Table des matières

Les fichiers de configurations

Configuration de la résolution de noms

Le fichier **/etc/host.conf** contient des informations spécifiques pour la configuration de la bibliothèque de résolution de noms. Elle doit contenir un mot-clé de configuration par ligne, suivi par l'information appropriée. Les mots-clés reconnus sont **order**, trim, **multi**, nospoof et reorder (faire un man host.conf pour obtenir plus d'informations) :

- **order** : Ce mot-clé indique dans quel ordre la résolution des noms d'hôtes doit avoir lieu. Il doit être suivi par une ou plusieurs méthodes séparées par des virgules. Ces méthodes sont **bind, hosts et nis**.
- **multi** : Les valeurs valides sont on et off. Si cette option est sur on, la bibliothèque resolv+ renverra toutes les adresses valides pour un hôte apparaissant dans le fichier /etc/hosts plutôt que de ne renvoyer que la première. Par défaut elle est sur off car cela peut causer des dégradations sensibles des performances sur les sites ayant un gros fichier hosts.

```
$ ls -l /etc/host.conf
-rw-r--r--    2 root     root           26 fév 12  2003 /etc/host.conf

$ cat /etc/host.conf
order hosts,bind
multi on
```

Correspondances statiques de noms d'hôtes

Le fichier **/etc/hosts** est un fichier de texte simple qui associe les adresses IP avec les noms d'hôtes, une ligne par adresse IP. Pour chaque hôte, une unique ligne doit être présente, avec les informations suivantes :

Adresse_IP Nom_officiel [Alias...]

Les divers champs de la ligne sont séparés par un nombre quelconque d'espaces ou de tabulations.
Les noms d'hôtes peuvent contenir n'importe quel caractère alphanumérique, le signe moins ("-") et les points (".") . Ils doivent commencer par un caractère alphabétique et se terminer par un caractère alphanumérique. Les alias permettent de disposer de noms différents, d'orthographes simplifiées, de surnoms plus courts, ou de noms d'hôte générique (comme localhost). Le format de la table des noms d'hôtes est décrit dans la RFC 952.

Le système Berkeley Internet Name Domain (BIND) implémente un serveur de noms Internet (**DNS**) pour les systèmes Unix. Il remplace ou complète le fichier /etc/hosts ou la recherche des noms d'hôtes, et libère un hôte de la nécessité de disposer d'un fichier /etc/hosts complet et à jour.

Dans les systèmes modernes, même si la table des hôtes a été remplacée par DNS, ce mécanisme est encore largement employé pour :

- initialiser une machine : beaucoup de systèmes ont un petit fichier contenant le nom et l'adresse des hôtes important sur le réseau local. Ceci est utile lorsque le DNS n'est pas en marche, notamment lors de la mise en route des systèmes.
- NIS : les sites employant NIS utilisent la table d'hôtes comme entrée pour la base de données des hôtes NIS. Même si NIS peut être employé avec un DNS, la plupart des sites NIS conservent encore la table des noms d'hôtes avec une entrée pour toutes les machines locales, à des fins de secours.
- noeud isolés : les petits sites, isolés des réseaux importants emploient la table d'hôtes à la place du DNS. Si les informations locales changent rarement, et si le réseau n'est pas connecté à Internet, le DNS n'offre pas beaucoup d'intérêt.

```
$ ls -l /etc/hosts
-rw-r--r--    1 root     root           38 sep  6 10:58 /etc/hosts

$ cat /etc/hosts
127.0.0.1       localhost
...
```

Base de données adresses Ethernet - adresses IP

Le fichier **/etc/ethers** contient des adresses Ethernet sur 48 bits et leur adresse IP correspondante, une ligne par adresses IP : Adresse-Ethernet Adresse-IP
L'adresse-Ethernet est écrite sous la forme x:x:x:x:x:x, où x est une valeur héxadecimale comprise entre 0 et ff représentant un octet de l'adresse. L'Adresse-IP peut être soit un nom d'hôte résolu par DNS ou une adresse en notation décimale pointée.
Si ce fichier existe, il sera lu par le script de démarrage du service réseau afin d'initialiser la table ARP de la machine.

Fichier de configuration de la résolution de noms

Le fichier de configuration de la résolution de noms **/etc/resolv.conf** contient des informations qui sont lues par les routines de résolution la première fois qu'elles sont invoquées par un processus. Le fichier est prévu pour être lisible par des humains et contient une liste de mots-clés avec des valeurs.
Les différentes options de configuration sont :

- **nameserver** : adresse Internet (en notation pointée) du serveur de noms qui sera interrogé.
- **domain** : nom du domaine local
- search : liste de recherche pour les noms d'hôte
- ...

```
$ ls -l /etc/resolv.conf
-rw-r--r--    1 root     tv             83 déc  9 08:35 /etc/resolv.conf

$ cat /etc/resolv.conf
domain btsiris.net
nameserver 192.168.52.83
```

Fichier de définition des protocoles internet

/etc/protocols est un fichier de texte ASCII, décrivant les différents protocoles internet DARPA qui sont disponibles avec le sous-système TCP/IP.
Ces numéros se trouvent dans le champ protocole des en-têtes IP.

Chaque ligne a le format suivant :

protocole numéro alias ...

où les champs sont séparés par des espaces ou des tabulations.

La description des champs est la suivante :

- protocole : le nom original du protocole, par exemple ip, tcp, ou udp.
- numéro : le numéro officiel du protocole, tel qu'il apparaîtra dans l'en-tête IP.
- alias : des synonymes éventuels pour le nom du protocole.

Les fonctions de la bibliothèque C getprotoent, getprotobyname, getprotobynumber, setprotoent et endprotoent permettent d'accéder aux protocoles de ce fichier depuis un programme.

```
$ ls -l /etc/protocols
-rw-r--r--   1 root     root      715 Jan  1  1980 /etc/protocols

$ cat /etc/protocols
ip      0       IP      # Protocole internet
icmp    1       ICMP    # Protocole internet pour messages de contrôle
igmp    2       IGMP    # Protocole internet pour multi-cast.
ggp     3       GGP     # Protocole passerelle/passerelle
tcp     6       TCP     # Protocole de contrôle de transmission
pup     12      PUP     # Protocole universel PARC
udp     17      UDP     # Protocole pour Datagrammes
raw     255     RAW     # Interface IP directe.
...
```

Liste des services internet

/etc/services est un fichier de texte ASCII fournissant une correspondance entre un nom décrivant un service internet et l'ensemble numéro de port / protocole utilisé.
Chaque programme réseau devrait consulter ce fichier pour obtenir le numéro de port et le protocole sous-jacent au service qu'il fournit.
Les fonctions de la bibliothèque C getservent, getservbyname, getservbyport, setservent, et endservent(3) permettent d'interroger ce fichier depuis un programme.
Les numéros de ports sont affectés par le IANA (*Internet Assigned Numbers Authority*), et leur politique actuelle consiste à assigner à la fois les protocoles TCP et UDP à chaque numéro de port. Ainsi la plupart des services auront deux entrées, même si elles n'utilisent que le protocole TCP.
Les numéros de ports en-dessous de 1024 ne peuvent être assignés à une socket que par un programme Super-User.

Chaque ligne décrivant un service est de la forme :

service-name port/protocole [alias ...]

où :

- service-name : est le nom intelligible du service. La différence majuscule/minuscule est importante. Souvent le programme client possède un nom rappelant celui du service.
- port : est le numéro de port (en décimal) utilisé pour ce service.
- protocol : est le type de protocole utilisé, il doit s'agir d'un nom déclaré dans le fichier /etc/protocols. Les protocoles les plus courants sont tcp et udp.
- alias : est une liste éventuelle d'autres noms se référant au même service, séparés par des espaces ou des tabulations. Encore une fois, la différence majuscule/minuscule est importante.

```
$ ls -l /etc/services
-rw-r--r--   1 root   root   3432 Feb 16  1996 /etc/services

$ cat /etc/services
...
netstat         15/tcp
qotd            17/tcp          quote
msp             18/tcp          # message send protocol
msp             18/udp          # message send protocol
chargen         19/tcp          ttytst source
chargen         19/udp          ttytst source
ftp             21/tcp
telnet          23/tcp
...
```

Récapitulatif

Fichier	*Description*
/etc/host.conf	Configuration de la résolution de noms
/etc/hosts	Correspondances statiques de noms d'hôtes
/etc/ethers	Base de données adresses Ethernet - adresses IP
/etc/resolv.conf	Fichier de configuration de la résolution de noms
/etc/protocols	Fichier de définition des protocoles internet
/etc/services	Liste des services internet

Les commandes de base

Commandes	*Description*
hostname	affiche ou définit le nom d'hôte du système
domainname	affiche le nom de domaine NIS/YP du système
dnsdomainname	affiche le nom de domaine du système
nisdomainname	affiche ou définit le nom de domaine NIS/YP du système
ypdomainname	affiche ou définit le nom de domaine NIS/YP du système
nodename	affiche ou définit le nom de domaine DECnet du système

Commandes	*Description*
finger, pinky	rechercher des informations sur un utilisateur
rup, ruptime	affiche l'état des machines du réseau local
rusers	affiche les utilisateurs connectés sur les machines du réseau local
rwho	affiche les utilisateurs connectés sur les machines du réseau local
w	affiche les utilisateurs connectés et ce qu'ils font
lsof	liste les fichiers ouverts

Commandes	*Description*
ifcfg	Configuration file for network interfaces
ifconfig	**configure une interface réseau**
ifdown	désactive une interface réseau
ifup	active une interface réseau
netstat	**Affiche les connexions réseau, les tables de routage, les statistiques des interfaces, les connexions masquées, les messages netlink, et les membres multicast.**
nmap	**Outil d'exploration réseau et analyseur de sécurité**
ping, ping6	**envoyer des datagrammes ICMP ECHO_REQUEST à des hôtes sur un réseau**
traceroute	**print the route packets take to network host**
tracepath, tracepath6	**traces path to a network host discovering MTU along this path**
route	**affiche et manipule la table de routage IP**
arp	**manipule la table ARP du système**
arping	**send ARP REQUEST to a neighbour host**
ip	**TCP/IP interface configuration and routing utility**

Autres commandes concernant le réseau:

```
# apropos network
dhclient-script      (8)  - DHCP client network configuration script
nameif               (8)  - name network interfaces based on MAC addresses
netconf              (8)  - network client and server configuration
```

```
netreport           (1)  - request notification of network interface changes
rdisc               (8)  - network router discovery daemon
routed              (8)  - network routing daemon
slattach            (8)  - attach a network interface to a serial line
socket              (n)  - Open a TCP network connection
usernetctl          (8)  - allow a user to manipulate a network interface if permitted
zcip                (8)  - zero configure network interface
```

Il est aussi possible d'ajouter des outils spécifiques, comme par exemple **ethtool** qui permet d'afficher ou configurer les paramètres d'une carte Ethernet (*package* ethtool-1.6-1mdk.rpm pour une Mandrake) :

```
# ethtool -i eth0
driver: ne2k-pci
version: 1.02
firmware-version:
bus-info: 00:09.0

# ethtool eth0
Settings for eth0:
No data available
```

L'interface proc

/proc est un pseudo-système de fichiers qui est utilisé comme interface avec les structures de données du noyau. La plupart des fichiers sont en lecture seule, mais quelques uns permettent la modification de variables du noyau. On peut aussi utiliser la commande sysctl pour configurer les paramètres du noyau.
Le répertoire **/proc/net** regroupe divers pseudo-fichiers relatifs aux fonctionnalités réseau. Chaque fichier fournit des informations concernant une couche particulière. Ces fichiers sont en ASCII et sont donc lisibles grâce à la commande **cat**, mais le programme standard **netstat** fournit un accès plus propre à ces données.

```
# ls -l /proc/net/
-r--r--r--    1 root     root            0 déc  9 11:51 arp
dr-xr-xr-x    2 root     root            0 déc  9 11:51 atm/
-r--r--r--    1 root     root            0 déc  9 11:51 dev
-r--r--r--    1 root     root            0 déc  9 11:51 dev_mcast
dr-xr-xr-x    2 root     root            0 déc  9 11:51 drivers/
-r--r--r--    1 root     root            0 déc  9 11:51 igmp
...
-r--r--r--    1 root     root            0 déc  9 11:51 netlink
-r--r--r--    1 root     root            0 déc  9 11:51 netstat
-r--r--r--    1 root     root            0 déc  9 11:51 packet
-r--r--r--    1 root     root            0 déc  9 11:51 psched
-r--r--r--    1 root     root            0 déc  9 11:51 raw
-r--r--r--    1 root     root            0 déc  9 11:51 route
dr-xr-xr-x    2 root     root            0 déc  9 11:51 rpc/
...
-r--r--r--    1 root     root            0 déc  9 11:51 snmp
-r--r--r--    1 root     root            0 déc  9 11:51 sockstat
-r--r--r--    1 root     root            0 déc  9 11:51 softnet_stat
-r--r--r--    1 root     root            0 déc  9 11:51 tcp
-r--r--r--    1 root     root            0 déc  9 11:51 tr_rif
-r--r--r--    1 root     root            0 déc  9 11:51 udp
-r--r--r--    1 root     root            0 déc  9 11:51 unix
-r--r--r--    1 root     root            0 déc  9 11:51 wireless
```

- /proc/net/arp : ce fichier contient un affichage ASCII lisible des tables ARP du noyau servant à la résolution d'adresse. Il indique à la fois les entrées apprises dynamiquement et celles pré-programmées. Le format est le suivant : Adresse IP Matériel Attribut Adresse matérieuue Masque Périph.

```
# cat /proc/net/arp
IP address       HW type     Flags       HW address            Mask     Device
```

- /proc/net/dev : ce pseudo-fichier contient des informations d'état sur les périphériques réseau. On y trouve les nombres de paquets émis et reçus, le nombre d'erreurs et de collisions, ainsi que d'autres données statistiques. Ce fichier est utilisé par le programme **ifconfig**.

```
# cat /proc/net/dev
Inter-|   Receive                                                |  Transmit
face |bytes    packets errs drop fifo frame compressed multicast|bytes    packets errs drop fifo colls carrier compressed
lo:   27552     396    0    0    0     0          0         0    27552     396    0    0    0     0       0          0
eth0:     0       0    0    0    0     0          0         0    96629       0  688    0    0 11696    1376          0
```

Les paramètres réseaux du noyau sont dans le répertoire **/proc/sys/net/** :

```
802/          core/         ethernet/     ipv4/         token-ring/   unix/
```

Modifier une option réseau du noyau, par exemple :

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

Le démarrage du service réseau

L'interface eth0

Avec la commande **dmesg** qui affiche le tampon des messages du noyau, on peut vérifier par exemple la présence de l'interface eth0 :

```
# dmesg | grep eth0
eth0: RealTek RTL-8029 found at 0xec00, IRQ 16, 00:00:21:CB:7A:54.
```

La gestion de la carte réseau est réalisée par un *driver* généralement écrit sous forme de modules chargeables que l'on peut lister avec la commande **lsmod** :

```
# lsmod | grep $(ethtool -i eth0 | grep driver | cut -d : -f2)
ne2k-pci                6752   1  (autoclean)
8390                    7916   0  (autoclean) [ne2k-pci]
```

Le driver de la carte réseau sera chargé dynamiquement lorsqu'une action sur l'interface eth0 sera réalisée. Pour une gestion automatique du chargement, on ajoutera un alias dans le fichier modules.conf :

```
# cat /etc/modules.conf | grep $(ethtool -i eth0 | grep driver | cut -d : -f2)
alias eth0 ne2k-pci
```

Le service network

Le script de démarrage **`rc`** récupère le niveau de démarrage x dans **`/etc/inittab`** et va lancer les différents scripts du répertoire **`/etc/rc.d/rcx.d/`** (où x est le niveau de démarrage du système).

La présence **S10network**, qui est un lien vers le script network, indique que le service réseau est lancé dans le niveau 3 :

```
# ll /etc/rc.d/rc3.d/
total 0
...
lrwxrwxrwx    1 root     root           17 jun 15 22:46 S10network -> ../init.d/network
...
```

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

```
# /etc/init.d/network
Utilisation : network {start|stop|restart|reload|status}

# /etc/init.d/network status
Périphériques configurés :
lo eth0 ppp0
Périphériques actuellement actifs :
lo eth0
```

Sur une **Mandriva**, le script **network** lit le fichier **/etc/sysconfig/network** qui contient des options de configuration :

```
# cat /etc/sysconfig/network
HOSTNAME="thius"
NETWORKING=yes
GATEWAY="192.168.52.42"
```

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

Avec l'option **start**, le script **network** réalise les actions suivantes :

- utilise la commande **sysctl** pour configurer les paramètres réseaux du noyau, en fonction des options définis dans **/etc/sysctl.conf** :

```
# cat /etc/sysctl.conf
# Kernel sysctl configuration file for Mandrake Linux
# For binary values, 0 is disabled, 1 is enabled.  See sysctl(8) and
# sysctl.conf(5) for more details.

# Controls IP packet forwarding
net.ipv4.ip_forward = 0
# Disables IP dynaddr
net.ipv4.ip_dynaddr = 0
# Disable ECN
net.ipv4.tcp_ecn = 0
# Controls source route verification
net.ipv4.conf.default.rp_filter = 1

# Controls the System Request debugging functionality of the kernel
#kernel.sysrq = 0

# Controls whether core dumps will append the PID to the core filename.
# Useful for debugging multi-threaded applications.
kernel.core_uses_pid = 1
net.ipv4.icmp_ignore_bogus_error_responses=0
net.ipv4.conf.all.rp_filter=0
net.ipv4.icmp_echo_ignore_broadcasts=0
net.ipv4.icmp_echo_ignore_all=0
net.ipv4.conf.all.log_martians=0
kernel.sysrq=1
```

- lit le fichier **/etc/ethers** et initialise la table **ARP**
- active l'interface ***loopback***
- configure la partie **IPX** (si $IPX = "yes" ou "true")
- active le **FORWARD IPv4** (si $FORWARD_IPV4 = "yes" ou "true")
- configure les réseaux virtuels **VLAN 802.1Q** (si $VLAN = "yes" ou "true")

- active avec **ifup** l'ensemble des interfaces décrites par un fichier de configuration **ifcfg*** présent dans le répertoire **/etc/sysconfig/network-scripts** :

```
# ll /etc/sysconfig/network-scripts/
...
-rwxr-xr-x    1 root     root     362 sep  6 10:58 ifcfg-eth0*
lrwxrwxrwx    1 root     root     22 jun 15 22:33 ifcfg-lo -> ../networking/ifcfg-
lo
-rw-r--r--    1 root     root     373 jun 15 22:49 ifcfg-ppp0
lrwxrwxrwx    1 root     root     20 jun 15 22:33 ifdown -> ../../../sbin/ifdown*
lrwxrwxrwx    1 root     root     18 jun 15 22:33 ifup -> ../../../sbin/ifup*
...

# cat /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE="eth0"
BOOTPROTO="none"
IPADDR="192.168.1.1"
NETMASK="255.255.255.0"
ONBOOT="yes"
...
```

  ONBOOT="yes" indique que l'interface sera activée au démarrage

- ajoute les routes statiques décrites dans le fichier **/etc/sysconfig/static-routes**

- lance la gestion d'**IP version 6** (si $NETWORKING_IPV6 = "yes")

La commande **chkconfig** offre un moyen simple pour définir les niveaux pour lesquels un service doit être démarré ou arrêté. Elle permet de visualiser, d'ajouter ou de retirer un service pour un ou plusieurs niveaux de fonctionnement :

```
chkconfig --list [service]
chkconfig --add service
chkconfig --del service
chkconfig [--level niveaux] service <on|off|reset>
```

Pour le service réseau :

```
# chkconfig --list network
network         0:Arrêt 1:Arrêt 2:Marche        3:Marche        4:Marche        5:Marche        6:Arrêt
```

La commande ifconfig

ifconfig est utilisé pour configurer et maintenir les interfaces réseau résidentes dans le noyau. Il est utilisé lors du boot pour configurer la plupart d'entre-elles et ainsi rendre le système opérationnel.

Si aucun argument n'est donné, ifconfig affiche simplement l'état des interfaces actuellement définies. Si seul le paramètre interface est donné, il affiche seulement l'état de l'interface correspondante; si seul le paramètre -a est fourni, il affiche l'état de toutes les interfaces, même celles qui ne sont pas actives.

Autrement, il considère qu'il faut positionner de nouvelles valeurs :

```
ifconfig [interface]
ifconfig interface [aftype] options | adresse ...
```

Quelques options :

- interface : le nom de l'interface réseau. C'est généralement un nom de pilote suivi d'un numéro d'ordre comme eth0 pour la première interface Ethernet.
- up : cette option active l'interface. Elle est implicitement spécifiée si une nouvelle adresse est affectée à l'interface.
- down : cette option arrête le fonctionnement du pilote pour cette interface.
- [-]arp : valide ou invalide l'utilisation du protocole ARP sur cette interface. Si le signe moins - est présent, l'option est invalidée.
- [-]promisc : valide ou invalide le mode promiscuous, toutes les trames seront reçues sur cette interface.
- mtu N : ce paramètre définit le MTU (*Maximum Transfer Unit*) d'une interface.

...

Faire un `man ifconfig` pour obtenir plus d'informations sur la commande `ifconfig`.

La trame 802.3 (*FRAME* 802.3)

Le bloc d'information ou **trame** au niveau de la sous-couche MAC (de la couche 2 LIAISON) est le suivant :

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

Remarques :

- Le **préambule**, composé d'une succession de 1 et de 0, assure la synchronisation du récepteur sur la trame émise.
- Le **délimiteur de trame** 10101011 permet de trouver le début du champ d'adresses (les 2 derniers bits émis sont à 1).

On a tendance à considérer que le préambule fait 8 octets et qu'il ne fait pas partie de la trame : il n'est pas capturé par les analyseurs réseaux et on n'en tient logiquement pas compte dans le calcul du CRC.

La trame Ethernet_II

Le bloc d'information ou **trame** est le suivant :

Préambule	Délimiteur de trame	Adresse destination	Adresse source	Type protocole de niveau supérieur	**Données (MTU)**	FCS CRC
7 octets	1 octet	6 octets	6 octets	2 octets	**46 à 1500 octets**	4 octets

Quelques valeur du champ **type** :

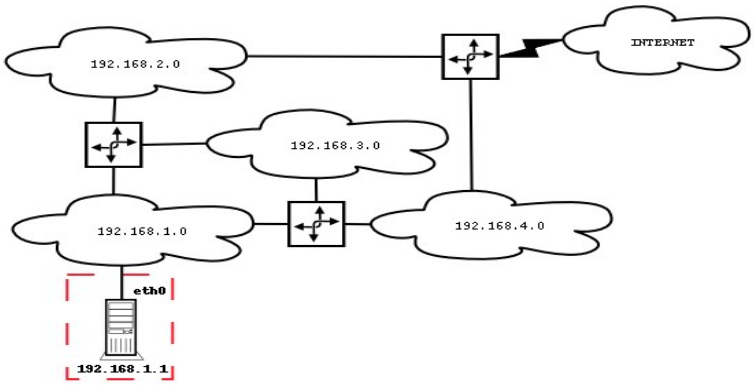
Type	**Protocole**
0x0800	IP
0x0806	ARP
0x809B	AppleTalk
0x8137	IPX
0x8191	NetBeui
0x0805	X25

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 équipement (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

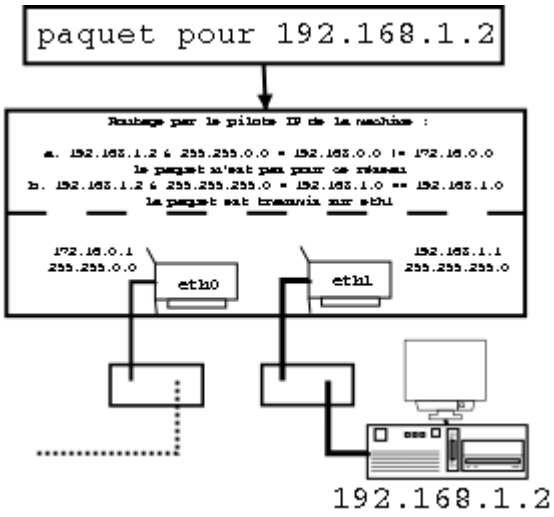
Routage statique et dynamique

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

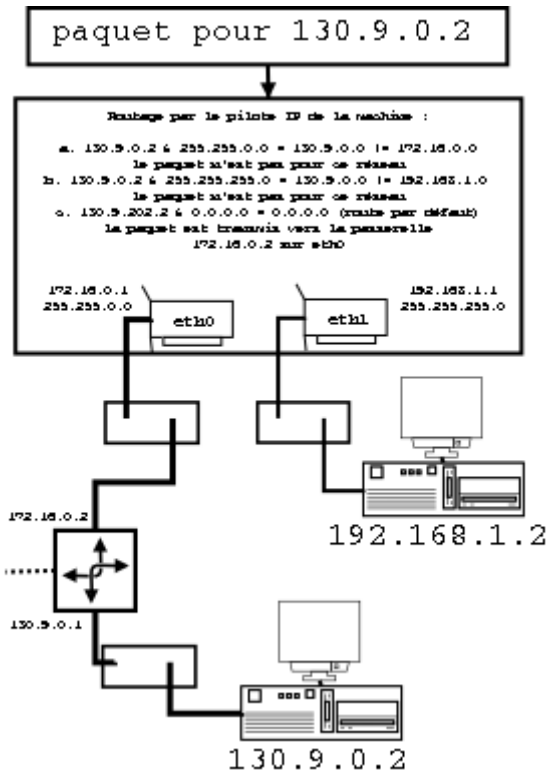


La commande **ifconfig** permet la configuration du routage direct en associant une adresse IP à une carte réseau.

```
$ 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.

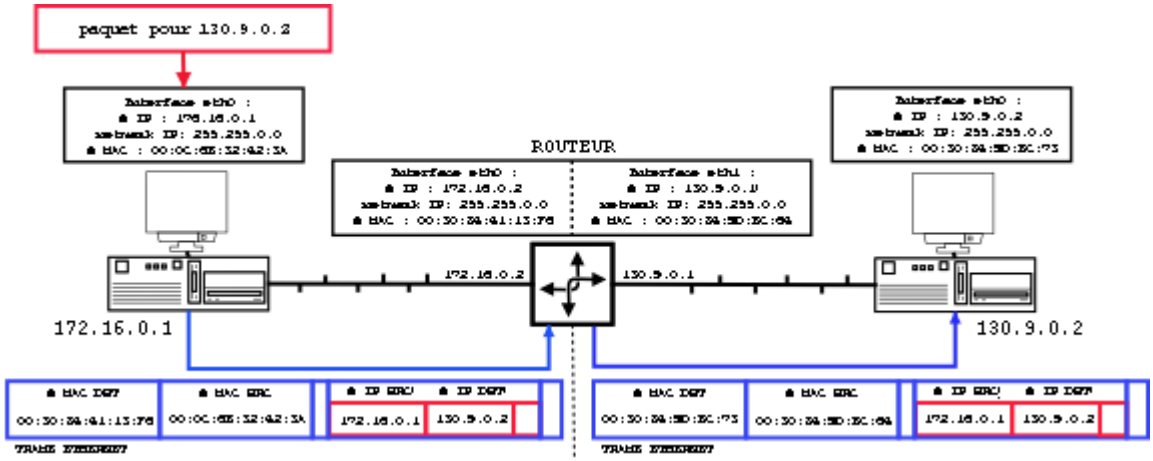
```
# route add default gw 172.16.0.2 dev eth0

$ 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 unreacheable"
          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 unreacheable"
     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 vénus.

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

Commandes	*Description*
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
tracepath, tracepath6	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

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 :

Aller vers	***Passer par***
la destination (hôte ou réseau)	**la route**
Champs: Destination et Genmask	*Champs:* Passerelle et Iface

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 :

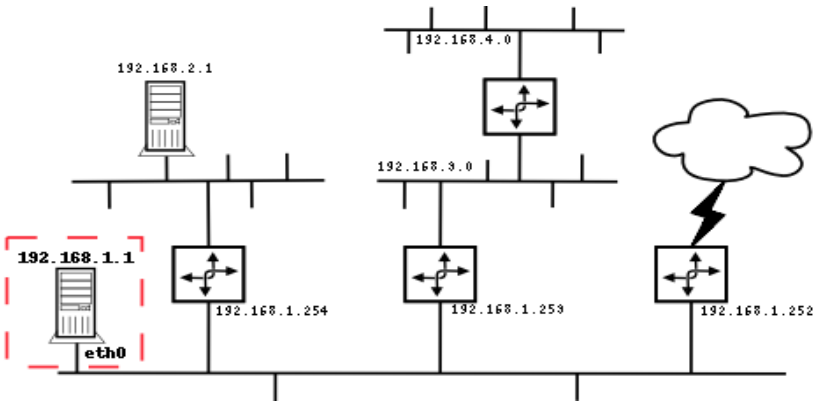
- le champ Métrique (*Metric*) indique la distance, en nombre de passerelles, pour atteindre la destination
- le champ Ref spécifie le nombre de références à cette route (non utilisé par le noyau Linux)
- le champ Use comptabilise le nombre de recherches associées à cette route
- le champ MSS indique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

Le script **/etc/init.d/network** permet de gérer le service réseau.
Le script **network** lit le fichier **/etc/sysconfig/network** dont quelques options concernent le routage :

```
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
```

Le fichier **/etc/sysconfig/static-routes** est utilisé pour ajouter les routes statiques.
La syntaxe du fichier **/etc/sysconfig/static-routes** dépend du script **network** (et donc souvent de la distribution utilisée, ici une Mandrake/Mandriva) 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`**

Routage statique

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

Avantages	*Inconvénients*
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

Remarques: 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 (FAI 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, couche 3 du modèle OSI).

Avantages	*Inconvénients*
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	

Remarques: quelques protocoles de routage dynamique comme RIP (*Routing Information Protocol*), IGRP (*Internet Gateway Routing Protocol*) de la société CISCO, OSPF (*Open Shortest Path First*), ...

La commande route

Il est important de savoir que tout PC/Linux, fonctionnant sous TCP/IP, possède une table de routage accessible par la commande **`route`**.

Table de routage (statique) d'un poste quelconque :

```
# route
Table de routage IP du noyau
Destination     Passerelle      Genmask         Indic Metric Ref    Use Iface
192.168.1.0     *               255.255.255.0   U     0      0        0 eth0
127.0.0.0       *               255.0.0.0       U     0      0        0 lo
```

Signification :

Destination	Adresse de destination de la route
Gateway	Adresse ip de la passerelle pour atteindre la route, * sinon
Genmask	Masque à utiliser
Flags	Indicateur d'état (U - Up, H - Host - G - Gateway, D - Dynamic, M - Modified)
Metric	Coût métrique de la route (0 par défaut)
Ref	Nombre de routes qui dépendent de celle-ci
Use	Nombre d'utilisation dans la table de routage
Iface	Interface eth0, ethl, lo...

Faire un `man route` pour obtenir plus d'informations sur la commande `route`.

La fragmentation

Sur toute machine ou passerelle mettant en oeuvre TCP/IP, une unité maximale de transfert **MTU** (*Maximum Transfer Unit*) définit la taille maximale d'un datagramme transporté sur le réseau physique correspondant.

Si on prend comme exemple un réseau Ethernet, on aura un MTU par défaut de 1500 octets, ce qui correspond à la longueur maximale du champ DATA d'une trame Ethernet (le minimum étant de 46 octets). On peut modifier la valeur du MTU pour une interface eth0 avec la commande ifconfig.

Ce n'est pas le rôle de la couche Liaison du modèle OSI (ou Interface pour le modèle DoD) de réaliser la fragmentation, mais c'est elle qui fixe la valeur du MTU puisqu'elle a en charge le transport des trames sur le support physique.

Lorsque la taille du datagramme initial est plus grand que la valeur du MTU, la machine ou la passerelle le fragmente en un certain nombre de fragments transportés par autant de trames sur le support physique. Le destinataire final reconstitue le datagramme initial à partir de l'ensemble des fragments reçus. On rappelle que les datagrammes peuvent emprunter des chemins différents pour atteindre une machine destinatrice.

Remarques:
La taille de ces fragments correspond au plus petit MTU emprunté sur le réseau.
Si un seul des fragments est perdu, le datagramme initial est considéré comme perdu.

Dans un datagramme IP, les champs propres a la fragmentation sont :

- **IDentification** : entier sur 16 bits qui identifie le datagramme initial.

- **FRAGMENT OFFSET** : valeur sur 13 bits indiquant le déplcement des données contenues dans le fragment par rapport au datagramme initial. Cette valeur est toujours un multiple de 8 octets, la taille du fragment est donc aussi un multiple de 8 octets.

- **FLAGS** : 3 bits dont le bit de poids fort n'est pas utilisé x DF MF

    - DF : Don't Fragment
    - **MF : More Fragments**, MF = 1 il y a d'autres fragments
      MF = 0 dernier fragment

Chaque fragment a donc une structure identique à celle du datagramme initial, seul les champs FLAGS et FRAGMENT OFFSET sont spécifiques à chaque fragment.

Le paquet IP

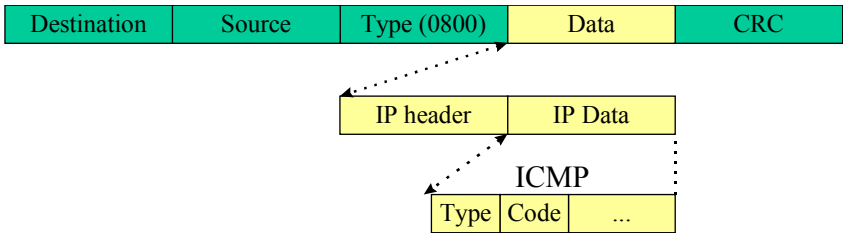
Le format unique du paquet IP, ou datagramme IP, est organisé en champs de 32 bits :

<table>
<tr><td colspan="8">0</td><td colspan="8">7</td><td colspan="8">8</td><td colspan="8">15</td><td colspan="8">16</td><td colspan="8">23</td><td colspan="8">24</td><td colspan="8">31</td></tr>
<tr><td colspan="8">Version</td><td colspan="8">IHL</td><td colspan="16">TOS</td><td colspan="32">Total length</td></tr>
<tr><td colspan="32">Id.</td><td>0</td><td>D F</td><td>M F</td><td colspan="13">Fragment offset</td></tr>
<tr><td colspan="16">TTL</td><td colspan="16">Protocole</td><td colspan="32">Header checksum</td></tr>
<tr><td colspan="64">Adresse Source</td></tr>
<tr><td colspan="64">Adresse Destination</td></tr>
<tr><td colspan="64">Options + padding</td></tr>
<tr><td colspan="64">DONNEES</td></tr>
</table>

- 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 "*Total Length*" (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.
- "Fragment 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 , ... (voir /etc/protocols)
- Le "*Header checksum"* ou champ de contrôle de l'en-tête (16 bits) contient le "complément à un" du total "en complément à un" de tous les mots de 16 bits de l'en-tête.
- 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*" (bourrage) est habituellement rempli de 0 de manière à aligner le début des données sur un multiple de 32 bits.

Le protocole ICMP (*Internet Control Message Protocol*)

Le protocole ICMP (*Internet Control Message Protocol* -- RFC 792) utilise les datagrammes IP pour transporter ses messages.



Le protocole ICMP permet par exemple :

- le contrôle de flux : le récepteur débordé par un émetteur trop rapide, envoie un message ICMP *Source Quench* pour arrêter temporairement l'émission
- la détection de destinations inaccessibles dénoncée par un message *Destination Unreachable*
- la redirection de routes pour avertir une machine hôte d'utiliser un autre *gateway.*

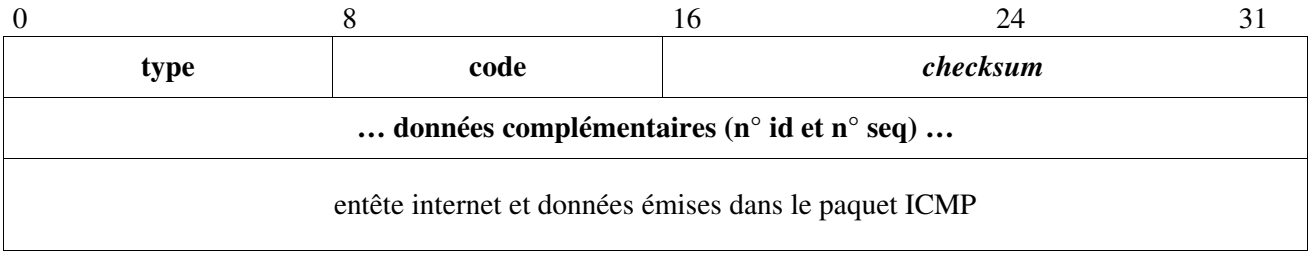
ICMP fournit d'intéressantes données pour le diagnostic d'opérations du réseau. ICMP utilise des datagrammes IP pour véhiculer des messages aller-retour entre noeuds concernés. Un message d'erreur ICMP est généré par une machine hôte réalisant qu'il y a un problème de transmission et renvoyé à l'adresse de départ du datagramme ayant provoqué le problème.

Le protocole **ICMP** est utilisé notamment par la commande **ping** qui :

- emet un paquet ICMP "demande d'écho" (type=8 et code=0) et
- reçoit, si la machine distante est active (*alive*), un paquet ICMP "réponse d'écho" (type=0 et code=0).

Format du paquet 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 :



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
3	Destination inaccessible
4	Source Quench
5	Redirection
8	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
- 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.

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.

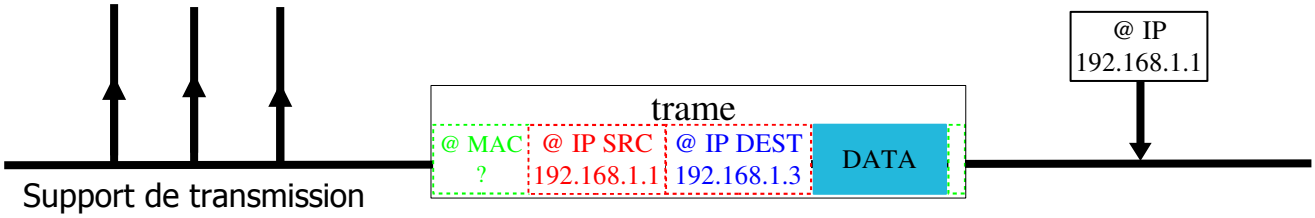
17-18

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 protocole ARP (*Address Resolution Protocol*)

Le protocole ARP sert à traduire une adresse réseau IP en une adresse physique.

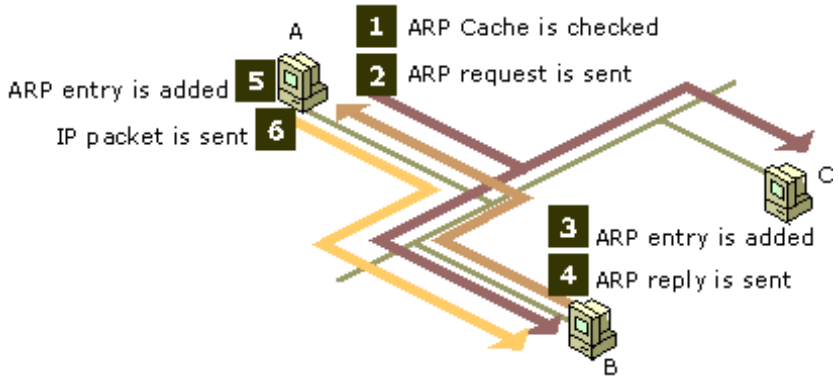
Un poste désire envoyer un paquet IP à un poste appartenant au même réseau physique que lui. Il doit connaître l'adresse physique du destinataire. Or, il ne connait que son adresse IP.



Le protocole ARP va lui permettre de trouver l'adresse physique du poste destinataire. Ce mécanisme est transparent pour l'utilisateur.

Une table de conversion est générée dynamiquement sur chaque hôte dans ce qu'on appelle l'"***ARP cache***". Quand ARP reçoit une demande de conversion, il consulte sa table et retourne l'adresse physique si elle s'y trouve sinon il envoie un paquet spécial *ARP Request Packet* à tous les hôtes du même réseau physique incluant l'adresse IP à rechercher et en utilisant l'adresse broadcast MAC FF FF FF FF FF FF.

La machine possédant l'adresse réseau IP demandée répond en lui renvoyant donc son adresse physique qui est alors placée dans la table ARP.



Si aucune réponse n'est reçue dans un délai imparti, la requête est envoyée à nouveau.

Le contenu de l'*ARP Cache* est généralement conservé jusqu'à l'extinction de la machine hôte. Lors du démarrage de la machine, l'*ARP Cache* est donc vide :

```
# arp -v
Entrées: 0      Ignorées: 0     Trouvées: 0

# cat /proc/net/arp
IP address       HW type     Flags       HW address            Mask     Device
```

Remarques:

- ### *Routeurs et requêtes ARP*

Les requêtes ARP ne passent pas les routeurs, qui relaient des informations au niveau de la couche réseau mais pas du trafic *broadcast* MAC.

- ### *ARP Spoofing (ou ARP Redirect)*

Si une machine non fiable a accès au réseau physique et émet de faux messages ARP pour corrompre le cache ARP d'une machine cible et d'en détourner tout le trafic vers elle-même afin d'en d'écouter et/ou modifier les données (attaque par ARP *spoofing*). Des générateurs de paquets ARP comme **arpspoof** ou **nemesis** permettent ce type d'attaque. La machine pirate se rend transparente en reroutant le trafic en ayant activé l'*IP Forwarding*
(`echo 1 > /proc/sys/net/ipv4/ip_forward`). Il existe donc un certain risque lié à l'utilisation du protocole ARP, même sur des réseaux segmentés par des *switch*.

- ### *Proxy ARP*

Le proxy-ARP permet de résoudre le problème posé par des ***firewall*** installés sur un même réseau d'adresse. Les requêtes ARP, étant faites par *broadcast*, seront donc normalement bloquée par la machine filtrante. L'activation de la fonctionnalité Proxy-ARP permet de palier à ce problème en demandant explicitement à ce que les requêtes et réponses ARP arrivant par une carte soient propagées vers l'autre et vice-versa.
Sous Linux, on activera le Proxy-ARP pour chacune des 2 interfaces de la manière suivante :

```
# echo 1 > /proc/sys/net/ipv4/conf/eth0/proxy_arp
# echo 1 > /proc/sys/net/ipv4/conf/eth1/proxy_arp
```

Le protocole RARP (Reverse ARP)

Le protocole RARP permet d'associer une adresse réseau à une adresse physique. Ce protocole était utilisé avant l'adoption du protocole DHCP. RARP était alors utilisé sous Unix par des stations *diskless* (sans disque) ou des terminaux pour leur attribuer une adresse IP à partir d'un serveur. Le protocole *Inverse* ARP est similaire à RARP.

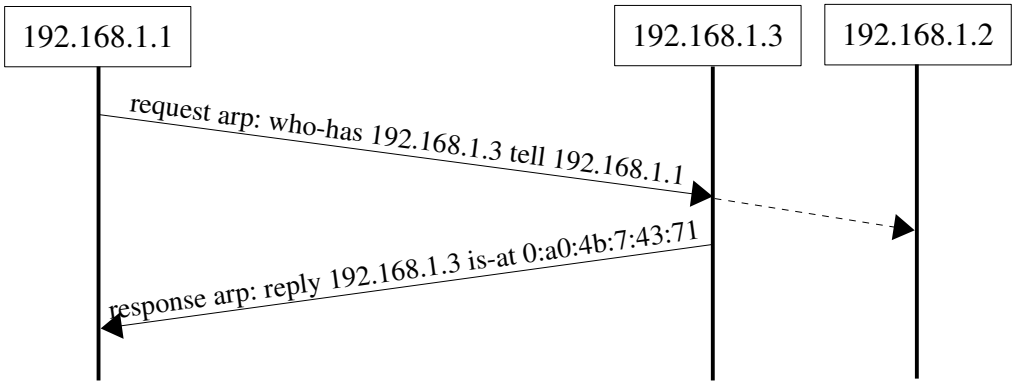
Les RFCs

RFC	***Contenu***
RFC 826	ARP
RFC 903	RARP
RFC 1122	Prérequis réseau
RFC 1433	ARP direct
RFC 1868	Extension UnARP
RFC 2131	DHCP et DHCP ARP
RFC 2390	Inverse ARP

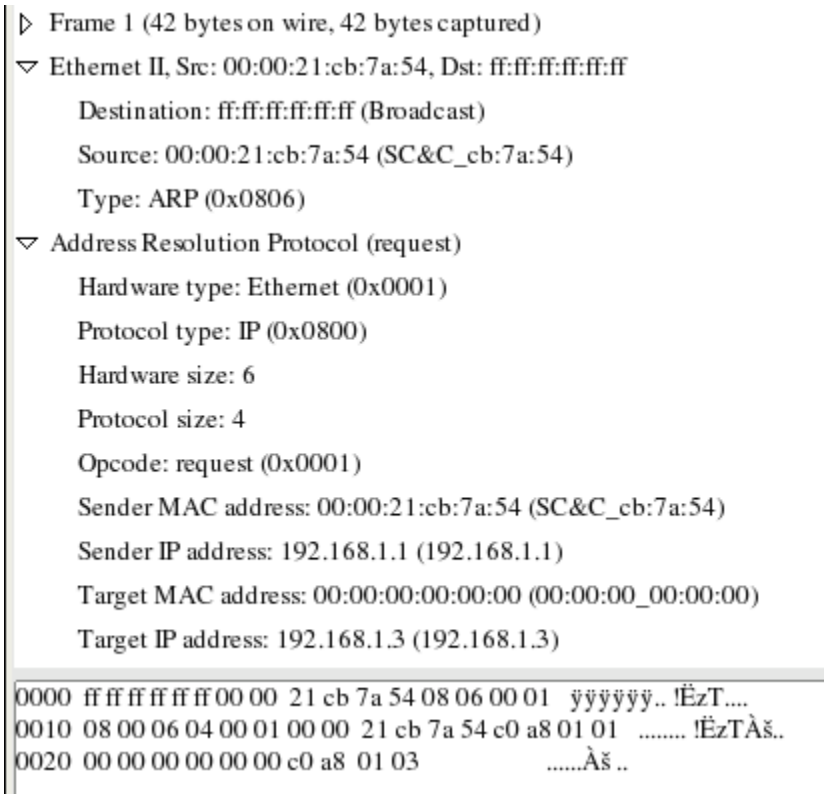
Exemple

On vient de démarrer le poste d'adresse IP 192.168.1.1 et le chache ARP est vide. On fait un ping vers le poste d'adresse IP 192.168.1.3. On capture l'échange ARP qui s'en suit avec **tcpdump** :

```
# tcpdump -vv arp
tcpdump: listening on eth0
08:51:26.702516 arp who-has 192.168.1.3 tell 192.168.1.1
08:51:26.702747 arp reply 192.168.1.3 is-at 0:a0:4b:7:43:71
```

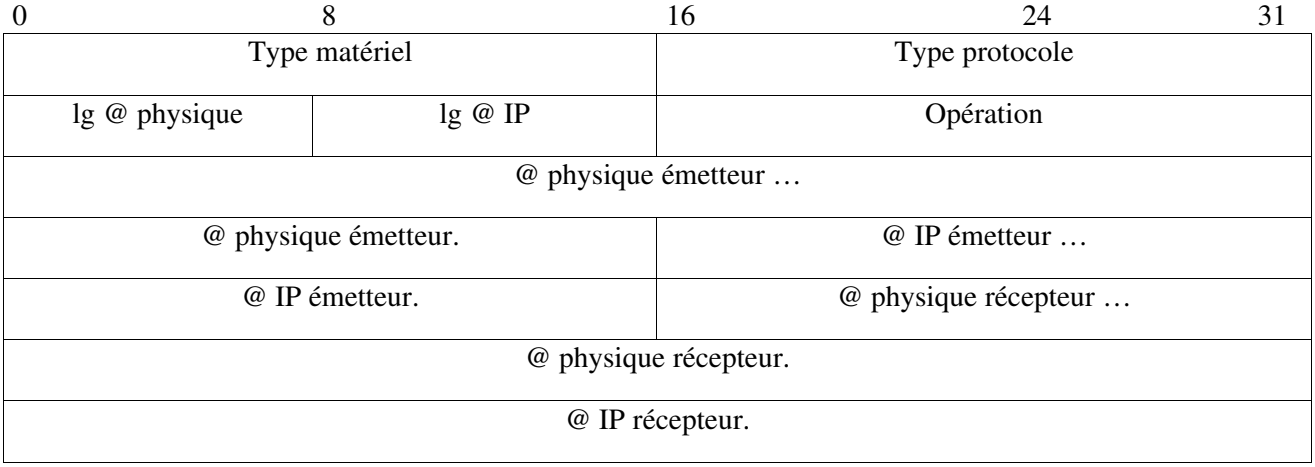


Dans le même échange, on capture et on visualise la première trame avec **ethereal** :



La trame ARP (Address Resolution Protocol)

ARP a été conçu pour fonctionner avec n'importe quel protocole, cela implique l'utilisation d'un format de paquet souple. Par exemple, le "paquet" ARP est encapsulé dans une trame Ethernet_II avec l'adresse de diffusion générale (*broadcast*) et le numéro de protocole désignant un paquet ARP (0x806). Les champs spécifiques à ARP sont :



Pour rappel:

- les adresse physiques ont une taille de 6 octets
- les adresses réseaux IP v4 ont une taille de 4 octets

Quelques valeurs du champ **Type matériel** :

Type matériel	**Protocole**
1	**Ethernet**
6	IEEE 802
20	ATM

Le champ **Type protocole** désigne le protocole dont on veut résoudre les adresses : toujours 0x800 pour IP.
Le champ **lg @ physique** indique la longueur de l'adresse physique soit 6 octets pour Ethernet ;
Le champ **lg @ protocole** indique la longueur de l'adresse réseau soit 4 octets pour IPv4 ;

Le champ **Opération** définit la signification du paquet, soit :

Opération	**Message**
1	**Requête ARP**
2	**Réponse ARP**
3	Requête RARP
4	Réponse RARP
8	Requête *Inverse* ARP
9	Réponse *Inverse* ARP

Les commandes et outils de base (Linux/Windows)

Description	*Linux*	*Windows*
Configurer une interface réseau	**ifconfig**	**ipconfig**
Afficher les connexions réseau, les tables de routage, les statistiques des interfaces, ...	**netstat**	**netstat**
Envoyer des datagrammes ICMP ECHO_REQUEST à des hôtes sur un réseau	**ping**	**ping**
Afficher le chemin qu'un paquet IP va prendre pour aller d'une machine A à une machine B	**traceroute**	**tracert**
Suivre le chemin qu'un paquet IP va prendre pour aller d'une machine A à une machine B pour découvrir le MTU à utiliser sur ce chemin	**tracepath**	
Afficher et manipuler la table de routage IP	**route**	**route**
Manipuler la table ARP du système	**arp**	**arp**
Outil d'analyse et de capture réseau	**ethereal/wireshark tcpdump**	**ethereal/wireshark windump**
Outil d'exploration réseau et analyseur de sécurité	**nmap**	**nmap**
Fournir un moyen de communication TCP bi-directionnel et orienté octet (caractère)	**telnet**	**telnet, putty**
Lire et écrire en utilisant TCP ou UDP	**netcat**	**netcat**
Remarque :		
Accès aux options des commandes	**nom_commande -h nom_commande --help**	**nom_commande /?**
Accès à la documentation	**man nom_commande**	

Description	*Linux*	*Windows*
Configuration réseau	**/etc/ /proc/net**	**Base de registre (regedit)**

Remarque :
/proc est un pseudo-système de fichiers qui est utilisé comme interface avec les structures de données du noyau. La plupart des fichiers sont en lecture seule, mais quelques uns permettent la modification de variables du noyau. On peut aussi utiliser la commande sysctl pour configurer les paramètres du noyau.
Le répertoire **/proc/net** regroupe divers pseudo-fichiers relatifs aux fonctionnalités **réseau**. Chaque fichier fournit des informations concernant une couche particulière. Ces fichiers sont en ASCII et sont donc lisibles grâce à la commande **cat**, mais le programme standard **netstat** fournit un accès plus propre à ces données.