Faculté des Sciences & Techniques — Université de Limoges

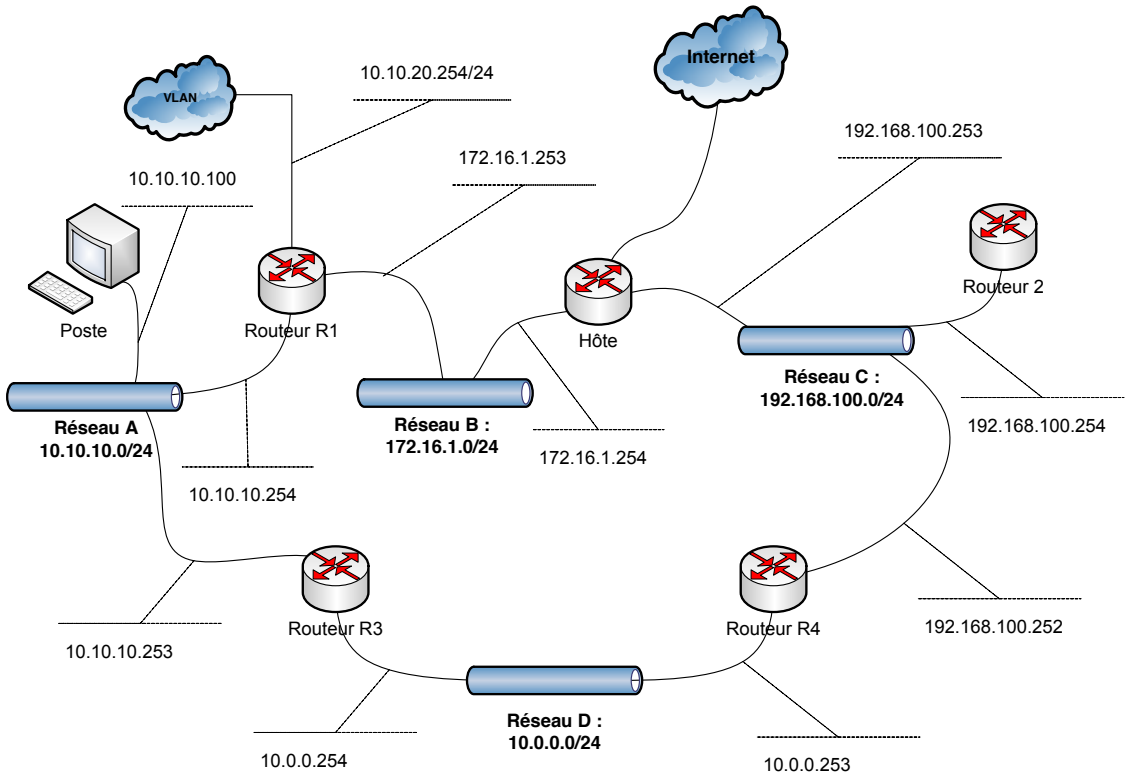
Master 1ère année

Infrastructure Réseaux

TP n°2

Routage dynamique avec RIP & OSPF

Routage à l'aide de RIP & OSPF — Extension du réseau d'interconnexion de la fiche de TP n°1



But de la simulation

- ▷ étendre le réseau proposé dans la fiche de TP n°1 ;
- ▷ configurer et déployer le protocole RIP dans cette nouvelle configuration ;
- ▷ étudier les paquets échangés du protocole RIP dans cette nouvelle configuration ;
- ▷ générer une panne due à la perte d'une liaison entre routeurs ;
- ▷ analyser les paquets échangés et le fonctionnement du protocole suite à la détection de la panne ;

Travail

1. vous finaliserez la configuration de la simulation étendue :
   - ◇ vous réaliserez la configuration de RIP sur « routeur3 » et « routeur4 » ;
   - ◇ vous vérifierez que les tables de routage de « routeur3 » et « routeur4 » sont bien configurées ;
2. Par quel chemin est accessible le réseau `10.0.0.0/24` depuis l'hôte ? Est-ce normal ?

```
xterm
Every 2,0s: ip r

default via 192.168.127.2 dev eth0  metric 100
10.0.0.0/24 via 192.168.100.252 dev bridge_reseauC  proto zebra  metric 2
10.10.10.0/24 via 172.16.1.253 dev bridge_reseauB  proto zebra  metric 2
10.10.20.0/24 via 172.16.1.253 dev bridge_reseauB  proto zebra  metric 2
172.16.1.0/24 dev bridge_reseauB  proto kernel  scope link  src 172.16.1.254
192.168.100.0/24 dev bridge_reseauC  proto kernel  scope link  src 192.168.100.253
```

Le réseau `10.0.0.0/24` est accessible par «Routeur 4», c-à-d le plus court chemin en nombre de saut ou de passage au travers d'un routeur.

3. Questions sur le fonctionnement de RIP :
   - quel est la table de routage de « routeur3 » ?

     xterm
     ```
     default via 10.10.10.254 dev eth0  proto zebra  metric 3
     10.0.0.0/24 dev eth1  proto kernel  scope link  src 10.0.0.254
     10.10.10.0/24 dev eth0  proto kernel  scope link  src 10.10.10.253
     10.10.20.0/24 via 10.10.10.254 dev eth0  proto zebra  metric 2
     172.16.1.0/24 via 10.10.10.254 dev eth0  proto zebra  metric 2
     192.168.100.0/24 via 10.0.0.253 dev eth1  proto zebra  metric 2
     ```
   - vous sniffererez les paquets RIP reçu sur « routeur1 » en provenance de « routeur3 » :

     ```
     root@routeur1:~/# tcpdump -nv -i eth2 '(udp and port 520) and host 10.10.10.253'
     ```
   - Existent-ils des différences entre la table de « routeur3 » et la table qu'il diffuse vers « routeur1 » ? Pourquoi ? *Routeur 3 ne diffuse pas les réseaux qu'il atteint par l'interface où il diffuse. Les différences sont dues à la technique du « split horizon ».*

4. Vous exécuterez depuis l'hôte, la commande :

   ```
   rezo@ishtar:~/$ traceroute 10.0.0.254
   ```

   Le résultat est-il correct ?

   xterm
   ```
   rezo@ishtar:~/RésAvII/RIP_OSPF$ traceroute 10.0.0.254
   traceroute to 10.0.0.254 (10.0.0.254), 30 hops max, 60 byte packets
    1  192.168.100.252 (192.168.100.252)  3.802 ms  3.751 ms  3.739 ms
    2  10.0.0.254 (10.0.0.254)  3.695 ms  3.686 ms  3.674 ms
   ```

   Oui, on passe bien par Routeur 4 pour atteindre l'interface « `10.0.0.254` » du Routeur 3.

5. Identifiez le nom des différentes interfaces et leur connexion pour « routeur1 », « routeur3 » et « routeur4 » ;

6. Installez les éléments de surveillance suivants :
   - sur l'hôte, vous surveillerez sa configuration de routage :

     ```
     rezo@ishtar:~/$ watch ip route
     ```
   - sur « routeur1 », vous continuerez votre surveillance avec `tcpdump` ;
   - sur « routeur3 », où vous substituerez le nom de l'interface identifiée au paramètre `INTERFACE` :

     ```
     root@routeur3:~/# tcpdump -nvX -i INTERFACE '(udp and port 520) and host 10.0.0.253'
     ```
   - vous créerez un « incident » de routage sur « Routeur4 » :

     ```
     root@routeur4:~/# ip link set dev eth1 down
     ```

     Vous exécuterez depuis l'hôte, la commande :

     ```
     rezo@ishtar:~/$ traceroute 10.0.0.254
     ```

     Le résultat est-il correct ?

     xterm
     ```
     rezo@ishtar:~/RésAvII/RIP_OSPF$ traceroute 10.0.0.254
     traceroute to 10.0.0.254 (10.0.0.254), 30 hops max, 60 byte packets
      1  192.168.100.252 (192.168.100.252)  3.802 ms  3.751 ms  3.739 ms
      2  * * *
     ```

     Non, on remarque que l'on a pas de réponse.
   - Que pouvez vous observez sur la capture par tcdump sur « Routeur3 » lorsque l'interface sur « Routeur4 » tombe ?

     xterm
     ```
     02:12:26.504157 IP (tos 0xc0, ttl 1, id 0, offset 0, flags [DF], proto UDP (17), length
     92)
         10.0.0.253.520 > 224.0.0.9.520:
        RIPv2, Response, length: 64, routes: 3
          AFI IPv4,          0.0.0.0/0 , tag 0x0000, metric: 16, next-hop: self
          AFI IPv4,       172.16.1.0/24, tag 0x0000, metric: 16, next-hop: self
          AFI IPv4,    192.168.100.0/24, tag 0x0000, metric: 16, next-hop: self
        0x0000:  45c0 005c 0000 4000 0111 8dcb 0a00 00fd  E..\..@.........
        0x0010:  e000 0009 0208 0208 0048 3e54 0202 0000  .........H>T....
        0x0020:  0002 0000 0000 0000 0000 0000 0000 0000  ................
        0x0030:  0000 0010 0002 0000 ac10 0100 ffff ff00  ................
        0x0040:  0000 0000 0000 0010 0002 0000 c0a8 6400  ..............d.
        0x0050:  ffff ff00 0000 0000 0000 0010            ............
     ```

     Les métriques sont à l'infini, c-à-d 16.

Et sur « Routeur1 » ?

xterm
```
02:12:26.504271 IP (tos 0xc0, ttl 1, id 0, offset 0, flags [DF], proto UDP (17), length
52)
    10.10.10.253.520 > 224.0.0.9.520:
   RIPv2, Response, length: 24, routes: 1
     AFI IPv4,   192.168.100.0/24, tag 0x0000, metric: 16, next-hop: self
02:12:28.852745 IP (tos 0xc0, ttl 1, id 0, offset 0, flags [DF], proto UDP (17), length
72)
    10.10.10.253.520 > 224.0.0.9.520:
   RIPv2, Response, length: 44, routes: 2
     AFI IPv4,        10.0.0.0/24, tag 0x0000, metric: 1, next-hop: self
     AFI IPv4,   192.168.100.0/24, tag 0x0000, metric: 16, next-hop: self
```

- Combien de temps faut-il pour que l'hôte se rende compte du problème ? *Longtemps...*

xterm
```
Every 2,0s: ip r

default via 192.168.127.2 dev eth0  metric 100
10.0.0.0/24 via 172.16.1.253 dev bridge_reseauB  proto zebra  metric 3
10.10.10.0/24 via 172.16.1.253 dev bridge_reseauB  proto zebra  metric 2
10.10.20.0/24 via 172.16.1.253 dev bridge_reseauB  proto zebra  metric 2
172.16.1.0/24 dev bridge_reseauB  proto kernel  scope link  src 172.16.1.254
192.168.100.0/24 dev bridge_reseauC  proto kernel  scope link  src
192.168.100.253
```

7. En réactivant l'interface sur « Routeur4 » :

```
root@routeur4:~/# ip l set dev eth1 up
```

Est-ce que l'hôte mets beaucoup de temps à se mettre à jour ? Pourquoi ?

8. Vous lancerez un « ping » depuis l'hôte :

```
rezo@ishtar:~/$ ping 10.0.0.254
```

Vous ferez de nouveau tomber l'interface sur « Routeur4 ».
Combien de paquets `icmp` sont perdus avant que la route ne soit rétablie ? *140 paquets, soient plus de deux minutes.*

xterm
```
rezo@ishtar:~/RésAvII/RIP_OSPF$ traceroute 10.0.0.254
traceroute to 10.0.0.254 (10.0.0.254), 30 hops max, 60 byte packets
 1  192.168.100.252 (192.168.100.252)  3.802 ms  3.751 ms  3.739 ms
 2  10.0.0.254 (10.0.0.254)  3.695 ms  3.686 ms  3.674 ms
rezo@ishtar:~/RésAvII/RIP_OSPF$ ping 10.0.0.254
PING 10.0.0.254 (10.0.0.254) 56(84) bytes of data.
64 bytes from 10.0.0.254: icmp_req=1 ttl=63 time=0.327 ms
64 bytes from 10.0.0.254: icmp_req=2 ttl=63 time=0.060 ms
64 bytes from 10.0.0.254: icmp_req=3 ttl=63 time=0.062 ms
64 bytes from 10.0.0.254: icmp_req=4 ttl=63 time=0.053 ms
64 bytes from 10.0.0.254: icmp_req=5 ttl=63 time=0.059 ms
64 bytes from 10.0.0.254: icmp_req=6 ttl=63 time=0.069 ms
64 bytes from 10.0.0.254: icmp_req=7 ttl=63 time=0.068 ms
64 bytes from 10.0.0.254: icmp_req=8 ttl=63 time=0.067 ms
64 bytes from 10.0.0.254: icmp_req=9 ttl=63 time=0.065 ms
64 bytes from 10.0.0.254: icmp_req=10 ttl=63 time=0.070 ms
64 bytes from 10.0.0.254: icmp_req=11 ttl=63 time=0.070 ms
64 bytes from 10.0.0.254: icmp_req=12 ttl=63 time=0.058 ms
64 bytes from 10.0.0.254: icmp_req=13 ttl=63 time=0.058 ms
From 192.168.100.253 icmp_seq=64 Destination Host Unreachable
From 192.168.100.253 icmp_seq=65 Destination Host Unreachable
From 192.168.100.253 icmp_seq=66 Destination Host Unreachable
From 192.168.100.253 icmp_seq=67 Destination Host Unreachable
From 192.168.100.253 icmp_seq=68 Destination Host Unreachable
From 192.168.100.253 icmp_seq=69 Destination Host Unreachable
From 192.168.100.253 icmp_seq=71 Destination Host Unreachable
...
From 192.168.100.253 icmp_seq=137 Destination Host Unreachable
From 192.168.100.253 icmp_seq=138 Destination Host Unreachable
From 192.168.100.253 icmp_seq=139 Destination Host Unreachable
From 192.168.100.253 icmp_seq=140 Destination Host Unreachable
64 bytes from 10.0.0.254: icmp_req=153 ttl=63 time=8.36 ms
64 bytes from 10.0.0.254: icmp_req=154 ttl=63 time=0.055 ms
64 bytes from 10.0.0.254: icmp_req=155 ttl=63 time=0.052 ms
```

Travail sur OSPF

1. Quel est le coût d'une laison par défaut ? *En ethernet c'est 10.*

2. Vous sniffierez les paquets échangés par OSPF à l'aide de la commande suivante (OSPF est associé au protocole 89 dans le datagramme IP) :

```
root@Routeur3:~/# tcpdump -nv -i eth1 'ip[9] == 89'
```

3. Sur l'hôte vous essaierez la commande `traceroute` :

```
root@ishtar:~/RésAvII/RIP_OSPF# traceroute 10.0.0.254
```

xterm

```
traceroute to 10.0.0.254 (10.0.0.254), 30 hops max, 60 byte packets
 1  192.168.100.252 (192.168.100.252)  0.045 ms  0.006 ms  0.005 ms
 2  10.0.0.254 (10.0.0.254)  1.795 ms  1.758 ms  1.738 ms
```

Quel est le chemin emprunté ?
Le plus court par Routeur 4.
Consultez la table de routage, est-ce conforme ?

xterm

```
Every 2,0s: ip r

default via 192.168.127.2 dev eth0  metric 100
10.0.0.0/24 via 192.168.100.252 dev bridge_reseauC  proto zebra  metric 20
10.10.10.0/24 via 172.16.1.253 dev bridge_reseauB  proto zebra  metric 20
10.10.20.0/24 via 172.16.1.253 dev bridge_reseauB  proto zebra  metric 20
172.16.1.0/24 dev bridge_reseauB  proto kernel  scope link  src 172.16.1.254
192.168.100.0/24 dev bridge_reseauC  proto kernel  scope link  src
192.168.100.253
```

4. Vous modifierez le coût associé au lien de « Routeur4 » sur son interface `eth1` :

```
ospfd> enable
ospfd# configure terminal
ospfd(config)# interface eth1
ospfd(config-if)# ospf cost 30
ospfd(config-if)# exit
ospfd(config)# exit
```

Vous ferez de même pour l'interface « `bridge_reseauC` » sur l'hôte.
Pourquoi doit-on faire la modification sur « Routeur4 » et l'hôte ? *Ce sont des liens passant par Ethernet et non pt-à-pt, il n'y a pas d'information partagée entre les deux routeurs.*
Est-ce que le `traceroute` donne le même résultat ?
Non, le chemin emprunte maitnenant le Routeur 1.
La table de routage a-t-elle été modifiée ?
Oui.
Est-ce conforme à la théorie ?
Oui, car la métrique est privilégiée pour le calcul de la route la plus courte par rapport au nombre de sauts.

5. Vous rétablirez les coûts sur « Routeur4 » et sur l'hôte.

   En recommançant une surveillance d'affichage de la table de routage de l'hôte :

```
$ watch ip route
```

   Vous ferez tomber l'interface `eth1` sur « Routeur4 ».

```
root@Routeur4:~/RésAvII/RIP_OSPF# ip link set dev eth1 down
```

   Est-ce que la modification est rapide ?
   Très.
   Vous réactiverez l'interface et vous lancerez un ping depuis l'hôte :

```
root@ishtar:~/RésAvII/RIP_OSPF# ping 10.0.0.254
```

   Puis de nouveau de désactiver l'interface : combien de paquets `icmp` sont perdus ?
   Aucun, on note seulement un ralentissement.