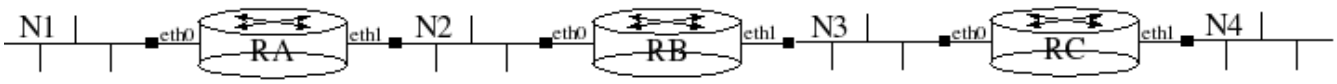
Table des matières

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

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

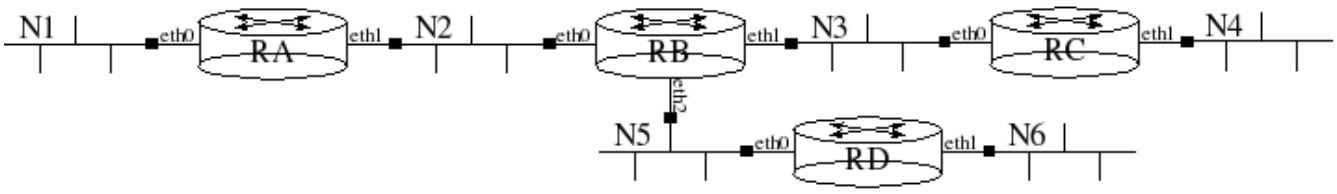
Routeur A				Routeur B				Routeur C			
Destination	Passerelle	Metric	Iface	Destination	Passerelle	Metric	Iface	Destination	Passerelle	Metric	Iface
N1	*	0	eth0								
N2	*	0	eth1								

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				

Remarque:

- **RB a ajouté une route directe vers N5 pour son interface eth2**

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			

Remarque:

- **RD a deux routes directes vers N5 et N6 pour ses interfaces eth0 et eth1**

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)

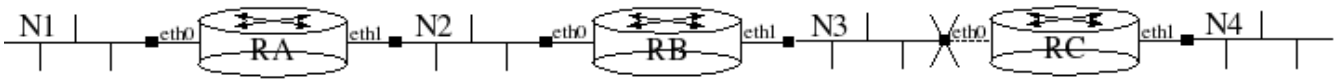
Routeur A					Routeur C			
Destination	Passerelle	Metric	Iface		Destination	Passerelle	Metric	Iface
N1	*	0	eth0		N1			
N2	*	0	eth1		N2			
N3	RB	1	eth1		N3			
N4	RB	2	eth1		N4			
N5					N5			
N6					N6			

Remarque:

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

Exercice 3 – *Split Horizon* et *Reverse Poison*

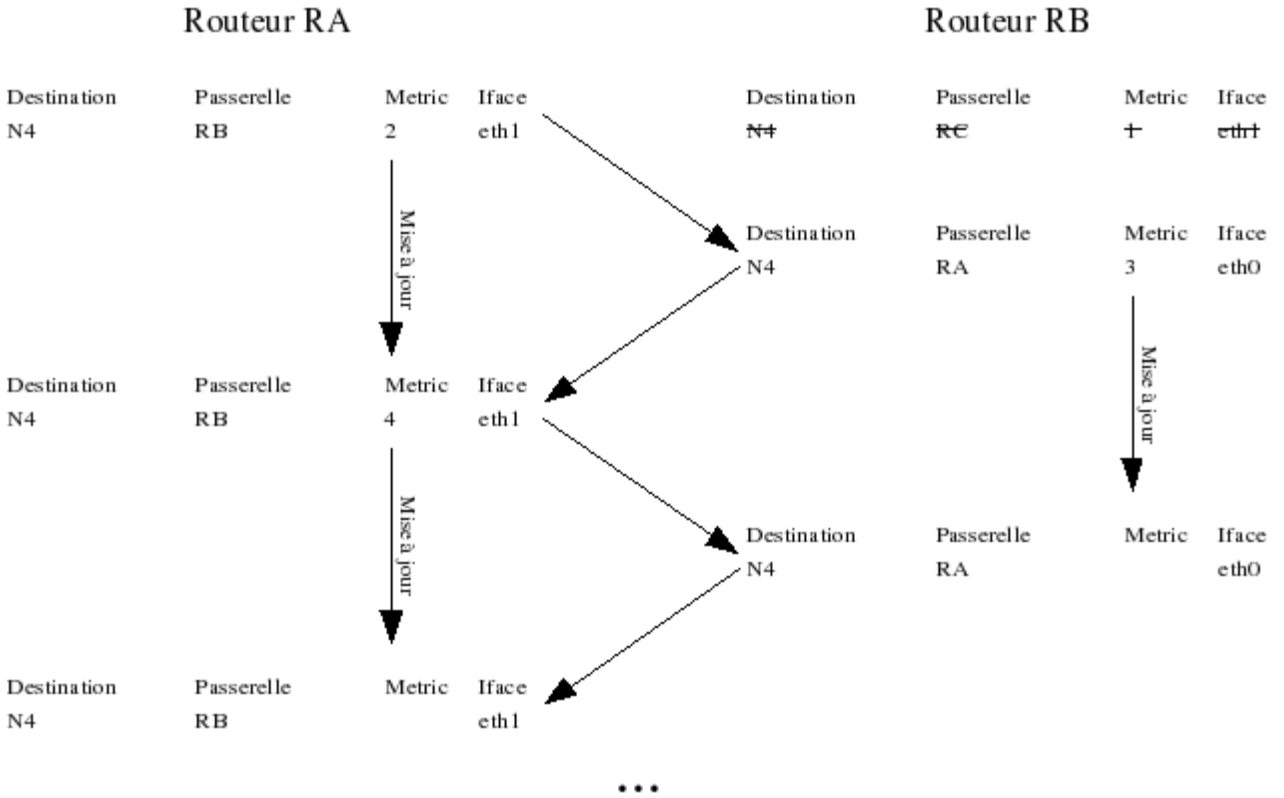
On va observer le fonctionnement d'un routage dynamique de type *Distant-Vector* (exemple RIP).
On reprend la maquette de l'exercice n°1 :



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



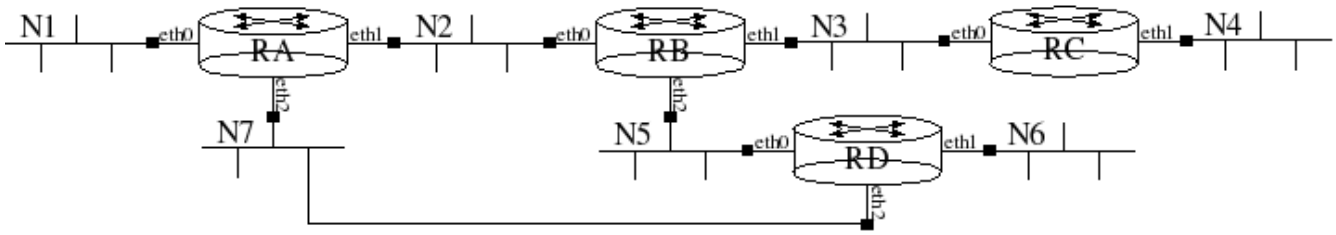
Problème: la *metric* pour atteindre N4 augmente indéfiniment (la convergence prend un temps infini).
Solution 1: il faut fixer une limite. Dans RIP, la *metric* limite est fixée à 16 et correspond à une route inaccessible.

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.

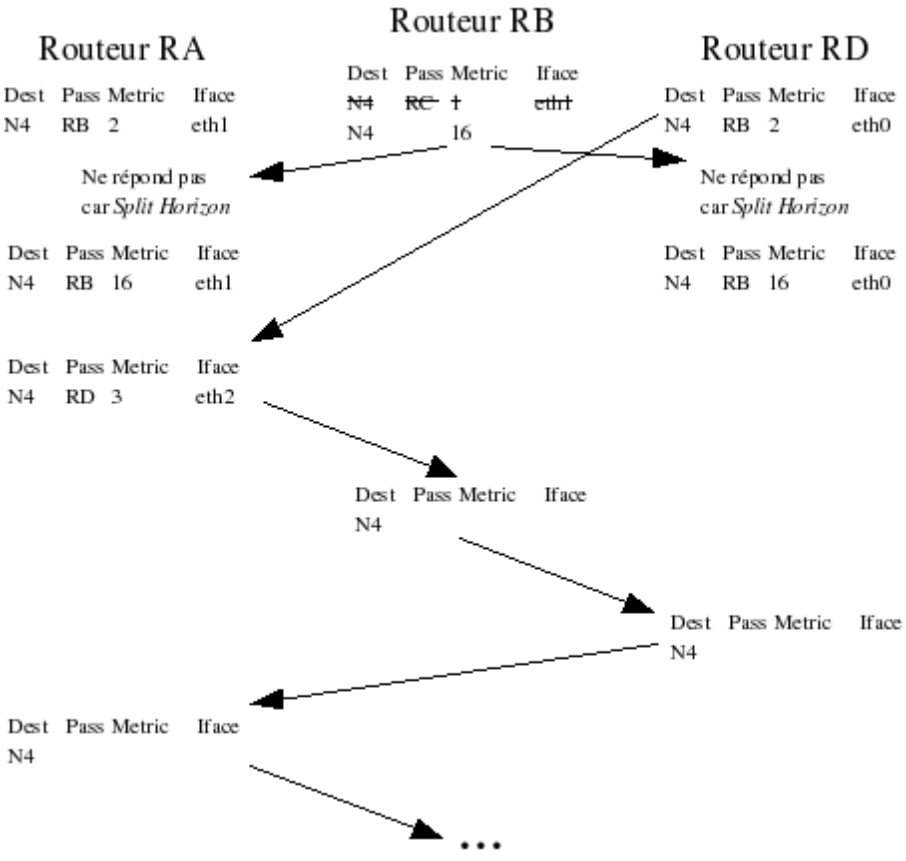
Amélioration: *Poison Reverse* (Retour Empoisonné)
Les routes en provenance d'un voisin lui sont ré-annoncées avec une métrique infinie

Est-ce suffisant ?
Le même incident sur le routeur C (son interface eth0 est *down*) mais sur le domaine suivant :

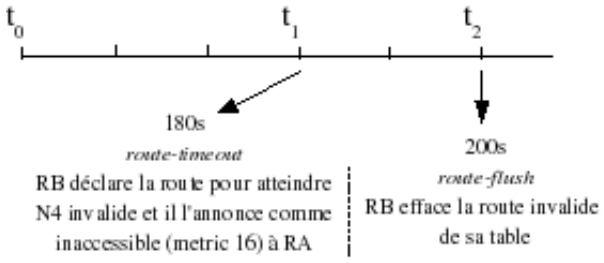


Donc, le routeur RB prévient RA et RD que la route pour joindre NA est inaccessible (*metric* de 16). RA et RC ne répondent pas grâce au Split Horizon. Par contre, RD a communiqué à RA qu'il a un moyen d'atteindre N4 en passant par RB. Puis, RA indique à RB qu'il a une route pour atteindre N4 en passant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



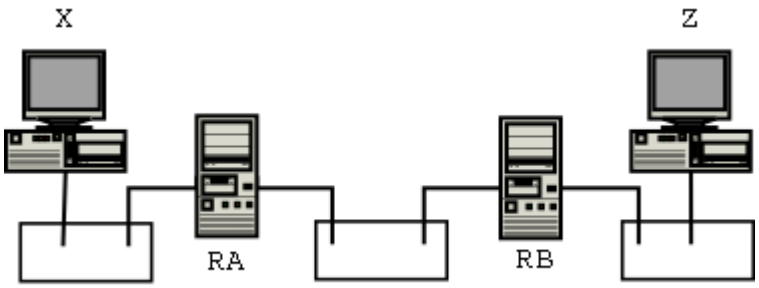
Problème: le temps de convergence est (trop) long.

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
eth0	*eth0*	*eth1*	*eth0*	*eth1*	*eth0*

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

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				

1 . Réaliser les tests suivants à partir du poste X

Rappel: un *ping* correspond à l'émission d'un paquet **ICMP *echo request*** et à la réception d'un paquet **ICMP *echo reply***

ping à partir du poste X	Résultat (OUI/NON)	Commentaires et solution proposée en cas d'échec
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.

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

Remarque: **La redirection** (*ICMP Redirect*)

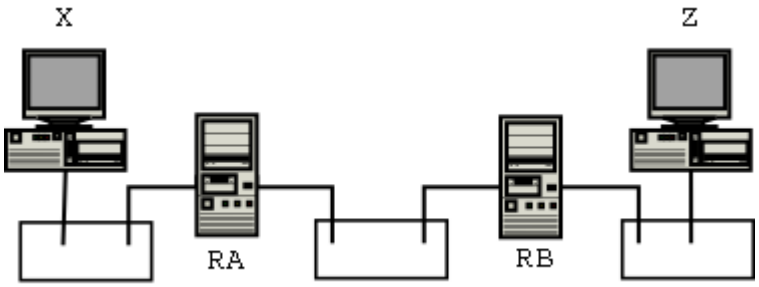
Le plus souvent, la configuration d'un poste (qui n'est pas routeur) se limite à indiquer la route par défaut qui lui permet de sortir de son réseau local. Sur ce réseau local, il peut y avoir plusieurs routeurs (cas de Y, RA et RB), mais dans la mesure, où le routeur par défaut possède une tale de routage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

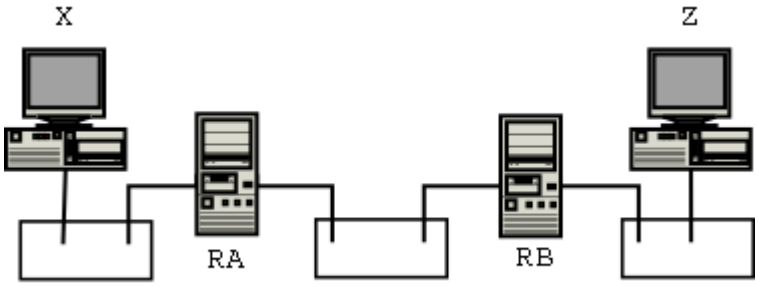
Poste X	**Routeur RA**		**Routeur RB**		**Poste Z**
130.9.202.81	130.9.202.45	192.168.8.44	192.168.8.24	192.168.16.25	192.168.16.46
eth0	*eth0*	*eth1*	*eth0*	*eth1*	*eth0*

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
```



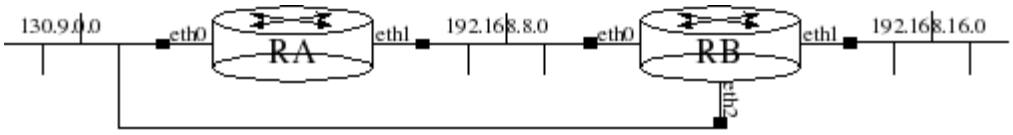
1 . Compléter les champs adresses IP source et destination, TLL et le type ICMP manquants.

```
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
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
```

3 . En conclusion, décrire brièvement le principe de la commande *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,v 1.11 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]  Inteface 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 eth0 -nt -s 0
tcpdump: listening on eth0
(a) 192.168.8.2.520 > 224.0.0.9.520:  RIPv2-req 24 (DF) [ttl 1]

(b) 192.168.8.2 > 224.0.0.22: igmp v3 report, 1 group record(s) (DF) [tos
0xc0]  [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 . Quelles sont les routes annoncées par RB ? Combien de routes sont apprises par RA ?

2 . Justifier l'utilisation du champ TLL à 1.

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 irtt 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
```

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

b . ethereal sur eth0 de RB

01	00	5E	00	00	09	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
00	01	00	02	00	00	C0	A8	10	00	FF	FF	FF	00	00	00
00	00	00	00	00	01										

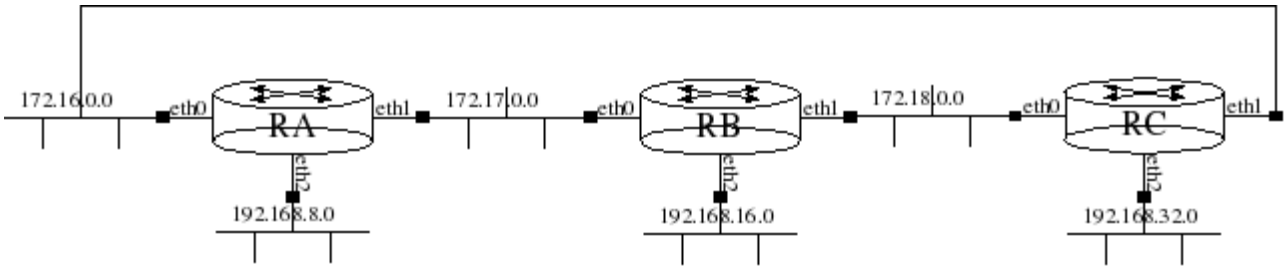
1 . Décoder cette trame.

2 . Représenter les protocoles encapsulés dans cette trame.

3 . 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 :



Les adresse IP des interfaces de chacun des routeurs sont:

Routeur RA		**Routeur RB**		**Routeur RC**	
172.16.0.1	172.17.0.1	172.17.0.2	172.18.0.1	172.18.0.2	172.16.0.2
eth0	*eth1*	*eth0*	*eth1*	*eth0*	*eth1*

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 !).

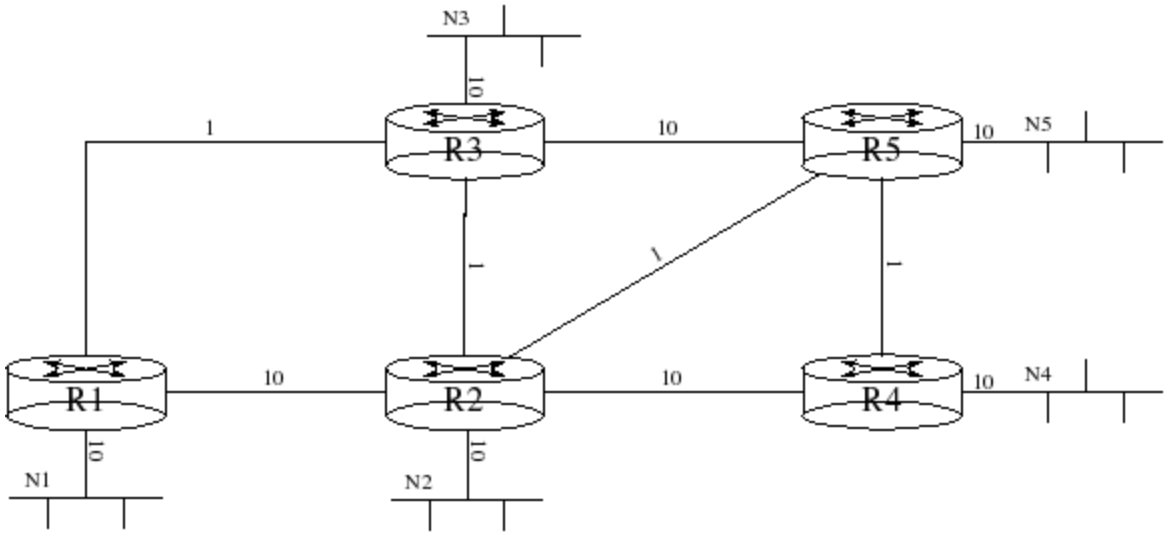
1 . Après un délai de convergence, donner la nouvelle table de routage de RA.

Routeur RA			
Destination	Passerelle	*Metric*	*Iface*

2 . Avec sa nouvelle table de routage, l'ensemble des réseaux sont-ils accessibles à partir de RA ?

Exercice 8 – OSPF

En reprenant la topologie suivante :



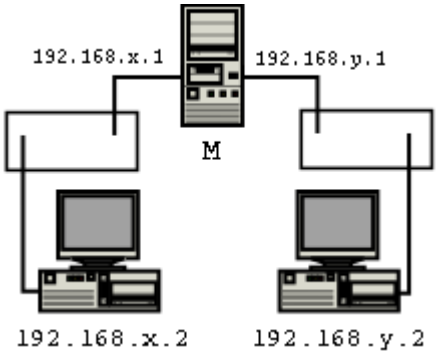
1 . Proposer une découpage en aires.

2 . Donner la carte topologique pour le routeur R4.

Exercice 9 - Routeur NAT

Remarque: lire l'annexe 9 sur le masquage et la translation d'adresses.

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.

Configuration d'iptables

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

```
# iptables -t nat -A POSTROUTING -s 192.168.x.0/24 -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 192.168.x.2 vers 192.168.y.2 soit X vers Y** et on capture le trafic sur les deux interfaces de M:

```
# tcpdump -e -vvv -i eth0
tcpdump: listening on eth0
0:c:6e:32:42:3a 0:30:84:9d:ec:49 ip 98: 192.168.x.2 > 192.168.y.2: 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.y.2 > 192.168.x.2: 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.y.1 > 192.168.y.2: 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.y.2 > 192.168.y.1: icmp:
echo reply (ttl 64, id 57316, len 84)
```

Remarque: Les trames 1 et 4 sont capturées sur ***eth0*** et les trames 2 et 3 sur ***eth1***.

On visualise la table NAT après le *ping* de 192.168.x.2 vers 192.168.y.2 :

```
# 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     192.168.x.0/24 anywhere
Chain OUTPUT (policy ACCEPT 1 packets, 68 bytes)
 pkts bytes target     prot opt in     out     source   destination
```

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).

trame 1		***paquet IP***		***ICMP***
@ MAC DEST	@ MAC SRC	@ IP SRC	@ IP DEST	Type
				echo request

trame 2		***paquet IP***		***ICMP***
@ MAC DEST	@ MAC SRC	@ IP SRC	@ IP DEST	Type
				echo request

trame 3		***paquet IP***		***ICMP***
@ MAC DEST	@ MAC SRC	@ IP SRC	@ IP DEST	Type
				echo reply

trame 4		***paquet IP***		***ICMP***
@ MAC DEST	@ MAC SRC	@ IP SRC	@ IP DEST	Type
				echo reply

2 . Commenter les différences de fonctionnement entre un routeur classique et la maquette utilisant la passerelle M. Pour vous aider, compléter le même tableau mais dans le cas d'un simple routeur **R** et non un routeur NAT M.

Utilisez les lettres **R**, X et Y en lieu et place des adresses:

trame 1		***paquet IP***		***ICMP***
@ MAC DEST	@ MAC SRC	@ IP SRC	@ IP DEST	Type
				echo request

trame 2		***paquet IP***		***ICMP***
@ MAC DEST	@ MAC SRC	@ IP SRC	@ IP DEST	Type
				echo request

trame 3		***paquet IP***		***ICMP***
@ MAC DEST	@ MAC SRC	@ IP SRC	@ IP DEST	Type
				echo reply

trame 4		***paquet IP***		***ICMP***
@ MAC DEST	@ MAC SRC	@ IP SRC	@ IP DEST	Type
				echo reply

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

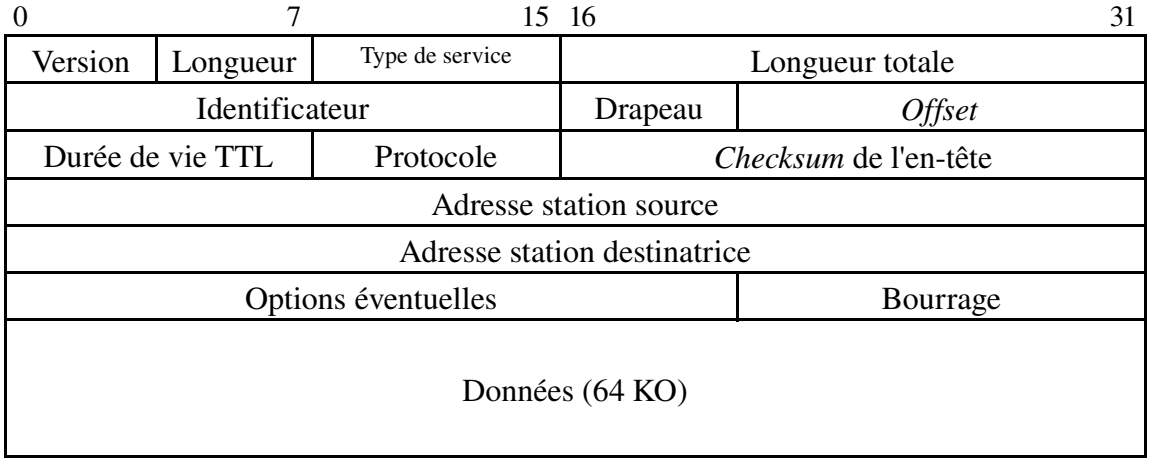
Trame Ethernet_II

Préambule*	Délimiteur de trame*	Adresse destination	Adresse source	Type protocole de niveau supérieur	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 :



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 pâque 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 (*echo reply*)
3	Destination inaccessible
4	Source Quench
5	Redirection
8	Echo request (*echo request*)
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

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 sollicitation*, 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** (*chechsum*) 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 contient 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 (= 1) ou une réponse (= 2). 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:

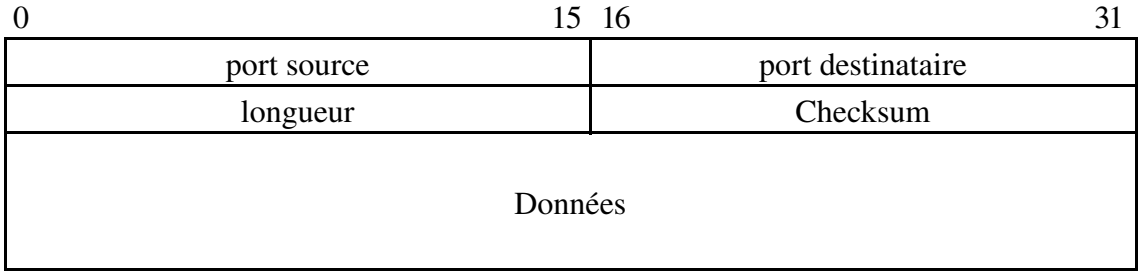
Un préfixe est constitué de l'ensemble {adresse du réseau , masque du réseau}.

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 séquencement) :



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

Annexe 6 - TCP

<table>
<tr><td colspan="32">0 7 8 15 16 23 24 31</td></tr>
<tr><td colspan="16">PORT SOURCE</td><td colspan="16">PORT DESTINATION</td></tr>
<tr><td colspan="32">NUMERO DE SEQUENCE (SEQ)</td></tr>
<tr><td colspan="32">NUMERO D'ACCUSE DE RECEPTION (ACK)</td></tr>
<tr><td colspan="4">offset</td><td colspan="6">Réservé</td><td>U
R
G</td><td>A
C
K</td><td>P
S
H</td><td>R
S
T</td><td>S
Y
N</td><td>F
I
N</td><td colspan="16">FENETRE (WINDOW)</td></tr>
<tr><td colspan="16">Checksum</td><td colspan="16">Pointeur de données urgentes</td></tr>
<tr><td colspan="24">Options</td><td colspan="8">Bourrage</td></tr>
<tr><td colspan="32">DONNEES</td></tr>
</table>

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 envoi 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 joindre et quitter le groupe à tout moment.
- La capacité des hôtes à se joindre aux groupes multidestinataires est exploitée via l'envoi de messages IGMP.
- Les groupes ne sont pas limités en taille et les membres peuvent être répandus 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 multidestinataires 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 (*Open Shortest Path First*)** **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

Annexe 9 - 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

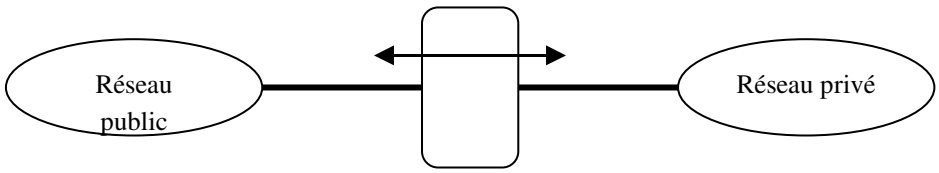
Le besoin se pose essentiellement lorsqu'on interconnecte un réseau public (Internet) avec un réseau privé. La situation courante veut qu'on utilise des adresse privées pour le réseau local.

Le réseau privé ne pourra pas utiliser n'importe lequel 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 adresse réservées peuvent être découpées en sous-réseaux privés.
Par contre, les postes privés ont peut être le 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 Adress Translation*)

La machine, ayant a 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 (**S**NAT)
- la modification d'adresse destination (**D**NAT)

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 privilégiée 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 set en particulier à réaliser des proxy transparents, comme par exemple la redirection du trafic HTTP vers un proxy local (*squid* par exemple)