TP Réseau : Routage Dynamique

Les TP d'acquisition des fondamentaux visent à construire un socle de connaissances de base, à appréhender des concepts, des notions et des modèles qui sont fondamentaux. Ce sont des étapes indispensables pour aborder d'autres apprentissages. Les TP sont conduits de manière fortement guidée pour vous placer le plus souvent dans une situation de découverte et d'apprentissage.

Objectifs

Les objectifs de ce TP sont de découvrir le routage dynamique avec le protocole RIP sous *GNU/Linux*.

TP Réseau : Routage Dynamique

Travail préparatoire

Installation du TP

Le TP3 est disponible dans l'archive `/home/user/sujets-tp/tp3-rip.tgz` :

```
host> cd /home/user/
host> tar zxvf sujets-tp/tp3-rip.tgz
host> cd /home/user/tp3-rip
host> lstart -s
```

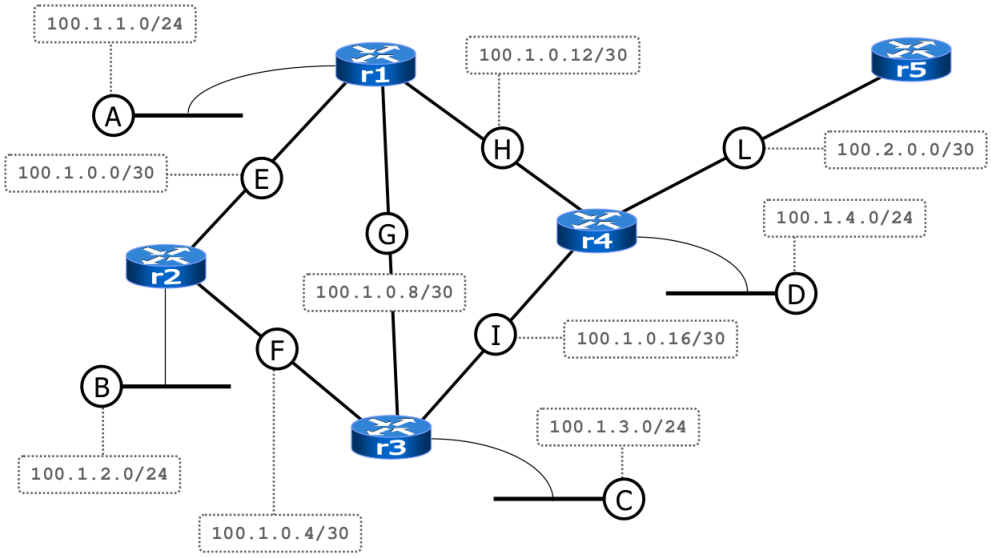
La maquette

Dans ce TP, la maquette NetKit est la suivante :

```
r1[0]="H"
r1[1]="G"
r1[2]="E"
r1[3]="A"

r2[0]="E"
r2[1]="F"
r2[2]="B"

r3[0]="F"
r3[1]="G"
r3[2]="I"
r3[3]="C"

r4[0]="L"
r4[1]="D"
r4[2]="I"
r4[3]="H"

r5[0]="L"

pc1[0]="A"
pc2[0]="B"
pc3[0]="C"
pc4[0]="D"
```

Le fichier lab.conf

Ce qui donne l'architecture suivante :



Cela peut représenter la topologie d'une AS avec **r5** comme routeur de bordure. Pour les tests, des **pc** ont été placés dans les domaines `A`, `B`, `C` et `D`.

Rappels

Le **routage** consiste à déterminer la route qu'un paquet doit prendre pour atteindre une destination. Cette tâche est réalisée au niveau de la couche RESEAU du modèle à couches. Dans cette couche, on utilise un adressage qui permet de spécifier à quel réseau appartient un équipement (hôte ou routeur) : dans le modèle TCP/IP, on utilise l'adressage IP.

Les fonctions au niveau de la couche RESEAU sont :

- **acheminer** (**hôte** ou **routeur**) : envoyer un paquet vers une destination (hôte ou routeur)
- **relayer** (**routeur**) : acheminer un paquet d'un réseau vers un autre réseau (*forwarding*)

Pour déterminer la route à prendre, le pilote IP utilise sa **table de routage** qui indique pour chaque destination (hôte, réseau ou sous-réseau), la route (interface ou passerelle) à prendre : routage de proche en proche.

Une **table de routage** indique pour chaque destination (hôte, réseau ou sous-réseau) la route (interface et passerelle) qu'il faut prendre.

Pour visualiser la table de routage, on utilisera la commande `route` ou `nestat -r`. L'option `-n` désactive pour ces deux commandes la résolution de noms.

Un équipement (hôte ou routeur) aura plusieurs possibilités pour construire sa table de routage :

- manuellement : routes statiques
- automatiquement par échange de routes (protocoles RIP, OSPF, ...) avec ses voisins ou par connaissance directe : routes dynamiques

Routage dynamique

Le routage dynamique est assuré par les routeurs eux-même en s'échangeant des informations sur leurs tables de routage. Cela nécessite donc la mise en œuvre d'un protocole de routage.

La reconfiguration manuelle des tables de routage en cas de défaillance d'un routeur peut vite devenir fastidieuse. La mise en œuvre un protocole de routage permet de mettre à jour dynamiquement les tables.

Il faut distinguer deux types de domaine de routage :

- IGP (*Interior Gateway Protocol*) : protocole de routage interne utilisé au sein d'une même unité administrative (AS, *Autonomous System*). Les protocoles les plus utilisés sont : RIP et OSPF.
- EGP (*Exterior Gateway Protocol*) : protocole de routage externe utilisé entre passerelles appartenant à des unités administratives différentes (AS). Le protocole utilisé est BGP (*Border Gateway Protocol*).

Pour IGP (*Interior Gateway Protocol*), il existe essentiellement deux types d'algorithme de routage :

- *Distant Vector* : Un protocole de type distance-vecteur sélectionne une route si elle est la plus courte en terme de distance (ou *Metric*) en se basant sur l'algorithme de *Belleman-Ford*. La distance est le nombre de routeurs (sauts ou « *hops* ») pour joindre une destination, chaque routeur ne connaît que son voisinage et propage les routes qu'il connaît à ses voisins. RIP (*Routing Information Protocol*) est un protocole basé sur un algorithme de type *Distant-Vector*, créé à l'Université de Berkeley (RFC 1058).
- *Link State* : Un protocole de type état de liens ou SPF (*Shortest Path First*) repose sur la recherche de la route la plus courte en se basant sur l'algorithme de *Dijkstra*. Cet algorithme implique une vision globale du réseau : chaque routeur ayant une vision topologique du réseau et l'état de l'ensemble des liens. OSPF (*Open Shortest Path First*) est un protocole ouvert de routage interne de type *Link-State*, élaboré par l'IETF (RFC 1247). Il existe aussi IS-IS qui est un protocole ISO à état de liens utilisé dans des grands réseaux de fournisseurs de services.

Pour ces deux protocoles, les routes émises par le routeur sont diffusées périodiquement en *multicast* (toutes les 30 s en *multicast* `224.0.0.9` pour RIPv2 et toutes les 10 s en *multicast* `224.0.0.5` pour OSPF).

RIPv2

RIP (*Routing Information Protocol*) est un protocole de routage IP de type *Vector Distance* (à vecteur de distances) s'appuyant sur l'algorithme de détermination des routes décentralisé *Bellman-Ford*.

À chaque route est associée une métrique (`M`) qui est sa distance exprimée en nombre de routeurs à traverser (sauts ou « *hops* »), `16` étant l'infini.

Avec RIPv2, chaque routeur envoie à ses voisins (adresse *multicast* `224.0.0.9`) ses informations de routage (les réseaux qu'il sait router et métriques associées) : toutes les 30 secondes systématiquement. Un message RIPv2 comprend un en-tête suivi de 1 à 25 enregistrement(s) de route (24 si un message d'authentification est requis).

Si un routeur reçoit d'un voisin ses informations de routage, il calcule les métriques locales des routes apprises (M → M+1), sélectionne les meilleures routes, en déduit sa table de routage et envoie à ses voisins ses nouvelles informations de routage (si elles ont changées), mécanisme *Triggered Updates* (mises à jour déclenchées).

Dans ce TP, on utilisera le protocole RIPv2 avec *GNU Zebra*, une suite logicielle de routage qui prend en charge plusieurs protocoles et permet de transformer une machine *Unix* en routeur.

GNU Zebra

GNU Zebra est une suite logicielle de routage qui prend en charge plusieurs protocoles et permet de transformer une machine *Unix* en routeur. *GNU Zebra* propose une configuration similaire aux systèmes *Cisco* IOS.

Cisco Systems est une entreprise informatique américaine spécialisée, à l'origine, dans le matériel réseau (routeurs et commutateurs ethernet), et depuis 2009 dans les serveurs. La spécificité de la gamme des produits *Cisco* est l'uniformité de son système d'exploitation, l'entreprise privilégiant pour la majorité de ses produits un **système d'exploitation propriétaire** nommé IOS.

Le logiciel libre *GNU Zebra* est toujours disponible selon les termes de la licence GNU GPL, mais n'est plus maintenu. Il a été remplacé par *Quagga*, un *fork* de *GNU Zebra*. *Quagga* est une suite de logiciels de routage implémentant les protocoles OSPF (v2 et v3), RIP (v1, v2 et v3), BGP (v4) et IS-IS pour les plates-formes de type *Unix*.

La configuration de *Zebra* est située dans `/etc/zebra` tandis que la configuration de *Quagga* est elle située dans `/etc/quagga`.

Le logiciel supportant plusieurs protocoles, il est possible de choisir les services qui les gèreront dans le fichier `daemons`. Pour le TP, on démarrera les *daemons* `zebra` et `ripd` :

```
zebra=yes
bgpd=no
ospfd=no
ospf6d=no
ripd=yes
ripngd=no
```

Le fichier `zebra.conf` permet de définir le nom et mot de passe à utiliser avec *Zebra/Quagga* (et eventuellement des routes statiques) :

```
! zebra configuration file
!
hostname zebra
password zebra
enable password zebra
!
! Static default route sample.
!
!ip route 0.0.0.0/0 203.181.89.241
!
log file /var/log/zebra/zebra.log
```

Le fichier `ripd.conf` permet de définir les routes (connectées directement aux interfaces) à diffuser en RIP (ici le `100.1.0.0/16`) :

```
! ripd configuration file
!
hostname ripd
password zebra
enable password zebra
!
router rip
redistribute connected
network 100.1.0.0/16
!
log file /var/log/zebra/ripd.log
```

Dans la maquette du TP, cette configuration est appliquée aux routeurs **r1**, **r2**, **r3** et **r4**. Au démarrage de la maquette, *Zebra/Quagga* n'est pas lancée.

Pour démarrer *Quagga*, il faut taper la commande :

```
r1:~# /etc/init.d/quagga start
```

Ensuite, il est possible de se connecter à *Zebra/Quagga* :

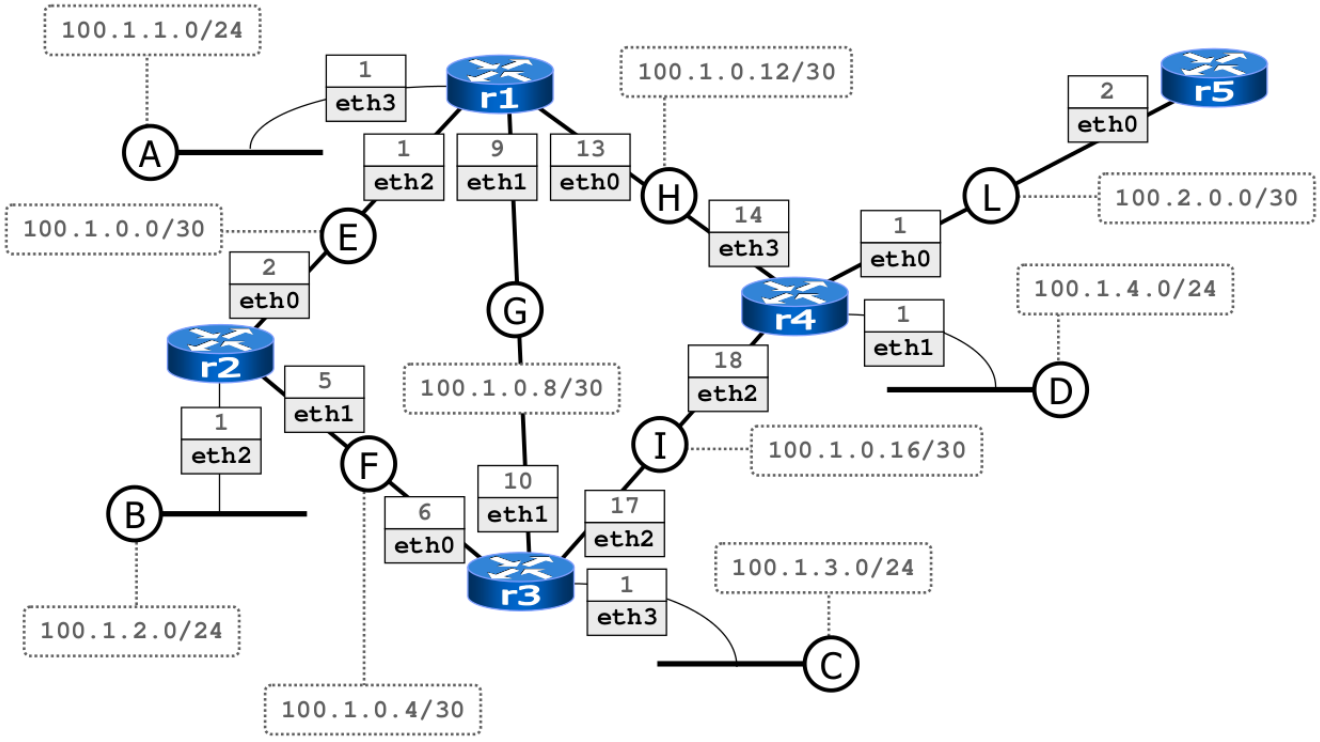
```
r1:~# telnet localhost zebra
Trying 127.0.0.1...
Connected to r1.

Hello, this is Quagga (version 0.99.10).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

User Access Verification
Password: zebra

zebra> ?
  echo     Echo a message back to the vty
  enable   Turn on privileged mode command
  exit     Exit current mode and down to previous mode
  help     Description of the interactive help system
  list     Print command list
  quit     Exit current mode and down to previous mode
  show     Show running system information
  terminal Set terminal line parameters
  who      Display who is on vty
zebra> list
  enable
  exit
  help
  list
  quit
  show debugging zebra
  show history
  show interface [IFNAME]
  show ip forwarding
  show ip prefix-list
  ...
  show ip protocol
  show ip route
  show ip route (bgp|connected|isis|kernel|ospf|rip|static)
  show ipv6 route X:X::X:X
  show ipv6 route X:X::X:X/M
  show ipv6 route X:X::X:X/M longer-prefixes
  show logging
  show memory
  show memory all
  show table
  show thread cpu [FILTER]
  show version
  show work-queues
```

```
  terminal length <0-512>
  terminal no length
  who
zebra> quit
r1:~#
```

Travail demandé

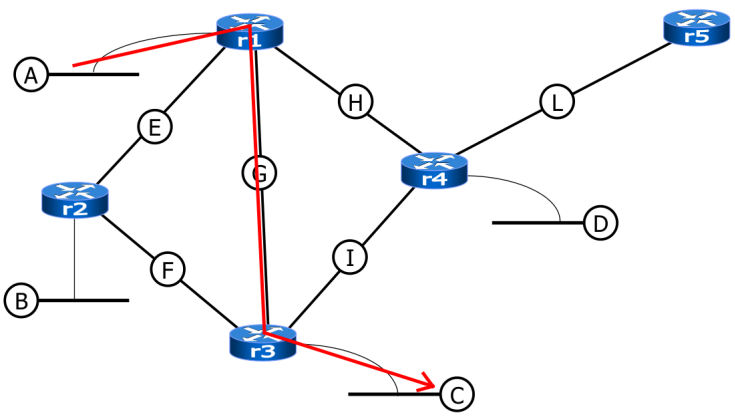
Routage dynamique

L'architecture réseau est la suivante :



Vérifier que les paramètres IP des interfaces de l'ensemble des machines sont bien configurés.

Question 1. Tester une communication depuis **pc1** vers **pc3** puis depuis **r1** vers **pc3**. Est-ce-que **pc3** est « joignable » ?



Question 2. Visualiser la table de routage de **r1**. Existe-t-il une route pour joindre le réseau de **pc3** ?

Question 3. Au lieu d'ajouter une route statique, activer *Zebra/Quagga* sur **r1** et **r3**.

Question 4. Après quelques secondes, tester une communication depuis **pc1** vers **pc3**. Est-ce-que **pc3** est maintenant « joignable » ? Si oui, tracer la route pour atteindre **pc3**.

Question 5. Visualiser la table de routage de **r1** et de **r3**. Existe-t-il de nouvelles routes ? Comment ont-elle été obtenues ?

Capturer avec `wireshark` les échanges de trames concernant RIP du domaine `G`.

```
host> vdump G | wireshark -i - -k &
```

Question 6. Quelles sont les routes diffusées par **r1** ? Quelles sont les routes diffusées par **r3** ? À quelle adresse de destination sont elles envoyées ?

Question 7. D'après la capture, quels sont les protocoles encapsulant RIP ? Quelles sont les informations de routage émises dans cette trame ? Justifier l'utilisation du champ TLL à `1`.

Zebra/Quagga

Question 8. Connectez vous à *Zebra/Quagga* depuis **r1**.

```
r1:~# telnet localhost zebra
Trying 127.0.0.1...
Connected to r1.

Hello, this is Quagga (version 0.99.10).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

User Access Verification
Password: zebra

zebra>
```

Question 9. Visualiser les paramètres de l'interface `eth1` et les routes.

```
zebra> show interface eth1
zebra> show ip route
```

Question 10. Connectez vous à `ripd` depuis **r1**.

```
r1:~# telnet localhost ripd
Trying 127.0.0.1...
Connected to r1.

Hello, this is Quagga (version 0.99.10).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

User Access Verification
Password: zebra
ripd>
```

Question 11. Visualiser les routes sur **r1**. Combien y-a-t-il de routes apprises par RIP ?

```
ripd> show ip rip
```

Question 12. Activer *Zebra/Quagga* sur **r2** et **r4**.

Question 13. Après quelques secondes, visualiser les routes sur **r1**. Combien y-a-t-il maintenant de routes apprises par RIP ?

Routage statique

Le réseau étudié est un réseau de bout car il n'a qu'une seule route avec le routeur externe **r5**. Par conséquent, les routes statiques sont suffisantes pour le connecter à Internet.

r5 a déjà une route pour le trafic entrant à destination du réseau `100.1.0.0/16` en passant par la passerelle (*gateway*) **r4** (`100.2.0.1`) :

```
r5:~# route
Kernel IP routing table
Destination    Gateway        Genmask         Flags Metric Ref  Use Iface
100.2.0.0      *              255.255.255.252 U   0      0        0 eth0
100.1.0.0      100.2.0.1      255.255.0.0     UG    0      0        0 eth0
```

Question 14. Tester la connectivité pour le trafic entrant depuis **r5** vers **pc1**.

Pour le trafic sortant, il suffit d'ajouter une **route par défaut** sur **r4** vers la passerelle (*gateway*) **r5** (`100.2.0.2`) puis de la **diffuser autres routeurs**.

Question 15. Ajouter une route par défaut sur **r4** vers la passerelle (*gateway*) **r5** (`100.2.0.2`).

Question 16. Configurer RIP sur **r4** pour propager la route par défaut.

```
r4:~# telnet localhost ripd
Trying 127.0.0.1...
Connected to r4.

Hello, this is Quagga (version 0.99.10).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

User Access Verification
Password: rebra

ripd> enable
Password: zebra

ripd# configure terminal
ripd(config)# router rip
ripd(config-router)# route 0.0.0.0/0
ripd(config-router)# quit
ripd(config)# quit
ripd# disable
ripd> exit
```

Après quelques instants, vérifier qu'une route par défaut vers **r4** a été ajoutée à la table de routage de **r1**.

```
r1:~# route | grep default
default         100.1.0.14      0.0.0.0         UG    2      0        0 eth0
```

Capturer avec `wireshark` les échanges de trames du domaine `L`.

```
host> vdump L | wireshark -i - -k &
```

Question 17. Tester la connectivité pour le trafic sortant depuis **pc1** vers une adresse Internet (`8.8.8.8` par exemple). Vérifier que **r5** reçoit bien les demandes d'echo (*echo resuest*).

Tolérance aux pannes

Relever la table de routage de **r1**. Effectuer un `traceroute` de **pc1** vers une adresse Internet (`8.8.8.8` par exemple) puis vers **pc4**.

```
pc1:~# traceroute 8.8.8.8
pc1:~# traceroute 100.1.4.2
r1:~# route
```

Question 18. Pour simuler une panne sur le routeur **r1**, désactiver son interface `eth0` (`ifconfig eth0 down`). Devez-vous reconfigurer les tables de routage des routeurs ?

Question 19. Après quelques instants, relever la table de routage de **r1**. Effectuer un `traceroute` de **pc1** vers une adresse Internet (`8.8.8.8` par exemple) puis vers **pc4**. Y-a-t-il eu des changements ? Expliquer.

Question 20. Après quelques instants, l'ensemble des reseaux sont-ils accessibles à partir de **pc1**. Que conclure sur RIP ?

Aller plus loin

Élimination des boucles *poison reverse* : les routes en provenance d'un voisin lui sont ré-annoncées avec une métrique infinie
Le *split horizon* : la métrique maximum est de 15 (une route égale à 16 sauts spécifie qu'elle est devenue inaccessible)
Utilisation de 3 minuteurs : *routing-update* (30 secondes +/- 0 à 5 secondes), *route-timeout* (180 secondes) et *route-flush* (120 secondes)
Mécanisme *Triggered Updates* (mises à jour déclenchées)