

© Copyright 2005 tv <tvtsii@free.fr>

Permission is granted to copy, distribute and/or modify this document under the terms of the **GNU Free Documentation License**, Version 1.1 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover Texts, and with no Back-Cover. You can obtain a copy of the GNU General Public License : write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Permission est accordée de copier, distribuer et/ou modifier ce document selon les termes de la **Licence de Documentation Libre GNU** (GNU Free Documentation License), version 1.1 ou toute version ultérieure publiée par la Free Software Foundation ; sans Sections Invariables ; sans Texte de Première de Couverture, et sans Texte de Quatrième de Couverture. Vous pouvez obtenir une copie de la GNU General Public License : écrire à la Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Table des matières

Le routage.....	3
Introduction.....	3
Routage statique et dynamique.....	4
Routage direct.....	4
Routage indirect.....	4
Adresses physiques et réseaux.....	5
Algorithme de routage.....	5
Les commandes de base.....	6
La table de routage.....	7
Présentation.....	7
Exemple.....	8
Configurer un poste Linux en routeur.....	8
Le service network.....	9
Routage statique.....	10
Routage dynamique.....	10
Les protocoles de routage dynamique.....	11
Architecture de routage Internet.....	12
Le modèle hiérarchique historique.....	12
Le nouveau modèle.....	12
BGP (Border Gateway Protocol).....	12
Les algorithmes de routage dynamique.....	13
Distant Vector.....	13
Link State.....	13
Le protocole de routage RIP (Routing Information Protocol).....	13
Historique.....	13
Caractéristiques.....	13
Les minuteurs.....	14
Fonctionnement.....	14
Différence entre RIP et RIPng.....	14

Le protocole de routage OSPF (Open Shortest Path First).....	15
Historique.....	15
Caractéristiques.....	15
Exemple.....	16
Les aires.....	18
Comparaison RIP – OSPF.....	19
Solutions de routage pour Linux.....	19
Le protocole IRDP (ICMP Router Discover Protocol).....	20
Principe.....	20
Solution IRDP pour Linux.....	20
Masquage et Translation d'adresse.....	21
Mise en situation.....	21
IP Masquerade.....	21
NAT (Network Address Translation).....	22
Proxy.....	22
La traduction d'adresse réseau avec iptables.....	22
Modification d'adresse source.....	22
Modification d'adresse destination.....	22
Exercice 1 – Routage direct et indirect.....	23
Exercice 2 – Routage dynamique.....	24
Exercice 3 – Split Horizon et Reverse Poison.....	26
Exercice 4 - Ping Pong.....	29
Exercice 5 - traceroute.....	31
Exercice 6 – Décodage RIP.....	33
Exercice 7 – Tolérance aux pannes avec RIP.....	39
Exercice 8 – OSPF.....	40
Exercice 9 - Routeur NAT.....	41
Annexe 1 - La trame Ethernet_II et 802.3.....	44
Annexe 2 - IP.....	45
Annexe 3 - ICMP.....	46
Annexe 4 - RIP-II.....	48
Annexe 5 - UDP	49
Annexe 6 - TCP	50
Annexe 7 – IGMP.....	51
Annexe 8 - La multidiffusion IP.....	52

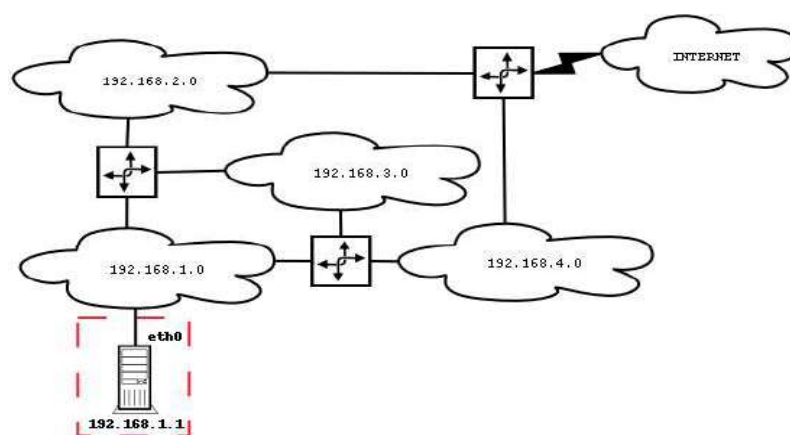
Note: Ce document traite de l'aspect réseau du système GNU/Linux.

Le routage

Introduction

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).



Les équipements (hôtes ou routeurs) qui se situent sur des réseaux différents devront utiliser les services d'un **routeur** (*gateway* dans la terminologie IP) pour communiquer.

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

Question: sur quelle interface réseau **acheminer** le paquet ? (un équipement peut posséder une ou plusieurs interfaces)

Réponse: il faut chercher un élément dans le paquet à acheminer pour prendre sa décision

De manière générale, on distingue deux approches possibles au niveau de la couche RESEAU :

- service orienté connexion : circuit virtuel X25 (non abordé dans ce document)
- service sans connexion : adresse destination du datagramme IP

Donc, chaque équipement (hôte ou routeur) achemine un paquet en fonction de l'**adresse IP destination** uniquement.

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.

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

- **manuellement**
- **automatiquement** par échange de routes avec ses voisins ou par connaissance directe

Routage statique et dynamique

On distingue :

- **routage statique** si les routes sont fixées manuellement par l'administrateur réseau
- **routage dynamique** si les tables de routages sont automatiquement mises à jour pour tenir compte d'une modification du réseau global (panne de routeur, nouvelle route, ...)

Routage direct

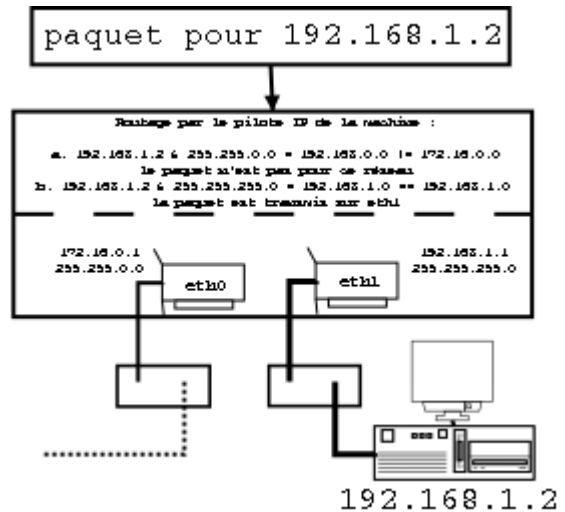
Délivrance d'un paquet à un hôte qui appartient au même réseau physique

La commande **ifconfig** permet la configuration du routage direct en associant une adresse IP à une carte réseau.

```
$ ifconfig eth0 172.16.0.1 netmask 255.255.0.0  
$ ifconfig eth1 192.168.1.1 netmask 255.255.255.0
```

```
$ netstat -nr
```

```
Table de routage IP du noyau  
Destination Passerelle Genmask Indic Iface  
172.16.0.0 0.0.0.0 255.255.0.0 U eth0  
192.168.1.0 0.0.0.0 255.255.255.0 U eth1  
127.0.0.0 0.0.0.0 255.0.0.0 U lo
```



Routage indirect

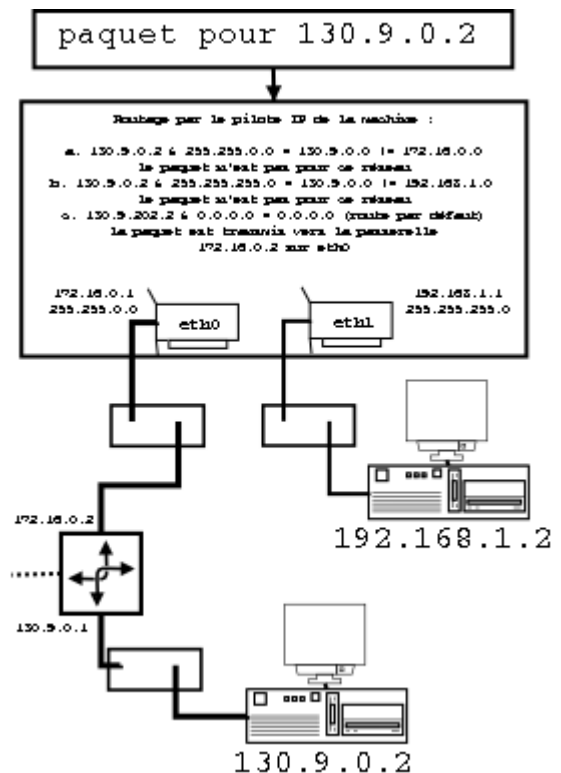
Délivrance d'un paquet à un hôte qui appartient à un réseau physique différent

La commande **route** permet la configuration du routage indirect en permettant l'ajout et la suppression de route vers un hôte, un réseau ou une route par défaut.

```
# route add default gw 172.16.0.2 dev eth0
```

```
$ netstat -nr
```

```
Table de routage IP du noyau  
Destination Passerelle Genmask Indic Iface  
172.16.0.0 0.0.0.0 255.255.0.0 U eth0  
192.168.1.0 0.0.0.0 255.255.255.0 U eth1  
127.0.0.0 0.0.0.0 255.0.0.0 U lo  
0.0.0.0 172.16.0.2 0.0.0.0 UG eth0
```



Remarques

Les tables de routage doivent être configurées sur l'ensemble des équipements (hôtes et routeurs).

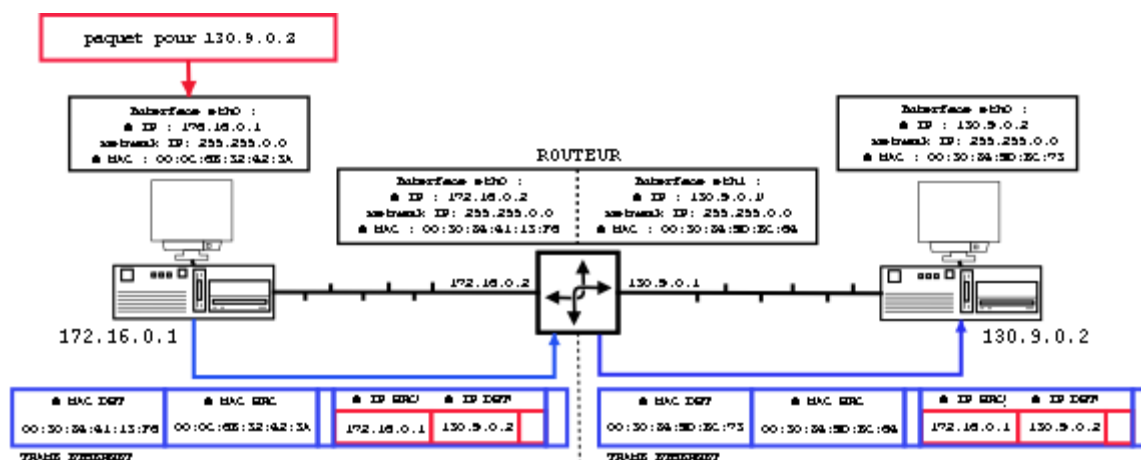
Cas des hôtes:

Les tables de routages des postes se limitent souvent à une route par défaut : vers le routeur (*gateway*, donc souvent passerelle en français) qui permettra de sortir du réseau physique.

Cas des routeurs:

Les tables de routages sont donc configurées principalement au niveau des routeurs : manuellement (routage statique) ou automatiquement acquises par dialogue entre routeurs (routage dynamique).

Adresses physiques et réseaux



Remarque: la résolution des adresses MAC est réalisée par le protocole ARP (*Address Resolution Protocol*).

Algorithme de routage

Le routage consiste à **déterminer la route qu'un paquet doit prendre pour atteindre une destination en fonction de l'adresse IP de destination et des routes contenues dans la table de routage.**

L'équipement recherche le réseau de destination à partir de l'adresse IP de destination. On rappelle que l'adresse réseau s'obtient en effectuant un ET binaire entre l'adresse IP et un masque de réseau (*netmask*).

On distingue :

➤ **Classfull** : basé sur les classes réseaux

Le principe du routage présenté précédemment est connu sous le nom de *classfull*. En résumé :

```

interfaceTrouvée  délivrer le paquet par routage direct
SI !interfaceTrouvée
    ALORS
        routeTrouvée  délivrer par routage indirect
    SI !routeTrouvée
        ALORS  délivrer la paquet par la passerelle par défaut
        SINON  renvoyer message : "Destination unreachabile"
    FSI
FSI
    
```

FSI

➤ **Classless** : sans classe

L'algorithme a évolué pour tenir compte des réseaux dont la taille est supérieure à la taille déduite de la classe (sur-réseaux). Ceci conduit à l'abandon de la notion de classe, seul le *netmask* détermine la taille du réseau. On utilise aussi le terme **CIDR** (*Classless InterDomain Routing*).

```
POUR une adresse IP destination
  trouvé rechercher dans la table de routage le préfixe le plus long qui correspond à
  l'adresse destination
  SI trouvé
    ALORS envoyer le paquet
    SINON renvoyer le message : "Destination unreachable"
  FSI
FPOUR
```

Exemple de la règle du plus grand préfixe:

Table de routage
10.0.0.0/8 passé par venus
10.0.0.0/16 passé par mars

Un paquet destiné à 10.0.1.1 passera par mars.
Un paquet destiné à 10.3.1.1 passera par venus.

Route par défaut:

La route par défaut est notée **0.0.0.0** et correspond donc au masque de longueur nulle : toutes les adresses destinations correspondront. Suivant la règle du plus grand préfixe utilisée dans le routage, cette correspondance, étant la plus petite (0.0.0.0/0), sera donc bien testée et choisie en dernier (donc par défaut).

Les commandes de base

<i>Commandes</i>	<i>Description</i>
netstat	Affiche les connexions réseau, les tables de routage, les statistiques des interfaces, les connexions masquées, les messages netlink, et les membres multicast.
ping, ping6	envoyer des datagrammes ICMP ECHO_REQUEST à des hôtes sur un réseau
traceroute	affiche la route prise par des paquets sur le réseau afin d'atteindre une destination
tracpath, tracpath6	traces path to a network host discovering MTU along this path
route	affiche et manipule la table de routage IP permet de paramétrer le routage indirect
ifconfig	permet la configuration des interfaces et du routage direct
ip	TCP/IP interface configuration and routing utility

Rappel: Ce document traite de l'aspect réseau du système GNU/Linux.

La table de routage

Présentation

Les commandes **netstat -r** et **route** affichent la table de routage d'un hôte.

Une table de routage indique pour chaque destination (hôte, réseau ou sous-réseau) la route (interface ou passerelle) qu'il faut prendre. Les informations pour chaque route sont donc les suivantes :

<i>Aller vers</i>	<i>Passer par</i>
la destination (hôte ou réseau)	la route
<i>Champs: Destination et Genmask</i>	<i>Champs: Passerelle et Iface</i>

Le champ Indic (*Flags*) :

- U (*Up*) : la route est active
- H (*Host*) : la route conduit à un hôte
- G (*Gateway*) : la route passe par une passerelle (voisine)

Les informations complémentaires :

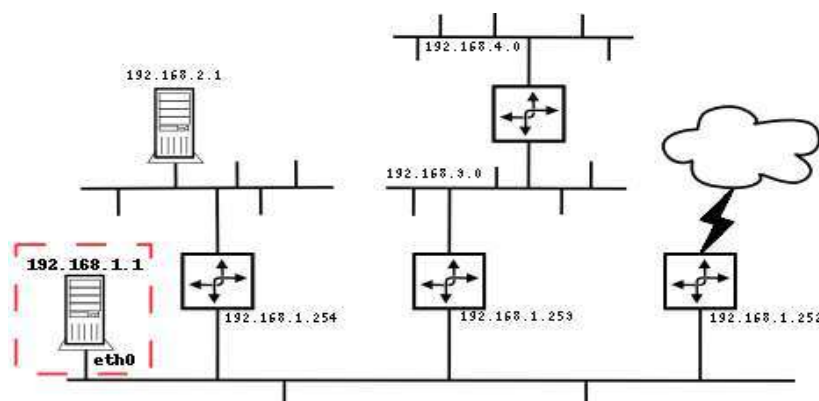
- le champ Métrique (*Metric*) indique la distance, en nombre de passerelles, pour atteindre la destination
- le champ Ref spécifie le nombre de références à cette route (non utilisé par le noyau Linux)
- le champ Use comptabilise le nombre de recherches associées à cette route
- le champ MSS indique la taille maximale des segments TCP sur cette route
- le champ Fenêtre (*Window*) indique la taille de la fenêtre sur cette route
- le champ irrtr indique le paramètre IRRT d'une connexion TCP pour cette route

Le fichier ASCII **/proc/net/route** donne la table de routage du noyau, mais le programme standard **netstat** fournit un accès plus propre à ces données.

La table de routage est renseignée par :

- ◆ la commande **route**(routage indirect)
- ◆ la commande **ifconfig**(routage direct)
- ◆ des fichiers au démarrage de la machine (**route statique**)

Exemple



Ajout d'une route vers un poste

```
# route add 192.168.2.1 gw 192.168.1.254 dev eth0
```

Ajout d'une route vers un réseau

```
# route add -net 192.168.3.0 gw 192.168.1.253 netmask 255.255.255.0 dev eth0
```

Ajout d'une route par défaut

```
# route add default gw 192.168.1.252 dev eth0
```

Visualisation de la table de routage de l'hôte

```
# netstat -nr
```

Destination	Passerelle	Genmask	Indic	MSS	Fenêtre	irtt	Iface
192.168.2.1	192.168.1.254	255.255.255.255	UGH	0	0	0	eth0
192.168.3.0	192.168.1.253	255.255.255.0	UG	0	0	0	eth0
192.168.1.0	0.0.0.0	255.255.255.0	U	0	0	0	eth0
127.0.0.0	0.0.0.0	255.0.0.0	U	0	0	0	lo
0.0.0.0	192.168.1.252	0.0.0.0	UG	0	0	0	eth0

Configurer un poste Linux en routeur

Par défaut, un poste Linux ne délivre que les paquets qui proviennent du poste ou qui lui sont destinés. Les autres paquets sont ignorés et donc le poste ne peut faire fonction de routeur.

Pour devenir routeur, le poste Linux doit **relayer** (retransmettre) les paquets qui ne lui sont pas destinés. Pour cela, il faut modifier l'option **ip_forward** du noyau :

- Activer le routage:

```
# echo "1" > /proc/sys/net/ipv4/ip_forward // OU  
# sysctl -n -w net.ipv4.ip_forward=1
```

- Désactiver le routage:

```
# echo "0" > /proc/sys/net/ipv4/ip_forward // OU  
# sysctl -n -w net.ipv4.ip_forward=0
```

Activer le routage de manière permanente (prise en compte au démarrage du service réseau) :

FORWARD_IPV4=yes dans le fichier /etc/sysconfig/network (voir aussi /etc/sysctl.conf)

Le service *network*

Le script **network** permet de gérer le service réseau :

```
# /etc/init.d/network
```

```
Utilisation : network {start|stop|restart|reload|status}
```

Le service **network** est exécuté automatiquement par le système pour les niveaux de démarrage suivants :

```
# chkconfig --list network
```

```
network          0:Arrêt 1:Arrêt 2:Marche          3:Marche          4:Marche          5:Marche          6:Arrêt
```

Remarque: le script **network** a déjà été étudié lors du TP Réseaux n°1 sous Linux.

Le script **network** lit le fichier `/etc/sysconfig/network` qui contient des options de configuration :

```
# cat /etc/sysconfig/network
HOSTNAME="thius"
NETWORKING=yes
```

Si l'option **NETWORKING** est égale à **no** ou si l'utilitaire de configuration **ip** n'est pas présent, le script **network** sort immédiatement.

Les options du fichier `/etc/sysconfig/network` concernant le routage sont :

```
FORWARD_IPV4=      // yes (pour un routeur) ou no (pour un hôte)
```

```
GATEWAY=           // adresse ip de la passerelle IP
```

```
GATEWAYDEV=        // une interface, par exemple eth0 ou ppp0
```

Avec l'option **start**, le script **network** réalise entre autres des actions concernant le routage :

- active le **FORWARD IPv4** (si `FORWARD_IPV4 = "yes" ou "true"`)
- ajoute les routes statiques décrites dans le fichier `/etc/sysconfig/static-routes`

La syntaxe du fichier `/etc/sysconfig/static-routes` dépend du script **network** qui réalise le traitement suivant :

```
# Add non interface-specific static-routes.
if [ -f /etc/sysconfig/static-routes ]; then
    grep "^any" /etc/sysconfig/static-routes | while read ignore args ; do
        /sbin/route add -$args
    done
fi
```

Donc, chaque ligne du fichier `/etc/sysconfig/static-routes` doit absolument commencer par **any** et préciser les arguments à ajouter à la commande **route add -**, par exemple :

```
any net 192.168.0.0 gw 192.168.1.254 netmask 255.255.255.0 dev eth1
```

Remarques:

- Voir aussi le script `/etc/sysconfig/network-scripts/ifup-routes`
- Les informations fournies dépendent évidemment de la distribution Linux utilisée. Celles présentées ici ont été testées sur une Mandrake 9.x.

Routage statique

Le routage statique est réalisé manuellement par l'administrateur réseau.

<i>Avantages</i>	<i>Inconvénients</i>
Utilisation de fichiers de configuration donc stabilité de la configuration	Si le réseau comporte de nombreux routeurs : - tâche fastidieuse - risque d'erreur important
	Impossibilité pour gérer les routes redondantes

Le routage statique est utilisé dans les réseaux de petite taille pour :

- les postes de travail (route par défaut)
- un routeur avec une route par défaut vers le Fournisseur d'Accès Internet (ou ISP: *Internet Service Provider*)

Routage dynamique

Le routage dynamique est assuré par les routeurs eux-même en s'échangeant des informations sur leurs tables de routage (nécessité d'un protocole de routage).

<i>Avantages</i>	<i>Inconvénients</i>
Simplicité de la configuration	Dépend du protocole de routage utilisé et de la taille du réseau : - consommation de la bande passante - temps de convergence - sécurité
Adaptabilité à l'évolution du réseau	
Optimisation (sélection des meilleurs routes)	
Elimination des boucles de routage	

Les protocoles de routage dynamique

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) ;
- **EGP** (*Exterior Gateway Protocol*) : protocole de routage externe utilisé entre passerelles appartenant à des unités administratives différentes (AS)

	<i>Internet</i>	<i>ISO</i>
Routage intra-domaines IGP Taille < 100 routeurs	On distingue deux types de protocoles : - distance vecteur (<i>distant vector</i>) : RIP (<i>Routing Information Protocol</i>), IGRP (<i>Internet Gateway Routing Protocol</i>) de la société CISCO (le protocole a été amélioré sous le nom EIGRP) - état de liens (<i>link state</i>) : OSPF (<i>Open Shortest Path First</i>)	IS-IS (<i>Intermediate System to Intermediate System</i>)
Routage inter-domaines EGP Taille = Internet	EGP (<i>Exterior Gateway Protocol</i>) : obsolète, remplacé par BGP (<i>Border Gateway Protocol</i>)	IDRP (<i>Inter Domain Routing Protocol</i>)
Entre équipement et routeur	ICMP <i>Redirect</i>	IS-ES

<i>Taille du réseau privé</i>	<i>Protocole de routage</i>
< 5 routeurs	Routage statique ou RIP
< 15 routeurs	RIP
< 100 routeurs	OSPF
Réseau vers plusieurs ISP	Optimisation des routes vers Internet par BGP

Architecture de routage Internet

L'architecture du réseau Internet a changé au fil des années notamment pour s'adapter à sa croissance ...

Le modèle hiérarchique historique

Dans la structure initiale d'Internet, il existait une hiérarchie de passerelles. Le réseau était bâti sur le réseau ARPANET existant, appelé le coeur (*core*), et les passerelles permettant de s'y connecter étaient appelées les passerelles centrales.

Dans ce modèle hiérarchique, les informations de routage concernant l'ensemble du réseau Internet étaient envoyées aux passerelles centrales qui devaient maintenir un routage central : toute route devait absolument être déterminée par le coeur du réseau. Evidemment, ce modèle, qui impliquait une charge de travail importante pour le système central, n'était plus adapté l'évolution grandissante du réseau Internet.

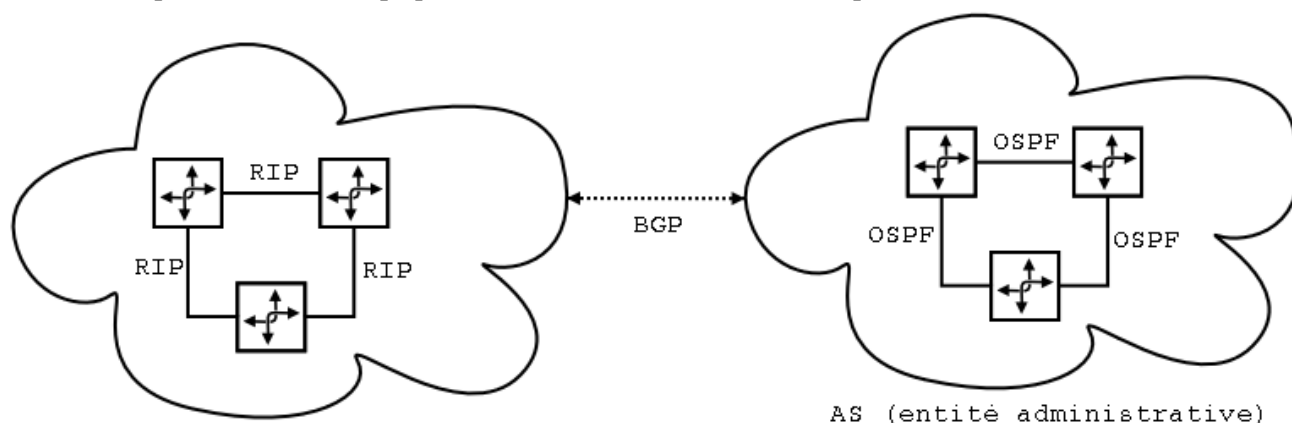
Le nouveau modèle

Le nouveau modèle de routage est basé sur une ensemble de systèmes autonomes (AS *Autonomous System* RFC 1930) appelés domaine de routage.

Ces domaines de routage échangent des informations de routage en utilisant un protocole externe comme BGP. A la différence du modèle hiérarchique, ce modèle ne dépend pas d'un seul système central mais il permet de choisir les meilleures routes et il est donc plus évolutif.

Un système autonome (AS) est :

- un ensemble de routeurs sous une administration unique
- identifié par un entier sur 16 bits
- géré par un protocole de routage interne (IGP) et des métriques communes (unique protocole de routage dynamique au sein d'un AS) pour router les paquets à l'intérieur du domaine
- capable de router les paquets vers d'autres AS en utilisant un protocole externe (EGP) comme BGP.



BGP (Border Gateway Protocol)

BGP est le protocole de routage externe le plus utilisé sur Internet. Il est basé sur le protocole IRDP de l'OSI. BGP gère le routage basé sur une politique (non technique, c'est-à-dire en tenant compte de considérations politiques, organisationnelles ou de sécurité) pour prendre des décisions en matière de routage.

BGP améliore la capacité d'un AS à choisir entre différentes routes sans se baser sur une autorité centrale de routage (on n'utilise plus le modèle hiérarchique). Seules les passerelles, qui interconnectent un AS avec un autre AS, utilisent un protocole externe tel que BGP. BGP se place au-dessus de TCP/IP en utilisant le port TCP 179.

Les algorithmes de routage dynamique

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 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 (RFC1058).

Link State

Un protocole de type *link-state* 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).

Le protocole de routage RIP (*Routing Information Protocol*)

Historique

Les étapes de la standardisation IETF pour RIP:

- RIPv1 : RFC1058 (06/88)
- RIPv2 : RFC1387, RFC1388 (01/93), RFC1723 (04/94) → routage CIDR, diffusion *multicast*, authentification des routeurs
- RIPng : RFC2080 (01/97), RFC2453 (11/98) → adaptation pour IPv6

Caractéristiques

- Utilisation d'une métrique fixe égale à 1 pour les réseaux directement connectés à une interface du routeur
- Fonctionne au dessus des ports UDP 520 (IPv4), 521 (IPv6)
- Adresse *multicast* 224.0.0.9
- Temps de convergence de quelques minutes acceptable
- Élimination des boucles ***poison reverse*** : les routes en provenance d'un voisin lui sont ré-annoncées avec une métrique infinie
- ***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)

Les minuteurs

- **routing-update** (30 secondes \pm 0 à 5 secondes) : période maximale entre deux annonces

Si tous les routeurs utilisaient des minuteurs *routing-update* paramétrés avec la même valeur de 30 secondes par exemple, il se produirait au bout d'un certain un phénomène de synchronisation de leurs annonces RIP. Pour éviter ce phénomène qui conduirait à des rafales de paquets et des risques de congestion cycliques, les valeurs effectives des minuteurs sont perturbées aléatoirement de 0 à 5 secondes.

- **route-timeout** (180 secondes) : durée de vie associée à chacune des routes apprises par RIP

Après expiration de ce minuteur, la route est marquée comme invalide dans la table des informations RIP. Elle ne sera effacée que lorsque le minuteur *route-flush* expire. Ce mécanisme permet à un routeur de propager l'information de route invalide vers ses voisins (pour tenir compte d'une interface réseau qui devient inopérante par exemple). Si pendant ce temps une nouvelle route vers ce préfixe est apprise, elle remplace la route invalide.

- **route-flush** ou *garbage-collection* (120 secondes) : périodicité de nettoyage de la table RIP

Les routes marquées comme invalides sont effacées.

Fonctionnement

À chaque route (@IP + netmask) est associée une métrique (M) qui est sa distance exprimée en nombre de routeurs à traverser.

Chaque routeur envoie à ses voisins ses informations de routage (les réseaux qu'il sait router et métriques associées) : toutes les 30 secondes systématiquement.

Si un routeur reçoit d'un voisin ses informations de routage, il calcule les métriques locales des routes apprises ($M \rightarrow M+1$), 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).

Différence entre RIP et RIPng

RIPng est une adaptation de RIP pour IPv6, il n'y a pas eu d'évolutions conceptuelles mais juste les modifications suivantes :

- Usage d'un nouveau numéro de port UDP 521 (au lieu de 520)
- Modification du format du paquet d'annonce de routes
- Suppression du mécanisme d'authentification spécifique
- Limitation du nombre de routes annoncées par le MTU uniquement
- Possibilité d'annoncer spécifiquement un *next hop*

Le protocole de routage OSPF (*Open Shortest Path First*)

Historique

Protocole ouvert de routage interne de type *Link-State*, élaboré par l'IETF en s'inspirant du protocole OSI IS-IS :

- OSPFv1 : RFC1131 (10/89) puis RFC1247 (07/91)
- OSPFv2 : RFC 2328 (04/98)
- OSPFv3 : RFC 2740 (12/99)
- Adaptation pour IPv6

Caractéristiques

- Repose sur la recherche de la meilleure route en se basant sur l'algorithme SPF de Dijkstra
- Fonctionne au dessus d'IP (protocole n° 89)
- Utilise les adresses multicast : 224.0.0.5 (pour tous les routeurs du lien) et 224.0.0.6 (pour le routeur principal et de secours)
- Le nombre de routeurs n'est pas limité
- Attribue un coût (*metric*) à chaque lien afin de privilégier l'élection des meilleures routes
- Valeur par défaut de la *metric* : $10^8 / (\text{débit nominal du lien})$
- Authentification possible

Chaque routeur doit :

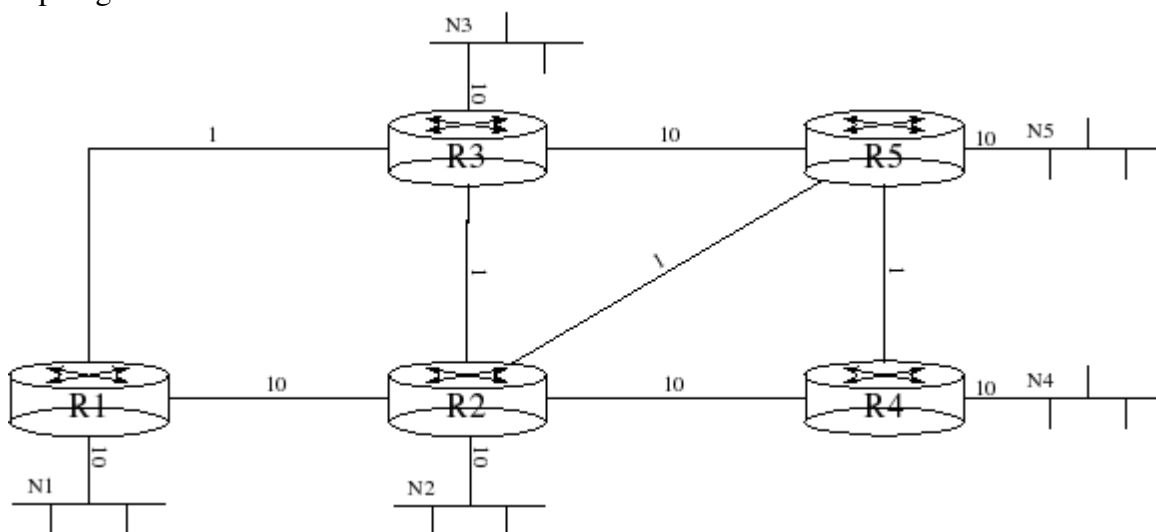
- établir la liste des routeurs (en s'annonçant à ses voisins avec des paquets HELLO toutes les 10 secondes en *multicast* 224.0.0.5)
- élire le routeur désigné (*Designated Router*) et le routeur désigné de secours (*Backup Designated Router*), c'est-à-dire celui avec la priorité sur 8 bits (par défaut à 1) la plus élevée (en cas d'égalité, celui qui a la plus grande adresse IP). Le routeur désigné est le référent (échanges entre chaque routeur et le DR permet de réduire le trafic) de la base de données unique de la carte topologique.
- découvrir les routes (paquets LSA *Link State Advertisement*, LSR *Link State Request* et LSU *Link state Update*) afin de constituer la carte topologique (relation maître/esclave entre le DR et chaque routeur)
- élire les routes à utiliser (à partir de la carte topologique et avec l'algorithme SPF, le routeur construit sa table de routage)

Pour maintenir la carte topologique du réseau , chaque routeur :

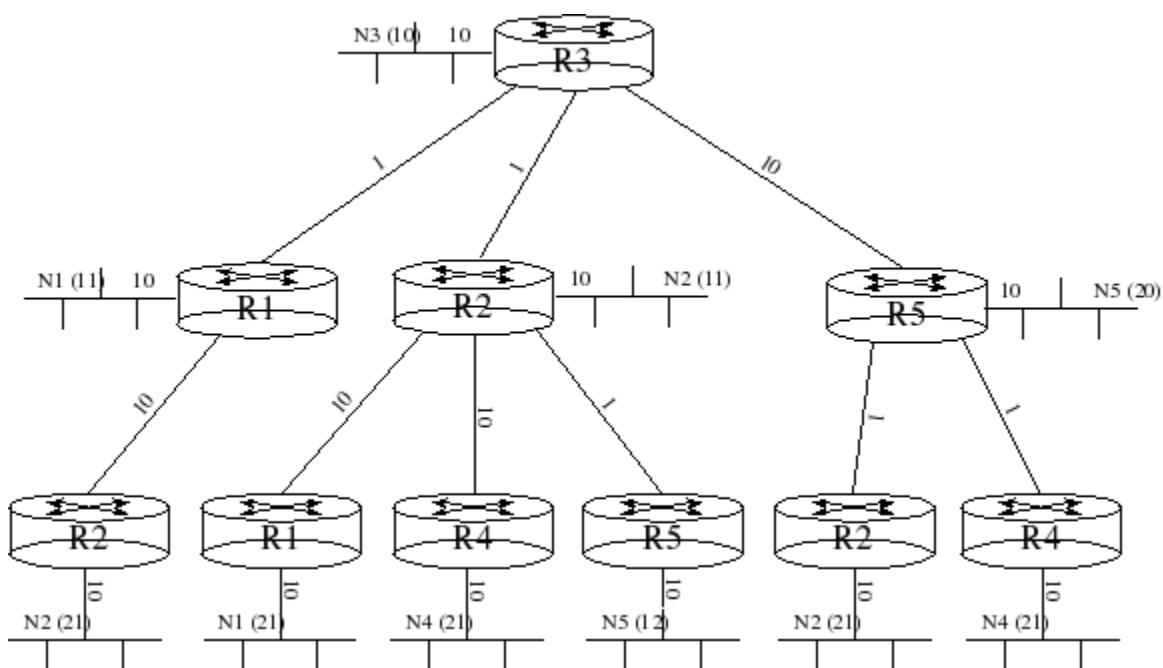
- surveille activement l'état de toutes ses liaisons (liens)
- diffuse (paquets LSU *Link state Update*) cet état au routeur désigné (*multicast* 224.0.0.6), puis celui-ci diffuse l'information aux routeurs (*multicast* 224.0.0.5)
- reconstruit à partir de ces informations une carte topologique complète du réseau
- recalcule les routes de plus court chemin (algorithme SPF de Dijkstra)

Exemple

Soit la topologie suivante :



Le coût des liaisons tient compte du débit des lignes (1 pour 100 Mbits/s et 10 pour 10 Mbits/s). Sans utiliser l'algorithme SPF, on a la carte topologique suivante du point de vue de R3 :



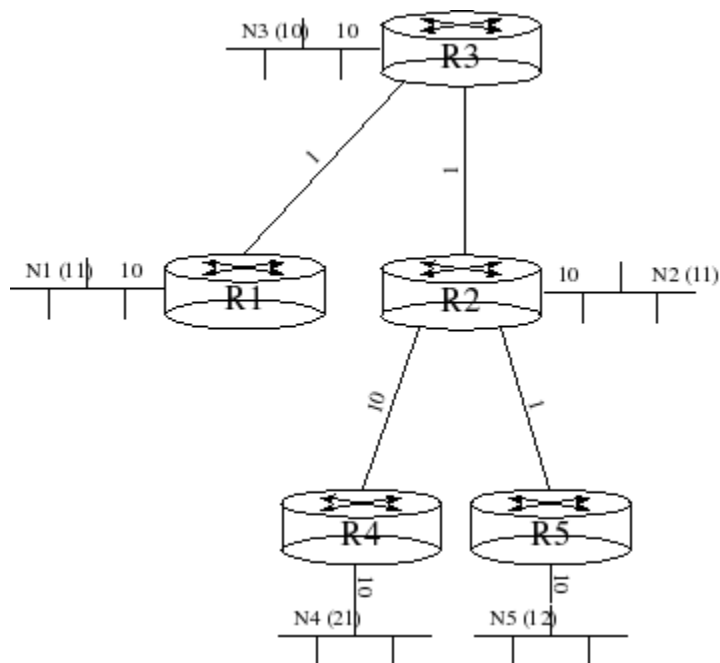
Remarque: pour atteindre N5

- RIP choisirait la route R3-R5 avec une *metric* de 2.
- OSPF choisira la route R3-R2-R5 qui a un coût plus faible ($12 < 20$).

- Le routage IP -

L'algorithme SPF va permettre de privilégier les routes qui ont un coût faible (qui sont donc supposées rapides).

Après traitement, chaque routeur se voit comme la racine de l'arbre, ici pour R3 :



R3 peut maintenant construire sa table de routage ...

Les aires

Dans un grand domaine de routage, chaque changement provoque une diffusion de la table de l'état des liens de tous les routeurs, ce qui provoque une consommation de bande passante importante et une charge CPU importante sur les routeurs. Alors que la portée d'une modification reste assez localisée.

Pour optimiser OSPF, le domaine de routage est découpé en **aires** : chaque aire est plus simple à gérer, plus stable.

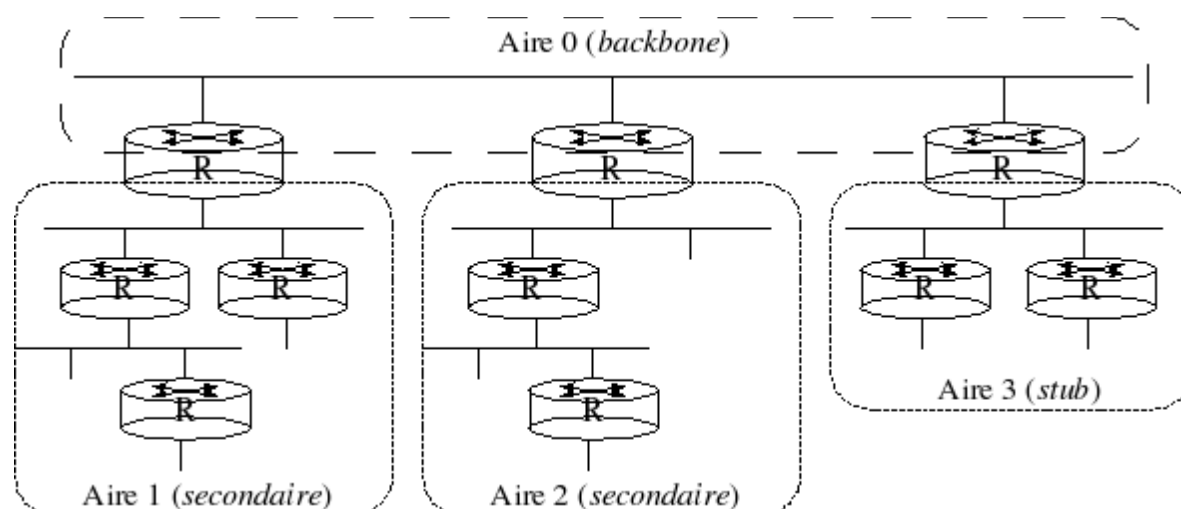
Pour garder une cohérence globale, il y aura toujours (et une seule) une **aire principale (backbone)** qui :

- relie toutes les aires entre elles
- connaît toutes les infos de routage, mais ne diffuse que des condensés (en agrégeant les routes, si c'est possible)

C'est le fait de lui attribuer le numéro 0 qui rend une aire principale dans un domaine de routage OSPF. Si on ne veut pas découper le domaine, tous les interfaces de tous les routeurs seront dans l'aire 0.

Les autres aires sont de type **secondaire** (numérotées de 1 à 4294967295 ($2^{31}-1$)).

Il existe aussi des aires secondaires de type **terminal (stub)**. Ce type d'aire a une gestion simplifiée car l'unique point de sortie vers l'aire principale permet de gérer les routeurs en leur diffusant seulement une route par défaut. Le passage d'une aire à une autre se fait par des routeurs de bord.



En résumé, on distingue trois types d'aires :

- l'**aire principale backbone (area 0)** : chemin obligatoire pour passer d'une aire à l'autre
- les **aires secondaires** : tous les routeurs ont une vue complète de la carte du réseau et calculent localement la meilleure route entre une source et une destination
- les **aires terminales (stub area)** : même comportement que les aires secondaires sauf que toutes les routes externes sont récapitulées dans une route par défaut (donc les routeurs ne mémorisent pas les informations sur les routes externes)

Comparaison RIP – OSPF

RIP (<i>Distance-Vector</i>)		OSPF (<i>Link-State</i>)	
Avantages	Inconvénients	Avantages	Inconvénients
simplicité calculs légers	lenteur de convergence et risques de bouclage infini ne peut y avoir de chemins multiples taille des messages échangés proportionnelle au nombre de routeurs seul critère de la métrique de distance	convergence rapide et sans boucle chemins multiples possibles calcul des routes indépendant des autres routeurs les messages ne sont proportionnels au nombre de routeurs le critère de la métrique n'est pas limitée à la distance	complexité calculs lourds

En conclusion, les algorithmes de type *Link-State* (OSPF) sont plus complexes mais plus performants et mieux adaptés aux réseaux importants que les algorithmes de type *Distance-Vector* (RIP).

Solutions de routage pour Linux

Applications	Commentaires
routed	logiciel traditionnel des systèmes Unix
gated	supporte la plupart des protocoles de routage dynamique (notamment EGP)
zebra	permet de transformer une machine Linux en routeur dédié (<i>CISCO IOS-like</i>) en supportant notamment les protocoles RIP, OSPF, EGP pour IPv4 et IPv6

Le protocole IRDP (*ICMP Router Discover Protocol*)

Le protocole **IRDP** permet à un hôte de découvrir les différentes passerelles présentes sur son réseau local et ainsi de mettre à jour sa route par défaut.

Principe

Un hôte se met à l'écoute du groupe *multicast* **224.0.0.1** et attend les messages **ICMP Router Advertisement** émis par les routeurs présents sur le réseau local.

L'hôte peut émettre des messages **ICMP Router Solicitation** vers **224.0.0.2** pour demander aux routeurs de se faire connaître.

Solution IRDP pour Linux

La commande **rdisc** ou **in.rdisc** (paquetage *iputils*) permet à un poste Linux de recevoir les messages IRDP émis par un routeur.

Masquage et Translation d'adresse

Les deux besoins distincts sont :

- ◆ partager une ou plusieurs adresses valides d'un réseau d'interconnexion
- ◆ masquer les adresses réelles des postes derrière un routeur ou un pare-feu

Plusieurs techniques sont disponibles pour répondre à ces besoins :

- ◆ le masquage d'adresse réseau (Masquerading)
- ◆ la translation d'adresse réseau (NAT)
- ◆ les services d'un proxy

Mise en situation

Le besoin se pose essentiellement lorsqu'on interconnecte un réseau public (Internet) avec un réseau privé. La situation courante veut qu'on utilise des adresses privées pour le réseau local.

Le réseau privé ne pourra pas utiliser n'importe lequel d'adressage interne essentiellement à cause des tables de routage et des risques de conflits avec l'extérieur.

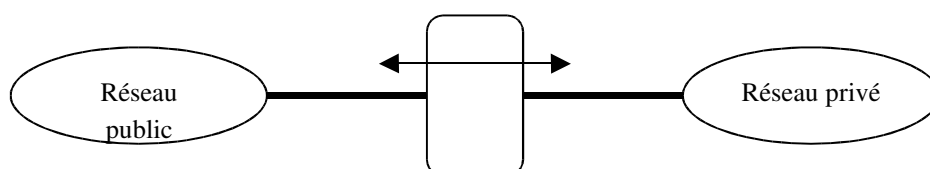
Il existe des adresses réservées, qui ne seront donc pas routées, à utiliser pour les réseaux privés :

Classe A : 10.0.0.0	à	10.255.255.255
Classe B : 172.16.0.0	à	172.31.255.255
Classe C : 192.168.0.0	à	192.168.255.255

Remarques:

Ces adresses réservées peuvent être découpées en sous-réseaux privés.

Par contre, les postes privés ont peut-être besoin d'accéder aux services proposés par le réseau public interconnecté :



IP Masquerade

L'IP masquerade, consiste à masquer les adresses IP du réseau interne et à n'utiliser que l'adresse du routeur sur le réseau d'interconnexion.

Cette technique est utilisée pour connecter tout un réseau local sur l'accès d'un prestataire qui ne fournit qu'une seule adresse IP (souvent dynamique) : partage d'une connexion internet le plus souvent.

La différence majeure entre le MASQ et NAT est que le serveur MASQ n'a besoin que d'une adresse IP valide (celle de l'interface de sortie).

La différence majeure entre le MASQ et un PROXY est que le serveur MASQ n'a pas besoin de changer la configuration des machines clientes. Il faut juste que les machines clientes déclarent leur passerelle par défaut.

NAT (Network Address Translation)

La machine, ayant à sa disposition un certain nombre d'adresses IP valides, pourra utiliser NAT pour les partager avec un groupe de postes ne possédant que des adresses IP privés.

Lorsqu'une connexion vers l'extérieur est requise par un utilisateur interne, cette machine associe une adresse IP valide disponible à l'IP privée qui requiert la connexion.

Le problème principal du NAT est qu'une fois que toutes les adresses IP publiques sont utilisées, ceux qui veulent obtenir une connexion ensuite ne le pourront pas tant qu'une adresse ne se libère pas.

Proxy

Un serveur proxy utilise seulement une adresse IP, comme IP MASQ, et sert de passerelle pour les clients du réseau privé.

Toutes les applications clientes doivent supporter les services du proxy (SOCKS par exemple) et être configurées pour les utiliser.

Au delà du masquage d'adresse, le proxy a la possibilité d'ajouter une fonction de cache pour améliorer les performances globales.

La traduction d'adresse réseau avec iptables

La table NAT supporte deux types d'action :

- ◆ la modification d'adresse source (SNAT)
- ◆ la modification d'adresse destination (DNAT)

Modification d'adresse source

Cette modification est réalisée juste avant l'envoi du paquet (POSTROUTING). En effet, il est plus judicieux de ne pas perdre la véritable adresse IP source avant la fin de tous les traitements.

Deux cibles permettent de réaliser cette modification :

- ◆ SNAT : la cible SNAT pour la spécification de l'adresse source
- ◆ MASQUERADE : cette cible est associée explicitement à l'interface de sortie (MASQUERADE n'est qu'un cas particulier de SNAT). Si cette interface tombe en panne ou change d'adresse IP (cas d'adresse IP dynamique), toutes les sessions en cours sont détruites (ce qui n'est pas le cas avec la cible SNAT qu'il faudra donc privilégier dans la majorité des cas). La cible MASQUERADE est souvent utilisée dans le cas d'adresse dynamique (DHCP) et notamment pour partager une connexion internet.

Modification d'adresse destination

Cette modification se fait au niveau des chaînes PREROUTING et OUTPUT, c'est-à-dire avant un point de routage interne à Netfilter (*Routing Decision*) puisque la modification de l'adresse peut modifier la manière dont sera routé le paquet.

Deux cibles permettent de réaliser cette modification :

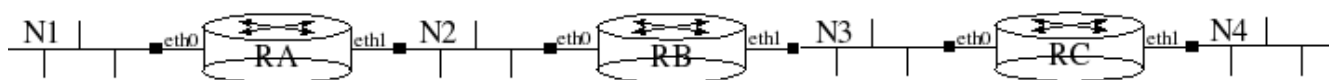
- ◆ DNAT : pour la spécification d'une adresse destination
- ◆ REDIRECT : qui envoie les paquets vers *localhost*, ce qui est équivalent à un DNAT sur 127.0.0.1.

Remarque :

La cible REDIRECT sert en particulier à réaliser des proxy transparents, comme par exemple la redirection du trafic HTTP vers un proxy local (*squid* par exemple)

Exercice 1 – Routage direct et indirect

La maquette du réseau est la suivante :



1 . Donner pour les trois routeurs leur table de routage uniquement pour les routes directes

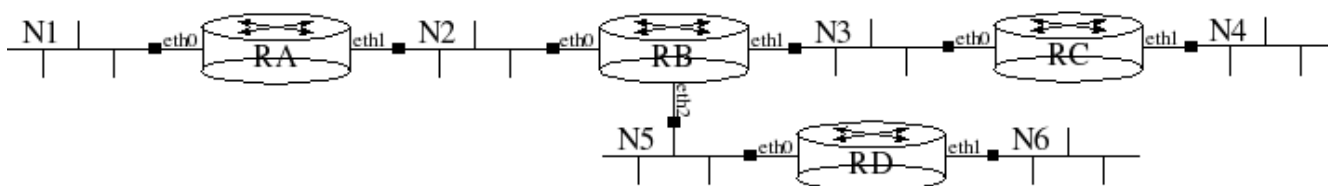
Routeur A				Routeur B				Routeur C			
Destination	Passerelle	Metric	Iface	Destination	Passerelle	Metric	Iface	Destination	Passerelle	Metric	Iface
N1	*	0	eth0								
N2	*	0	eth1								

2 . Donner pour les trois routeurs leur table de routage (les routes directes et indirectes)

Routeur A				Routeur B				Routeur C			
Destination	Passerelle	Metric	Iface	Destination	Passerelle	Metric	Iface	Destination	Passerelle	Metric	Iface
N1	*	0	eth0	N1				N1			
N2	*	0	eth1	N2				N2			
N3	RB	1	eth1	N3				N3			
N4	RB	2	eth1	N4				N4			

Exercice 2 – Routage dynamique

En reprenant la maquette du réseau précédente, on ajoute un routeur D et deux réseaux N5 et N6 :



Après configuration des routeurs RB et RD et avant tout échange de routes, on a les tables de routages suivantes :

Routeur B				Routeur D			
Destination	Passerelle	Metric	Iface	Destination	Passerelle	Metric	Iface
N2	*	0	eth0	N5	*	0	eth0
N3	*	0	eth1	N6	*	0	eth1
N5	*	0	eth2				
N1	RA	1	eth0				
N4	RC	1	eth1				

Remarques:

- RB a ajouté une route directe vers N5 pour son interface eth2
- RD a deux routes directes vers N5 et N6 pour ses interfaces eth0 et eth1

1 . Donner les tables de routage après que RB envoie sa nouvelle table de routage à ses voisins (RA, RC et RD)

Routeur A				Routeur C				Routeur D			
Destination	Passerelle	Metric	Iface	Destination	Passerelle	Metric	Iface	Destination	Passerelle	Metric	Iface
N1	*	0	eth0	N1				N1			
N2	*	0	eth1	N2				N2			
N3	RB	1	eth1	N3				N3			
N4	RB	2	eth1	N4				N4			
N5				N5				N5			
								N6			

2 . Donner la table de routage de RB après que RD lui envoie sa table de routage

Routeur B			
Destination	Passerelle	Metric	Iface
N1			
N2			
N3			
N4			
N5			
N6			

3 . Donner les tables de routage de RA et RC après que RB envoie sa nouvelle table de routage à ses voisins (RA et RC)

Routeur A				Routeur C			
Destination	Passerelle	Metric	Iface	Destination	Passerelle	Metric	Iface
N1	*	0	eth0	N1			
N2	*	0	eth1	N2			
N3	RB	1	eth1	N3			
N4	RB	2	eth1	N4			
N5				N5			
N6				N6			

Remarque:

Il y a eu un certain nombre d'échanges de table de routage entre tous les routeurs du domaine. Après un certain temps, appelé **temps de convergence**, les routeurs possèdent les routes pour atteindre tous les réseaux du domaine.

Exercice 3 – Split Horizon et Reverse Poison

On va observer le fonctionnement d'un routage dynamique de type *Distant-Vector* (exemple RIP).

On reprend la maquette de l'exercice n°1 :

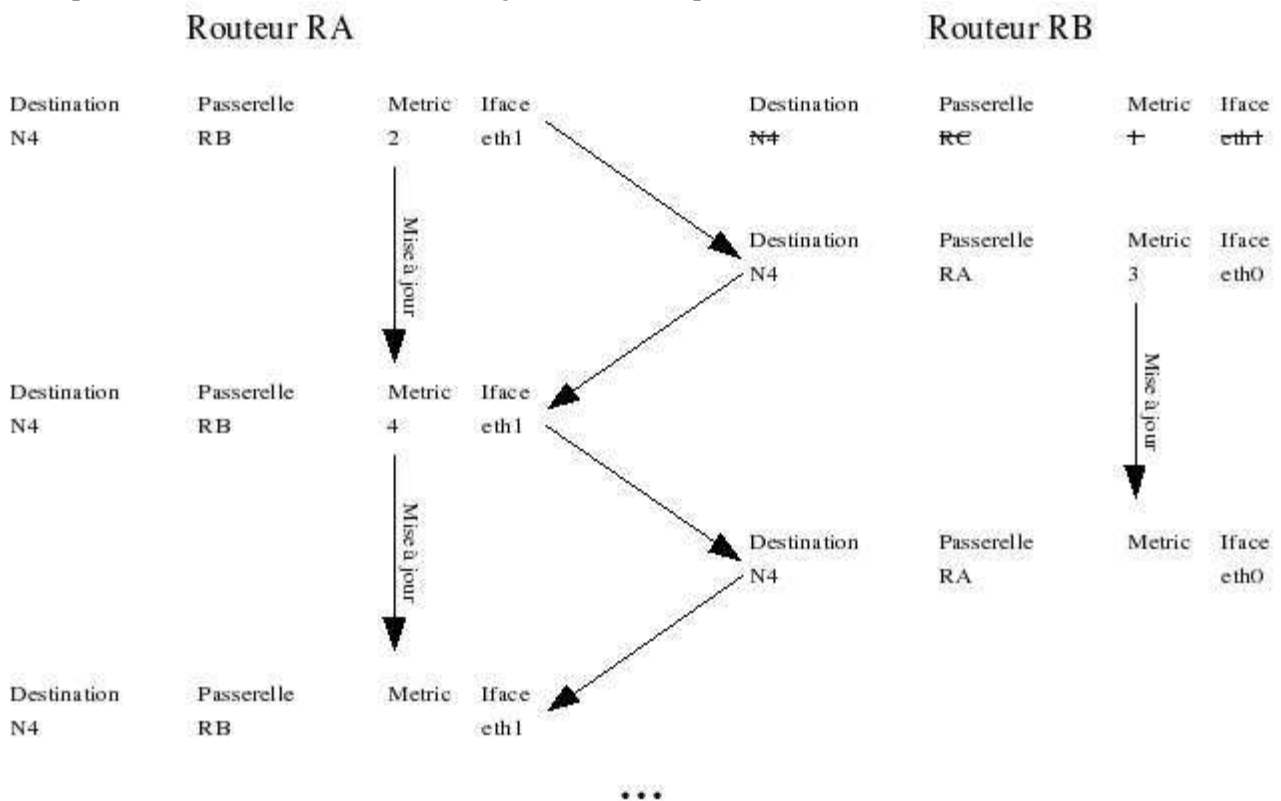


Un **incident** s'est produit sur le routeur C : son interface eth0 est *down*.

En observant le domaine de manière globale, on constate tout de suite que le réseau N4 n'est plus accessible pour les réseaux N1, N2 et N3. **Qu'en est-il des routeurs de ces réseaux ?**

Le routeur RB va supprimer sa route vers N4 mais RA va lui indiquer qu'il a une route pour atteindre ce réseau (en fait c'est la route qui passe par RB). RB va donc mettre sa table de routage à jour puis envoyer sa nouvelle table à RA. Le routeur RA va devoir mettre à jour sa table en tenant compte que la *metric* pour atteindre N4 a augmenté de 1 ... etc ... *déception mutuelle* !

1 . Compléter l'évolution des tables de routage de RA et RB pour la route N4



Problème: la *metric* pour atteindre N4 augmente indéfiniment (la convergence prend un temps infini).

Solution 1: il faut fixer une limite. Dans RIP, la *metric* limite est fixée à 16 et correspond à une route inaccessible.

2 . Calculer approximativement le temps de convergence pour déclarer une route inaccessible dans RIP en sachant que les mises à jour des tables sont réalisées toutes les 30 secondes.

Solution 2 : Split Horizon (Horizon coupé)

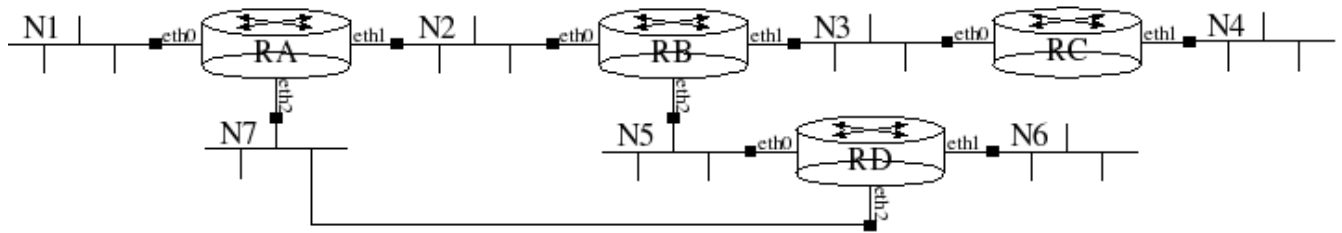
Un routeur ne doit pas envoyer à un routeur une route qu'il a apprise par lui.

Amélioration: Poison Reverse (Retour Empoisonné)

Les routes en provenance d'un voisin lui sont ré-annoncées avec une métrique infinie

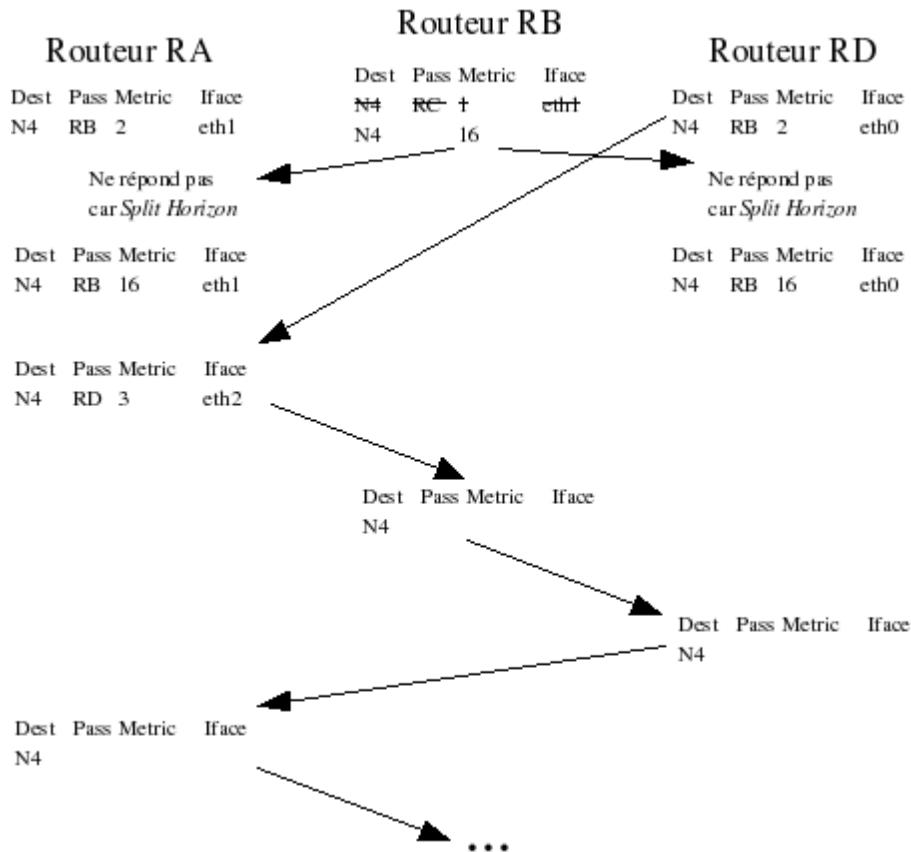
Est-ce suffisant ?

Le même incident sur le routeur C (son interface eth0 est *down*) mais sur le domaine suivant :



Donc, le routeur RB prévient RA et RD que la route pour rejoindre N4 est inaccessible (*metric* de 16). RA et RC ne répondent pas grâce au Split Horizon. Par contre, RD a communiqué à RA qu'il a un moyen d'atteindre N4 en passant par RB. Puis, RA indique à RB qu'il a une route pour atteindre N4 en passant par RD (RB ne peut pas savoir que cette route, en fait, passe par lui !) ...

3 . Compléter l'évolution des tables de routage de RA, RB et RD pour la route N4



4 . Calculer, en théorie, le temps de convergence pour déclarer une route inaccessible dans RIP en sachant que les mises à jour des tables sont réalisées toutes les 30 secondes.

Problème: le temps de convergence est (trop) long.

Solution: *Triggered Updates* (Mises à jour déclenchées)

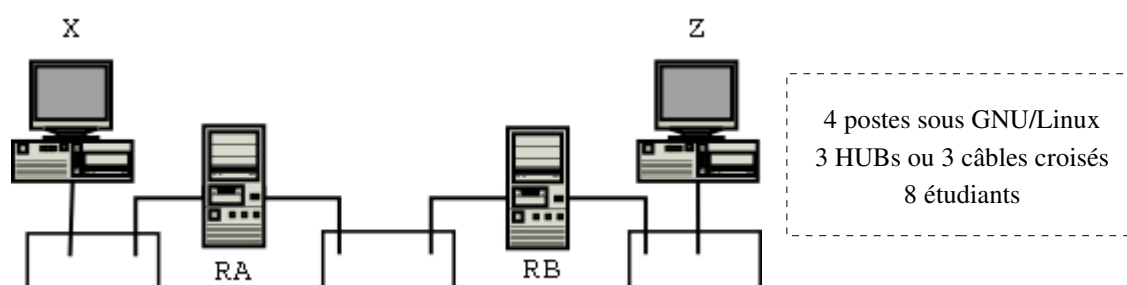
Une mise à jour des tables entraîne un envoi immédiat (des changements) sans attendre les 30 s.

Conclusion:

Les techniques du *Split Horizon*, *Reverse Poison* et *Triggered Updates* éliminent la plupart des problèmes (cas de bouclages directs) mais restent inefficace dans le cas d'une succession de routeurs. Pour résoudre ce problème, il faudrait que les routeurs aient une connaissance complète du domaine. Les routeur de type *Distant-Vector* (exemple RIP) ne peuvent pas puisqu'ils ne connaissent que leurs voisins immédiats.

Exercice 4 - Ping Pong

La maquette du réseau est la suivante :



On suppose les interfaces réseaux configurées pour tous les équipements et que les fonctions de routage sont activées sur les postes RA et RB.

Reporter sur le schéma ci-dessus l'ensemble des adresses IP :

Poste X	Routeur RA		Routeur RB		Poste Z
130.9.202.81	130.9.202.45	192.168.8.44	192.168.8.24	192.168.16.25	192.168.16.46
<i>eth0</i>	<i>eth0</i>	<i>eth1</i>	<i>eth0</i>	<i>eth1</i>	<i>eth0</i>

La station X ne reçoit pas de réponse lorsqu'elle « ping » la station Z.

1 . Proposer une procédure détaillée de test.

- Le routage IP -

On a les tables de routage suivantes pour les deux postes :

Poste X				Poste Z			
Destination	Passerelle	Metric	Iface	Destination	Passerelle	Metric	Iface
130.9.0.0	*	0	eth0	192.168.16.0	*	0	eth0
0.0.0.0	130.9.202.45	0	eth0				

Et on a les tables de routage suivantes pour les deux routeurs :

Routeur A				Routeur B			
Destination	Passerelle	Metric	Iface	Destination	Passerelle	Metric	Iface
130.9.0.0	*	0	eth0	192.168.8.0	*	0	eth0
192.168.8.0	*	0	eth1	192.168.16.0	*	0	eth1
0.0.0.0	192.168.8.24	0	eth1				

2 . Compléter le tableau suivant dans le cas d'un *ping* à partir du poste X

ping à partir du poste X	Résultat (OUI/NON)	Solution proposée ou commentaires
vers 130.9.202.45	OUI	Sur le même réseau IP
vers 192.168.8.44		
vers 192.168.8.24		
vers 192.168.16.25		
vers 192.168.16.46		

On place maintenant un poste Y sur le réseau 192.168.8.0.

3 . Quelle configuration faudra-t-il faire sur ce poste ?

Remarque: La redirection (ICMP Redirect)

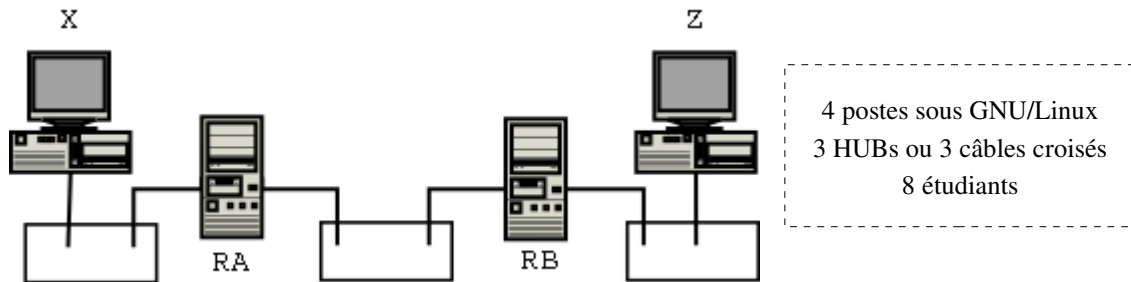
Le plus souvent, la configuration d'un poste (qui n'est pas routeur) se limite à indiquer la route par défaut qui lui permet de sortir de son réseau local. Sur ce réseau local, il peut y avoir plusieurs routeurs (cas de Y, RA et RB), mais dans la mesure, où le routeur par défaut possède une table de routage complète, le routage s'effectuera correctement.

Si le routeur par défaut s'aperçoit d'une meilleure route, il prévient alors le poste par un message *ICMP Redirect* qui lui demande de mettre à jour sa table de routage pour les futures envois.

L'administrateur du réseau peut décider de d'interdire ce trafic supplémentaire.

Exercice 5 - traceroute

La maquette du réseau est la suivante :



On suppose les interfaces réseaux et les tables de routages configurées pour tous les équipements.

Reporter sur le schéma ci-dessus l'ensemble des adresses IP :

Poste X	Routeur RA		Routeur RB		Poste Z
130.9.202.81	130.9.202.45	192.168.8.44	192.168.8.24	192.168.16.25	192.168.16.46
<i>eth0</i>	<i>eth0</i>	<i>eth1</i>	<i>eth0</i>	<i>eth1</i>	<i>eth0</i>

On trace la route de X vers Z (donc de 130.9.202.81 vers 192.168.16.46) avec la commande suivante :

```
# traceroute 192.168.16.46 -nI
```

```
traceroute to 192.168.16.46 (192.168.16.46), 30 hops max, 38 byte packets
```

```
1 130.9.202.45 0.242 ms 0.094 ms 0.090 ms
2 192.168.8.24 0.207 ms 0.152 ms 0.348 ms
3 192.168.16.46 0.274 ms 0.458 ms 0.424 ms
```

On capture le trafic ICMP pour analyser le traceroute avec la commande suivante :

```
# tcpdump -vvv -x ip
```

1. Compléter les champs adresses IP source et destination, TLL et le type ICMP manquants.

```
11:36:49.028014 130.9.202.81 > 192.168.16.46: icmp: echo request [ttl 1] (id 46140, len 38)
```

```
4500 0026 b43c 0000 0101 e869 8209 ca51
c0a8 102e 0800 a6ea b43b 0001 0101 b129
8e40 5c6d 0000
```

```
11:36:49.028205 130.9.202.45 > 130.9.202.81: icmp: time exceeded (ttl 64, id 49995, len 66)
```

```
45c0 0042 c34b 0000 4001 1e1e 8209 ca2d
8209 ca51 0b00 f4ff 0000 0000 4500 0026
b43c 0000 0101 e869 8209 ca51 c0a8 102e
0800 a6ea b43b 0001 0101 b129 8e40 5c6d
0000
```

```
11:36:49.032431 _____ > _____: icmp: _____ (ttl __, id 46143, len 38)
```

```
4500 0026 b43f 0000 0201 e766 8209 ca51
c0a8 102e 0800 53d5 b43b 0004 0402 b129
8e40 ac7e 0000
```

- Le routage IP -

11:36:49.032618 _____ > _____ : icmp: _____ (ttl 63, id 50545, len 66)

45c0 0042 c571 0000 3f01 a06e c0a8 0818
8209 ca51 0b00 f4ff 0000 0000 4500 0026
b43f 0000 0101 e866 8209 ca51 c0a8 102e
0800 53d5 b43b 0004 0402 b129 8e40 ac7e
0000

11:36:49.037086 _____ > _____ : icmp: _____ (ttl __, id 46146, len 38)

4500 0026 b442 0000 0301 e663 8209 ca51
c0a8 102e 0800 21bf b43b 0007 0703 b129
8e40 db90 0000

11:36:49.037340 _____ > _____ : icmp: _____ (ttl 62, id 46752, len 38)

4500 0026 b6a0 0000 3e01 a905 c0a8 102e
8209 ca51 0000 29bf b43b 0007 0703 b129
8e40 db90 0000 0000 0000 0000 0000

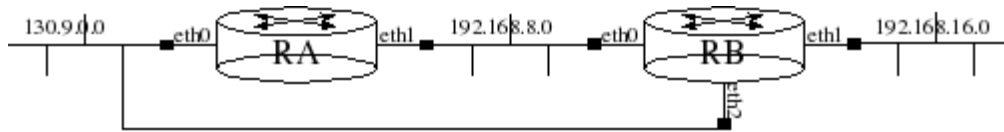
2 . Avec la capture précédente, reconstruire le diagramme d'échanges en précisant les champs adresses IP source et destination, TTL et le type ICMP.

3 . En conclusion, décrire brièvement le principe de la commande traceroute.

4 . Quel autre protocole aurait-on pu utiliser en lieu et place des paquets ICMP émis par traceroute ?

Exercice 6 – Décodage RIP

On utilise la maquette suivante avec un routage dynamique RIP sur les deux routeurs RA et RB :



Le logiciel Zebra (*CISCO IOS-like*) est installé et configuré sur les deux routeurs en suivant ce mini-HOWTO :

I . Installation

a . zebra

```
# rpm -ivh /mnt/cdrom2/zebra-0.93b-2mdk.i586.rpm --force --nodeps
Preparing... ##### [100%]
 1:zebra ##### [100%]
```

```
# vim /etc/zebra/zebra.conf
hostname routeurB
password password
```

b . ripd

```
# cp /usr/share/doc/zebra-0.93b/ripd.conf.sample /etc/zebra/ripd.conf
# vim /etc/zebra/ripd.conf
! *- rip *-
!
! RIPd sample configuration file
!
! $Id: ripd.conf.sample,v1.11 1999/02/19 17:28:42 developer Exp $
!
hostname routeurB(RIP)
password password
router rip
```

II . Démarrage des services

```
a . zebra : # /etc/init.d/zebra start
```

```
b . ripd : # /etc/init.d/ripd start
```

c . vérification

```
# ps -x
11601 ? S 0:00 /usr/sbin/zebra -d
11627 ? S 0:00 /usr/sbin/ripd -d
11637 pts/1 R 0:00 ps x
```

III . Configuration

a . zebra

Connexion par telnet sur le port 2601 (zebra)

```
# telnet localhost 2601
Trying 127.0.0.1...
Connected to localhost (127.0.0.1).
Escape character is '^]'.
Hello, this is zebra (version 0.93b).
Copyright 1996-2002 Kunihiro Ishiguro.
User Access Verification
Password: *****
a . passage en mode configuration
routeurB> enable
```

Liste des commandes

```
routeurB# ?
  configure  Configuration from vty interface
  copy       Copy configuration
  debug      Debugging functions (see also 'undebug')
  disable    Turn off privileged mode command
  end        End current mode and change to enable mode.
  exit       Exit current mode and down to previous mode
  help       Description of the interactive help system
  list       Print command list
  no         Negate a command or set its defaults
  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
  write      Write running configuration to memory, network, or terminal

routeurB# show ip ?
  access-list List IP access lists
  forwarding  IP forwarding status
  route       IP routing table
routeurB# show interface ?
  [IFNAME]   Interface name
```

Visualisation de la configuration des interfaces

```
routeurB# show interface
Interface lo
...
Interface eth0
  index 2 metric 1 mtu 1500 <UP, BROADCAST, RUNNING, MULTICAST>
  HWaddr: 00:4f:4e:08:25:1a
  inet 192.168.8.2/24 broadcast 192.168.8.255
...
```

Visualisation de la table de routage

```
routeurB# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
      B - BGP, > - selected route, * - FIB route
```

```
K * 127.0.0.0/8 is directly connected, lo
C>* 127.0.0.0/8 is directly connected, lo
C>* 130.9.0.0/16 is directly connected, eth2
C>* 192.168.8.0/24 is directly connected, eth0
C>* 192.168.16.0/24 is directly connected, eth1
```

Visualisation de la configuration en mémoire

```
routeurB# show running-config
```

Relayage des paquets ?

```
routeurB# show ip forwarding
IP forwarding is on
```

```
routeurB# quit
Connection closed by foreign host.
```

Vérification:

```
# cat /proc/sys/net/ipv4/ip_forward
1
```

b . ripd

- diffusion des routes directement connectée sur le réseau 192.168.8.0 (eth0)
- les messages envoyés vers l'adresse IP destination : 224.0.0.9 (*multicast*)
- ici RB envoie deux vecteurs de distance vers RA : 130.9.0.0 et 192.168.16.0

Remarque: show ip rip affiche la table diffusée, ce qui n'est donc pas la table de routage !

Connexion par telnet sur le port 2602 (ripd)

```
# telnet localhost 2602
Trying 127.0.0.1...
Connected to localhost (127.0.0.1).
Escape character is '^]'.
Hello, this is zebra (version 0.93b).
Copyright 1996-2002 Kunihiro Ishiguro.
User Access Verification
Password: *****
routeurB(RIP)> enable
routeurB(RIP)# show ip rip
Codes: R - RIP, C - connected, O - OSPF, B - BGP
      (n) - normal, (s) - static, (d) - default, (r) - redistribute,
      (i) - interface
```

Network	Next Hop	Metric	From	Time
routeurB(RIP)# conf t				
routeurB(RIP) (config)# router rip				
routeurB(RIP) (config-router)# redistribute				
bgp				
	Border Gateway Protocol (BGP)			

```
connected Connected
kernel Kernel routes
ospf Open Shortest Path First (OSPF)
static Static routes
```

Configuration de la redistribution

Toutes les routes directes (connectées) vers le réseau 192.168.8.0 (cad eth0 de RB)

```
routeurB(RIP) (config-router) # redistribute connected
routeurB(RIP) (config-router) # network 192.168.8.0/24
routeurB(RIP) (config-router) # end
routeurB(RIP) # show ip rip
```

Codes: R - RIP, C - connected, O - OSPF, B - BGP
(n) - normal, (s) - static, (d) - default, (r) - redistribute,
(i) - interface

	Network	Next Hop	Metric	From	Time
C(r)	130.9.0.0/16	0.0.0.0	1	self	
C(i)	192.168.8.0/24	0.0.0.0	1	self	
C(r)	192.168.16.0/24	0.0.0.0	1	self	

```
routeurB(RIP) # show ip protocols
```

Routing Protocol is "rip"

Sending updates every 30 seconds with +/-50%, next due in 5 seconds

Timeout after 180 seconds, garbage collect after 120 seconds

Outgoing update filter list for all interface is not set

Incoming update filter list for all interface is not set

Default redistribution metric is 1

Redistributing: connected

Default version control: send version 2, receive version 2

Interface	Send	Recv	Key-chain
eth0	2	2	

Routing for Networks:

192.168.8.0/24

Routing Information Sources:

Gateway	BadPackets	BadRoutes	Distance	Last Update
---------	------------	-----------	----------	-------------

Distance: (default is 120)

Remarque: les phases I, II et III sont répétés sur le routeur RA.

IV . Capture

a . tcpdump sur RB

```
# tcpdump -i eth2 -nt -s 0
```

```
tcpdump: listening on eth0
```

```
(a) 192.168.8.2.520 > 224.0.0.9.520: RIPv2-req 24 (DF) [ttl 1]
```

```
192.168.8.2 > 224.0.0.22: igmp v3 report, 1 group record(s) (DF) [tos 0xc0] [ttl 1]
```

```
(b) 192.168.8.2.520 > 224.0.0.9.520: RIPv2-resp [items 2]:
```

```
{130.9.0.0/255.255.0.0} (1) {192.168.16.0/255.255.255.0} (1) (DF) [ttl 1]
```

```
(c) 192.168.8.2.520 > 224.0.0.9.520: RIPv2-resp [items 2]:
```

```
{130.9.0.0/255.255.0.0} (1) {192.168.16.0/255.255.255.0} (1) (DF) [ttl 1]
```

- 1 . Décrire les trames a et b émises par le routeur RB. Quel est l'intérêt de ces trames ?
- 2 . Quel est l'intérêt de la trame c ?
- 3 . Justifier l'utilisation du champ TLL à 1.
- 4 . Quelles sont les routes annoncées par RB ? Combien de routes sont apprises par RA ?

On visualise la table de routage sur le routeur RA :

// avant RIP

```
# netstat -rn
```

Table de routage IP du noyau

Destination	Passerelle	Genmask	Indic	MSS	Fenêtre	irrtt	Iface
192.168.8.0	0.0.0.0	255.255.255.0	U	0	0	0	eth1
130.9.0.0	0.0.0.0	255.255.0.0	U	0	0	0	eth0
127.0.0.0	0.0.0.0	255.0.0.0	U	0	0	0	lo

// apres RIP

```
routeurA# show ip route
```

Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
B - BGP, > - selected route, * - FIB route

```
K * 127.0.0.0/8 is directly connected, lo
```

```
C>* 127.0.0.0/8 is directly connected, lo
```

```
C>* 130.9.0.0/16 is directly connected, eth0
```

```
C>* 192.168.8.0/24 is directly connected, eth1
```

```
R>* 192.168.16.0/24 [120/2] via 192.168.8.24, eth1, 00:03:12
```

Remarque: 120 est une distance administrative ajoutée par Zebra afin de distinguer le type des routes apprises (120 correspond à la valeur affectée à RIP, 0 pour C, 1 pour S, 20 pour BGP et 110 pour OSPF).

```
# route
```

Table de routage IP du noyau

Destination	Passerelle	Genmask	Indic	Metric	Ref	Use	Iface
192.168.16.0	192.168.8.24	255.255.255.0	UG	2	0	0	eth1
192.168.8.0	*	255.255.255.0	U	0	0	0	eth1
130.9.0.0	*	255.255.0.0	U	0	0	0	eth0
127.0.0.0	*	255.0.0.0	U	0	0	0	lo

- 5 . A votre avis, quelles seraient les routes annoncées par le routeur RA ?

b . ethereal sur eth0 de RB

```
01 00 5E 00 00 9 00 4F 4E 08 25 1A 08 00 45 00
00 48 73 DA 00 00 01 11 76 EB C0 A8 08 02 E0 0
00 09 02 08 02 08 00 34 27 76 02 02 00 00 00 02
00 00 82 09 00 00 FF FF 00 00 00 00 00 00 00
00 01 00 02 00 00 C0 A8 10 00 FF FF 00 00 00 00
00 00 00 00 00 01
```

1 . Décoder cette trame.

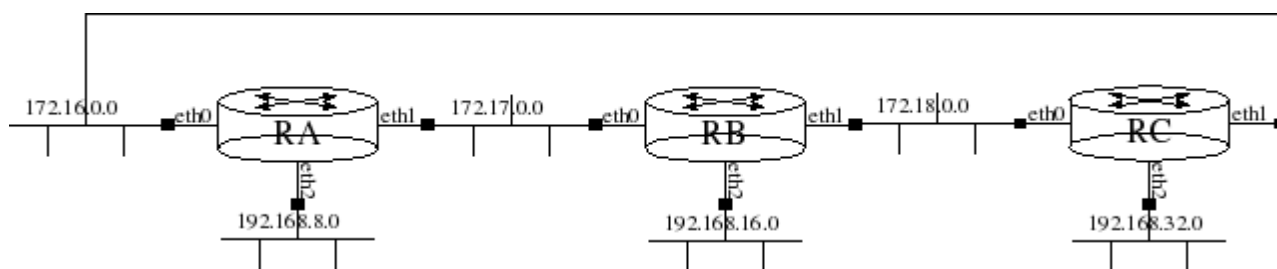
2 . Représenter les protocoles encapsulés dans cette trame.

3 . Quels sont les ports de couche 4 utilisés ?

4 . Quelles sont les informations de routage émises dans cette trame ?

Exercice 7 – Tolérance aux pannes avec RIP

On utilise la maquette suivante avec un routage dynamique RIP sur les trois routeurs :



On donne la table de routage de RA :

```
routeurA# show ip route
```

Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF

```
C>* 172.16.0.0/16 is directly connected, eth0
```

```
C>* 172.17.0.0/16 is directly connected, eth1
```

```
C>* 192.168.8.0/24 is directly connected, eth2
```

```
R>* 172.18.0.0/16 [120/2] via 172.17.0.2, eth1
```

```
R>* 192.168.16.0/24 [120/2] via 172.17.0.2, eth1
```

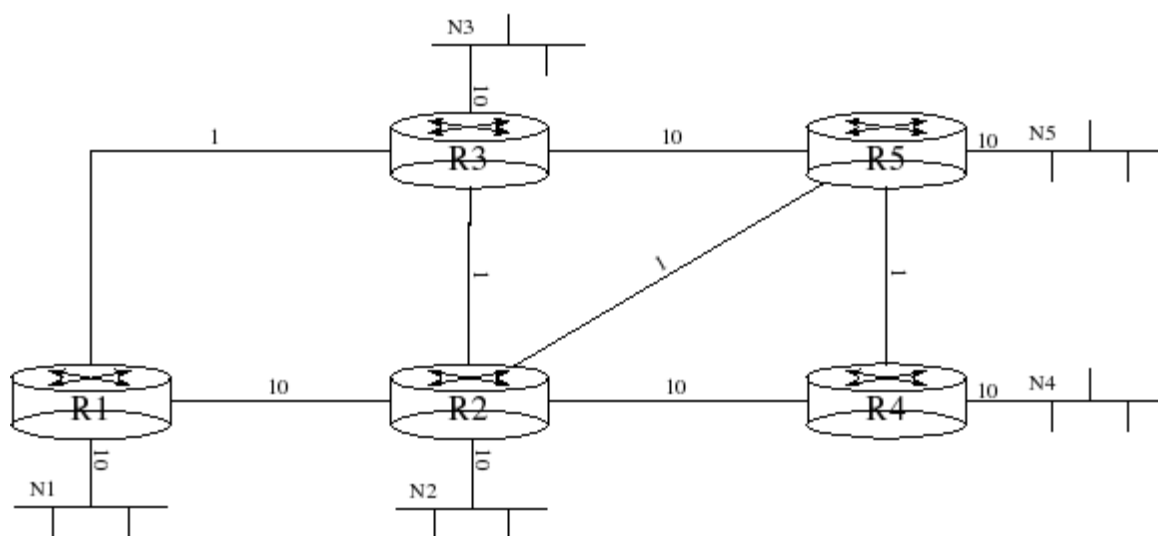
```
R>* 192.168.32.0/24 [120/3] via 172.17.0.2, eth1
```

Incident: l'interface eth1 de RA tombe en panne (le câble réseau est déconnecté !).

1 . Après un délai de convergence, donner la nouvelle table de routage de RA.

Exercice 8 – OSPF

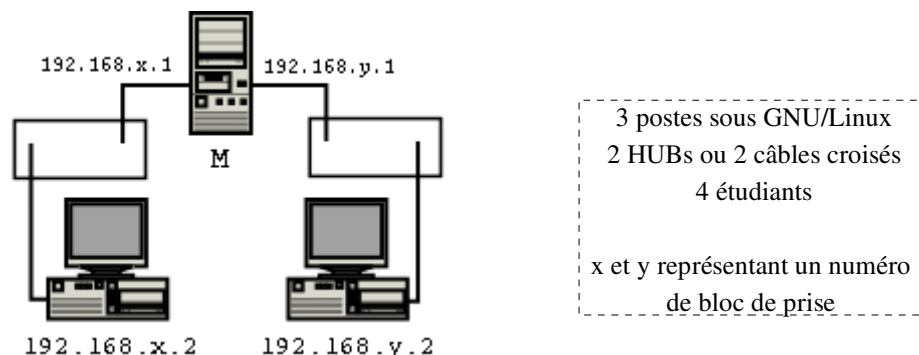
En reprenant la topologie suivante :



- 1 . Proposer une découpage en aires.
- 2 . Donner la carte topologique pour le routeur R4.

Exercice 9 - Routeur NAT

On utilise la maquette suivante en transformant la machine M en routeur NAT (*Network Address Translation*) :



Pour cela, on va ajouter le service **netfilter/iptables** sur la passerelle M pour utiliser la fonctionnalité de masquage d'adresse.

On utilisera la configuration suivante:

poste X	Routeur NAT		poste Y
130.9.202.81	130.9.202.45	192.168.8.44	192.168.8.46
<i>eth0</i>	<i>eth0</i>	<i>eth1</i>	<i>eth0</i>

Configuration d'iptables

La commande iptables qui permet de valider la fonction de masquage d'adresse :

```
# iptables -t nat -A POSTROUTING -s 130.9.0.0/16 -o eth1 -j MASQUERADE
```

La maquette est maintenant prête pour analyser le principe du masquage d'adresse à partir d'un **ping** de x.2 vers y.2. Pour capturer le trafic entre les deux postes, on utilisera **tcpdump** et/ou **ethereal** sur la machine M.

On réalise un **ping de 130.9.202.81 vers 192.168.8.46 soit X vers Y**

On capture le trafic sur les deux interfaces :

```
# tcpdump -e -vvv -i eth0
tcpdump: listening on eth0
0:c:6e:32:42:3a 0:30:84:9d:ec:49 ip 98: 130.9.202.81 > 192.168.8.46:
icmp: echo request (DF) (ttl 64, id 0, len 84)
0:30:84:9d:ec:49 0:c:6e:32:42:3a ip 98: 192.168.8.46 > 130.9.202.81:
icmp: echo reply (ttl 63, id 57317, len 84)

# tcpdump -e -vvv -i eth1
tcpdump: listening on eth1
0:30:84:9d:ea:56 0:30:84:9d:ec:70 ip 98: 192.168.8.44 > 192.168.8.46:
icmp: echo request (DF) (ttl 63, id 0, len 84)
0:30:84:9d:ec:70 0:30:84:9d:ea:56 ip 98: 192.168.8.46 > 192.168.8.44:
icmp: echo reply (ttl 64, id 57316, len 84)
```

On visualise la table NAT après le ping de 130.9.202.81 vers 192.168.8.46 :

```
# iptables -L -t nat -v
Chain PREROUTING (policy ACCEPT 54 packets, 7452 bytes)
 pkts bytes target      prot opt in      out     source
 destination

Chain POSTROUTING (policy ACCEPT 1 packets, 68 bytes)
 pkts bytes target      prot opt in      out     source
 destination
    2  168 MASQUERADE all  --  any    eth1    130.9.0.0/16
 anywhere

Chain OUTPUT (policy ACCEPT 1 packets, 68 bytes)
 pkts bytes target      prot opt in      out     source
 destination
```

- Le routage IP -

1 . Compléter, en tenant compte des captures réalisées, les tableaux ci-dessous (utilisez les lettres M, X et Y en lieu et place des adresses).

<i>trame 1</i>		<i>paquet IP</i>		<i>ICMP</i>
@ MAC DEST	@ MAC SRC	@ IP SRC	@ IP DEST	Type
<i>Signification</i>				
				echo request

<i>trame 2</i>		<i>paquet IP</i>		<i>ICMP</i>
@ MAC DEST	@ MAC SRC	@ IP SRC	@ IP DEST	Type
<i>Signification</i>				
				echo request

<i>trame 3</i>		<i>paquet IP</i>		<i>ICMP</i>
@ MAC DEST	@ MAC SRC	@ IP SRC	@ IP DEST	Type
<i>Signification</i>				
				echo reply

<i>trame 4</i>		<i>paquet IP</i>		<i>ICMP</i>
@ MAC DEST	@ MAC SRC	@ IP SRC	@ IP DEST	Type
<i>Signification</i>				
				echo reply

2 . Commenter les différences de fonctionnement entre un routeur classique et la maquette utilisant la passerelle M.

Annexe 1 - La trame Ethernet_II et 802.3

Trame 802.3

Préambule*	Délimiteur de trame*	Adresse destination	Adresse source	Longueur des données	Données	FCS
7 octets	1 octet	6 octets	6 octets	2 octets	46 à 1500 octets	4 octets

Trame Ethernet_II

Préambule*	Délimiteur de trame*	Adresse destination	Adresse source	Type <small>protocole de niveau supérieur</small>	Données	FCS
7 octets	1 octet	6 octets	6 octets	2 octets 0x0800 pour IP	46 à 1500 octets	4 octets

Remarque: * non capturé par un analyseur de protocoles

Annexe 2 - IP

Le format unique du paquet IP, ou datagramme IP, est organisé en champs de 32 bits :

0	7	15	16	31
Version	Longueur	Type de service	Longueur totale	
Identificateur			Drapeau	Offset
Durée de vie TTL	Protocole		Checksum de l'en-tête	
Adresse station source				
Adresse station destinatrice				
Options éventuelles			Bourrage	
Données (64 KO)				

Les différents champs de l'en-tête sont :

- Le champ Version (4 bits) identifie la version du protocole IP. Elle est fixée actuellement à 4.
- Un champ Internet Header Length (4 bits) spécifie la longueur de l'en-tête en mots de 32 bits. Cette longueur IHL varie de 5 à 15, 5 étant la longueur normale lorsqu'aucune option n'est utilisée.
- Le champ "type de service" TOS (8 bits) définit la priorité du paquet et le type de routage souhaité. Cela permet à un logiciel de réclamer différents types de performance pour un datagramme : délai court, haut débit, haute fiabilité ou bas prix.
- Le champ "longueur totale" (16 bits) définit le nombre d'octets contenus dans le paquet y compris l'en-tête IP. Puisque ce champ est codé sur 16 bits, la taille max. d'un paquet IP est de 65535 octets (64 KO).
- Le champ "Identification" (16 bits) contient une valeur entière utilisée pour identifier les fragments d'un datagramme. Ce champ doit être unique pour chaque nouveau datagramme.
- "Flags" (3 bits) est utilisé pour contrôler la fragmentation des paquets. Le bit de poids faible à zéro indique le dernier fragment d'un datagramme et est baptisé "More Flag" ou MF bit. Le bit du milieu est appelé "Do not Fragment flag" ou DF bit. Le bit de poids fort n'est pas utilisé.
- "Offset" (13 bits) sert à indiquer la position qu'occupait les données de ce fragment dans le message original.
- Le TTL ou "Time To Live" (8 bits) est l'expression en secondes de la durée maximale de séjour du paquet dans un réseau. Il existe 2 manières de faire baisser cette valeur : lors du réassemblage du paquet dans un routeur, sa valeur est décrémentée chaque seconde ou alors chaque routeur qui traite ce paquet décrémente le TTL d'une unité (compteur de routeurs). Si le TTL devient nul, son paquet IP n'est plus relayé : c'est souvent l'indication d'une erreur de paquet qui boucle. Une utilisation détournée de ce champ permet de tracer la route empruntée par un paquet. En mettant le champ TTL à 0, le premier routeur rencontré rejette le paquet et signale sa présence en retournant un paquet ICMP d'erreur vers l'émetteur. On renvoie alors le paquet avec le champ TTL à 1 afin d'atteindre le routeur suivant et ainsi de suite. A chaque fois, on récupère l'adresse IP du routeur.
- Le champ "protocole" (8 bits) identifie la couche de transport propre à ce datagramme :
17 pour UDP, 6 pour TCP, 1 pour ICMP, 8 pour EGP, 89 pour OSPF
- Le *checksum* ou champ de contrôle de l'en-tête (16 bits)
- Les adresses IP source et destination sont codées sur 32 bits
- A la rubrique "Options", sont stockées des demandes spéciales pour requérir un routage particulier pour certains paquets.
- Le champ "*padding*" est habituellement rempli de 0 de manière à aligner le début des données sur un multiple de 32 bits.

Annexe 3 - ICMP

Le paquet ICMP (*Internet Control Message Protocol* -- RFC 792), encapsulé dans un paquet IP (dont le champ protocole vaut 1 pour ICMP), a la structure suivante :

0	8	16	24	31
type	code	checksum		
... données complémentaires (n° id et n° seq) ...				
entête internet et données émises dans le paquet ICMP				

L'en-tête d'un paquet ICMP a une longueur de 8 octets dont les champs sont les suivants :

- Le champ **type** sur 1 octet, définis par les RFC 792 et 1256, dont les valeurs indiquant le type de message sont :

0	Réponse d'écho (<i>echo reply</i>)
3	Destination inaccessible
4	Source Quench
5	Redirection
8	Echo request (<i>echo request</i>)
9	Annonce de routeur
10	Sollicitation de routeur
11	TTL expiré
12	Problème de paramètre
13	Requête Horodatage
14	Réponse d'horodatage
15	Demande d'information
16	Réponse d'information
17	Requête de masque d'adresse
18	Réponse de masque d'adresse

- Le champ **code** sur 1 octet indique la sous-catégorie du message :

0-8

Les messages ICMP les plus courants sont le couple de type 0 et 8 générés par le programme de test "ping". Ping envoie un datagramme de type 8 (*echo request*) à un noeud dont il attend en retour un message de type 0 (*echo reply*) renvoyant les données incluses dans la requête.

3
Quand le "type" est par exemple 3 pour destination inaccessible, le "code" précise si c'est le réseau, l'hôte, le protocole ou le port qui sont inaccessibles :

0	Network unreachable
1	Host unreachable
2	Protocol unreachable
3	Port unreachable
4	Fragmentation needed and do not fragment bit set
5	Source route failed
7	Destination Host unknown
11	Network unreachable for type of service
12	Host unreachable for type of service
13	Communication administratively prohibited
14	Host precedence violation
15	Precedence cut-off in effect

- Le routage IP -

4

Un datagramme *Source Quench* est identique à celui du type *Destination Unreachable*. Il sert à contrôler un flux d'informations. Si un routeur détecte que son réseau ou son processeur ne peut suivre le débit d'une machine hôte émettrice, il envoie à celle-ci un message ICMP incluant la cause du dépassement de capacité.

0	Redirect datagram to go to that network
1	Redirect datagram to reach that host
2	Redirect datagram for that network with that TOS
3	Redirect datagram for that host with that TOS

5

Le datagramme *Route change request* est utilisé par les routeurs qui connaissent une meilleure route pour atteindre une destination particulière.

9-10

Le Router discovery protocol permet à un système d'être averti dynamiquement de la présence de tous les routeurs disponibles immédiatement sur un réseau LAN. Les messages de type 9, *router advertisement*, permettent à des routeurs de s'annoncer sur un réseau à intervalles de 7 à 10 minutes suite à un message de type 10, *router solicitation*, émis par une machine hôte.

11

Le message *Time exceeded for datagram* utilise un datagramme identique à celui du type *Destination Unreachable*. Un routeur l'utilise pour signaler à la machine source que la valeur TTL (Time To Live) d'une en-tête IP a été décrémentée jusqu'à la valeur d'expiration 0, ce qui revient à dire que le paquet a été écarté probablement à cause d'une boucle infinie dans le routage.

12

Le message ICMP *Parameter Problem* indique qu'un argument invalide a été utilisé dans le champ Options d'une en-tête IP.

13-14

Le type ICMP 13 pour *Time Stamp Request* et 14 pour *Time Stamp Reply* sont utilisés pour interroger l'horloge d'un système distant afin de s'y synchroniser ou récolter des informations statistiques.

15-16

Les messages *Information Request* est envoyé pour obtenir l'adresse réseau d'une machine hôte donnée. C'est la méthode utilisée par le protocole SLIP (Serial Line IP) pour allouer une adresse IP à la machine appelante.

17-18

Les messages *Address Mask Request* sont utilisés parallèlement à l'adressage en sous-réseau pour découvrir le masque de sous-réseau d'une machine hôte.

- Le champ **somme de contrôle** (*checksum*) sur 2 octets permet de valider les données
- Le champ **donnée complémentaire** sur 4 octets est divisé en deux champs de 16 bits contenant :
 - un numéro d'identification du paquet (pour distinguer 2 ping simultanément) ;
 - un numéro de séquence pour mesurer les temps aller et retour sur le réseau et les pertes.

Les données du paquet ICMP contiennent d'abord l'en-tête du paquet IP à l'origine du message (de 20 à 60 octets) puis des données quelconques.

Annexe 4 - RIP-II

Lorsque RIP est activé sur un routeur, il envoie une requête de mise à jour en diffusion et attend la réponse. Si un autre système RIP se trouve sur le même réseau physique, il envoie un message décrivant sa table de routage.

31		15	0
Commande	Version	Domaine de routage	
Identificateur de la famille d'adresse (AFI)		Route Tag	
Adresse du réseau			
Masque du réseau			
Adresse du routeur cible			
Métrique			
... (répété jusqu'à 25 fois) *			

Description des champs:

Commande : indique si le paquet est une requête ou une réponse. La requête est une demande pour obtenir la table de routage. La réponse peut être non sollicitée (cas des émissions régulières faites par les routeurs) ou sollicitée par une requête

Version : 2 actuellement (la version 1 de RIP n'est plus utilisée)

AFI (*Address Family Identifier*) : type de protocole

Route tag : marqueur qui peut être utilisé pour distinguer les routes apprises par RIP des routes apprises par d'autres protocoles (ex. OSPF)

Adresse du réseau : Adresse IP donnant le préfixe

Masque du réseau : champ binaire dont les bits positionnés à 1 donnent la longueur du préfixe

Adresse du routeur cible : adresse IP où il faut router les paquets à destination du réseau cible **Métrique** : valeur de la métrique (nombre compris entre 1 et 15)

Remarques:

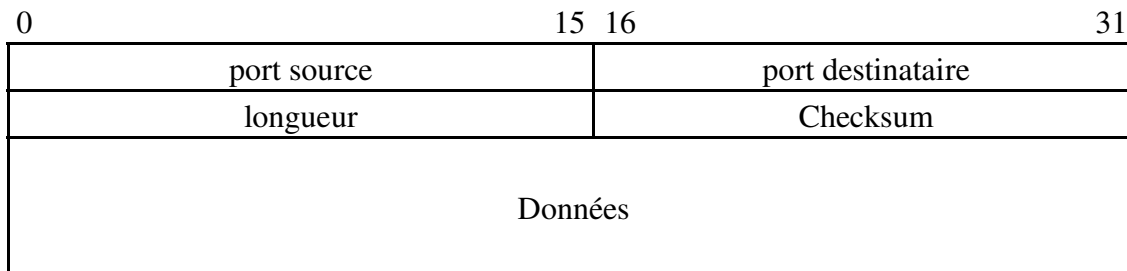
Un préfixe est constitué de l'ensemble {adresse du réseau , masque du réseau}.

Une route est constituée de l'ensemble des informations {AFI, route tag, préfixe, adresse IP du routeur cible, métrique}.

*Les paquets de type réponse peuvent contenir jusqu'à 25 routes par paquet. S'il y a plus de 25 routes à envoyer, plusieurs paquets sont émis.

Annexe 5 - UDP

UDP (*User Datagram Protocol*) permet à une application d'envoyer des messages à une autre application avec un minimum de fonctionnalités (pas de garantie d'arrivée, ni de contrôle de séquençement) :



Remarque: UDP n'accepte pas de datagramme de taille supérieure à 8KO.

Annexe 6 - TCP

0						7	8							15	16							23	24							31	
PORT SOURCE																PORT DESTINATION															
NUMERO DE SEQUENCE (SEQ)																															
NUMERO D'ACCUSE DE RECEPTION (ACK)																															
offset	Réservé							U	A	P	R	S	F	FENETRE (WINDOW)																	
							R	C	S	S	Y	I																			
							G	K	H	T	N	N																			
Checksum																Pointeur de données urgentes															
Options																							Bourrage								
DONNEES																															

Les différents champs de l'en-tête sont :

- *Source Port* (16 bits)
- *Destination port* (16 bits)
- *Sequence Number* (32 bits) compte les octets du flux de transmission de manière à identifier la position du premier octet de données d'un segment dans le flot des données initiales
- *Acknowledgment Number* (32 bits) contient le n° de séquence du prochain octet attendu
- *Offset* le déplacement en mots de 32 bits du début des données de l'application. La valeur par défaut est 5.
- *Flags* :
 - URG Urgent Pointer vers des données urgentes
 - ACK Acknowledge Field
 - PSH Push Flag pour passer immédiatement les données à la couche application
 - RST Reset Flag force la clôture d'une connexion après une erreur irrécupérable.
 - SYN Synchronize Flag pour synchroniser le démarrage d'une connexion entre deux noeuds.
 - FIN pour terminer une connexion
- *Window* (16 bits) annonce le nombre d'octet que le récepteur peut accepter.
- Le *checksum* ou champ de contrôle de l'en-tête (16 bits)
- *Urgent Pointer* (16 bits) pointe à la fin d'un champ de données considéré comme urgent
- Options (longueur variable)
- Le champ "*padding*" est habituellement rempli de 0 de manière à aligner le début des données sur un multiple de 32 bits.

Annexe 7 – IGMP

Le protocole IGMP (*Internet Group Management Protocol*) permet de gérer les déclarations d'appartenance à un ou plusieurs groupes auprès des routeurs *multicast*. Les inscriptions sont soit spontanées soit après requête du routeur. Pour cela, l'hôte envoie une trame IGMP destinées à ce ou ces groupes.

Il existe 2 version du protocole IGMP :

- IGMP version 1 : RFC 1112
- IGMP version 2 : RFC 2236

Structure de l'entête

Type	Temps de réponse max	Checksum	Adresse de groupe
1 octet	1 octet	2 octets	4 octets

Les différents champs de l'en-tête sont :

- **Type** : le champ Type est codé sur 8 bits et détermine la nature du message IGMP. Les 4 types de messages existant :
 - 11 - 00001011 - Requête pour identifier les groupes ayant des membres actifs.
 - 12 - 00001100 - Rapport d'appartenance au groupe émis par un membre actif du groupe (IGMP version 1)
 - 16 - 00010000 - Rapport d'appartenance au groupe émis par un membre actif du groupe (IGMP version 2)
 - 17 - 00010001 - Un membre annonce son départ du groupe
- **Temps de réponse max** : ce champ n'est utilisé que pour les messages de type 11. Il indique le temps d'attente maximum pour un client avant l'émission du rapport d'appartenance. L'unité utilisée est le 1/10 de seconde. Pour les autres types, ce champ est marqué à 0.
- **Checksum** : le champ Checksum est codé sur 16 bits
- **Adresse du groupe** : le champ Adresse du groupe est codé sur 32 bits et contient une adresse IP. Celle ci représente l'adresse du groupe d'appartenance ou 0 si l'inscription n'a pas encore eu lieu. Le type 11 place ce champ à 0 et les autres types marquent l'IP.

Annexe 8 - La multidiffusion IP

Le trafic IP multidestinataire (*multicast*) est envoyé vers une seule adresse IP mais est traité par plusieurs hôtes. Seuls les ordinateurs hôtes appartenant au groupe multidestinataire reçoivent et traitent le trafic IP envoyé à l'adresse IP réservée du groupe. L'ensemble des hôtes écoutant sur une adresse IP multidiffusion spécifique est appelé groupe multidiffusion.

Aspects importants de la multidiffusion IP :

- Les membres du groupe sont dynamiques, ce qui permet aux hôtes de rejoindre et quitter le groupe à tout moment.
- La capacité des hôtes à se joindre aux groupes multidestinaires est exploitée via l'envoi de messages IGMP.
- Les groupes ne sont pas limités en taille et les membres peuvent être répartis à travers plusieurs réseaux IP (si les routeurs de connexion prennent en charge la propagation du trafic IP multidiffusion et des informations sur les membres du groupe).
- Un hôte peut envoyer le trafic IP vers l'adresse IP du groupe même s'il n'appartient pas au groupe correspondant.

Adresses *multicast*

Les adresses IP multidestinaires sont réservées et affectées à partir de la plage d'adresses de la **classe D** comprise entre **224.0.0.0** et **239.255.255.255**.

Le tableau suivant présente une liste partielle d'adresses de la classe D connues et enregistrés avec l'IANA (*Internet Assigned Numbers Authority*) :

224.0.0.0	Adresse de base (réservée)
224.0.0.1	Le groupe multidestinataire de tous les hôtes qui contient tous les systèmes sur le même segment de réseau.
224.0.0.2	Le groupe multidestinataire de tous les routeurs qui contient tous les routeurs sur le même segment de réseau.
224.0.0.5	Adresse AllSPFRouters OSPF (<i>Open Shortest Path First</i>) Permet d'envoyer des informations sur le routage OSPF vers tous les routeurs OSPF d'un même segment de réseau.
224.0.0.6	Adresse AllDRouters OSPF Permet d'envoyer des informations sur le routage OSPF vers les routeurs OSPF d'un même segment de réseau.
224.0.0.9	Adresse du groupe RIP version 2 Permet d'envoyer des informations sur le routage RIP vers tous les routeurs RIP v2 d'un même segment de réseau.
224.0.1.24	Adresse du groupe de serveur WINS. Permet de prendre en charge une récupération automatique et une configuration dynamique de réplication pour les serveurs WINS