuration de la section précédente avec répartition du trafic sur ces deux mêmes liens.

Toujours comme dans le cas de la section précédente sur la répartition de trafic, la tolérance aux pannes sur les passerelles de l'aire OSPF est assurée par le protocole de routage dynamique. En effet, OSPF étant un protocole à état de liens, l'ajout ou le retrait d'un lien entraîne un nouveau calcul de la topologie par chacun des routeurs de l'aire. On a vu précédemment, que les deux passerelles de l'aire OSPF sont publiées par R1 et R3. Les états des deux liens entre l'aire et le routeur ISP sont donc intégrés dans les calculs de topologie.

Une fois encore, sans configuration particulière le routeur ISP ne dispose pas de fonctionnalité permettant de faire évoluer sa table de routage en fonction de l'état des liens sur lesquels il est directement connecté.

On se propose de compenser cette lacune en plusieurs étapes.

- Au lieu de modifier une unique table de routage statique, on va créer plusieurs tables de routage que l'on utilisera suivant le lien d'arrivée du trafic sur le routeur ISP.
- Pour identifier le trafic issu d'un lien, on va marquer ce trafic de façon à ce que le trafic retour emprunte le même lien. Le point clé ici étant que le routeur ISP ne peut pas recevoir de trafic via un lien inactif.

5.1. Tables de routage multiples sur le routeur ISP

Avec la distribution Debian GNU/Linux, le paquet iproute fournit les outils nécessaires aux manipulations sur les tables de routages multiples.

1. On commence par éditer le fichier /etc/iproute2/rt_tables qui permet de faire la correspondance entre l'identifiant numérique de table et le nom que l'on souhaite lui donner. Ici, on choisit de numéroter les deux tables de routage supplémentaires avec les numéros des VLANs de chacun des liens de raccordement de l'aire OSPF au routeur ISP.

```
$ cat /etc/iproute2/rt_tables
#
# reserved values
#
255     local
254     main
253     default
0       unspec
#
# local
#
101     link101
103     link103
```

2. Une fois les deux tables de routage supplémentaires disponibles, on saisit la route de synthèse (***static summary route***) dans chacune.

```
# ip route add 10.1.0.0/19 via 10.1.30.1 src 10.1.30.2 table link101
# ip route add 10.1.0.0/19 via 10.1.30.9 src 10.1.30.10 table link103
```

Du point de vue configuration système, ces deux instructions sont placées dans le fichier de configuration des interfaces réseau du routeur ISP.

3. On définit ensuite les conditions dans lesquelles ces tables de routage seront utilisées.

```
# ip rule add fwmark 101 table link101
# ip rule add fwmark 103 table link103
```

Ainsi, tout paquet IP portant la marque 101 sera routé via la table link101. Il en ira de même avec la marque 103 et la table link103.

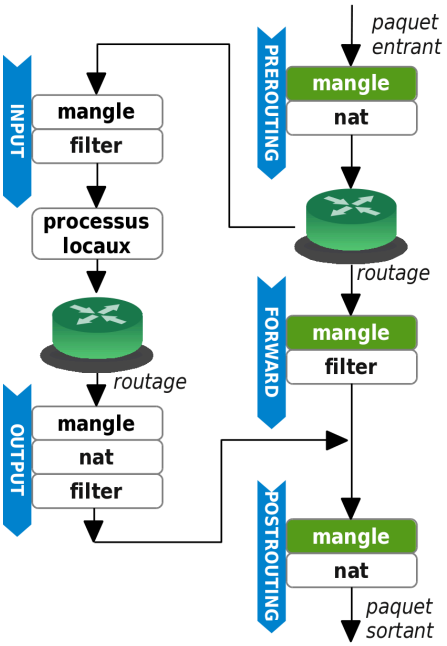
Dans les fonctions réseau du noyau Linux on parle de ***routage avancé*** dès que l'on ne se contente plus de l'examen de l'adresse IP destination des paquets reçus sur une interface réseau. L'utilisation d'un marquage des paquets pour orienter le routage entre bien dans cette catégorie.

Les fonctions de routage avancé utilisent une base de données de politiques de routage ou ***Routing Policy DataBase*** (RPDB). En utilisant cette base, il est possible de prendre une décision d'acheminement des paquets sur les différents champs des en-têtes de paquet IP, les protocoles réseau, les protocoles de transports et même la charge utile des paquets. Dans notre cas, c'est le marquage des paquets qui permet d'identifier les liens par lesquels ils ont transité. Ce marquage est contrôlé par les outils de filtrage réseau du noyau Linux : netfilter/iptables.

5.2. Marquage des paquets sur le routeur ISP

On distingue deux parties dans les outils de filtrage réseau du noyau Linux. La partie noyau, baptisée netfilter, comprend l'ensemble des fonctions disponibles ainsi que le découpage en tables et en chaînes. Ces fonctions sont appelées et utilisées à l'aide de la partie utilisateur baptisée iptables. Généralement, la partie utilisateur est fournie avec un paquet du même nom. Pour plus de détails sur l'organisation et la configuration du système de filtrage, consulter le guide *Tutoriel iptables*.

Dans le cas de cet article, le marquage des paquets utilise la notion d'altération de paquet ou ***packet mangling***.



Comme le trafic émis et reçu par l'hôte host ne fait que transiter par le routeur ISP, on s'intéresse plus particulièrement à la chaîne PREROUTING de la table mangle. Les processus locaux du routeur ne sont donc pas concernés par le marquage.

> **Avertissement**
>
> Le propos de cette section est le marquage et non le filtrage des paquets routés par ISP. La politique de filtrage est donc maintenue dans son état par défaut : «tout ce qui n'est pas interdit est autorisé». Ici, aucun trafic n'est interdit.

Une fois que la table et la chaîne sont choisies, il faut définir la correspondance à utiliser. C'est à ce niveau qu'intervient une fonction très intéressante : CONNMARK. Cette correspondance permet d'associer le marquage des paquets au mécanisme de suivi des communications réseau. Ainsi, un flux entrant dans le routeur ISP via les interfaces de connexion à l'aire OSPF (eth1.101 ou eth1.103) est marqué ***et enregistré*** dans la table de suivi des communications. La marque peut alors être restituée à l'arrivée du flux retour sur l'interface externe (eth0). Les instructions correspondant aux choix effectués sont les suivantes.

- Marquage des flux sortants de l'aire OSPF.

```
# iptables -t mangle -A PREROUTING -i eth1.101 -j MARK --set-mark 0x65
# iptables -t mangle -A PREROUTING -i eth1.103 -j MARK --set-mark 0x67
# iptables -t mangle -A PREROUTING -i eth1.101 -j CONNMARK --save-mark
# iptables -t mangle -A PREROUTING -i eth1.103 -j CONNMARK --save-mark
```

- Restauration du marquage des flux retours à destination de l'aire OSPF.

```
# iptables -t mangle -A PREROUTING -i eth0 -j CONNMARK --restore-mark
```

Du point de vue configuration système, les règles appliquées à l'aide de la commande iptables peuvent être enregistrées dans un fichier de configuration et restaurées à chaque initialisation du routeur. Voir Annexe D, *Configuration du filtrage réseau*.

Le fonctionnement du marquage peut être caractérisé à plusieurs niveaux. En reprenant l'exemple de la consultation de pages ***web*** depuis l'hôte `host` et en consultant les résultats du marquage des paquets, on obtient les informations suivantes.

Consultation de la base FIBCommutation de paquets IP

On retrouve les informations de marquage au niveau réseau dans la ***Forwarding Information Base***. La base FIB est une table de hachage contenant une trace l'ensemble des décisions de routage. La taille de la base augmente rapidement avec le nombre des flux réseau routés par `ISP`. C'est la raison pour laquelle on ne donne pas une copie complète de cette base ci-dessous.

L'instruction de consultation de la base FIB est la suivante.

```
$ ip route ls cache
```

Deux exemples d'entrées portant la marque `101` ou `0x65`.

```
192.0.32.8 from 10.1.20.2 via 192.0.2.1 dev eth0  src 10.1.30.2  mark 0x65
    cache  iif eth1.101
10.1.20.2 from 192.0.32.8 via 10.1.30.1 dev eth1.101  src 192.0.2.3  mark 0x65
    cache  ipid 0x0be6 rtt 7ms rttvar 6ms cwnd 10 iif eth0
```

La première ligne correspond à la décision d'acheminement des paquets émis par `host` (`10.1.20.2`) à destination de l'adresse `192.0.32.8` reçus sur l'interface `eth1.101`. La seconde ligne correspond à la décision symétrique.

Deux exemples d'entrées portant la marque `103` ou `0x67`.

```
192.0.32.9 from 10.1.20.2 via 192.0.2.1 dev eth0  src 10.1.30.2  mark 0x67
    cache  iif eth1.103
10.1.20.2 from 192.0.32.9 via 10.1.30.9 dev eth1.103  src 192.0.2.3  mark 0x67
    cache  ipid 0x0be6 rtt 7ms rttvar 6ms cwnd 10 iif eth0
```

Comme dans le cas des deux entrées précédentes, la première ligne correspond à la décision d'acheminement des paquets émis par `host` (`10.1.20.2`) à destination de l'adresse `192.0.32.9` reçus sur l'interface `eth1.103` et la seconde ligne correspond à la décision symétrique.

Consultation de la base conntrackTable de suivi des communications

Les mêmes informations de marquage se retrouvent au niveau de la table de suivi des communications réseau. Pour consulter les entrées de cette table, on peut utiliser l'outil conntrack fourni avec le paquet du même nom. Là encore, le nombre des entrées augmente rapidement avec l'apparition de nouveaux flux réseau.

L'instruction de consultation de la table de suivi des communications réseau est la suivante.

```
# conntrack -L
```

Deux exemples d'entrées portant la marque `101`.

```
tcp      6 68 TIME_WAIT src=10.1.20.2 dst=192.0.32.8 sport=37926 dport=80 src=192.0.32.8 dst=10.1.20.2 \
                                      sport=80 dport=37926 [ASSURED] mark=101 use=1
tcp      6 81 TIME_WAIT src=10.1.20.2 dst=192.0.32.8 sport=37927 dport=80 src=192.0.32.8 dst=10.1.20.2 \
                                      sport=80 dport=37927 [ASSURED] mark=101 use=1
```

Un autre exemple d'entrée portant la marque `103`.

```
tcp      6 92 TIME_WAIT src=10.1.20.2 dst=192.0.32.9 sport=56108 dport=80 src=192.0.32.9 dst=10.1.20.2 \
                                      sport=80 dport=56108 [ASSURED] mark=103 use=1
```

La table `conntrack` fait apparaître les conditions d'identification d'un flux retour en identifiant en plus du protocole de couche transport les paires d'adresses IP en communication ainsi que les paires de numéros de ports utilisés.

Maintenant que les fonctions de marquage des paquets sont en place, les liens actifs entre l'aire OSPF et le routeur de niveau supérieur sont identifiés. Il reste à qualifier la tolérance aux pannes à l'aide de l'analyse réseau.

5.3. Analyse réseau sur le routeur ISP avec tolérance aux pannes

Pour caractériser la tolérance aux pannes sur les liens réseau entre l'aire OSPF et le routeur `ISP`, on procède de la façon suivante.

1. On lance la capture de trafic dans un terminal dédié sur le routeur `ISP`.
2. On lance le téléchargement d'un image iso de DVD dans un terminal dédié sur l'hôte `host`. L'idée est d'effectuer les interruptions de liens pendant qu'une transaction réseau est en cours.
3. De retour sur `ISP`, on identifie le lien utilisé pour le téléchargement à l'aide de la commande conntrack. C'est ce lien que l'on doit interrompre en premier de façon à provoquer le basculement d'un lien vers l'autre en cas de panne.
4. Partant du routeur `ISP`, on se connecte sur `R3` puis `R1` pour faire «tomber» le lien correspondant au VLAN et à la marque `101`. On rétablit ensuite ce même lien.
5. Partant du routeur `ISP`, on se connecte sur `R1` puis `R3` pour faire «tomber» le lien correspondant au VLAN et à la marque `103`. On rétablit ensuite ce même lien.
6. On retourne sur l'hôte `host` pour constater que le téléchargement est toujours en cours et se poursuit normalement. Ce téléchargement peut maintenant être arrêté.
7. On arrête la capture de trafic sur `ISP` et récupère le fichier d'enregistrement des paquets capturés pour l'analyser.

Voici une trace des commandes exécutées sur les différents routeurs et sur l'hôte à l'initiative du trafic.

- Sur le routeur `ISP`, la capture de trafic est le point important.

```
$ screen tshark -i eth1 ! port 22  -w /var/tmp/failover-host.pcap
```

- Sur l'hôte `host`, on lance le téléchargement «long».

```
$ screen wget http://cdimage.debian.org/debian-cd/6.0.3/amd64/iso-dvd/debian-6.0.3-amd64-DVD-1.iso
```

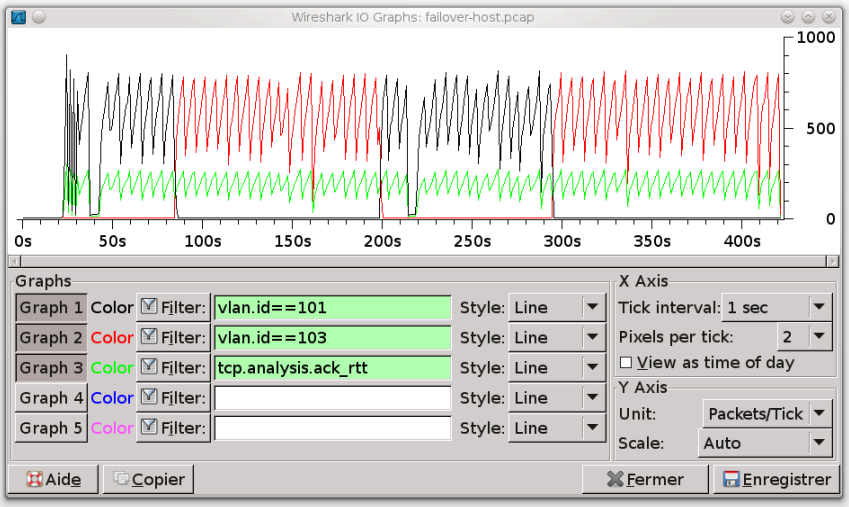
- Sur le routeur `R1`, on désactive puis réactive le lien vers `ISP`.

```
# ifconfig eth0.101 down
<attendre un certain temps.../>
# ifconfig eth0.101 up
# ip ro add default via 10.1.30.2
```

- De même, sur le routeur `R3`, on désactive puis réactive le lien vers `ISP`.

```
# ifconfig eth0.103 down
<attendre un certain temps.../>
# ifconfig eth0.103 up
# ip ro add default via 10.1.30.10
```

À la différence de la section Section 4.2, « Analyse réseau sur le routeur ISP avec répartition de trafic », le volume de trafic capturé est beaucoup plus important. On ne propose donc ici qu'une représentation graphique caractérisant le basculement d'un lien sur l'autre à chaque désactivation d'interface.



Il y aurait beaucoup de remarques à faire sur cette capture de téléchargement. Simplement :

- La courbe ***noire*** correspond à l'utilisation du lien entre `R1` et `ISP` : le VLAN `101`.
- La courbe ***rouge*** correspond à l'utilisation du lien entre `R3` et `ISP` : le VLAN `103`.
- La courbe ***verte*** est là pour indiquer que certains délais observés lors des échanges sont dus à la communication de bout en bout entre l'hôte `host` et le serveur ***web*** et non au système de tolérance aux pannes réseau.

6. Tolérance aux pannes réseau sur les flux descendants - *customized keepalive*

Dans la section précédente, qui traite aussi de la tolérance aux pannes réseau, on a uniquement caractérisé les flux sortants de l'aire OSPF. Autrement dit, cette tolérance n'est effective que pour les flux émis à partir des hôtes présents dans cette aire. Ces hôtes sont donc nécessairement assimilés à des ***clients*** dans le modèle client/serveur.

Considérons maintenant le cas où ces hôtes sont des serveurs dont le contenu doit être accessible depuis l'Internet. Le trafic correspondant transite toujours par le routeur `ISP` mais celui-ci ne dispose d'aucun moyen de détection de l'état des liens vers les routeurs `R1` et `R3`.

Les fonctions de détection de l'état des liens sont intrinsèques aux protocoles de routage de type ***link-state***. Une fois que la convergence initiale est atteinte, les nouveaux calculs de topologie n'interviennent que si un lien connu au préalable est désactivé ou réactivé. La section 4.3 du standard RFC2328 *OSPF Version 2* définit les 5 types de paquets du protocole de routage. Le type 1 est justement le ***Hello packet*** dont la fonction est de découvrir et de maintenir les adjacences avec les routeurs OSPF voisins. Sur un réseau de diffusion, comme dans le cas de cet article, ces paquets sont émis toutes les 10 secondes et utilisent une adresse IP destination multicast propre au protocole : `224.0.0.5`.

Comme le routeur `ISP` n'appartient pas au même domaine d'administration que les trois autres routeurs de l'aire OSPF, aucune adjacence n'est formée sur les deux liens de sortie de cette aire. On se propose donc de compenser l'absence de contrôle d'état à l'aide de scripts maison.

6.1. Scripts de contrôle d'état de lien - *keepalive shell scripts*

L'objectif ici n'est pas de reproduire un fonctionnement aussi sophistiqué que celui offert par un protocole de routage dynamique, mais de trouver une solution à moindre coût qui permette de manipuler les routes par défaut des routeurs `R1` et `R3` à partir des réponses aux requêtes ICMP de type `8` (***echo***).

Avant d'aller plus loin dans la présentation de la technique utilisée ici, il faut rappeler que l'architecture étudiée ne comprend que des interfaces Ethernet sur lesquelles transitent des trames avec balises IEEE 802.1Q. On ne dispose donc pas d'informations provenant de la couche physique sur le fonctionnement de ces interfaces. Si les deux liens à contrôler étaient de type série, les scripts présentés ci-dessous seraient totalement inutiles puisque les composants contrôleurs des liaisons série détectent directement les interruptions de transmission.

Il faut aussi ajouter que, même avec des réseaux locaux Ethernet, d'autres solutions existent. Ainsi, il est possible d'utiliser le protocole PPPoE qui offre toutes les fonctions du protocole PPP sur des liaisons Ethernet. Pour un exemple de mise en œuvre, voir l'article *Routage inter-VLAN et protocole PPPoE*.

Avec l'architecture correspondant à la vue topologie logique présentée au début à la Section 2, « Présentation de la topologie réseau étudiée », on place deux scripts développés en langage ***shell*** qui fonctionnent en permanence sur les routeurs `R1` et `R3`. Le code ce ces deux scripts est donné dans l'Annexe E, *Scripts de contrôle d'état de liens*.

La partie intéressante de ces deux scripts est extraite ci-dessous.

```
while $(true) ; do
  msg=$(/bin/ping -W 2 -n -c 1 ${neighbor} 2>&1 | egrep '(bytes from|100% packet loss)') ❶

  if [[ "${msg}" =~ "bytes from" ]] && [[ ! -e /tmp/${neighbor}_up ]]; then ❷
    ip route add default via ${neighbor} ❸
    touch /tmp/${neighbor}_up ❹
    keepalive_debug "${neighbor} is reachable, default route added" ❺
  fi

  if [[ "${msg}" =~ "100% packet loss" ]] && [[ -e /tmp/${neighbor}_up ]]; then ❻
    ip route del default via ${neighbor} ❼
    rm -f /tmp/${neighbor}_up ❽
    keepalive_debug "${neighbor} is not reachable, default route deleted" ❾
  fi

  sleep ${delay} ❿
done
```

❶ La collecte du message d'information ICMP se fait à l'aide d'une seule requête dont on a limité le temps d'attente de réponse à 2 secondes. En fonction de cette réponse on extrait deux chaînes de caractères : "`bytes from`" si la réponse est positive ou "`100% packet loss`" si aucune réponse n'a été obtenue dans le temps imparti.

La réponse stockée dans la variable `${msg}`.

❷❻ Suivant le contenu de la variable `${msg}` et de l'état précédemment enregistré du lien, on ajoute ou supprime la route par défaut vers le routeur `ISP`.

❸❼ La commande d'ajout et de suppression de la route par défaut suppose que le propriétaire du processus a la capacité à utiliser cette instruction.

La variable `${neighbor}` contient l'adresse IP de la passerelle à atteindre.

❹❽ Le fichier `/tmp/${neighbor}_up` est utilisé comme variable d'état. Il est créé lors de l'ajout de la route par défaut lorsque le lien vers le routeur `ISP` est à nouveau actif. Il est effacé lorsque ce même lien est perdu.

❺❾ Les deux scripts utilisent une fonction de journalisation système embryonnaire `keepalive_debug` qui permet d'enregistrer les mouvements sur les liens. Une copie des messages générés à l'aide de cette fonction est donnée à la fin de la section suivante sur l'analyse réseau.

❿ Dans le cas de cette maquette, le délai a été fixé à 10 secondes par l'intermédiaire de la variable `${delay}`. L'idée étant d'avoir un comportement qui se rapproche de la période des ***Hello packets*** du protocole OSPF.

6.2. Analyse réseau sur le routeur ISP avec tolérance aux pannes sur les flux entrants

Pour caractériser la tolérance aux pannes sur les flux entrants dans l'aire OSPF via le routeur `ISP`, on suit la séquence d'instructions suivante.

1. Sur le système hôte, on créé un petit script de téléchargement en boucle que l'on exécute dans un terminal dédié.

```
$ cat download.sh
#!/bin/bash

while true
do
  wget http://10.1.20.2/debian-6.0.3-amd64-CD-1.iso
  rm debian-6.0.3-amd64-CD-1.iso
done
```

```
$ screen sh -x download.sh
```

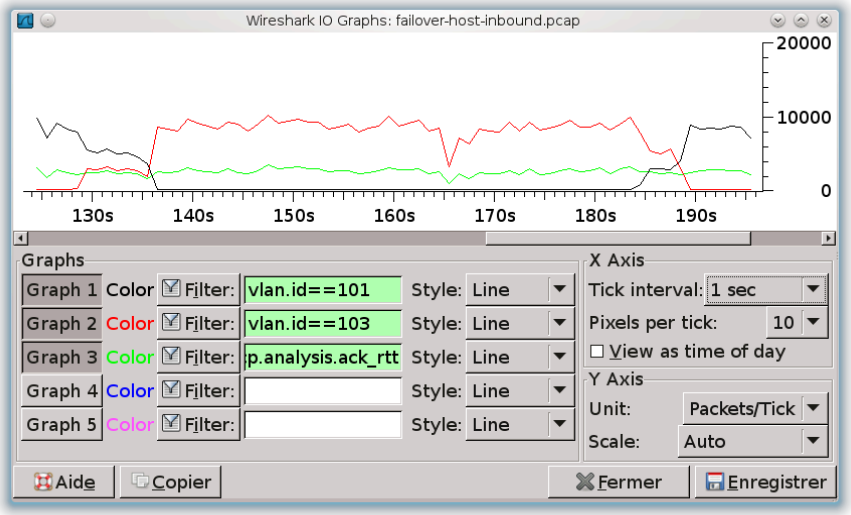
2. Sur le routeur `ISP`, on commence par lancer la capture de trafic dans un terminal dédié. Ensuite, on désactive puis réactive les interfaces réseau connectées aux liaisons avec l'aire OSPF.

```
$ screen tshark -i eth1 ! port 22 -a filesize:8192000 -w /var/tmp/failover-host-inbound.pcap
```

   Le choix du lien à désactiver dépend de l'enregistrement des communications après marquage des paquets provenant de l'aire OSPF. Dans le cas ci-dessous, c'est le lien entre `R1` et `ISP` qui était utilisé avant la désactivation de l'interface `eth1.101`.

```
# conntrack -L
tcp      6 299 ESTABLISHED src=192.0.2.1 dst=10.1.20.2 sport=50403 dport=80 \
                           src=10.1.20.2 dst=192.0.2.1 \
                           sport=80 dport=50403 [ASSURED] mark=101 use=1
<snipped/>
# ip link set dev eth1.101 down
<attendre au moins 10s/>
# conntrack -L
tcp      6 299 ESTABLISHED src=192.0.2.1 dst=10.1.20.2 sport=50405 dport=80 \
                           src=10.1.20.2 dst=192.0.2.1 \
                           sport=80 dport=50405 [ASSURED] mark=103 use=1
<snipped/>
# ip link set dev eth1.101 up
<attendre au moins 10s/>
# conntrack -L
tcp      6 299 ESTABLISHED src=192.0.2.1 dst=10.1.20.2 sport=50405 dport=80 \
                           src=10.1.20.2 dst=192.0.2.1 \
                           sport=80 dport=50405 [ASSURED] mark=103 use=1
<snipped/>
# ip link set dev eth1.103 down
<attendre au moins 10s/>
# conntrack -L
tcp      6 299 ESTABLISHED src=192.0.2.1 dst=10.1.20.2 sport=50405 dport=80 \
                           src=10.1.20.2 dst=192.0.2.1 \
                           sport=80 dport=50405 [ASSURED] mark=101 use=1
<snipped/>
# ip link set dev eth1.103 up
```

Comme le téléchargement est effectué entre le système hôte et une instance de machine virtuelle, les débits disponibles sont importants et le volume de trafic capturé pour caractériser le basculement d'un lien sur l'autre augmente très rapidement. Le graphique ci-dessous est un extrait obtenu à partir d'une capture d'environ 1.2Go.



Les choix de couleurs de courbes sont les mêmes que dans la section précédente.

- La courbe ***noire*** correspond à l'utilisation du lien entre `R1` et `ISP` : le VLAN `101`.
- La courbe ***rouge*** correspond à l'utilisation du lien entre `R3` et `ISP` : le VLAN `103`.
- La courbe ***verte*** est là pour indiquer que les interruptions de trafic correspondent à la rotation des fichiers téléchargés sur le système hôte depuis le serveur ***web*** installé sur l'hôte `host`.

Suite aux manipulations effectuées sur les interfaces du routeur `ISP`, les journaux système des deux routeurs en vis-à-vis donnent les résultats suivants.

- Vu du routeur `R1` :

```
$ cat /var/log/r1-keepalive.log
janv. 04 22:05:01: START
janv. 04 22:05:01: daemonize ./r1-keepalive.sh
janv. 04 22:05:01: daemonized ./r1-keepalive.sh
janv. 04 22:05:01: alive_ping
janv. 04 22:05:01: 10.1.30.2 is reachable, default route added
janv. 04 22:12:36: 10.1.30.2 is not reachable, default route deleted
janv. 04 22:13:11: 10.1.30.2 is reachable, default route added
```

- Vu du routeur `R3` :

```
$ cat /var/log/r3-keepalive.log
janv. 04 22:06:34: START
janv. 04 22:06:34: daemonize ./r3-keepalive.sh
janv. 04 22:06:34: daemonized ./r3-keepalive.sh
janv. 04 22:06:34: alive_ping
janv. 04 22:13:29: 10.1.30.10 is not reachable, default route deleted
janv. 05 16:26:22: 10.1.30.10 is reachable, default route added
```

7. Pour conclure

Cet article est une illustration de quelques notions de routage avancé avec quatre instances de machines virtuelles. Il peut paraître un peu long ; comme la plupart des documents publiés sur ***inetdoc.net***. Il se distingue cependant par le fait qu'à chaque élément de configuration avancé, on précise la démarche suivie pour caractériser la fonction implantée et on donne un extrait des résultats obtenus.

Même s'il est question de routage «avancé», la section sur la répartition de trafic illustre parfaitement le cours sur les modélisations. On caractérise le fait que chaque paquet IP peut suivre un chemin propre au niveau réseau tout en préservant une connexion de bout en bout au niveau transport. On retrouve ainsi les notions de réseau à commutation de paquets et de protocole orienté connexion.

Le marquage de paquets et l'enregistrement des communications sont aussi très intéressants et montrent qu'il est possible d'utiliser ces mécanismes sans nécessairement avoir recours à la traduction d'adresses. Sujet au combien polémique.

Enfin, l'architecture étudiée n'est qu'un exemple de ce qu'il est possible de faire avec des systèmes GNU/Linux. Il existe bien d'autres solutions ou variations pour aboutir au même résultat !

A. Configuration de la commutation

Les interfaces réseau des instances de systèmes virtualisés sont un brassées sur une instance de commutateur, virtuel lui aussi, ***vde***. Ce commutateur est fourni par le paquet `vde2` et il est lancé lors de l'initialisation de l'interface `tap0` sur le système hôte. Voici un extrait du fichier `/etc/network/interfaces`.

```
auto tap0
iface tap0 inet static
        address 192.0.2.1
        netmask 255.255.255.192
        network 192.0.2.0
        broadcast 192.0.2.63
        vde2-switch -
```

Le brassage correspondant à la vue topologie logique est donné dans le tableau ci-dessous.

Tableau A.1. Brassage commutateur virtuel

Port VDE	Hôte	Interface(s)	Liaison
1	Système hôte	tap0	commutateur vde
2	ISP	eth0	système hôte | Internet
3	ISP	eth1.101	link101
		eth1.103	link103
4	R1	eth0.101	link101
		eth0.13	trunk R1 + R3
		eth0.12	trunk R1 + R2
5	R2	eth0	host
		eth0.12	trunk R2 + R1
		eth0.23	trunk R2 + R3
6	R3	eth0.103	link103
		eth0.13	trunk R3 + R1
		eth0.23	trunk R3 + R2
7	host	eth0	R2

Le fichier de configuration à charger au lancement du commutateur se présente comme suit.

```
vlan/create 12
vlan/create 13
vlan/create 23
vlan/create 20
vlan/create 101
vlan/create 103
vlan/create 999

vlan/addport 12 4
vlan/addport 12 5

vlan/addport 13 4
vlan/addport 13 6

vlan/addport 23 5
vlan/addport 23 6

vlan/addport 20 5
vlan/addport 20 7

vlan/addport 101 3
vlan/addport 101 4

vlan/addport 103 3
vlan/addport 103 6

vlan/addport 999 3
vlan/addport 999 4
vlan/addport 999 6
```

B. Configuration des interfaces

Les configurations des interfaces réseau des routeurs sont données ci-dessous. Dans la liste des interfaces des différents routeurs, on retrouve une sous-interface par VLAN. Ces sous-interfaces sont configurées en utilisant la commande ip fournie avec le paquet iproute. Comme cet article, s'appuie déjà énormément sur iproute, on utilise aussi la commande ip pour l'affectation des numéros de VLANs. Par défaut, les scripts spécifiques à la distribution Debian GNU/Linux utilisent la commande vconfig fournie avec le paquet vlan. Cette dernière commande est dorénavant considérée comme obsolète.

Routeur ISP

```
# The loopback network interface
auto lo
iface lo inet loopback

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

auto eth1.101
iface eth1.101 inet static
  address 10.1.30.2
  netmask 255.255.255.248
  network 10.1.30.0
  broadcast 10.1.30.7
  pre-up ip link set dev eth1 up
  pre-up ip link add link eth1 name eth1.101 type vlan id 101
  pre-up ip link set dev eth1.101 address de:ad:be:ef:01:01
  post-up ip route add 10.1.0.0/19 via 10.1.30.1 src 10.1.30.2 table link101
  post-up ip rule add fwmark 101 table link101
  post-down ip rule del fwmark 101 table link101
  post-down ip route del 10.1.0.0/19 via 10.1.30.1 src 10.1.30.2 table link101
  post-down ip link del link eth1 name eth1.101 type vlan id 101

auto eth1.103
iface eth1.103 inet static
  address 10.1.30.10
  netmask 255.255.255.248
  network 10.1.30.8
  broadcast 10.1.30.15
  pre-up ip link set dev eth1 up
  pre-up ip link add link eth1 name eth1.103 type vlan id 103
  pre-up ip link set dev eth1.103 address de:ad:be:ef:01:03
  post-up ip route add 10.1.0.0/19 via 10.1.30.9 src 10.1.30.10 table link103
  post-up ip rule add fwmark 103 table link103
  post-down ip rule del fwmark 103 table link103
  post-down ip route del 10.1.0.0/19 via 10.1.30.9 src 10.1.30.10 table link103
  post-down ip link del link eth1 name eth1.103 type vlan id 103
```

Routeur R1

```
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet manual

auto eth0.101
iface eth0.101 inet static
  address 10.1.30.1
  netmask 255.255.255.248
  network 10.1.30.0
  pre-up ip link set dev eth0 up
  pre-up ip link add link eth0 name eth0.101 type vlan id 101
  post-up ip ro add default via 10.1.30.2

auto eth0.12
iface eth0.12 inet static
  address 10.1.12.1
  netmask 255.255.255.192
  network 10.1.12.0
  pre-up ip link set dev eth0 up
  pre-up ip link add link eth0 name eth0.12 type vlan id 12

auto eth0.13
iface eth0.13 inet static
  address 10.1.13.1
  netmask 255.255.255.192
  network 10.1.13.0
  pre-up ip link set dev eth0 up
  pre-up ip link add link eth0 name eth0.13 type vlan id 13
```

Routeur R2

```
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
  address 10.1.20.1
  netmask 255.255.255.192
  network 10.1.20.0

auto eth0.12
iface eth0.12 inet static
  address 10.1.12.2
  netmask 255.255.255.192
  network 10.1.12.0
  pre-up ip link add link eth0 name eth0.12 type vlan id 12

auto eth0.23
iface eth0.23 inet static
  address 10.1.23.2
  netmask 255.255.255.192
  network 10.1.23.0
  pre-up ip link add link eth0 name eth0.23 type vlan id 23
```

Routeur R3

```
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0.103
iface eth0.103 inet static
  address 10.1.30.9
  netmask 255.255.255.248
  network 10.1.30.8
  pre-up ip link set dev eth0 up
  pre-up ip link add link eth0 name eth0.103 type vlan id 103
  post-up ip route add default via 10.1.30.10

auto eth0.13
iface eth0.13 inet static
  address 10.1.13.3
  netmask 255.255.255.192
  network 10.1.13.0
  pre-up ip link set dev eth0 up
  pre-up ip link add link eth0 name eth0.13 type vlan id 13

auto eth0.23
iface eth0.23 inet static
  address 10.1.23.3
  netmask 255.255.255.192
  network 10.1.23.0
  pre-up ip link set dev eth0 up
  pre-up ip link add link eth0 name eth0.23 type vlan id 23
```

C. Configuration du routage statique et dynamique

Les configurations des démons `zebra` des trois routeurs sont données ci-dessous. La seule particularité de ces configurations tient à la définition des bandes passantes sur chaque interface. Ces bandes passantes sont utilisées par les démons `ospfd` pour calculer les métriques de chacun des liens.

```
hostname R1-Zebra
password zebra
enable password zebra
log file /var/log/quagga/zebra.log
!
interface eth0
 bandwidth 1000000
 ipv6 nd suppress-ra
!
interface eth0.12
 bandwidth 1000000
 ipv6 nd suppress-ra
!
interface eth0.13
 bandwidth 1000000
 ipv6 nd suppress-ra
!
interface eth0.101
 bandwidth 1000000
 ipv6 nd suppress-ra
!
interface lo
!
ip forwarding
!
line vty
```

```
hostname R2-Zebra
password zebra
enable password zebra
log file /var/log/quagga/zebra.log
!
interface eth0
 bandwidth 1000000
 ipv6 nd suppress-ra
!
interface eth0.12
 bandwidth 1000000
 ipv6 nd suppress-ra
!
interface eth0.23
 bandwidth 1000000
 ipv6 nd suppress-ra
!
interface lo
!
ip forwarding
!
line vty
```

```
hostname R3-Zebra
password zebra
enable password zebra
log file /var/log/quagga/zebra.log
!
interface eth0
 bandwidth 1000000
 ipv6 nd suppress-ra
!
interface eth0.13
 bandwidth 1000000
 ipv6 nd suppress-ra
!
interface eth0.23
 bandwidth 1000000
 ipv6 nd suppress-ra
!
interface eth0.103
 bandwidth 1000000
 ipv6 nd suppress-ra
!
interface lo
!
ip forwarding
!
line vty
```

Les configurations des démons `ospfd` des trois routeurs sont données ci-dessous. Les deux routeurs de bordure `R1` et `R3` publient une route par défaut à l'aide de l'instruction `default-information originate`.

```
hostname R1-ospfd
password zebra
enable password zebra
log file /var/log/quagga/ospfd.log
!
interface eth0
!
interface eth0.12
!
interface eth0.13
!
interface eth0.101
!
interface lo
!
router ospf
 ospf router-id 0.0.0.1
! Important: ensure reference
! bandwidth is consistent across
! all routers
 auto-cost reference-bandwidth 1000
 network 10.1.12.0/26 area 0.0.0.0
 network 10.1.13.0/26 area 0.0.0.0
 default-information originate
!
line vty
```

```
hostname R2-ospfd
password zebra
enable password zebra
log file /var/log/quagga/ospfd.log
!
interface eth0
!
interface eth0.12
!
interface eth0.23
!
interface lo
!
router ospf
 ospf router-id 0.0.0.2
! Important: ensure reference
! bandwidth is consistent across
! all routers
 auto-cost reference-bandwidth 1000
 network 10.1.12.0/26 area 0.0.0.0
 network 10.1.20.0/26 area 0.0.0.0
 network 10.1.23.0/26 area 0.0.0.0
!
line vty
```

```
hostname R3-ospfd
password zebra
enable password zebra
log file /var/log/quagga/ospfd.log
!
interface eth0
!
interface eth0.13
!
interface eth0.23
!
interface eth0.103
!
interface lo
!
router ospf
 ospf router-id 0.0.0.3
! Important: ensure reference
! bandwidth is consistent across
! all routers
 auto-cost reference-bandwidth 1000
 network 10.1.13.0/26 area 0.0.0.0
 network 10.1.23.0/26 area 0.0.0.0
 default-information originate
!
line vty
```

D. Configuration du filtrage réseau

Les règles de marquage des flux réseau transitant par le routeur `ISP` sont enregistrées dans le fichier `/var/lib/iptables/active` dont la copie est donnée ci-dessous. Ce fichier est initialement créé à l'aide de l'instruction suivante.

```
# iptables-save >/var/lib/iptables/active
```

Il est ensuite possible de l'éditer et de le compléter au besoin.

```
# cat /var/lib/iptables/active
# Table filter ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
COMMIT
# Table mangle ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
*mangle
:PREROUTING ACCEPT [0:0]
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
-A PREROUTING -i eth0 -j CONNMARK --restore-mark --nfmask 0xffffffff --ctmask 0xffffffff
-A PREROUTING -i eth1.101 -j MARK --set-xmark 0x65/0xffffffff
-A PREROUTING -i eth1.103 -j MARK --set-xmark 0x67/0xffffffff
-A PREROUTING -i eth1.101 -j CONNMARK --save-mark --nfmask 0xffffffff --ctmask 0xffffffff
-A PREROUTING -i eth1.103 -j CONNMARK --save-mark --nfmask 0xffffffff --ctmask 0xffffffff
-A POSTROUTING -o eth0 -p tcp -m tcp --syn -m tcpmss --mss 1400:1536 -j TCPMSS --clamp-mss-to-pmtu
COMMIT
# Table nat    ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
*nat
:PREROUTING ACCEPT [0:0]
:INPUT ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
COMMIT
```

Les règles contenues dans ce fichier sont appliquées à l'aide de l'instruction suivante.

```
# iptables-restore </var/lib/iptables/active
```

On peut intégrer cette instruction dans les scripts d'initialisation des interfaces réseau de façon à rendre obligatoire l'application des règles. Par exemple :

```
# cat /etc/network/if-up.d/iptables
#!/bin/sh

if [ -f "/var/lib/iptables/active" ]
then
  iptables-restore </var/lib/iptables/active
fi

exit 0
```

E. Scripts de contrôle d'état de liens

Les deux scripts ci-dessous assurent un contrôle permanent sur les deux liens entre l'aire OSPF et le routeur `ISP` à l'aide des messages d'information ICMP ***echo/reply***. Le propriétaire du processus doit avoir la capacité à ajouter ou supprimer une route au niveau noyau.

- Sur le routeur `R1`, la passerelle à joindre est à l'adresse `10.1.30.2`. Le code du script est téléchargeable à partir du lien suivant : r1-keepalive.sh.

```
#!/bin/bash

#* Adaptation d'un script publié par Lukas Ruf en 2006
#* License: GPL3

neighbor=10.1.30.2

scriptname=`basename $0`
scriptname=${scriptname%.*}
delay=10 # sleep time
logfile=/var/log/$scriptname.log  # file for debug messages
lockfile=/var/lock/$scriptname.lock # locking to avoid races

# control where to log debug messages to:
# debugfile = 0 : log to /dev/null
# debugfile = 1 : log to file specified in ${logfile}
debugfile=1

# log a message to the log-file specified by debugfile
function keepalive_debug() {

  local msg="$1"
  local dat=$(date "+%b %d %H:%M:%S") # time stamp

  if [ "${debugfile}" == "1" ]; then
    lockfile -r 1 ${lockfile}
    if [ "$?" == "0" ]; then
      touch ${logfile} >/dev/null 2>&1
      chmod 0644 ${logfile} >/dev/null 2>&1
      echo -n "${dat}: " >>${logfile} 2>&1
      echo "${msg}" >>${logfile} 2>&1
      rm -f ${lockfile} >/dev/null 2>&1
    fi
  else
    echo "${msg}" >/dev/null 2>&1
  fi
}

# redirect messages from stdin to debug
function redirect() {

  local msg="$(cat -)"

  keepalive_debug "${msg}"
}

# ping the ${neighbor} and sleep ${delay} seconds
function alive_ping() {

  local msg=""

  keepalive_debug "alive_ping"
  while $(true) ; do
    msg=$(/bin/ping -W 2 -n -c 1 ${neighbor} 2>&1 | egrep '(bytes from|100% packet loss)')

    if [[ "${msg}" =~ "bytes from" ]] && [[ ! -e /tmp/${neighbor}_up ]]; then
      ip route add default via ${neighbor}
      touch /tmp/${neighbor}_up
      keepalive_debug "${neighbor} is reachable, default route added"
    fi

    if [[ "${msg}" =~ "100% packet loss" ]] && [[ -e /tmp/${neighbor}_up ]]; then
      ip route del default via ${neighbor}
      rm -f /tmp/${neighbor}_up
      keepalive_debug "${neighbor} is not reachable, default route deleted"
    fi

    sleep ${delay}
  done
}

# start up and handle proper daemonization
function main() {

  local argument="daemon"

  if [ "$1" == "${argument}" ]; then
    alive_ping
  else
    keepalive_debug "START"
    keepalive_debug "daemonize $0"
    nohup /bin/bash $0 ${argument} $* >/dev/null 2>&1 &
    keepalive_debug "daemonized $0"
  fi
}

main $*

exit 0
```

- Sur le routeur `R3`, la passerelle à joindre est à l'adresse `10.1.30.10`. Le code du script est téléchargeable à partir du lien suivant : r3-keepalive.sh.

  Le code de ce script est identique au précédent à l'exception de la valeur attribuée à la variable `${neighbor}`.