Protocole de routage

1 Routage

1.1 Topologie d'un réseau

Le **routage** est le mécanisme par lequel des chemins sont sélectionnés dans un réseau pour acheminer les données (ou paquets) d'un expéditeur jusqu'à leurs destinataires. Les **routeurs** sont les machines dont le rôle est de transiter les paquets d'une interface réseau vers une autre. L'interconnexion des routeurs forment la **topologie du réseau**.

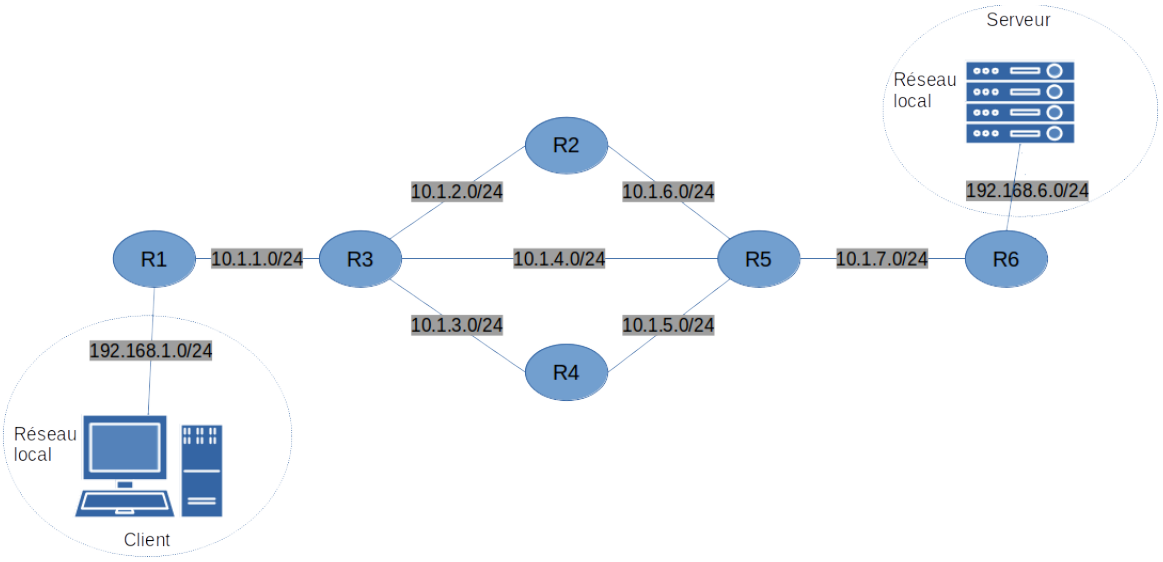


FIGURE 1 – Topologie d'un réseau

Sur le shéma de la figure 1, il y a six routeurs : les routeurs `R1` et `R6` sont des routeurs d'accès, les quatres autres routeurs `R2`, ..., `R5` sont des routeurs internes.
Les adresses IP utilisées par les machines sont indiquées sous la forme sous-réseau/masque :

- a.b.c.d/8 indique que les adresses allant de a.0.0.0 jusqu'à a.255.255.255 peuvent être utilisé pour associer des adresses IP aux machines dans le sous-réseau,
- a.b.c.d/16 pour indiquer les adresses allant de a.b.0.0 jusqu'à a.b.255.255,
- a.b.c.d/24 pour indiquer les adresses allant de a.b.c.0 jusqu'à a.b.c.255.

Par exemple, le réseau local `Client` peut utiliser toutes les adresses allant de `192.168.1.0` à `192.168.1.255`.

1.2 Table de routage

Lorsqu'un routeur reçoit un paquet, en fonction de l'adresse de destination, il choisit vers quel routeur voisin le transmettre. Par exemple, si le `Client` veut envoyer un message au `Serveur`, il le transmet au routeur `R1` qui le transmet à son tour au routeur `R3` auquel il est connecté. Ce dernier à plusieurs possibilités, soit transmettre le message à `R2`, soit à `R4` ou encore `R5`, ce qui donne différent chemins possibles :

- `R1` → `R3` → `R2` → `R5` → `R6`
- `R1` → `R3` → `R4` → `R5` → `R6`
- `R1` → `R3` → `R5` → `R6`

Pour déterminer à quel routeur voisin envoyer les paquets reçu, chaque routeur possède une **table de routage**.
Sur le shéma de la figure 1, les tables (simplifiées) de `R1` et `R3` sont :

Table de routage de `R1`

Réseau destinataire	Passerelle	Interface
192.168.1.0/24		192.168.1.1
10.1.1.0/24		10.1.1.1
10.1.2.0/24	10.1.1.3	10.1.1.1
10.1.3.0/24	10.1.1.3	10.1.1.1
10.1.4.0/24	10.1.1.3	10.1.1.1
10.1.5.0/24	10.1.1.3	10.1.1.1
10.1.6.0/24	10.1.1.3	10.1.1.1
10.1.7.0/24	10.1.1.3	10.1.1.1
192.168.6.0/24	10.1.1.3	10.1.1.1

Table de routage de `R3`

Réseau destinataire	Passerelle	Interface
10.1.1.0/24		10.1.1.3
10.1.2.0/24		10.1.2.14
10.1.3.0/24		10.1.3.5
10.1.4.0/24		10.1.4.42
192.168.1.0/24	10.1.1.1	10.1.1.3
10.1.5.0/24	10.1.3.4	10.1.3.5
10.1.6.0/24	10.1.2.5	10.1.2.14
10.1.7.0/24	10.1.4.8	10.1.4.42
192.168.6.0/24	10.1.4.8	10.1.4.42

Pour le routeur `R1`, les deux premières lignes ne contiennent pas de passerelle, les réseaux `10.1.1.0` et `192.168.1.0` sont directement reliés au routeur.
L'interface connectant `R1` au réseau `192.168.1.0` a pour adresse `192.168.1.1` et celle le connectant au réseau `10.1.1.0` a pour adresse `10.1.1.1`.
Tous les paquets destinés aux autres réseaux passent par l'interface `10.1.1.1` à destination de `R3` dont l'interface connectée au réseau `10.1.1.0` a pour adresse `10.1.1.3`.

Dans les deux paragraphes qui suivent nous allons présenter deux algorithmes implémentés dans les routeurs qui permettent de configurer leur table de routage. Ces algorithmes, appelés *protocoles de routage*, sont à la fois décentralisés et dynamiques.

2 Protocole RIP

Le protocole RIP (*Routing Information Protocol*) rentre dans la catégorie des protocoles à vecteur de distance. Un vecteur de distance est un couple (adresse, distance). Le principe de ce protocole est de chercher à minimiser le nombre de routeurs à traverser pour atteindre la destination (on minimise le nombre de sauts).
Nous allons nous intéresser à l'évolution des tables de routage des routeurs `R1` et `R3` de la figure 1, sur lesquels on a activé le protocole RIP.

Etape 0

Initialement, les routeurs ne connaissent que leurs voisins proches. La table de `R1` ressemble à ceci :

Réseau destinataire	Passerelle	Interface	Distance
192.168.1.0/24		wlan0	1
10.1.1.0/24		eth0	1

Et celle de `R3` :

Réseau destinataire	Passerelle	Interface	Distance
10.1.1.0/24		eth1	1
10.1.2.0/24		eth0	1
10.1.3.0/24		eth3	1
10.1.4.0/24		eth2	1

Etape 1

Au bout de 30 secondes, un premier échange intervient avec les voisins immédiats de chacun des routeurs. Lorsqu'un routeur reçoit une nouvelle route de la part d'un voisin, il y a quatre cas possibles :

1. Il découvre une route vers un nouveau sous-réseau inconnu : il l'ajoute à sa table.
2. Il découvre une route vers un réseau connu plus courte que celle qu'il possède dans sa table : il actualise sa table.
3. Il découvre une route vers un réseau connu plus longue que celle qu'il possède dans sa table : il ignore cette route.
4. Il reçoit une route vers un réseau connu en provenance d'un routeur déjà existant dans sa table : un problème est apparu sur son ancienne route, il met à jour sa table.

En appliquant ces règles, voici la table de routage de `R1` après une étape :

Réseau destinataire	Passerelle	Interface	Distance
192.168.1.0/24		wlan0	1
10.1.1.0/24		eth0	1
10.1.2.0/24	10.1.1.3	eth0	2
10.1.3.0/24	10.1.1.3	eth0	2
10.1.4.0/24	10.1.1.3	eth0	2

`10.1.1.3` est l'adresse IP du routeur `R3`. On ajoute à la table précédente les réseaux atteignables par `R3` en icrémentant la distance de 1. Si R1 veut atteindre le réseau `10.1.2.0`, il s'adressera à `R3` et atteindra le réseau cible en 2 sauts.
Voici la table de `R3` qui s'enrichit des informations envoyées par `R1` afin d'atteindre le réseau local, mais aussi des informations en provenance de `R2`, `R4` et `R5`. Il découvre ainsi quatre nouveaux réseaux.

Réseau destinataire	Passerelle	Interface	Distance
10.1.1.0/24		eth1	1
10.1.2.0/24		eth3	1
10.1.3.0/24		eth2	1
10.1.4.0/24		eth0	1
192.168.1.0/24	10.1.1.1	eth1	2
10.1.6.0/30	10.1.2.5	eth3	2
10.1.5.0/30	10.1.4.8	eth0	2
10.1.7.0/30	10.1.4.8	eth0	2

Etape Finale

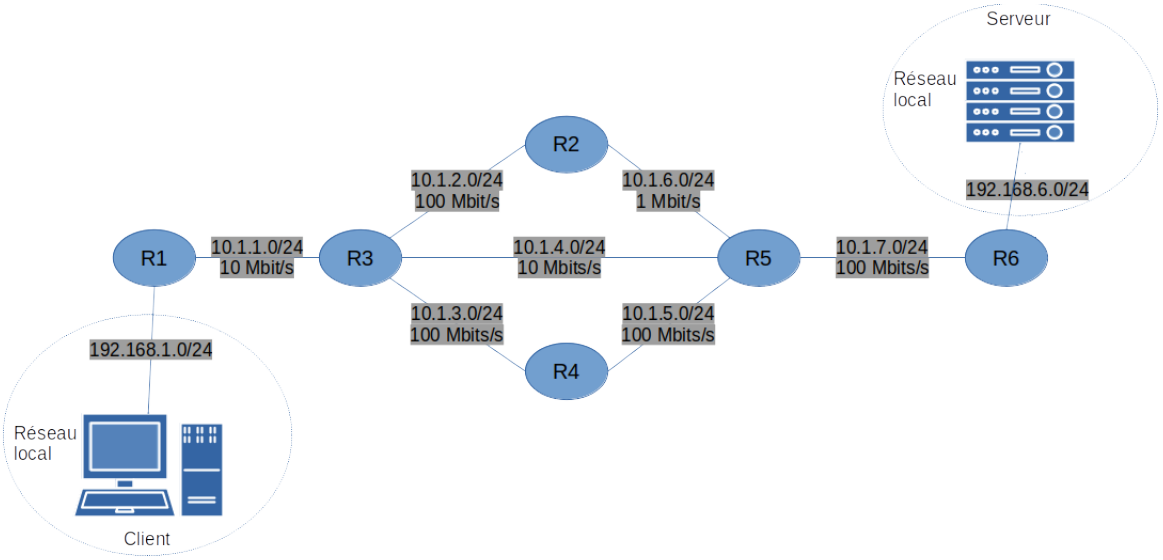
En répétant l'étape précédente, les routeurs vont avoir la même vue du réseau. La table de routage finale de `R1`, sera :

Réseau destinataire	Passerelle	Interface	Distance
192.168.1.0/24		wlan0	1
10.1.1.0/24		eth0	1
10.1.2.0/24	10.1.1.3	eth0	2
10.1.3.0/24	10.1.1.3	eth0	2
10.1.4.0/24	10.1.1.3	eth0	2
10.1.5.0/24	10.1.1.3	eth0	3
10.1.6.0/24	10.1.1.3	eth0	3
10.1.7.0/24	10.1.1.3	eth0	3
192.168.6.0/24	10.1.1.3	eth0	4

3 Protocole OSPF

Le protocole OSPF (*Open Shortest Path First*) est un protocole de routage «à état de liens», il cherche à optimiser le débit des liaisons epruntées.

On considère le réseau de la figure 1 sur lequel on a ajouté le débit des liaisons.



Le coût d'une liaison est donnée par la formule suivante :

$$\text{coût} = \frac{10^8}{\text{débit}}$$

où le débit est exprimé en *bits/s* (1 Gbit = 10^9 bits, 1 Mbit = 10^6 bits, 1 kbit = 10^3 bits).
Dans l'exemple précédent, on obtient le tableau suivant :

Liaison	R1-R3	R3-R2	R3-R4	R3-R5	R2-R5	R4-R5	R5-R6
Coût	10	1	1	10	100	1	1

FIGURE 2 – Topologie d'un réseau de routeurs OSPF

Etape 0 : Au démarrage du protocole, chaque routeur envoie un message (message `HELLO`) aux routeurs auxquels il est connecté et ainsi construire sa *table de voisinage*.

Liaison	sous-réseau	coût
R3-R1	10.1.1.0/24	10
R3-R2	10.1.2.0/24	1
R3-R5	10.1.4.0/24	10
R3-R4	10.1.3.0/24	1

FIGURE 3 – Table de voisinage de `R3`

Etape 1 : Chaque routeur va ensuite échanger des messages `LSA` (*Link State Advertisement*) contenant sa table de voisinage avec *tous* les autres routeurs du réseau et ainsi pouvoir reconstituer la topologie complète du réseau de la figure 2.

Etape 2 : Finalement chaque routeur calcul à l'aide de l'algorithme de Dijkstra le plus court chemin pour chaque destination dans le réseau et construit ainsi sa table de routage.

R3	R1	R2	R4	R5	R6
0-R3					
0-R3	10-R3	1-R3	1-R3	10-R3	
×		**1-**R3			
×		×	**1-**R3	2-R4	
×		×	×	**2-**R4	3-R5
×		×	×	×	**3-**R5
×	**10-**R3	×	×	×	×

FIGURE 4 – Algorithme de Dijkstra pour R3

Réseau destinataire	Passerelle	Interface	Coût
10.1.1.0/24		eth1	10
10.1.2.0/24		eth3	1
10.1.3.0/24		eth2	1
10.1.4.0/24		eth0	10
192.168.1.0/24	10.1.1.1 R1	eth1	10
10.1.6.0/30	10.1.2.2 R2	eth3	1
10.1.5.0/30	10.1.3.4 R4	eth2	2
10.1.7.0/30	10.1.3.4 R4	eth2	3

FIGURE 5 – Table de routage de R3

4 Commandes système

4.1 La commande ip

La commande ip remplace plusieurs commandes historiques dont l'utilisation est maintenant obsolète (*deprecated*), c'est un véritable couteau Suisse de l'administration réseau.

La commande ip addr permet d'afficher la liste des périfériques réseau aisi que les adresses IP qui leur sont associées.

```
pascal@linux:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state ...
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp4s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc ...
    link/ether a8:5e:45:e2:9d:73 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.86/24 brd 192.168.1.255 scope global ...
       valid_lft 71649sec preferred_lft 71649sec
    inet6 fe80::e89b:6f37:960c:88e8/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

La sortie de la commande `ip addr` ci-dessus nous indique que la machine possède deux interfaces :

- L'interface `lo`, de type *loopback* (ou boucle locale, une interface fictive) est associé à l'adresse IPv4 `127.0.0.1` et l'adresse IPv6 `::1`.
- L'interface `enp4s0`, est de type `ethernet` (`en`). L'adresse MAC (adresse matérielle ou adresse physique) est `a8:5e:45:e2:9d:73`. Son adresse IPv4 est `192.168.1.86` et son adresse IPv6 est `fe80::e89b:6f37:960c:88e8`

Un affichage similaire peut être obtenue avec `ifconfig` et avec `ipconfig` sou Windows.

La commande `ip route` affiche la table de routage actuellement configurée.

```
pascal@linux:~$ ip route
default via 192.168.1.1 dev enp4s0 proto dhcp metric 100
169.254.0.0/16 dev enp4s0 scope link metric 1000
192.168.1.0/24 dev enp4s0 proto kernel scope link src 192.168.1.86 metric 100
```

La *route par défaut* est la machine `192.168.1.1`, c'est le routeur d'accès (celui du réseau local). Tous les paquets qui ne sont pas à destination du réseau local seront transférés au routeur.
La dernière ligne indique que tout paquet à destination d'une adresse réseau `192.168.1.0/24` (c'est à dire toutes les adresses comprises entre `192.168.1.0` et `192.168.1.255`) sera difusé directement par l'interface `enp4s0`.

4.2 La commande `ping`

La commande `ping` permet de tester l'accessibilité d'une autre machine à travers un réseau IP. La commande mesure également le temps mis pour recevoir une réponse, appelé round-trip time (temps aller-retour). (*source : Wikipedia*)

```
pascal@linux:~$ ping www.google.fr
PING www.google.fr (142.250.178.131) 56(84) bytes of data.
64 bytes from par21s22-in-f3.1e100.net (142.250.178.131): icmp_seq=1 ttl=117 time=16.1 ms
64 bytes from par21s22-in-f3.1e100.net (142.250.178.131): icmp_seq=2 ttl=117 time=14.7 ms
64 bytes from par21s22-in-f3.1e100.net (142.250.178.131): icmp_seq=3 ttl=117 time=14.6 ms
...
```

Dans l'exemple ci-dessus, on apprend que l'adresse IP de `www.google.fr` est `142.250.178.131`, la première requête a reçu une réponse 16,1 ms après avoir été envoyé, la deuxième 14,7 ms.
Le TTL (*Time To Live*) indique combien de routeur un paquet peut encore traverser. A chaque passage dans un routeur, ce compteur est décrémenté. Quand un routeur reçoit un paquet avec un TTL de 0 il le détruit.

L'option `-c` permet de préciser le nombre de tentatives.

```
pascal@linux:~$ ping -c 1 www.google.fr
PING www.google.fr (216.58.198.195) 56(84) bytes of data.
64 bytes from par10s27-in-f3.1e100.net (216.58.198.195): icmp_seq=1 ttl=118 time=16.0 ms

--- www.google.fr ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
```

L'option -t permet de préciser le TTL des paquets envoyés.

```
pascal@linux:~$ ping -c 1 -t 5 www.google.fr
PING www.google.fr (216.58.198.195) 56(84) bytes of data.
From 125.10.136.77.rev.sfr.net (77.136.10.125) icmp_seq=1 Time to live exceeded

--- www.google.fr ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms
```

4.3 La commande `traceroute`

La commande `traceroute` permet de déterminer la route empruntée par un paquet IP pour atteindre une machine cible. Pour cela la commande envoie des paquets IP vers la destination avec un TTL croissant. Chaque routeur se trouvant sur la route décrémente le TTL d'une unité avant de le transmettre si il est non nul. Si le TTL est nul, le routeur détruit le paquet et émet vers la source un paquet ICMP d'erreur signalant que le message a été détruit et n'a pas atteint sa source. Cela permet à traceroute de savoir quels sont les routeurs atteints à l'aller.

```
pascal@linux:~$ traceroute www.google.fr
traceroute to www.google.fr (142.250.74.227), 64 hops max
  1   192.168.1.1  0,467ms  0,440ms  0,347ms
  2   109.6.26.23  2,541ms  2,735ms  2,709ms
  3   84.96.247.81  3,406ms  2,840ms  2,746ms
  4   77.136.10.125  15,349ms  16,803ms  16,684ms
  5   77.136.10.125  16,053ms  13,915ms  13,764ms
  6   72.14.194.30  13,882ms  13,734ms  18,384ms
  7   *  *  *
  8   108.170.245.1  16,945ms  16,549ms  16,473ms
  9   108.170.245.6  36,180ms  22,133ms  15,393ms
 10   *  *  142.250.74.227  15,098ms
```