



RES 3 :
ROUTAGE STATIQUE,
DYNAMIQUE AVEC
RIP v1, RIP v2 et OSPF

Sommaire

TP2 : Objectifs du TP 2	2
1. Configuration IP des machines.....	2
1.1. Sous linux	2
1.2. Sous windows 2000	3
2. Routage statique	5
2.1. Configuration.....	5
2.1.1. Sous Linux	5
2.1.2. Sous Windows 2000.....	6
2.2. Gestionnaire de routage Windows : Routage et accès distant.....	7
3. Routage dynamique avec RIPv1	8
3.1. Configuration.....	8
3.1.1. Sous Linux	8
3.1.2. Sous Windows.....	8
3.2. Convergence	9
3.3. Mise en évidence de boucles dans RIP	10
TP3 : Objectifs du TP 3	11
1. Optimisation RIP : utilisation de RIP version 2	11
1.1. Installation et configuration des applications	11
1.1.1. Sous Linux	11
1.1.2. Sous Windows.....	14
2. Routage dynamique avec OSPF	16
2.1. Mise en place du routage.....	16
2.2. Routage dynamique avec OSPF	17
2.2.1. Configuration du routeur AT	17
2.2.2. Sous Linux	18
2.2.3. Sous Windows.....	20
2.3. Routage hiérarchique	20
2.4. Mise en place de la redondance de liens OSPF	21
Conclusion générale	22

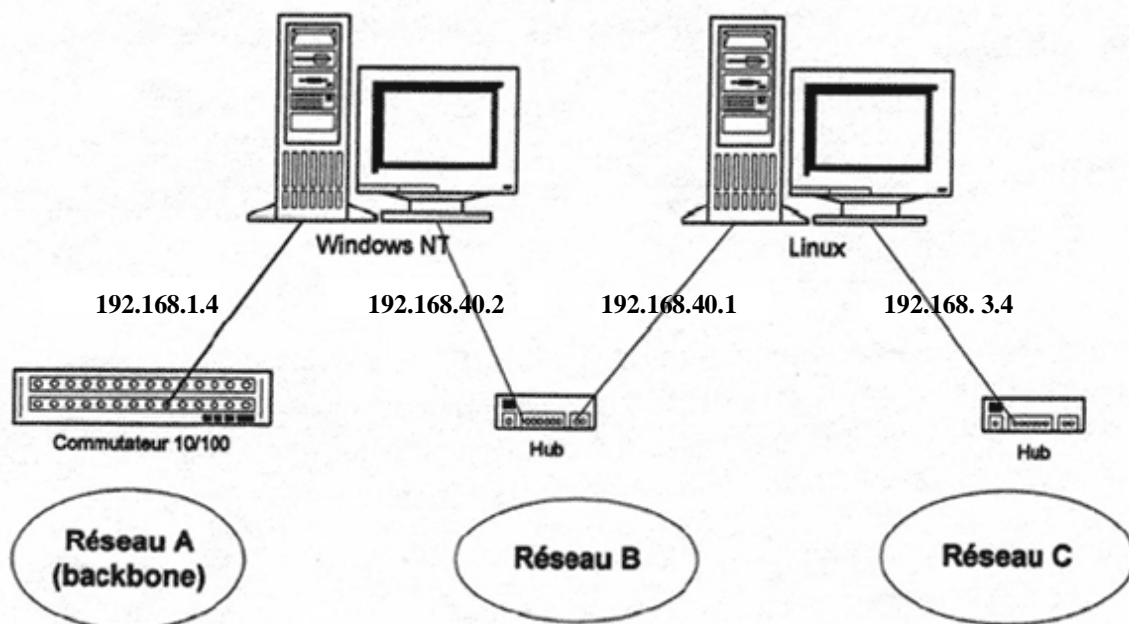
TP2 : Objectifs du TP 2

Le but de ce TP est de mettre en pratique nos connaissances sur le routage. Un routeur est une machine dédiée ou non qui permet à 2 réseaux (ou plus) différents de communiquer entre eux.

Nous allons au cours de ce TP apprendre à manipuler le routage statique et dynamique. Pour se faire, nous allons utiliser le protocole RIPv1 sur un inter-réseau composé de machines Linux et Windows 2000 NT. Puis nous examinerons dans les différents cas les échanges qui se produisent grâce aux analyseurs de protocole *Tcpdump* pour Linux et *Ethereal* pour Windows.

1. Configuration IP des machines

Nous avons configuré nos machines pour utiliser des IP de la classe C. 192.168.x.x
Nous étions la table N°4, le plan d'adressage réseau utilisé est donné sur le fascicule de TP :



1.1. Sous linux

Pour configurer les adresses IP sous Linux, nous avons utilisé la commande `ifconfig` :

```
$>ifconfig ethx adresse_IP netmask masque_de_sous_reseau up/down (exemple d'utilisation de la commande ifconfig sans argument)
```

```
ifconfig eth0 192.168.40.1 netmask 255.255.255.0  
ifconfig eth1 192.168.3.4 netmask 255.255.255.0
```

```

ServeurAdmin:/home/jhavez# ifconfig
eth0      Lien encap:Ethernet  HWaddr 00:30:F1:16:A0:93
          inet adr:192.168.1.125  Bcast:192.168.1.255  Masque:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:187939 errors:0 dropped:0 overruns:0 frame:0
          TX packets:27199 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 lg file transmission:1000
          RX bytes:24291866 (23.1 MiB)  TX bytes:13528572 (12.9 MiB)
          Interruption:11  Adresse de base:0xcc00

lo        Lien encap:Boucle locale
          inet adr:127.0.0.1  Masque:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:31745 errors:0 dropped:0 overruns:0 frame:0
          TX packets:31745 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 lg file transmission:0
          RX bytes:4325534 (4.1 MiB)  TX bytes:4325534 (4.1 MiB)

```

1.2. Sous windows 2000

En utilisant la propriété des cartes réseaux (TCP/IP)

Carte reliée à la Backbone (réseau A) :

Utiliser l'adresse IP suivante :

Adresse IP :	192 . 168 . 1 . 4
Masque de sous-réseau :	255 . 255 . 255 . 0
Passerelle par défaut :	. . .

Carte reliée au hub (réseau B) :

Utiliser l'adresse IP suivante :

Adresse IP :	192 . 168 . 40 . 2
Masque de sous-réseau :	255 . 255 . 255 . 0
Passerelle par défaut :	. . .

Pour vérifier la validité et le bon fonctionnement de ce réseau, nous avons fait appel à la commande ping. Nous avons testé dans un premier temps que les machines Windows et Linux communiquent bien entre elle directement via le hub du réseau B c'est-à-dire la communication entre les interfaces 192.168.40.1 et 192.168.40.2 :

Windows :

- * Ping 192.168.40.1 : machine Linux, réseau B
- * Ping 192.168.3.4 : impossible à atteindre

```
Envoi d'une requête 'ping' sur 192.168.3.4 avec 32 octets de données :
Délai d'attente de la demande dépassé.
Délai d'attente de la demande dépassé.
Délai d'attente de la demande dépassé.
Délai d'attente de la demande dépassé.

Statistiques Ping pour 192.168.3.4:
  Paquets : envoyés = 4, reçus = 0, perdus = 4 (perte 100%),
```

Linux :

- * Ping 192.168.40.2 : machine Windows, réseau B
- * Ping 192.168.1.4 : impossible à atteindre

Certaines adresses IP ne peuvent être atteintes car il n'existe pas de route directe entre 2 réseaux différents. Pour faire communiquer ces 2 réseaux, nous devons obligatoirement indiquer aux machines d'un réseau la *route* à suivre pour atteindre les autres machines de l'autre réseau. Sans ces indications, seules les machines situées sur le même réseau peuvent communiquer entre-elles (vérifications faites par des ping entre machines reliées au backbone ou entre stations linux.)

Nous pouvons donc conclure que nos réseaux A, B et C sont bien configurés et que toutes les communications aux seins d'un réseau sont faites, nous allons pouvoir mettre maintenant en place des routes statiques pour permettre une communication entre les différents réseaux.

2. Routage statique

2.1. Configuration

2.1.1. Sous Linux

Analysons tout d'abord la table de routage via la commande `route` sous Linux.

<i>Destination</i>	<i>Gateway</i>	<i>Genmask</i>	<i>Flags</i>	<i>Metric</i>	<i>Iface</i>
192.168.40.0	*	255.255.255.0	U	0	eth1
192.168.3.0	*	255.255.255.0	U	0	eth0

On peut remarquer que cette machine a accès à ses 2 réseaux adjacents via les interfaces eth1 et eth0. De plus, on peut noter que le flag est actif (U) dans les deux cas et que la métrique vaut 0. En effet, elle indique le nombre de sauts (hops) effectués pour atteindre une adresse destination en sachant qu'un saut est un passage par un routeur.

Sous Linux, l'ajout d'une route statique se fait par la commande `route add` :

```
route add -net 192.168.1.0 netmask 255.255.255.0 gw 192.168.40.2
```

Nous avons défini qu'un paquet devant atteindre le réseau 192.168.1.0/24 doit passer par la carte réseau de la machine Windows : 192.168.40.2.

Afin de vérifier que le réseau distant est maintenant accessible, il suffit de faire un ping et de vérifier la nouvelle table de routage :

```
ping 192.168.1.x
```

<i>Destination</i>	<i>Gateway</i>	<i>Genmask</i>	<i>Flags</i>	<i>Metric</i>	<i>Iface</i>
192.168.40.0	*	255.255.255.0	U	0	eth1
192.168.3.0	*	255.255.255.0	U	0	eth0
192.168.1.0	192.168.40.2	255.255.255.0	UG	1	eth1

La réponse de la commande `route` nous montre bien que la route a été ajoutée avec succès à la table de routage. De plus, le flag (G) indique que pour atteindre ce réseau il faut passer par un routeur (Gateway). C'est pourquoi la metric de cette route est à 1.

Exemple d'utilisation de la commande `route` (*machine Linux non configurée pour le tp*)

```
ServeurAdmin:/home/jhavez# route
Table de routage IP du noyau
Destination      Passerelle      Genmask         Indic Metric Ref     Use Iface
localnet         *               255.255.255.0  U        0      0       0 eth0
default          192.168.1.1    0.0.0.0        UG       0      0       0 eth0
```

2.1.2. Sous Windows 2000

Nous visualisons la table de routage via la commande `route print`.

<i>Destination Réseau</i>	<i>Adresse Passerelle</i>	<i>Masque Réseau</i>	<i>Métrieque</i>
192.168.40.0	192.168.40.2	255.255.255.0	1
192.168.1.0	192.168.1.4	255.255.255.0	1

Remarque : par rapport à la table de routage de linux, celle de Windows commence la numérotation des metrics à 1. Dans ce tableau nous voyons bien que la machine Windows peut communiquer au travers de ses 2 cartes réseau avec les réseaux A et B.

Nous voulons maintenant accéder à l'interface extérieure de la machine Linux, c'est-à-dire pouvoir accéder au réseau C (192.168.3.0). Pour se faire on va utiliser la commande décrite ci après :

```
route add/delete adresse_reseau mask masque adresseIP_passerelle
```

Nous allons pour se faire passer par l'interface de la machine Linux sur le réseau local B (192.168.40.1). Nous obtenons donc dans notre cas la commande suivante :

```
route add 192.168.3.0 mask 255.255.255.0 192.168.40.1
```

La commande `route print` indique alors :

<i>Destination Réseau</i>	<i>Adresse Passerelle</i>	<i>Masque Réseau</i>	<i>Métrieque</i>
192.168.40.0	192.168.40.2	255.255.255.0	1
192.168.1.0	192.168.1.4	255.255.255.0	1
192.168.3.0	192.168.40.1	255.255.255.0	2

Le réseau C est donc maintenant accessible à la machine Windows au travers de la machine Linux (192.168.40.1).

Exemple d'utilisation de la commande route print (*machine windows non configurée pour le tp*)

```
C:\WINDOWS\system32\cmd.exe
C:\Documents and Settings\Julien>route print
=====
Liste d'Interfaces
0x1 ..... MS TCP Loopback interface
0x2 ..4c 00 10 54 4c 30 ..... Carte réseau Fast Ethernet PCI Realtek RTL8139 F
amily #2 - Miniport d'ordonnancement de paquets
0x3 ..00 0e a6 2f 2a cb ..... 3Com Gigabit LOM (3C940) - Miniport d'ordonna
ment de paquets
=====
Itinéraires actifs :
Destination réseau      Masque réseau      Adr. passerelle    Adr. interface    Métrique
0.0.0.0                0.0.0.0            192.168.0.254      192.168.0.2        20
127.0.0.0              255.0.0.0          127.0.0.1          127.0.0.1          1
172.20.0.0             255.255.0.0        172.20.43.33       172.20.43.33       20
172.20.43.33          255.255.255.255    127.0.0.1          127.0.0.1          20
172.20.255.255         255.255.255.255    172.20.43.33       172.20.43.33       20
192.168.0.0            255.255.255.0      192.168.0.2        192.168.0.2        20
192.168.0.2           255.255.255.255    127.0.0.1          127.0.0.1          20
192.168.0.255         255.255.255.255    192.168.0.2        192.168.0.2        20
224.0.0.0              240.0.0.0          172.20.43.33       172.20.43.33       20
224.0.0.0              240.0.0.0          192.168.0.2        192.168.0.2        20
255.255.255.255       255.255.255.255    172.20.43.33       172.20.43.33       1
255.255.255.255       255.255.255.255    192.168.0.2        192.168.0.2        1
Passerelle par défaut : 192.168.0.254
=====
Itinéraires persistants :
Aucun
C:\Documents and Settings\Julien>
```

Suite à ça, nous analysons le bon fonctionnement du routage par des dump du trafic tcp/ip grâce à l'outil tcpdump et Ethereal.

Le moniteur réseau Ethereal nous présente les échanges entre nos deux machines. Il nous indique notamment le protocole utilisé (ICMP pour les ping) et les adresses de destination et de source de chaque paquet.

2.2. Gestionnaire de routage Windows : Routage et accès distant

Grâce à cet outil graphique, nous pouvons comme précédemment ajouter des routes statiques à la machine Windows. La différence étant qu'il est plus convivial de travailler en mode graphique qu'en mode console (selon les goûts de chacun).

Nous avons donc réussi, dans cette partie, à faire communiquer 2 réseaux bien différents en passant par un routage statique. Mais le problème du routage statique est qu'il faut reconfigurer chaque routeur à chaque changement de la topologie des réseaux. Dans la troisième partie de ce TP nous verrons donc comment, au travers d'outils, il est possible de laisser les routeurs s'auto-configurer (RIP).

3. Routage dynamique avec RIPv1

Le routage dynamique permet de nous éviter de rentrer les tables de routage à la main sur toutes les machines. Cela permet de gagner du temps lors de réseaux très importants ou encore lors de changements dans la topologie des réseaux.

Pour la réalisation de ce TP, nous avons utilisé le protocole RIP v1 (Routing Information Protocol).

3.1. Configuration

3.1.1. Sous Linux

Il faut au préalable réaliser plusieurs opérations afin de mettre en place notre configuration. Tout d'abord, il faut activer le routage des paquets grâce à la commande :
`#echo "1" > /proc/sys/net/ipv4/ip_forward.`

Puis il faut installer et démarrer le démon de routage avec la prise en charge d'interfaces multiples. Le démon de routage RIP sous Linux s'appelle *routed*.

Lors du démarrage de ce démon, et après quelques minutes, la table de routage de la machine Linux est automatiquement remplie par *routed*. Lors d'un `tcpdump`, nous voyons qu'un dialogue entre les différentes machines Linux s'est opéré à l'aide du protocole RIP pour s'informer des réseaux existants et des routes à prendre pour y arriver.

3.1.2. Sous Windows

La mise en place se réalise via l'outil graphique Routage et accès distant. Celui-ci nous permet de choisir entre plusieurs versions de protocoles ainsi que des options spécifiques à chaque version.

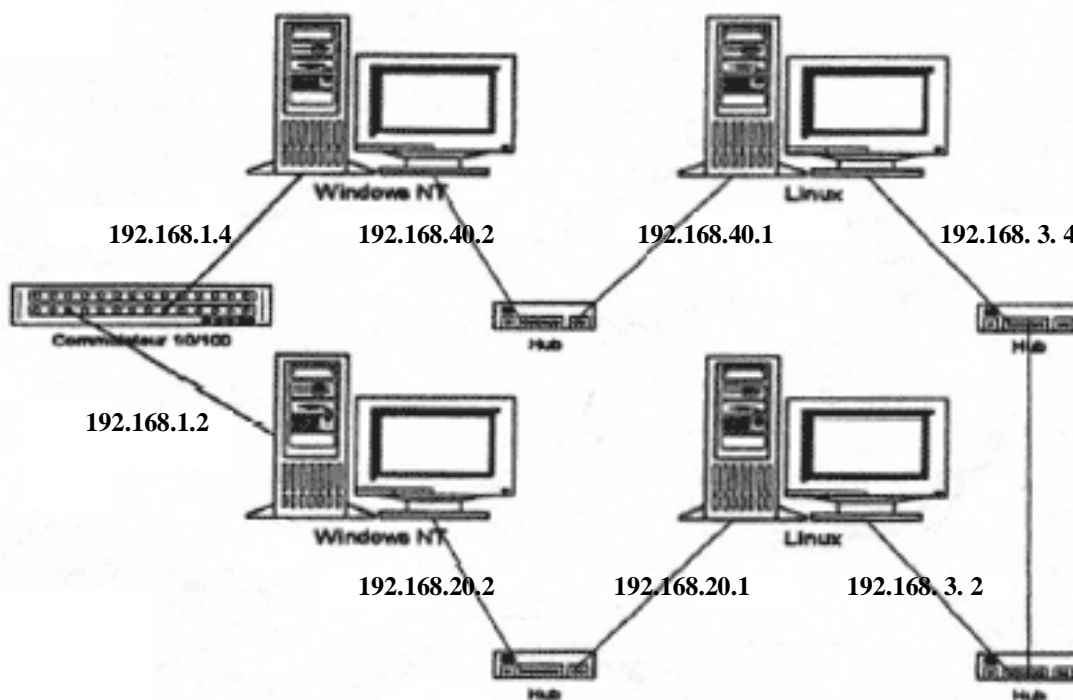
Une des options de RIP1 est le "routage silencieux". Il permet à notre machine Windows de récupérer les informations émises sur le réseau pour mettre à jour sa table de routage (protocole RIP) sans avoir elle-même à dialoguer avec d'autres routeurs.

Une fois les deux machines configurées, le routage fonctionne correctement après quelques minutes : la machine sous Linux accède au réseau A et celle sous Windows au réseau C.

Les outils comme `tcpdump` ou `Ethereal` nous informent sur les dialogues entre les différents routeurs.

3.2. Convergence

Pour vérifier la cohérence des tables d'adressage ainsi que le bon fonctionnement de RIPv1, nous formons avec nos voisins (table 2) un nouvel agrégat de réseaux. Suivant le schéma donné dans le fascicule du tp :



Lorsque les connexions ont été réalisées puis vérifiées, nous avons vérifié que les routeurs avaient effectivement bien mis à jour leur table de routage à l'aide du protocole RIPv1.

Grâce à tracert et tcpdump ou Ethereal nous avons vérifié que les paquets tcp/ip étaient effectivement bien routés au travers du réseau et nous avons pu voir quelles routes étaient attribuées pour chaque réseau.

Afin de simuler une panne de lien, nous avons débranché un câble tout en effectuant un ping continu vers une machine. Nous avons donc pu assister à la mise à jour automatique des tables de routage. Ceci a mis en évidence un temps de latence dû à la périodicité de vérification des routes valides. En effet, le protocole RIPv1 effectue plusieurs essais avant de constater l'erreur et de reconsidérer ses tables de routage (erreur lorsque : metric = 16). La route initiale est donc supprimée et remplacée par une route plus longue mais fonctionnelle.

3.3. Mise en évidence de boucles dans RIP

Si on désactive toutes les mesures de protection du protocole RIP :

- * Split horizon (découpage de l'horizon) : interdiction de renvoyer des tables de routage
- * Poison reverse (traitement anti-poison) : route infinie fixée à 16
- * Triggerred updates (mises à jour déclenchées) : émission des tables de routage dès que le réseau est modifié.

et qu'on modifie le temps de mise à jour sur les PC pour passer à 15s sur l'un des routeurs et 90s sur l'autre et on débranche ensuite un câble connecté sur une machine Linux.

Alors les routeurs font des boucles dans le réseau et ne passe plus forcément par le chemin le plus court (la metric associée au lien coupé augmente).

Toutefois, le protocole RIP coupe ce phénomène perpétuel lorsque la métrique atteint la valeur 16 en réinitialisant sa table de routage une fois cette valeur atteinte. Si l'on effectue une commande ping durant cette opération, on observera un temps de réponse plus élevé. Ce retard étant du à ce phénomène de boucle.

Nous pouvons donc dire que RIP est un protocole assez lent pour la convergence lors de la modification de la topologie du réseau et du re-calcule des nouvelles routes.

TP3 : Objectifs du TP 3

Le but de ce TP est d'étudier les protocoles de routage interne (RIPv2 et OSPF) et de voir les mécanismes d'apprentissage, de découverte de la topologie et de construction des tables de routage. Nous utiliserons l'application Zebra sous Linux pour la configuration RIP et OSPF.

1. Optimisation RIP : utilisation de RIP version 2

Avant de commencer toute configuration de routage, ou installation, il nous faut vérifier que les adresses IP sont celles que nous avons laissées la dernière fois. Ce qui n'était bien entendu pas le cas. Nous étions groupe 4 et nous devenons groupe 1.

Nos adresses Linux sont :

Nos adresses Windows sont :

Une fois ceci mis en place, et vérifié (ping des interfaces des machines), nous pouvons mettre en place des solutions de routage dynamique.

1.1. *Installation et configuration des applications*

1.1.1. **Sous Linux**

1.1.1.1. **Installation et paramétrage de zebra**

Zebra est un logiciel libre de routage disponible pour les systèmes linux et les principales instances de BSD. Son interface est fortement inspirée de la console des routeurs Cisco/IOS. Il fournit des implémentations de OSPF, RIP et BGP pour IPv4 et IPv6. C'est un logiciel modulaire qui fait appel à un démon (ospfd, ripd...) par protocole de routage, plus un démon (zebra) qui permet de configurer les adresses IP des interfaces, de spécifier des routes statiques..., et qui est nécessaire au fonctionnement des autres démons.

Il nous faut tout d'abord installer zebra via la commande : `apt-get install zebra`

Puis, il faut modifier son fichier de configuration avec `vi` (ou autre) afin de dé-commenter les mots de passes pour qu'ils soient valides. On pourra ainsi se connecter à la console de routage, et une fois le travail accompli, on pourra re-protéger le démon de routage en re-commentant ce mot de passe.

C'est également le démon zebra qui fait l'interface entre les protocoles de routage et le noyau, et qui gère le passage d'informations de routage d'un démon à l'autre. On lancera donc au moins deux démons pour avoir un protocole de routage opérationnel, par exemple `zebra -d` et `ospfd -d`.

La configuration se fait en utilisant un terminal telnet :

```
pc-gi-50:/etc/zebra# telnet localhost zebra
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.

```

```
Hello, this is zebra (version 0.92a).
Copyright 1996-2001 Kunihiro Ishiguro.

```

```
User Access Verification
Password: *****

```

Une fois connecté, il faut alors passer en mode super utilisateur et on configure les interfaces

```
Router> enable
Password: *****

```

```
zebra> enable
zebra # configure terminal
zebra (config)# interface eth0
zebra (config-if)# ip 192.168.3.1/24
zebra (config-if)# exit
zebra (config)# interface eth1
zebra (config-if)# ip 192.168.10.1/24
zebra (config-if)# write file

```

On vérifie l'installation des interfaces

```
Router# show interface

```

```
Interface lo
index 1 metric 1 mtu 16436 <UP,LOOPBACK,RUNNING>
inet 127.0.0.1/8
input packets 252, bytes 19009, dropped 0, multicast packets 0
input errors 0, length 0, overrun 0, CRC 0, frame 0, fifo 0, missed 0
output packets 252, bytes 19009, dropped 0
output errors 0, aborted 0, carrier 0, fifo 0, heartbeat 0, window 0
collisions 0

```

```
Interface eth0
index 2 metric 1 mtu 1500 <UP,BROADCAST,MULTICAST>
HWaddr: 00:60:08:29:c2:5c
inet 192.168.3.1/24 broadcast 192.168.3.255
input packets 139169, bytes 30193999, dropped 0, multicast packets 0
input errors 0, length 0, overrun 0, CRC 0, frame 0, fifo 0, missed 0
output packets 1225, bytes 83654, dropped 0
output errors 0, aborted 0, carrier 1129, fifo 0, heartbeat 0, window 0
collisions 0

```

```
Interface eth1
index 3 metric 1 mtu 1500 <UP,BROADCAST,RUNNING,MULTICAST>
HWaddr: 00:00:5a:9d:47:a5
inet 192.168.10.1/24 broadcast 192.168.10.255
input packets 175, bytes 21243, dropped 0, multicast packets 2
input errors 0, length 0, overrun 0, CRC 0, frame 0, fifo 0, missed 0
output packets 20, bytes 1562, dropped 0
output errors 0, aborted 0, carrier 0, fifo 0, heartbeat 0, window 0
collisions 0

```

1.1.1.2. RIPv2

Dans le but de déprotéger la configuration du routage RIP, on modifie le fichier de configuration de la même façon que celui de zebra : *ripd.conf*, ceci nous permettra de nous connecter pour le configurer on peut même vérifier que *routed* n'est plus présent grâce à *ps -afe* et on lance le démon suivant :

```
pc-gi-50:~# ripd -d -f /etc/zebra/ripd.conf
```

```
pc-gi-50:~# telnet localhost ripd
```

```
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
```

```
Hello, this is zebra (version 0.92a).
Copyright 1996-2001 Kunihiro Ishiguro.
```

Puis on active le mode de routage et les réseaux a router.

```
User Access Verification
```

```
Password:
```

```
ripd> enable
ripd# configure terminal
ripd(config)# router rip
ripd(config-router)# network 192.168.10.0/24
ripd(config-router)# network 192.168.3.0/24
ripd(config-router)# write file
Configuration saved to /etc/zebra/ripd.conf
```

On attend alors que les serveurs échangent leurs informations, si rien ne se passe, il faut relancer le serveur...

Pendant ce temps, on observe les paquets des différents réseaux qui se découvrent mutuellement

```
pc-gi-50:~# tcpdump
tcpdump: listening on eth0
15:29:46.680587 192.168.3.1.route > 224.0.0.9.route:  RIPv2-resp [items 4]:
{192.168.1.0/255.255.255.0}(3) {192.168.20.0/255.255.255.0}(4)[|rip] (DF)
[ttl 1]
15:29:59.197542 192.168.3.2.route > 224.0.0.9.route:  RIPv2-resp [items 4]:
{192.168.1.0/255.255.255.0}(3) {192.168.10.0/255.255.255.0}(4)[|rip] (DF)
[ttl 1]
15:30:03.238343 192.168.3.3.route > 224.0.0.9.route:  RIPv2-resp [items 4]:
{192.168.1.0/255.255.255.0}(3) {192.168.10.0/255.255.255.0}(4)[|rip] (DF)
[ttl 1]
15:30:25.690577 192.168.3.1.route > 224.0.0.9.route:  RIPv2-resp [items 4]:
{192.168.1.0/255.255.255.0}(3) {192.168.20.0/255.255.255.0}(4)[|rip] (DF)
[ttl 1]
15:30:27.248280 192.168.3.3.route > 224.0.0.9.route:  RIPv2-resp [items 4]:
{192.168.1.0/255.255.255.0}(3) {192.168.10.0/255.255.255.0}(4)[|rip] (DF)
[ttl 1]
```

```

15:30:40.207532 192.168.3.2.route > 224.0.0.9.route:  RIPv2-resp [items 4]:
{192.168.1.0/255.255.255.0}(3)  {192.168.10.0/255.255.255.0}(4)[|rip]  (DF)
[ttl 1]
15:30:43.700567 192.168.3.1.route > 224.0.0.9.route:  RIPv2-resp [items 4]:
{192.168.1.0/255.255.255.0}(3)  {192.168.20.0/255.255.255.0}(4)[|rip]  (DF)
[ttl 1]

```

Au final, on obtient alors la table de routage suivante :

```

pc-gi-50:~# route -n
Kernel IP routing table
Destination      Gateway          Genmask         Flags Metric Ref    Use Iface
192.168.20.0     192.168.10.2   255.255.255.0  UG    4      0      0 eth1
192.168.3.0      0.0.0.0         255.255.255.0  U      0      0      0 eth0
192.168.1.0      192.168.10.2   255.255.255.0  UG    3      0      0 eth1
192.168.30.0     192.168.10.2   255.255.255.0  UG    4      0      0 eth1
192.168.10.0     0.0.0.0         255.255.255.0  U      0      0      0 eth1
192.168.40.0     192.168.10.2   255.255.255.0  UG    4      0      0 eth1

```

On voit bien que tout passe par eth1 (vers Windows) sauf pour le réseau sur lequel se trouve eth0 (logique !)

Si on coupe une liaison (par exemple le câble liant la station Linux à son Windows de même groupe), le réseau se réorganise et une nouvelle route est choisie pour joindre la machine dont le lien à été perdu. Ici toutes les routes sont modifiées car les paquets transitaient auparavant via le routage de Windows, dorénavant les routes utilisées seront les routages des stations Linux, et les station Windows resteront toutes accessibles.

```

pc-gi-50:~# route -n
Kernel IP routing table
Destination      Gateway          Genmask         Flags Metric Ref    Use Iface
192.168.60.0     192.168.3.3     255.255.255.192 UG    5      0      0 eth0
192.168.20.0     192.168.3.3     255.255.255.0  UG    5      0      0 eth0
192.168.3.0      0.0.0.0         255.255.255.0  U      0      0      0 eth0
192.168.1.0      192.168.3.2     255.255.255.0  UG    4      0      0 eth0
192.168.30.0     192.168.3.2     255.255.255.0  UG    5      0      0 eth0
192.168.10.0     0.0.0.0         255.255.255.0  U      0      0      0 eth1
192.168.40.0     192.168.3.3     255.255.255.0  UG    5      0      0 eth0

```

On voit bien que tout passe par eth0 (vers d'autres Linux) sauf pour le réseau sur lequel se trouve eth1 (logique !)

1.1.2. Sous Windows

De la même façon que dans le TP2, nous avons mit en place le protocole RIPv2 sur notre machine Windows via le moniteur de Routage et accès à distance. Pour le cas présent, nous avons sélectionné uniquement ce protocole en mode multicast contrairement au mode broadcast du TP2.

Après avoir réalisé plusieurs ping pour tester la bonne configuration du réseau, nous avons visualisé le moniteur réseau pour voir les différentes routes présentes.

Si on analyse cette table de routage on peut s'apercevoir qu'il y a beaucoup de routes présentées et on ne prendra pas en compte les adresses de destination correspondant aux boucles. On peut s'apercevoir que l'on peut accéder aux réseaux suivants :

- Réseau A (BackBone) : 192.168.1.0 avec l'adresse Windows 192.168.1.1
- Réseau local B : 192.168.10.0 avec l'adresse Windows 192.168.10.254
- Réseau C : 192.168.3.0
- Réseaux locaux :
 - 192.168.20.0
 - 192.168.30.0
 - 192.168.40.0

On peut donc dire que le protocole RIP nous a bien permis d'accéder à tous les réseaux locaux des autres tables (avec une métrique de 3) via le réseau Backbone et plus précisément via chaque interface Windows du réseau A. Ceci traduit bien le bon fonctionnement et le bon paramétrage de notre réseau.

Nous pouvons donc conclure que RIPv2 propose un peu la même configuration que RIPv1 avec les mêmes inconvénients, par exemple convergence et boucles. Toutefois, RIPv1 fonctionnait en broadcast ce qui est un peu plus gênant puisque la diffusion se fera sur tout le réseau et par conséquent il y aura des collisions et encombrement du réseau.

A l'inverse RIPv2 fonctionne en multicast et l'on peut remarquer que sur la table de routage précédente on peut voir l'adresse de multicast utilisée : 224.0.0.0. Cette adresse est utilisée sur les deux interfaces de la machine Windows, elle caractérise une diffusion sur un réseau de classe D. Outre cette différence au niveau de la diffusion entre ces deux protocoles, RIPv2 possède en plus une authentification des messages.

RIPv2 constitue une amélioration de RIPv1, et il est important de savoir lesquelles afin de mieux en tirer partie :

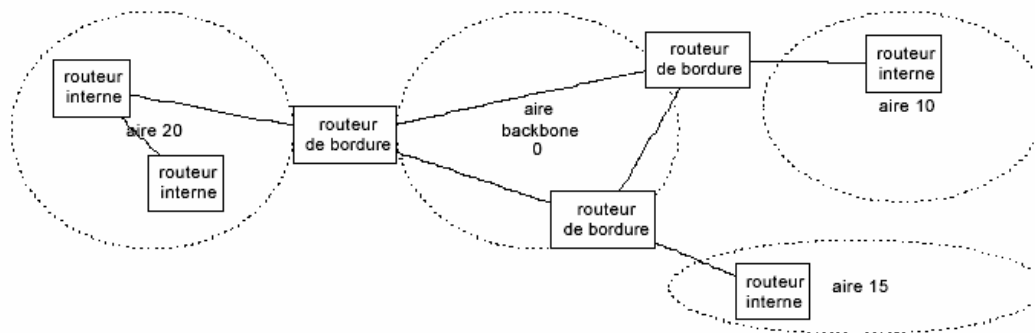
- Diffusion des masques de sous réseaux associés aux adresses réseaux (RIPv1 n'utilisait que les masques réseau par défaut).
- Utilisation d'une adresse de multicast pour diffuser les vecteurs de distance au lieu de l'adresse de broadcast ; ce qui réduit l'encombrement sur le réseau.
- Support de l'authentification en transportant un mot de passe crypté avec MD5 (non vu ici).
- Interopérabilité entre protocoles de routage en diffusant des routes apprises à partir d'autres protocoles.

Un des problèmes de ce protocole est qu'il est « bavard ». Toutes les 30 secondes il se manifeste ce qui génère du trafic.

2. Routage dynamique avec OSPF

OSPF est le protocole de routage interne de référence. C'est un protocole de type « état des liens » (link states). Il fonctionne donc en inondant le réseau de paquets donnant l'état des liens sur le réseau, ce qui permet à chaque routeur de construire la topologie du réseau.

Ensuite chaque routeur exécute en local un calcul des plus courts chemins – en utilisant en général un algorithme de Dijkstra – vers chaque destination (d'où l'acronyme : Open Shortest Path First). OSPF n'a pas les défauts de RIP, et en particulier converge aussi rapidement en cas d'apparition que de disparition d'un lien (ou augmentation du coût associé : problème du comptage à l'infini). Par contre OSPF est considérablement plus complexe que RIP même en ce qui concerne une utilisation basique. Une des raisons de cette complexité est qu'OSPF gère deux niveaux de hiérarchie, ce qui permet de l'utiliser dans des réseaux d'une centaine de routeurs.

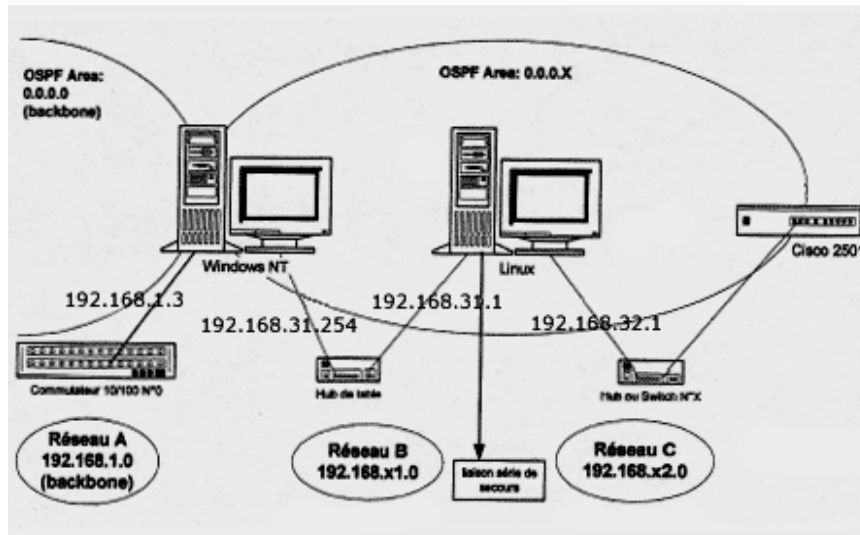


Exemple de réseau organisé avec OSPF

Dans ce type de routage, nous utilisons un **routeur AT** (5 interfaces Ethernet 10BaseT et 1 interface console pour y connecter un terminal). Le protocole OSPF (Open Shortest Path First) est un protocole plus performant que RIP puisqu'il analyse l'état du réseau et des liaisons afin de choisir la route la plus optimale. De plus, OSPF découpe l'ensemble du réseau en zones dites « area », 0.0.0.1 pour notre binôme.

2.1. Mise en place du routage

Etant la table de travail N°1 et grâce au plan d'adressage précédent, nous avons donc pu réaliser le réseau suivant :



Le routeur Cisco 2501 étant hors d'usage, nous avons utilisé un routeur AT.

Avant de commencer toute manipulation, il nous faut changer nos adresses IP en suivant le plan d'adressage ci-dessous (x notre numéro de table) :

Windows : La première carte réseau sera connectée au backbone via le commutateur 10/100. Son adresse IP sera 192.168.1.1/24 (réseau A). L'autre carte réseau aura pour IP 192.168.11.254/24 et sera reliée à la station linux via un hub de table (réseau B).

Linux : La première carte réseau sera connectée au routeur AT. Son adresse IP sera 192.168.12.1/24 (réseau C). L'autre carte réseau aura pour IP 192.168.11.1/24 et sera reliée à la station windows via un hub de table (réseau B).

2.2. Routage dynamique avec OSPF

2.2.1. Configuration du routeur AT

On se connecte sur le routeur via le port série RS232. On utilise la console HyperTerminal de windows (9600bps, 8 bits, pas de contrôle de flux, 1 bit de parité).

Le login est « manager » et le mot de passe « admin ».

`show ip interface` permet de voir la configuration IP des interfaces.

Configuration de notre plan d'adressage (affectation des IP aux ports)

```
add ip int=vlan1 ip=192.168.12.254 mask=255.255.255.0
add ip int=vlan2 ip=192.168.22.254 mask=255.255.255.0
add ip int=vlan3 ip=192.168.32.254 mask=255.255.255.0
add ip int=vlan4 ip=192.168.42.254 mask=255.255.255.0
```

Définition des zones gérées par OSPF:

```
add ospf area=0.0.0.1 stubarea=no
add ospf area=0.0.0.2 stubarea=no
add ospf area=0.0.0.3 stubarea=no
```

```
add ospf area=0.0.0.4 stubarea=no
```

```
add ospf range=192.168.12.0 mask=255.255.255.0 area=0.0.0.1
add ospf range=192.168.22.0 mask=255.255.255.0 area=0.0.0.2
add ospf range=192.168.32.0 mask=255.255.255.0 area=0.0.0.3
add ospf range=192.168.42.0 mask=255.255.255.0 area=0.0.0.4
```

```
add ospf interface=eth0 area=backbone
add ospf interface=vlan1 area=0.0.0.1
add ospf interface=vlan2 area=0.0.0.2
add ospf interface=vlan3 area=0.0.0.3
add ospf interface=vlan4 area=0.0.0.4
```

Permet au routeur d'échanger des informations ospf avec les autres routeurs. Mais pour cela, il faut avoir configuré les interfaces et les zones ospf, ce qui est fait.

```
enable ospf
```

Pour vérifier le bon fonctionnement d'ospf sur le routeur :

```
show ospf route
```

Pour afficher la table de routage du routeur :

```
show ip route
```

2.2.2. Sous Linux

Tout d'abord il faut arrêter le démon de routage RIPv2 :

```
ripd> enable
ripd# configure terminal
ripd(config)# no router rip
ripd(config-router)# write mem
Configuration saved to /etc/zebra/ripd.conf
```

Et ne pas oublier de commenter les mots de passes dans le fichier de configuration de ripd afin que celui-ci ne soit pas re-configurable par quelqu'un d'autre.

Puis il faut reconfigurer les interfaces sous zebra afin qu'elles aient les nouvelles adresses. On se reconnecte donc à zebra via telnet.

Une fois connecté, il faut alors passer en mode super utilisateur et on configure les interfaces

```
zebra> enable
zebra # configure terminal
zebra (config)# interface eth0
zebra (config-if)# ip 192.168.12.1/24
zebra (config-if)# exit
zebra (config)# interface eth1
zebra (config-if)# ip 192.168.11.1/24
zebra (config-if)# write file
```

On lance alors le démon de routage OSPF après avoir modifié le fichier de configuration pour activer la connexion telnet avec mot de passe :

```
pc-gi-50:/etc/zebra# ospfd -d -f /etc/zebra/ospfd.conf
```

Puis, de la même manière que pour zebra, on ouvre une session telnet pour configurer le démon de RIPv2

```
pc-gi-50:/etc/zebra# telnet localhost ospfd
```

Puis on active le mode de routage et les zones a router

```
ospfd > enable
ospfd# configure terminal
ospfd(config-router)# router ospf
ospfd(config-router)# network 192.168.11.0/24 area 0.0.0.1
ospfd(config-router)# network 192.168.12.0/24 area 0.0.0.1
ospfd(config-router)# write memory
```

Un tcpdump permet de voir les échanges d'informations entre les zones

```
16:33:20.861042 192.168.11.1 > 192.168.11.2: OSPFv2-dd 32: rtrid
192.168.12.1 area 0.0.0.1 E I/M/MS S 38833640 [ttl 1]
16:33:20.861579 192.168.11.2 > 192.168.11.1: OSPFv2-dd 152: area 0.0.0.1 E
S 38833640
16:33:20.863892 192.168.11.1 > 192.168.11.2: OSPFv2-dd 52: rtrid
192.168.12.1 area 0.0.0.1 E M/MS S 38833641 [ttl 1]
16:33:20.864172 192.168.11.2 > 192.168.11.1: OSPFv2-dd 32: area 0.0.0.1 E
S 38833641
16:33:20.864245 192.168.11.2 > 192.168.11.1: OSPFv2-ls_req 36: area
0.0.0.1
16:33:20.867707 192.168.11.1 > 192.168.11.2: OSPFv2-dd 32: rtrid
192.168.12.1 area 0.0.0.1 E MS S 38833642 [ttl 1]
16:33:20.867962 192.168.11.2 > 192.168.11.1: OSPFv2-dd 32: area 0.0.0.1 E
S 38833642
```

En effet, contrairement à RIP, OSPF est un protocole de routage non plus pour de simples réseaux, mais pour des interconnexions de réseaux. On ne définit ici non plus les réseaux à router, mais les zones (areas) à interconnecter entre elles.

Voici la table de routage alors obtenue :

```
pc-gi-50:~# route -n
Kernel IP routing table
Destination      Gateway         Genmask         Flags Metric Ref    Use    Iface
192.168.12.0     0.0.0.0        255.255.255.0  U        0     0      0     eth0
192.168.11.0     0.0.0.0        255.255.255.0  U        0     0      0     eth1
```

On peut ensuite modifier la priorité sur les routeurs afin que Windows soit toujours désigné sur 192.168.11.0, pour cela on augmente la priorité du routeur Linux en se connectant au démon ospfd :

```
ospfd(config)# interface eth1
ospfd(config-if)# ospf priority 2
ospfd(config-if)# write mem
```

2.2.3. Sous Windows

Nous avons tout d'abord supprimé le protocole RIP des deux interfaces de notre machine. De plus, nous avons mis en place le nouveau protocole OSPF dans le RAD afin de créer deux zones (« area ») OSPF : une première au niveau du backbone (zone 0.0.0.0) et une autre au niveau du réseau local B et du réseau C (zone 0.0.0.1).

Pour la zone 0.0.0.0, nous sélectionnons l'adresse 192.168.1.1 en précisant un réseau en mode diffusion dans la zone à cocher « Type de réseau ».

Nous réalisons une procédure similaire afin d'activer ce protocole sur l'autre zone (0.0.0.1) en précisant que cette zone comprend à la fois notre réseau local B (192.168.11.0) et le réseau C (192.168.12.0).

Puis, il faut préciser dans un second temps quelle est l'adresse IP utilisée pour le routeur AT, tout en indiquant bien les zones prise en compte (pour nous 0.0.0.0 et 0.0.0.1).

Une fois ces zones activées et après quelques instants de mise en place du protocole, on obtient une table de routage complète.

2.3. Routage hiérarchique

Le routage hiérarchique devient nécessaire lorsque le sous réseau est trop volumineux (en nombre de routeurs) :

Si vous avez tellement de routeurs qu'il vient difficile de les gérer dans une même zone (par exemple, la sortie de `show ip ospf database` devient trop longue). Certains routeurs très lents ont également du mal dès que la zone (et donc les informations à traiter) devient trop grande.

On dit souvent qu'il est recommandé de ne pas mettre plus de 50 ou 100 routeurs (selon leurs performances) par zone.

- Ou s'il existe plusieurs services dans l'organisation, que ces services disposent de compétence en routage et souhaitent un minimum d'autonomie. OSPF permet, dans une certaine mesure, d'isoler la backbone de ces zones.

Le poste NT est le routeur de bordure de chacune des aires délimitées. Son rôle peut être mis en évidence grâce à l'option « zone de stub » dans les propriétés de la zone OSPF qui permet de limiter la vision vers et depuis l'extérieur de l'intérieur de la zone.

Une fois ce paramètre activé sur les postes Linux, nous avons constaté la disparition de ces routeurs dans nos tables de routages.

Nous avons ainsi accès aux différentes zones uniquement via les routeurs frontières (postes sous Windows). Pour accéder directement aux postes sous Linux depuis l'extérieur, alors que

ceux-ci avaient l'option de zone « stub area » appliquée, nous devons créer une passerelle manuellement.

Nous pouvons remarquer que nous n'avons pas pu réaliser la mise en place des redondances de liens OSPF car le réseau et certains ordinateurs ne fonctionnaient pas dans les bonnes conditions.

2.4. *Mise en place de la redondance de liens OSPF*

Cette partie n'a pas eu être traitée faute d'outils pour la mettre en place sous les debian de la salle de TP.

Conclusion générale

Le TP2 et TP3 nous ont permis de comprendre le fonctionnement des protocoles RIPv1, RIPv2 ainsi que OSPF. Nous avons pu les mettre en pratique sous environnements Windows et Linux afin de voir les avantages et les inconvénients de ceux-ci.

D'une manière générale, le routage statique nous a montré ses limites puisque celui-ci est entièrement réalisé manuellement et ne permet par conséquent aucune ou peu d'évolution à l'inverse du routage dynamique.

RIP	OSPF
Calcule le coût en fonction du nombre de sauts On peut augmenter artificiellement un nombre de sauts pour privilégier une autre route	Calcule la route en fonction du nombre de sauts, du débit, du coût du lien
Diffuse des messages toutes les 30s Surcharge importante de trafic sur grands réseaux	Diffusion limitée Moins de surcharge
En cas de panne de route, il faut échanger toute la table de routage (long sur réseaux grande distance)	Plus rapide à trouver de nouvelles routes

Les différentes versions de RIP permettent de concevoir un système de routage simple et relativement efficace dans la majorité des cas même si celui se focalise seulement sur la métrique. Tandis que le protocole OSPF permet, par son fonctionnement et ses options, de pouvoir créer des architectures réseaux assez complexes, puisque maintenant nous interconnectons des zones composées de plusieurs réseaux. Il semble donc que OSPF soit un protocole plus performant que RIP s'adaptant aux réseaux et à ses évolutions.