Réseaux

Iproute2 et QoS

Sommaire

- Iproute2 est un ensemble d'utilitaires utilisés pour controler le trafic TCP, UDP et IP dans Linux. ;
- Il permet le management des paquets Ipv4 et Ipv6 ;
- Il est maintenu par Stephen Hemminger alors que l'auteur originel, Alexey Kuznetsov est actuellement responsable uniquement de la partie QoS.

Il a été conçu pour remplacer la suite entière des outils Unix appelé net-tools:

Utilisation	Outil net-tool	Commande iproute2
Configuration d'adresse et lien	ifconfig	ip addr, ip link
Tables de routage	route	ip route
Voisinage	arp	ip neighbour
VLAN	vconfig	ip link
Tunnels	iptunnel	ip tunnel
Multicast	ipmaddr	ip maddr
Statistiques	netstat	ss

- Listing des interfaces:

  # ip addr {show}

- Listing d'une interface:

  # ip addr show dev eth0

- Activation / désactivation d'une interface:

  # ip link set eth0 [**up | down**]

- Ajout /supression d'une adresse sur l'interface eth0:

  # ip addr [**add | del**] 192.168.0.1/24 dev eth0

- Listing des routes:

  # ip route {show}

  Listing des routes sur une table:

  # ip route show table {table}

- Ajout / suppression d'une route:

  # ip **ro** [**add | del**] 10.0.0.0/16 via 192.168.0.254

- Demande de résolution de routage:

  # ip ro get 10.0.0.1

  10.0.0.1 via 192.168.0.254 dev eth0 src 192.168.0.1
  cache mtu 1500 advmss 1460 hoplimit 64

- Listing du voisinage:

  # ip neighbourg {show}

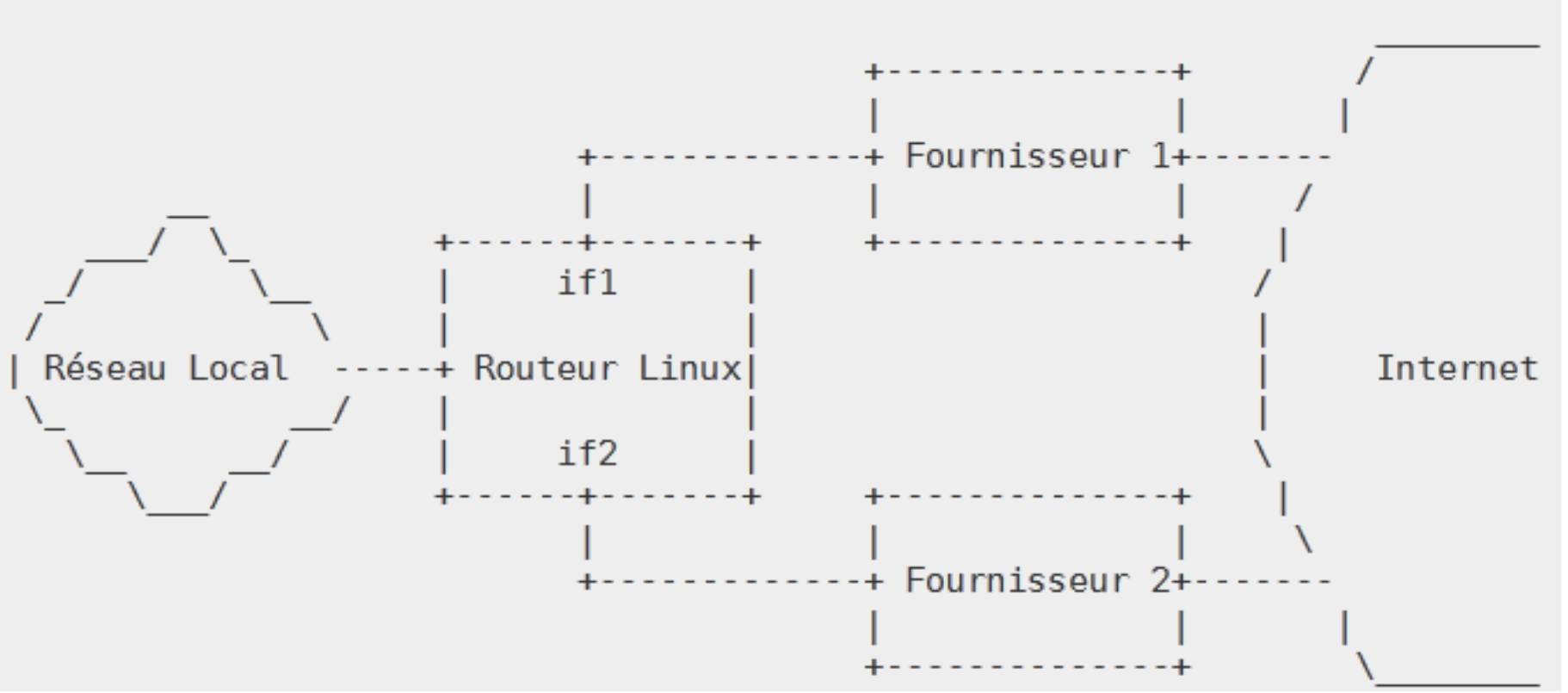
- Ajout / suppression / modification / renplacement d'un voisin:

  # ip neigh [**add | del | change | replace**]
  192.168.0.254 dev eth0 lladdr 00:02:a5:1f:cb:2d
  nud [permanent | stale | noarp | reachable]

- Vidange de la table de voisinage:

  # ip neigh flush

Imaginons que nous ayons deux accès Internet (deux FAI différents).



Commençons par définir quelques symboles:

- $IF1 est le nom de la première interface ;
- $IF2 est le nom de la deuxième interface ;

- $IP1 est l'adresse IP associée à $IF1 ;
- $IP2 est l'adresse IP associée à $IF2 ;

- $P1 est l'adresse IP de la passerelle du FAI1 ;
- $P2 est l'adresse IP de la passerelle du FAI2 ;

- $P1_NET est l'adresse réseau où se situe $P1 ;
- $P2_NET est l'adresse réseau où se situe $P2.

Commençons par créer deux tables de routages pour ne pas perturber les tables existantes:

```
# ip route add $P1_NET dev $IF1 src $IP1 table T1
# ip route add $P2_NET dev $IF2 src $IP2 table T2
```

Paramétrons les routes par défaut:

```
# ip route add default via $P1 table T1
# ip route add default via $P2 table T2
```

L'idéal est de router les éléments à destination d'un voisin direct à travers l'interface connectée à ce voisin.
C'est l'argument "src" qui assurent que la bonne adresse IP source sera choisie:

```
# ip route add $P1_NET dev $IF1 src $IP1
# ip route add $P2_NET dev $IF2 src $IP2
```

Il ne reste plus qu'à choisir une route par défaut:

```
# ip route add default via $P1
```

Il faut quand même s'assurer que toutes les réponses au trafic entrant sur une interface seront envoyées par cette même interface:

```
# ip rule add from $IP1 table T1
# ip rule add from $IP2 table T2
```

Il ne faut pas négliger la configuration du réseau local.

Soit $P0_NET est le réseau local et $IF0 son interface:

```
# ip route add $P0_NET dev $IF0 table T1
# ip route add $P0_NET dev $IF0 table T2

# ip route add $P2_NET dev $IF2 table T1
# ip route add $P1_NET dev $IF1 table T2

# ip route add 127.0.0.0/8 dev lo table T1
# ip route add 127.0.0.0/8 dev lo table T2
```

Pour l'instant notre configuration fonctionne sans balance de charge et il faut débrancher un lien pour basculer sur l'autre.

Pour répartir la charge de manière basique, on va jouer sur le paramètre 'weight':

```
# ip route add default scope global nexthop via $P1 dev $IF1 \
    weight 1 nexthop via $P2 dev $IF2 weight 1
```

Notez que cette balance de charge n'est pas parfaite car basée sur les routes et que celles-ci sont mises dans des caches.

Les routes vers les sites les plus souvent utilisés passeront toujours par le même fournisseur d'accès.

Imaginons que nous ayons deux interfaces et que nous voulions dédier une interface $IF1 ayant une adresse ip $IP1 pour le transit d'une machine $MACH ayant pour ip $IP2.

Tout d'abord, il faut créer une table de routage:

```
# ip rule add from $IP2 table $MACH
```

Il ne reste plus qu'à spécifier la paserelle par défaut de cette table:

```
# ip route add default via $IP1 dev $IF1 table $MACH
```

Bien sûr, ne pas oublier de vider le cache:

```
# ip route flush cache
```

Le Token Bucket Filter (TBF) est un gestionnaire de mise en file d'attente simple.

Il ne fait que laisser passer les paquets entrants avec un débit n'excédant pas une limite fixée administrativement.

L'envoi de courtes rafales de données avec un débit dépassant cette limite est cependant possible.

L'implémentation TBF consiste en un tampon (seau), constamment rempli par des éléments virtuels d'information appelés jetons, avec un débit spécifique (débit de jeton).

Le paramètre le plus important du tampon est sa taille, qui correspond au nombre de jetons qu'il peut stocker.

Chaque jeton entrant laisse sortir un paquet de la file d'attente et ce jeton est alors supprimé du seau.

L'association de cet algorithme avec les deux flux de jetons et de données, nous conduit à trois scénarios possibles :

- Les données arrivent dans TBF avec un débit **EGAL** au débit des jetons entrants. Chaque paquet entrant a son jeton correspondant et passe la file d'attente sans délai.
- Les données arrivent dans TBF avec un débit **PLUS PETIT** que le débit des jetons. Seule une partie des jetons est supprimée au moment où les paquets de données sortent de la file d'attente, de sorte que les jetons s'accumulent jusqu'à atteindre la taille du tampon.
- Les données arrivent dans TBF avec un débit **PLUS GRAND** que le débit des jetons. Ceci signifie que le seau sera dépourvu de jetons et provoquera l'arrêt de TBF pendant un moment. Ceci s'appelle « une situation de dépassement de limite » (overlimit situation). Si les paquets continuent à arriver, ils commenceront à être éliminés.

Même si vous n'aurez probablement pas besoin de les changer, TBF posséde les paramètres suivants:

- limit : c'est le nombre d'octets qui peuvent être mis en file d'attente en attendant la disponibilité de jetons.
  OU
- latency : c'est le temps maximal pendant lequel un paquet peut rester dans TBF. Ce dernier paramètre prend en compte la taille du seau, le débit, et s'il est configuré, le débit de crête (peakrate).

- burst : c'est la quantité maximale, en octets, de jetons dont on disposera simultanément. Plus les débits de mise en forme sont importants, plus le tampon doit être grand.

- mpu : c'est le nombre minimal de jetons à utiliser pour un paquet. Un paquet de taille nulle n'utilise pas une bande passante nulle (ethernet = 64 octets).

- rate : Le paramètre de la vitesse. Voir les remarques au-dessus à propos des limites !

- peakrate : c'est la vitesse à laquelle le seau est autorisé à se vider. Ceci est réalisé en libérant un paquet, puis en attendant suffisamment longtemps, pour libérer le paquet suivant. Le temps d'attente est calculé de manière à obtenir un débit égal au débit de crête.

Voici une configuration simple, mais très utile :

```
# tc qdisc add dev eth0 root tbf rate 220kbit latency 50ms burst 1540
```

Pourquoi est-ce utile ?

Si vous avez un périphérique réseau avec une grande file d'attente, comme un modem ou une Box, et que le dialogue se fait à travers une interface rapide, comme une interface ethernet, vous observerez que télécharger vers l'amont (uploading) dégrade complètement l'interactivité.

La ligne de commande au-dessus ralentit l'envoi de données à un débit qui ne conduit pas à une mise en file d'attente dans le modem. La file d'attente réside dans le noyau Linux, où nous pouvons lui imposer une taille limite.

Adapter la vitesse de 220Kbit par la vitesse du lien moins un petit pourcentage.

Stochastic Fairness Queueing (SFQ) est une implémentation simple de la famille des algorithmes de mise en file d'attente équitable.

Cette implémentation est moins précise que les autres, mais elle nécessite aussi moins de calculs tout en étant presque parfaitement équitable.

Le trafic est divisé en un grand nombre de files d'attente FIFO : une par conversation.

Le trafic est alors envoyé dans un tourniquet, donnant une chance à chaque session d'envoyer leurs données tour à tour.

Ceci empêche qu'une seule conversation étouffe les autres. SFQ est dit « Stochastic » car il n'alloue pas vraiment une file d'attente par session, mais utilse un algorithme qui divise le trafic à travers un nombre limité de files d'attente en utilisant un hachage.

A cause de ce hachage, plusieurs sessions peuvent finir dans le même seau, ce qui peut réduire de moitié les chances d'une session d'envoyer un paquet et donc réduire de moitié la vitesse effective disponible.

Pour empêcher que cette situation ne devienne importante, SFQ change très souvent son algorithme de hachage pour que deux sessions entrantes en collision ne le fassent que pendant un nombre réduit de secondes.

Configurer spécialement SFQ sur l'interface ethernet qui est en relation avec votre modem câble ou votre routeur DSL **est vain sans d'autres mises en forme du trafic !**

SFQ posséde les paramètres suivants:

- perturb : reconfigure le hachage une fois toutes les pertub secondes. S'il n'est pas indiqué, le hachage se sera **jamais** reconfiguré. 10 secondes est probablement une bonne valeur.

- quantum : c'est le nombre d'octets qu'un flux est autorisé à retirer de la file d'attente avant que la prochaine file d'attente ne prenne son tour. **Ne pas le configurez en dessous du MTU !**

Ci-dessous un exemple de configuration de SFQ :

```
# tc qdisc add dev eth0 root sfq perturb 10
```

Listons les détail de notre tourniquet:

```
# tc -s -d qdisc ls
      qdisc sfq 800c: dev ppp0 quantum 1514b limit 128p flows
128/1024 perturb 10sec
```

- 800c est un descripteur (handle) automatiquement assigné ;
- limit signifie que 128 paquets peuvent attendre dans la file d'attente ;
- Il y a 1024 « seaux de hachage » disponibles pour la comptabilité ;
- 128 peuvent être actifs à la fois ;
- Le hachage est reconfiguré toutes les 10 secondes.