Quelques informations sur la configuration réseau sous Solaris

Version 1.1

Auteur: Richard Luce
Correction : Benjamin Lisan

Equipe Déploiement BT Serveur

Table des matières

1. INTRODUCTION

Le but de ce document est de présenter les différents éléments d'un paramétrage réseau sous Solaris. Le style de document est volontairement peu littéraire. Il se veut plus dans l'esprit de « prise de note », permettant ainsi un usage au quotidien plus aisé. Ce document n'est pas exhaustif et devra évoluer dans le temps.

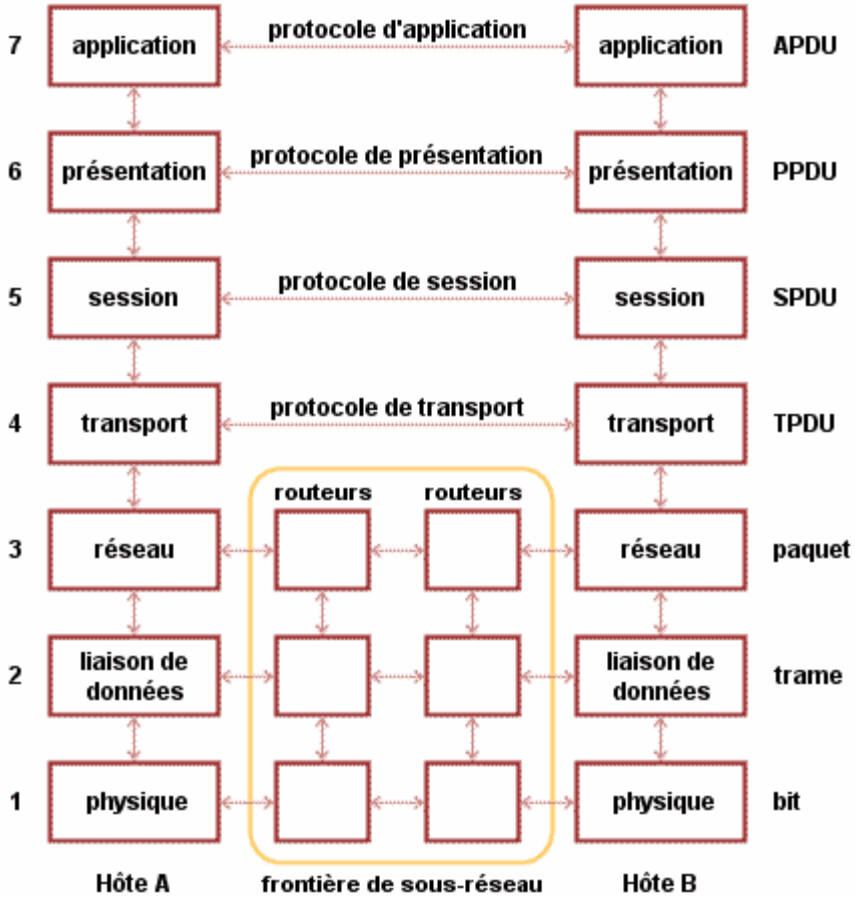
On verra par exemple :

- La commande « ndd » et la façon de la paramétrer,
- Positionner les bonnes vitesses par le « ndd » ou les fichiers de configuration,
- les adresses IP,
- le remplissage du fichier des adresses IP « hosts ».
- l'IPMP,
- les routages.

On examinera certains problèmes :

- le fait que le trafic sur la patte ne monte pas (et qui n'a rien a voir avec les masques).
- Des problèmes de « ping », en fonction des routes statiques initialisés ou non.

2. LE SCHEMA DES COUCHES OSI



2.1 La couche physique est la première couche du modèle OSI.

La couche physique est chargée de la transmission effective des signaux électriques ou optiques entre les interlocuteurs. Son service est typiquement limité à l'émission et la réception d'un bit ou d'un train de bits continu (notamment pour les supports synchrones comme la fibre optique). Cette couche est chargée de la conversion entre bits et signaux électriques ou optiques. Elle est en pratique toujours réalisée par un circuit électronique spécifique.

2.2 La couche liaison de données

Son rôle est un rôle de "liant": elle va transformer la couche physique en une liaison a priori exempte d'erreurs de transmission pour la couche réseau. Elle fractionne les données d'entrée de l'émetteur en trames, transmet ces trames en séquence et gère les trames d'acquittement renvoyées par le récepteur. Rappelons que pour la couche physique, les données n'ont aucune signification particulière. La couche liaison de données doit donc être capable de reconnaître les frontières des trames. Cela peut poser quelques problèmes, puisque les séquences de bits utilisées pour cette

reconnaissance peuvent apparaître dans les données.
La couche liaison de données doit être capable de renvoyer une trame lorsqu'il y a eu un problème sur la ligne de transmission. De manière générale, un rôle important de cette couche est la détection et la correction d'erreurs intervenues sur la couche physique. Cette couche intègre également une fonction de contrôle de flux pour éviter l'engorgement du récepteur.
L'unité d'information de la couche liaison de données est la trame qui est composées de quelques centaines à quelques milliers d'octets maximum.

2.3 La couche réseau

C'est la couche qui permet de gérer le sous-réseau, i.e. le routage des paquets sur ce sous-réseau et l'interconnexion des différents sous-réseaux entre eux. Au moment de sa conception, il faut bien déterminer le mécanisme de routage et de calcul des tables de routage (tables statiques ou dynamiques...).
La couche réseau contrôle également l'engorgement du sous-réseau. On peut également y intégrer des fonctions de comptabilité pour la facturation au volume, mais cela peut être délicat.
L'unité d'information de la couche réseau est le paquet.

3. CONFIGURER LA VITESSE DES PATTES DES CARTES RESEAU

Il y a normalement 3 façons d''initialiser les vitesses des pattes réseau sur nos serveurs :

1) On positionne les vitesses et les « modes », par des commandes « ndd » inscrite dans le fichier « /etc/rc2.d/S25conf-ethernet.ksh » [1] _ qui est exécuté a) en fin de boot du serveur, b) ou à la main.

2) On configure les vitesses des pattes et les « modes », au niveau d'un fichier de configuration du driver la carte réseau (fichier ce.conf, bge.conf … présents dans le répertoire /platform/sun4u/kernel/drv), dépendant du type de la carte (ce, bge …), qui est lu au moment de l'initialisation de ces vitesses au début du boot (initialisation faite au niveau du noyau Unix).

3) Il existe une troisième méthode qui consiste à positionner les vitesses dans « /etc/system ».

Recommandations SUN :
Pour les interfaces « ce », « bge » et « dmfe », **SUN** recommande d'utiliser le fichier de configuration du driver en lui-même. Les vitesses des ports seront donc positionnés dès le boot du system (par exemple pour « ce », dans le fichier « /platform/sun4u/kernel/drv/ce.conf ») [2].
La troisième méthode consistant à positionner les vitesses dans « /etc/system ». n'est pas recommandée par SUN.

Configurer ces derniers fichiers, obliger de rebooter le serveur.

3.1 Configurer la vitesse des pattes par la commande « ndd »

(par exemple au niveau du fichier « conf-ethernet.ksh »).

Les commandes « ndd » que nous passons se situe au niveau physique/liaison de données. Les commandes « ifconfig », « route » etc… sont au niveau de la couche réseau.

Conclusion: Les adaptations de vitesses sont donc indépendant des routages ou autres adresses ip.

C'est la commande qui permet de positionner, entre autres, les vitesses des cartes.

Note : Nous travaillons au niveau des différents drivers réseau, donc au niveau des différents devices. La numérotation des devices se fait dans l'ordre de découverte du hardware. Vous trouverez la correspondance entre le numéro de device et les chemins

[1] Ce fichier est, en fait, un lien sur le fichier « /etc/init.d/conf-ethernet.ksh ».
[2] Ce point est valable pour Solaris 8 et Solaris 10.

hardware dans /etc/path_to_inst. Ce fichier donne la liste de tous les chemins physiques (hard) :

Exemple d'un contenu de « /etc/path_to_inst » pour les drivers « ce » :

```
[su0967@root:/] grep ce /etc/path_to_inst
"/pci@1d,700000/pci@1/network@0" 0 "ce"
"/pci@1d,700000/pci@1/network@1" 1 "ce"
"/pci@1d,700000/pci@1/network@2" 2 "ce"
"/pci@1d,700000/pci@1/network@3" 3 "ce"
```

Note : dans cet exemple, tous ces « ce » sont sur la même carte physique.

Certains devices sont déclarés avec un numéro propre. Ce sont notamment les devices qui sont présents sur la carte mère. Exemple les ports « bge » d'un serveur SUN v240.

Exemple pour forcer l'interface ce6 en 100 full duplex sans auto négociation:

a) Positionner l'instance :

```
[su0967@root:/] ndd -set /dev/ce instance 6
```

```
b) Voir tous les paramètres existants pour le driver.
[su0967@root:/]  ndd -get /dev/ce ?
```

```
[su0967@root:/] ndd -get /dev/ce ?
adv_autoneg_cap                 (read and write)
adv_1000fdx_cap                 (read and write)
adv_1000hdx_cap                 (read and write)
adv_100T4_cap                   (read and write)
adv_100fdx_cap                  (read and write)
adv_100hdx_cap                  (read and write)
adv_10fdx_cap                   (read and write)
adv_10hdx_cap                   (read and write)
adv_asmpause_cap                (read and write)
adv_pause_cap                   (read and write)
master_cfg_enable               (read and write)
master_cfg_value                (read and write)
use_int_xcvr                    (read and write)
enable_ipg0                     (read and write)
ipg0                            (read and write)
ipg1                            (read and write)
ipg2                            (read and write)
rx_intr_pkts                    (read and write)
rx_intr_time                    (read and write)
red_dv4to6k                     (read and write)
red_dv6to8k                     (read and write)
red_dv8to10k                    (read and write)
red_dv10to12k                   (read and write)
tx_dma_weight                   (read and write)
rx_dma_weight                   (read and write)
infinite_burst                  (read and write)
disable_64bit                   (read and write)
accept_jumbo                    (read and write)
laggr_multistream               (read and write)
```

cc) exemple : forcer une interface ce 6 en 100full duplex :

```
# ndd -set  /dev/ce instance 6
# ndd -set /dev/ce adv_autoneg_cap 0
# adv_1000fdx_cap 0
# adv_1000hdx_cap 0
# adv_100T4_cap 0
```

```
# adv_100fdx_cap 1
# adv_100hdx_cap 0
# adv_10fdx_cap 0
```

Attention: *Avec du Gigabit il faut toujours laisser l'auto négociation*.
Note : souvent, la patte SVG (voire la patte CLI) sont en 1000 Mbps (Gigabit) avec autonegociation.

3.2 Configurer le fichier /platform/sun4u/kernel/drv/<drv>.conf

Ce fichier « /platform/sun4u/kernel/drv/<drv>.conf » peut être le fichier :
/platform/sun4u/kernel/drv/dmfe.conf
Ou encore /platform/sun4u/kernel/drv/ce.conf
Ou /platform/sun4u/kernel/drv/bge.conf

(Note : pour le fichier « bge.conf » voir paragraphe suivant 3.3).

Ce fichier contient les mêmes paramètres que ceux à positionner par la commande ndd.

Exemple0: positionner toutes les pattes de la carte « ce » à 1000 FD, autoneg=1.

```
#adv_autoneg_cap=1
#adv_1000fdx_cap=1
#adv_1000hdx_cap=0
#adv_100fdx_cap=0
#adv_100hdx_cap=0
#adv_10fdx_cap=0
#adv_10hdx_cap=0
```

Ces paramètres sont alors positionnés pour tous les ports (pattes) utilisant ce driver.

Exemple1: positionner le ce0 en 100 full duplex et le ce1 en 1000 full duplex avec autoneg :

Ici, il faut introduire des paramètres particulieers, dans le fichier de configuration :

Par exemple, positionner a) la patte ce0 en 100 full duplex et b) puis celle ce1 en 1000 full duplex avec autoneg, sur un serveur Sun, il faut :

A) d'abord extraire certaines informations utiles (chemins physiques) pour chaque patte dans le fichier « /etc/path_to_inst » :

```
    # grep ce /etc/path_to_inst
"/pci@9,700000/network@2" 0 "ce"     <= ligne contenant le path/chemin physique pour
la patte ce0
"/pci@9,600000/network@1" 1 "ce"     <= ligne contenant le path/chemin physique pour
la patte ce1
```

Autre exemple :

```
# grep ce /etc/path_to_inst
"/pci@8,700000/pci@2/network@0" 0 "ce"
"/pci@8,700000/pci@2/network@1" 1 "ce"
"/pci@8,700000/pci@2/network@2" 2 "ce"
"/pci@8,700000/pci@2/network@3" 3 "ce"
"/pci@8,700000/pci@3/network@0" 4 "ce"
```

```
"/pci@8,700000/pci@3/network@1" 5 "ce"
"/pci@8,700000/pci@3/network@2" 6 "ce"
"/pci@8,700000/pci@3/network@3" 7 "ce"
"/pci@9,700000/network@2" 8 "ce"
"/pci@9,600000/network@1" 9 "ce"
```

B) Puis dans le fichier « ce.conf », il faut créer les 2 lignes suivantes, dans l'exemple ci-après, définissant les paramètres (ou config) réseau des pattes ce0 et ce1.

La syntaxe de ces lignes est de la forme :

```
name="type-patte" parent="chemin_physique"  unit-address="numero_patte" ;
```

Avec type-patte = ce, bge, hme etc …
La partie en vert (un extrait du chemin physique) est à mettre après le mot clé « parent=" » et le nombre en rouge (le n° physique de la patte), est à mettre après le mot-clé « unit-address=" » comme ci-après :

```
name="ce" parent="/pci@9,700000" unit-address="2"
adv_autoneg_cap=0
adv_1000fdx_cap=0
adv_1000hdx_cap=0
adv_100fdx_cap=1
adv_100hdx_cap=0
adv_10fdx_cap=0
adv_10hdx_cap=0;

name="ce" parent="/pci@9,600000" unit-address="1"
adv_autoneg_cap=1
adv_1000fdx_cap=1
adv_1000hdx_cap=0
adv_100fdx_cap=0
adv_100hdx_cap=0
adv_10fdx_cap=0
adv_10hdx_cap=0;
```

Notes :

1) chaque ligne ou groupe de ligne ci-avant, définissant les paramètres réseau (vitesse, mode, …) d'une patte précise, dans le fichier « **/etc/path_to_inst** », **doit se terminer par un « ; »** (un point virgule).
2) Sinon, voici un exemple de fichier « ce.conf », configurant des vitesses différentes et des modes différents pour 3 pattes (SVG, CLI, ADM), donné en annexe de ce document.

3) On peut aussi extraire les paramètres inclus dans la ligne commençant par « **name=** » par :

```
[su0321@root:/platform/sun4u/kernel/drv] grep ce /etc/driver_aliases
vnex "SUNW,sun4v-virtual-devices"
vnex "SUNW,virtual-devices"
usb_mid "usb,device"
ce "pci108e,abba"
ce "pci100b,35"
pcelx "pccard101,556"
pcelx "pccard101,574"
pcelx "pccard101,589"
pcelx "pccard101,562,0"
```

ou bien

```
# grep bge /etc/driver_aliases
bge "SUNW,bge"
bge "pci108e,1647"
bge "pci108e,1648"
bge "pci108e,16a7"
bge "pci108e,16a8"
bge "pci14e4,1645"
```

```
bge "pci14e4,1647"
bge "pci14e4,16a7"
bge "pci14e4,16c7"
bge "pci14e4,1648"
bge "pci14e4,16a8"
bge "pci14e4,1649"
bge "pciex14e4,1659"
bge "pci14e4,1668"
bge "pci14e4,1669"
bge "pci14e4,1677"
bge "pci14e4,1678"
bge "pci14e4,167d"
```

Ce qui donne :

```
# patte ce0 CLI 1Go
name="pci108e,abba" parent="/pci@8,700000/pci@2" unit-address="0"
adv_autoneg_cap=1
adv_1000fdx_cap=1
adv_1000hdx_cap=0
adv_100fdx_cap=0
adv_100hdx_cap=0
adv_10fdx_cap=0
adv_10hdx_cap=0
;
# patte ce1 ADM 100 FD
name="pci108e,abba" parent="/pci@8,700000/pci@2" unit-address="1"
adv_autoneg_cap=0
adv_1000fdx_cap=0
adv_1000hdx_cap=0
adv_100fdx_cap=1
adv_100hdx_cap=0
adv_10fdx_cap=0
adv_10hdx_cap=0
;
# patte ce8 SVG 100 FD
name="pci108e,abba" parent="/pci@9,700000" unit-address="2"
adv_autoneg_cap=0
adv_1000fdx_cap=0
adv_1000hdx_cap=0
adv_100fdx_cap=1
adv_100hdx_cap=0
adv_10fdx_cap=0
adv_10hdx_cap=0
;
```

Exemple2: forcer l'interface 2 en gigabit:

```
[su0967@root:/] grep ce /etc/path_to_inst
"/pci@1d,700000/pci@1/network@0" 0 "ce"
"/pci@1d,700000/pci@1/network@1" 1 "ce"
"/pci@1d,700000/pci@1/network@2" 2 "ce"
"/pci@1d,700000/pci@1/network@3" 3 "ce"

name="ce" parent="pci@1d,700000/pci@1" unit-address="2" adv_autoneg_cap=1
adv_1000fdx_cap=1 adv_1000hdx_cap=0 adv_100fdx_cap=0 adv_100hdx_cap=0
adv_10fdx_cap=0 adv_10hdx_cap=0;
```

3.3 Vérifier les vitesses des cartes:

1) Il existe un script qui permet de vérifier les vitesses des cartes. Ce scripts se trouvent dans ***/opt/sfrsi/admin/scripts/nicstatus.ksh***

Attention: ce script fonctionne parfaitement en Solaris 10. En revanche il n'est pas 100 fiable en Solaris 8. Par exemple pour les interfaces « ce », en Solaris 8, le full duplex est « taggé » par le chiffre 1, alors qu'en Solaris 10 c'est le chiffre 2.

2) Vous pouvez aussi récupérer les informations à l'aide des commandes **ndd** et **kstat**.

Exemple1 avec **kstat** :

```
# kstat hme | egrep 'name|link_speed|link_up'
```

Exemple2 avec **ndd** :

```
# On positionne l'instance 4
# /usr/sbin/ndd -set /dev/qfe instance 4
# Pour afficher les valeurs des parametres
# ndd -get /dev/hme link_status #=> valeurs posssibles : 1 = Up,  0 = Down
# ndd -get /dev/hme link_speed  #=> valeurs posssibles : 1 = 100, 0 = 10
# ndd -get /dev/hme link_mode   #=> valeurs posssibles : 1 = full,0 = half
```

(Par exemple, en solaris 10 : **kstat** donne 2 pour le full duplex et 1 pour le half duplex. Le « **ndd -get** » donne 1 pour le full duplex).

Note : kstat est plus recommandé pour les cartes de type bge ou hme.

3) en Solaris 10, vous pouvez utiliser la commande « **dladm show-dev** ».

Exemple de la remonté d'information d'une carte ce en Solaris 8 et 10:

```
[root@su0428:/] uname -a
SunOS su0428 5.10 Generic sun4u sparc SUNW,Sun-Fire-V490
[root@su0428:/] ndd -set /dev/ce instance 5
[root@su0428:/] ndd -get /dev/ce ?
?                               (read only)
instance                        (read and write)
adv_autoneg_cap                 (read and write)
adv_1000fdx_cap                 (read and write)
adv_1000hdx_cap                 (read and write)
adv_100T4_cap                   (read and write)
adv_100fdx_cap                  (read and write)
adv_100hdx_cap                  (read and write)
adv_10fdx_cap                   (read and write)
adv_10hdx_cap                   (read and write)
adv_asmpause_cap                (read and write)
adv_pause_cap                   (read and write)
master_cfg_enable               (read and write)
master_cfg_value                (read and write)
use_int_xcvr                    (read and write)
enable_ipg0                     (read and write)
ipg0                            (read and write)
ipg1                            (read and write)
ipg2                            (read and write)
rx_intr_pkts                    (read and write)
rx_intr_time                    (read and write)
red_dv4to6k                     (read and write)
red_dv6to8k                     (read and write)
red_dv8to10k                    (read and write)
red_dv10to12k                   (read and write)
tx_dma_weight                   (read and write)
rx_dma_weight                   (read and write)
infinite_burst                  (read and write)
disable_64bit                   (read and write)
```

```
accept_jumbo                    (read and write)
laggr_multistream               (read and write)

[root@su0428:/] kstat -p | grep ce5
ce:5:ce5:adv_cap_1000fdx        0
ce:5:ce5:adv_cap_1000hdx        0
ce:5:ce5:adv_cap_100T4  0
……….
ce:5:ce5:link_duplex    2
ce:5:ce5:link_pause     0
ce:5:ce5:link_speed     100
ce:5:ce5:link_up        1
……
ce:5:ce5:xcvr_addr      1
ce:5:ce5:xcvr_id        536894584
ce:5:ce5:xcvr_inits     1
ce:5:ce5:xcvr_inuse     1

[root@barbadine:/] uname -a
SunOS barbadine 5.8 Generic_117350-06 sun4u sparc SUNW,Sun-Fire
[root@barbadine:/] ndd -set /dev/ce instance 0
[root@barbadine:/] ndd -get /dev/ce link_status
1
[root@barbadine:/]  ndd -get /dev/ce link_mode
1
[root@barbadine:/]   ndd -get /dev/ce link_speed
1

[root@barbadine:/] kstat -p | grep ce0
ce:0:ce0:alignment_err  0
ce:0:ce0:brdcstrcv      131131418
ce:0:ce0:brdcstxmt      6439
ce:0:ce0:cap_1000fdx    0
ce:0:ce0:cap_1000hdx    0
ce:0:ce0:cap_100T4      0
ce:0:ce0:cap_100fdx     1
ce:0:ce0:cap_100hdx     1
ce:0:ce0:cap_10fdx      1
ce:0:ce0:cap_10hdx      1
ce:0:ce0:cap_asmpause   0
ce:0:ce0:cap_autoneg    1
ce:0:ce0:cap_pause      0
ce:0:ce0:class  net
ce:0:ce0:code_violations        0
ce:0:ce0:collisions     0
ce:0:ce0:crc_err        0
ce:0:ce0:crtime 488.05444544
ce:0:ce0:excessive_collisions   0
ce:0:ce0:first_collision        0
ce:0:ce0:ierrors        0
ce:0:ce0:ifspeed        100000000
ce:0:ce0:ipackets       778383725
ce:0:ce0:ipackets64     778383725
ce:0:ce0:ipackets_cpu00 281931829
ce:0:ce0:ipackets_cpu01 154621492
ce:0:ce0:ipackets_cpu02 170835589
ce:0:ce0:ipackets_cpu03 170994815
ce:0:ce0:late_collisions        0
ce:0:ce0:lb_mode        0
ce:0:ce0:length_err     0
ce:0:ce0:link_T4        0
ce:0:ce0:link_asmpause  0
ce:0:ce0:link_duplex    2
ce:0:ce0:link_pause     0
ce:0:ce0:link_speed     100
ce:0:ce0:link_up        1

[su0260@root:/platform/sun4u/kernel/drv] dladm show-dev
ce0             link: unknown   speed: 0     Mbps       duplex: unknown
ce1             link: unknown   speed: 1000  Mbps       duplex: full
```

```
lpfc0           link: unknown   speed: 0     Mbps       duplex: unknown
lpfc1           link: unknown   speed: 0     Mbps       duplex: unknown
ce2             link: unknown   speed: 100   Mbps       duplex: full
ce3             link: unknown   speed: 0     Mbps       duplex: unknown
ce4             link: unknown   speed: 0     Mbps       duplex: unknown
ce5             link: unknown   speed: 0     Mbps       duplex: unknown
ce6             link: unknown   speed: 100   Mbps       duplex: full
ce7             link: unknown   speed: 100   Mbps       duplex: full
lpfc2           link: unknown   speed: 0     Mbps       duplex: unknown
lpfc3           link: unknown   speed: 0     Mbps       duplex: unknown
```

Conclusion:
Les vitesses des interfaces est indépendantes des adresses positionnées ou des routages. En général Il faut utiliser la commande « ndd » pour positionner les vitesses et le vérifier.

3.4 Exemples pour le fichier /platform/sun4u/kernel/drv/bge.conf

Il se trouve dans le répertoire : /platform/sun4u/kernel/drv

Exemple1:

```
name="pci108e,abba" parent="/pci@21c,700000/pci@1" unit-address="0"
adv_autoneg_cap=0 adv_1000fdx_cap=0 adv_1000hdx_cap=0
adv_100fdx_cap=1 adv_100hdx_cap=0 adv_100T4_cap=0 adv_10fdx_cap=0
adv_10hdx_cap=0;
```

Exemple2:

```
# interface bge0
    name="bge" parent="/pci@1f,700000" unit-address="2"
    adv_autoneg_cap=0 adv_1000fdx_cap=0 adv_1000hdx_cap=0
    adv_100fdx_cap=1 adv_100hdx_cap=0 adv_100T4_cap=0 adv_10fdx_cap=0
    adv_10hdx_cap=0;

    # interface bge1
    name="bge" parent="/pci@1f,700000" unit-address="2,1"
    adv_autoneg_cap=0 adv_1000fdx_cap=0 adv_1000hdx_cap=0
    adv_100fdx_cap=1 adv_100hdx_cap=0 adv_100T4_cap=0 adv_10fdx_cap=0
    adv_10hdx_cap=0;

    # interface bge2
    name="bge" parent="/pci@1d,700000" unit-address="2"
    adv_autoneg_cap=0 adv_1000fdx_cap=0 adv_1000hdx_cap=0
    adv_100fdx_cap=1 adv_100hdx_cap=0 adv_100T4_cap=0 adv_10fdx_cap=0
    adv_10hdx_cap=0;

    # interface bge3
    name="bge" parent="/pci@1d,700000" unit-address="2,1"
    adv_autoneg_cap=0 adv_1000fdx_cap=0 adv_1000hdx_cap=0
    adv_100fdx_cap=1 adv_100hdx_cap=0 adv_100T4_cap=0 adv_10fdx_cap=0
    adv_10hdx_cap=0;
```

avec :

```
$ grep bge /etc/path_to_inst
     "/pci@1f,700000/network@2" 0 "bge"
     "/pci@1f,700000/network@2,1" 1 "bge"
     "/pci@1d,700000/network@2" 2 "bge"
     "/pci@1d,700000/network@2,1" 3 "bge"
```

3.5 Exemples pour le fichier /platform/sun4u/kernel/drv/ce.conf

Il se trouve dans le répertoire : /platform/sun4u/kernel/drv

Exemple1:

```
# ce.conf
# GigaSwift (ce) driver configuration file
# Copyright (c) 2000 by Sun Microsystems, Inc.
# All rights reserved.
#
#ident  "@(#)ce.conf      1.0.5     041110 SMI ES-PTS-NET/Lor"
#
###############################################################################
# This file is not needed for most installations !
#
# It MAY be needed for rare special customizations. In this case,
# copy the file and edit the following example to suit your needs.
###############################################################################
# a mettre dans /platform/sun4u/kernel/drv/ce.conf
#
# Depending on your changes, this file overwrites the Sun[TM] recommended
# default values (To use the default values, you do not need this file).
# In particular, Sun's[TM] recommendation is to leave Ethernet Auto-negotiation
# ON at both link partners (this is the default). See Product Documentation and
# SunSolve Infodocs 41665, 72033 for proper use first.
#
# Most parameters must be set on both sides to have effect
# (e.g check also switch and remote partner(s) settings).
#
# To install:
# 1) On SPARC systems, copy ce.conf to /platform/sun4u/kernel/drv/ce.conf
#    On x86 Opteron systems, copy ce.conf to /kernel/drv/ce.conf
# 2) edit file to your needs. Default is to do nothing.
# 3) reboot the system
#
# For settings per interface, name, parent and unit-address
# must be specified. Obtain the hardware path from /etc/driver_aliases and
# /etc/path_to_inst as shown below (unit-address is NOT the instance !) :
#
# % grep ce /etc/driver_aliases
# ce "pci108e,abba"
#           |                       % grep ce /etc/path_to_inst
#           |                       "/pci@21c,700000/pci@1/network@0" 0 "ce"
#           |                              |                      |
#           V                              V                      V
# name="pci108e,abba" parent="/pci@21c,700000/pci@1" unit-address="0"
#
# after this, append the settings you want.
# Repeat for each interface to be configured. All other will be default.
# Don't forget the ";" after each section. Hardware path examples:
#
# name="pci108e,abba" parent="/pci@21c,700000/pci@1" unit-address="0"
# (GigaSwift NIC in Sun Fire 15000 hsPCI Board IO 16 PCI Slot 1)
#
# name="pci100b,35" parent="/pci@9,600000/pci@2/pci@0" unit-address="1"
# (2nd port of QGE NIC in Sun Fire V880 PCI Slot 7)
#
# name="pci108e,abba" parent="/pci@9,700000" unit-address="2"
# name="pci108e,abba" parent="/pci@9,600000" unit-address="1"
# (V480R onboard interfaces)
#
# For global settings, no hardware path needs to
# be specified, the settings are used for all ce interfaces.
###############################################################################
#
# Select hardware path (not needed if all interfaces set the same)
#name="" parent="" unit-address=""
```

```
#
## example: forced 1000 Mbit/s, 802.3x Flow Control send and receive
# Speed/Mode values, set the desired to 1, all other to 0:
# les donnees ci-dessous sont fournies par le fichier '/etc/path_to_inst'

# patte ce1 CLI 1Go FD (maj de cette patte le 31/1/08)
# info extraite de la ligne : '"/ssm@0,0/pci@1a,700000/pci@1/network@1" 1 "ce" '
name="pci108e,abba" parent="/ssm@0,0/pci@1a,700000/pci@1" unit-address="1"
adv_autoneg_cap=1
adv_1000fdx_cap=1
adv_1000hdx_cap=0
adv_100fdx_cap=0
adv_100hdx_cap=0
adv_10fdx_cap=0
adv_10hdx_cap=0
;
## patte ce2 ADM 100 FD
# info extraite de la ligne : '"/ssm@0,0/pci@1b,600000/pci@1/network@0" 2 "ce"'
name="pci108e,abba" parent="/ssm@0,0/pci@1b,600000/pci@1" unit-address="0"
adv_autoneg_cap=0
adv_1000fdx_cap=0
adv_1000hdx_cap=0
adv_100fdx_cap=1
adv_100hdx_cap=0
adv_10fdx_cap=0
adv_10hdx_cap=0
;
## patte ce6 SVGTEMP 100 FD, patte temporaire cree a la demande de Mathieu GABORO (BT
SAS) le 13/2/08
# info extraite de la ligne : '"/ssm@0,0/pci@1e,700000/pci@1/network@0" 6 "ce"'
name="pci108e,abba" parent="/ssm@0,0/pci@1e,700000/pci@1" unit-address="0"
adv_autoneg_cap=0
adv_1000fdx_cap=0
adv_1000hdx_cap=0
adv_100fdx_cap=1
adv_100hdx_cap=0
adv_10fdx_cap=0
adv_10hdx_cap=0
;
## patte ce7 SVG 100 FD => passe a 1000FD (1Go) a la demande de Raouf (BT Reseau)
# le 7/2/2008, on remet a 100FD (BL)
# info extraites de la ligne : '"/ssm@0,0/pci@1e,700000/pci@1/network@1" 7 "ce"'
name="pci108e,abba" parent="/ssm@0,0/pci@1e,700000/pci@1" unit-address="1"
adv_autoneg_cap=0
adv_1000fdx_cap=0
adv_1000hdx_cap=0
adv_100fdx_cap=1
adv_100hdx_cap=0
adv_10fdx_cap=0
adv_10hdx_cap=0
;
#
## Enable Ethernet Flow Control
#adv_asmpause_cap=0
#adv_pause_cap=1
#
# Clock Master values, change in forced 1000 Mbit/s back-to-back config only:
#master_cfg_enable=1 # enable usage of master_cfg_value (ce >1.118)
#master_cfg_value=0 # one link partner 1, the other 0 (ce >1.118)
#
# Advertize values above per Auto-negotiation.
# Should be set to 1 in most cases, 0 disables Auto-negotiation
adv_autoneg_cap=0
#
#accept-jumbo=1
```

3.6 Liste des principaux drivers sur Sun:

Type	Description	Configuration
* hme	- FEPS chip and 10/100baseT fast Ethernet	#ndd -set /dev/hme instance 0
* qfe	- Sun Quad FastEthernet	#ndd -set /dev/qfe instance 0
* eri	- Sun Ethernet Rio Interface 10/100 Base.	#ndd -set /dev/eri instance 0
* ce	- GigaSwift Gigabit Ethernet 1.x	#ndd -set /dev/ce instance 0
* vge	- Gigabit Ethernet 1.x ,	#ndd -set /dev/vge instance 0
* ge	- Gigabit Ethernet 2.x/3.x	#ndd -set /dev/ge instance 0
* gem	- Gigabit Ethernet MAC 1000 Base	#ndd -set /dev/ge instance 0
* bge	- The fourth generation of Sun Gigabit Ethernet products	#ndd -set /dev/bge0 adv_1000fdx_cap 1
* dmfe	- Davicom 10/100Mb Ethernet Drivers(use DM9102A Ethernet chipset)	#ndd -set /dev/dmfe0 adv_100fdx_cap 1
* e1000g	- Intel Gigabit and 82546EB based 1000-baseT driver	ndd /dev/e1000g0 link_status
* nge	- Gigabit Ethernet Driver in Solaris 10 x86	ndd -set /dev/nge0 adv_100fdx_cap 1

4. LES ADRESSES IP:

Afin de positionner correctement l'adressage ip sur un serveur nous devons modifier 4 types de fichiers.

4.1 /etc/inet/hosts :

C'est le fichier qui contient les alias pour chaque adresse ip.
Attention: Il est propre à chaque machine. Les noms, indiqués dans ce fichier, peuvent sans aucun problème être différents d'un serveur à l'autre.

Exemple :

```
[root@barbadine:/etc] more /etc/hosts
#
# Internet host table
#
127.0.0.1       localhost
164.17.41.3     barbadine       loghost
164.17.54.33    barbadine-srv barbadineadm
```

- l'adresse IPv4 127.0.0.1 est l'adresse de **loopback,** réservée à l'interface réseau utilisée par la machine locale pour sa communication inter-processus.

- Quand « syslogd » est démarré lors du boot, il évalue le fichier « /etc/hosts » pour vérifier l'adresse IP associé au nom de l'hôte par rapport à l'adresse IP associée à « loghost ». En bref, le champ « loghost » sert à préciser la machine de collecte des messages de « syslog ». L'indication « loghost » indique où est monté le « syslogd ».

Notes : le fichier « nswitch.conf », indique dans quel ordre, on va recherche les adresses IP. Si au début de fichier, il y a l'indication « FILE », on lit alors le fichier « hosts » en 1er pour aller chercher les adresses IP.

4.2 /etc/inet/ipnodes (cas spécifique Solaris 10)

Si une application prend en charge IPv6, le fichier « /etc/inet/ipnodes » est consulté en premier, puis le fichier « /etc/inet/hosts » est lu. Pour les applications IPv4, seul le fichier « /etc/inet/hosts » est lu.

Attention: dans le cas de Solaris 10 se fichier est systématiquement lu avant « /etc/inet/hosts » pour toute résolution de nom. Il faut donc que les valeurs dans les deux fichiers soient cohérentes.
En solaris 10, le fichier « /etc/inet/ipnodes » est lu avant le fichier « /etc/hosts ». Il faut normalement que le contenu de ces 2 fichiers soit identique.

4.3 /etc/hostname.<interface>

Ces fichiers sont lus au démarrage. Ils permettent de configurer une interface avec l'adresse ip contenu dans le fichier. Vous pouvez le remplir avec l'adresse Ip directement ou avec les noms compris dans « ipnodes/hosts ».

Exemples : ce0 : 10.10.10.10, ce1 : 10.10.10.20

Dans ce fichier nous pouvons déclarer la configuration ipmp.

Cas particulier: Il est possible de ne pas avoir d'adresse fixe sur un serveur Solaris (non conseillé) s'il existe un serveur de dhcp. Ce fichier sera le suivant: /etc/dhcp.<interface>

exemple:
client# touch /etc/dhcp.hme0
(ou encore /etc/dhcp.ce0).

4.4 /etc/netmasks :

Ce fichier est très important c'est lui qui permet de déterminer les découpages des sous-réseaux. Au moment des calculs d'adresse, il est effectué un & logique avec le masque de sous réseau. C'est donc la partie des 0 qui permettra de faire la différence entre les équipements/adresses.
Ce fichier sert à savoir, selon la patte de sortie, par quel sous-réseau, on sort. Le masque indique la plage d'adresses réseau autorisées (vers laquelle on peut aller à partir de la patte), pour cette patte.

exemple:

```
# cat /etc/netmasks
#               128.32.0.0 255.255.255.0
#
164.17.0.0      255.255.248.0
10.0.0.0        255.255.248.0
164.128.48.0    255.255.248.0
164.128.72.0    255.255.248.0
164.8.48.0      255.255.248.0
164.8.72.0      255.255.248.0
```

Exemple2 : Un cas d'école:

> soient les adresses de sous réseau suivantes à positionner sur 3 interfaces:

```
10.10.10.41  255.255.255.128
10.10.10.141 255.255.255.240
10.10.10.241 255.255.255.252
```

Valeur en décimal	Valeur en hexadécimal	Valeur en binaire
255	FF	1111 1111
128	80	1000 0000
240	F0	1111 0000
252	FC	1111 1100

Le calcul des plages de sous-réseau est le suivant:

- 256 - 128 =128 ==> soit 2 plages de 0 à 127 et 128 à 255 (avec 0 le sous réseau et 127 l'adresse de broadcast)
- 256 - 240 = 16 ==> soit les plages 0 à 15 ; 16 à 31 , 32 à 47 ...
- 256 – 252 = 4 ==> soit 0 à 3 , 4 à 7 , 8 à 11 ....

il ne reste plus qu'à déterminer dans quelle tranche de sous réseau nous nous trouvons.

- 41 se trouve dans la tranche de 0 à 127
- 141/16 = 8,81 soit 8x16= 128 pour le début du sous réseau
- 241/4 = 60,25 soit 4x60= 240 pour le début du sous réseau

le contenu de « /etc/netmasks » sera donc :

```
10.10.10.0  255.255.255.128
10.10.10.128 255.255.255.240
10.10.10.240 255.255.255.252
```

Nous adresserons les réseaux suivants et aucun autre réseau sans routage spécifiques.

- de 10.10.10.1 à 10.10.10.126 par la patte 1 (avec 10.10.10.0 en adresse de réseau et 10.10.10.127 en adresse de broadcast).
- de 10.10.10.129 à 10.10.10.142 par la patte 2 (avec 10.10.10.128 en adresse de réseau et 10.10.10.143 en adresse de broadcast).
- de 10.10.10.241 à 10.10.10.242 par la patte 3 (avec 10.10.10.240 en adresse de réseau et 10.10.10.243 en adresse de broadcast).

Conclusion : Il ne faut pas négliger l'importance du fichier netmasks. Souvent un mauvais fichier ne permettra pas la sortie des trames sur la bonne interface. Nous ne serons pas alors sur le bon réseau.

Notes : a) Le PAIP (le plan d'adresse IP) donne tous les informations pour les différents sous-réseaux se SFR, pour le « resolv.conf », le « defaultrouter », les passerelles, les masques (masks).
Par exemple dans le PAIP, l'indication 10.10.10.41 255.255.255.128 détermine une plage d'adresse de 0 à 127. Si c'est 255.255.255.240, ce mask détermine 15 adresses de 1 à 15.

b) Pour modifier le mask, soit par la mise à jour du fichier « /etc/netmasks » soit dynamiquement par la commande « ifconfig … ». Par exemple :
ifconfig qfe0 netmasks +
ifconfig qfe0 netmasks 255.255.255.0
ifconfig qfe0 netmasks 0xffffff00

5. LA CONFIGURATION IPMP:

La configuration ipmp (IP multi pathing) fournit à Solaris la capacité de lever un SPOF réseau (single point of failure). Si un port réseau, un câble, un switch tombe en panne et si la configuration est correcte (avec une redondance de cartes ou switch) nous basculerons sur l'autre interface sans perturbation de service.
Avec l'IPMP, on a 2 interfaces dans le même sous-réseau. On ne perd pas la liaison avec le serveur.

- Pour cela, s'assurer que le paramètre local-mac-address? est positionné à true.

=> adresse IP de service ou virtuelle (IP d'entrée) => 2 adresses réelles :
=>1ère entrée (sert d'adresse de test)
=> 2ème entrée (sert d'adresse de test).
L'adresse de test est différent de l'adresse de service.
Exemple :

```
@ test => ce0 10.10.10.30 | même sous réseau ⇔ adr.de service
@ test2=> ce0 10.10.10.31 | ou virtuelle : 10.10.10.32
```

Notes : un/des firewall(s) peut/peuvent bloquer l'IPMP.
Dans la DMZ rapide, on bloque les ping, donc les interfaces seront failed sur la patte du routeur par defaut.

5.1 3 types de configuration:

5.1.1 Le mode actif-passif:

C'est le mode le plus courant. C'est celui utilisé chez SFR. Il n'y a qu'une seule interface de services et deux interfaces de test. Les interfaces de test ping le routeur par défaut si il en existe un dans le sous réseau. A défaut un routeur ou un host déclaré dans le sous réseau ou encore à défaut des machines présente après une requête multicast.

```
# cat /etc/hostname.ce0
barbadine-tst1 netmask + broadcast +  deprecated -failover group adm up
addif barbadine-adm netmask + broadcast + up

# cat /etc/hostname.ce8
barbadine-tst2 netmask + broadcast +  deprecated -failover standby group adm
up
```

En Solaris 10, il n'y a plus d'adresse réelle visible, juste une adresse de service.
Pour la détection du lien (le "link detection"), en Solaris 10, on fait soit un ping vers le default router, soit on fait des pings sur les adresses de broadcast (si ping vers router par défaut interdit en IPMP, il faut rajouter des routes statiques … sorte de bricolage).
Avec l'IPMP, il faut 2 switches en face.

5.1.2 Le mode actif/actif

Le serveur possède deux adresses de service. Les utilisateurs pourront attaquer l'une ou l'autre de ces adresses. C'est une configuration gourmande en adresse. Il en faut 4. (sorte de load balancing). On aura par exemple, 1000 clients sur la 1ère adresse, et 1000 autres sur la 2ème. On peut « plumber » la 2ème adresse.

 # cat /etc/hostname.ce0
barbadine-tst1 netmask + broadcast + deprecated -failover group adm up
addif barbadine-adm netmask + broadcast + up

 # cat /etc/hostname.ce8
barbadine-tst2 netmask + broadcast + deprecated -failover group adm up
addif barbadine-adm2 netmask + broadcast + up

5.1.3 le mode link detection (Solaris 10 seulement):

C'est une fonctionnalité qui existe qu'en Solaris 10. Il n'y a qu'une seule adresse de service et aucune adresse de tests. L'avantage c'est que nous ne sommes pas gourmands en adresse (juste 1). L'inconvénient c'est que nous testons juste le link au niveau de la carte. Comme nous ne pingons aucun équipements le routeur ou le switch peut avoir un problème et ipmp ne le détectera pas.

 # cat /etc/hostname.ce0
192.168.10.10 netmask + broadcast + group ipmp0 up

 # cat /etc/hostname.ce8
group ipmp0 up

 # ifconfig -a
ce0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 4 inet 192.168.10.10 netmask ffffff00 broadcast 192.168.10.255 groupname ipmp0 ether 0:3:ba:93:90:fc

ce8: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 5 inet 0.0.0.0 netmask ff000000 broadcast 0.255.255.255 groupname ipmp0 ether 0:3:ba:93:91:35

5.2 Quelques Définitions

Deprecated : l'adresse ne doit pas être choisi comme adresse source pour les packets de sorties. Si une adresse non deprecated est présent c'est elle qui sera préféré plutôt que les autres deprecated. L'interface répondra à un ping et peut faire des connexions. L'interface n'est pas utilisé pour un traffic IP normal. Une adresse deprecated ne peut être utilise comme adresse source pour les packets de sortie a moins qu'il y ai pas

d'autre @ disponible sur l'interface ou si les applications ont cible cette adresse explicitement.
En résumé: adresse non utilisé par les applicatifs.
« depreciated » indique que l'adresse de service est différente de l'adresse de test.

No failover : l'interface ne basculera pas si une failure est détecté.

Standby : l'interface physique est marqué comme interface de standby pour l'interface de group. Une interface standby ne pourra pas être sélectionné pour envoyer des paquets en sorties a moins que les autres interfaces du group soient HS. Le statut apparaissant est ''standby inactive'' sauf quand il y a bascule juste ''standby''. Si tu le met pas l'interface physique peut sortir des paquets; même si tu as mis deprecated. En fait solaris fait du load sharing si tu le met pas.
On parle de load sharing plutôt que load balancing car le load balancing est contrôlable mais pas le load sharing. En résumé: standby = interface réseau non utilisé pour le trafic sortant.

in.mpathd :c'est le démon qui gerent l'ipmp. Il est chargé des detection et des bascules.

i**f_mpadm –d et if_mpadm –r** : permet d'arrêter /redémarrer une interface réseau sans la débrancher physiquement. C'est pratique pour tester l'ipmp.

6. LE ROUTAGE

6.1 Quelques fichiers:

/etc/defaultrouter:
Ce fichier contient l'adresse du routeur par défaut pour le serveur (sert au routage).

/etc/notrouter: (non positionné chez SFR)
Si le node ne doit pas être considéré comme routeur, effectuer l'action suivante
touch /etc/notrouter (cf /etc/rc2.d/S69inet : ip_forwarding)
Lors du redémarrage de la machine, un script de démarrage cherche la présence du fichier /etc/notrouter. Si ce fichier existe, ce script ne lance pas in.routed -s ou in.rdisc -s, et n'active pas ip_forwarding sur toutes les interfaces configurées "up" par ifconfig.

Le fichier /etc/gateways:(non positionné chez SFR)
Le processus « in.routed » lit le fichier optionnel /etc/gateways à l'initialisation pour construire sa table de routage. C'est une autre façon d'ajouter une (ou plusieurs) route(s) permanente(s) dans la table de routage (la première façon étant le routeur par défaut dans /etc/defaultrouter). Notez qu'ici, il ne s'agit pas de routes par défaut. Voici la syntaxe d'une ligne dans /etc/gateways :

```
net <dest.> gateway <routeur> metric <compteur> [passive][active]
```

/etc/rc2.d/S25conf-ethernet.ksh: (spécifique SFR)
Ce fichier contient des routages spécifiques autres que le routage par défaut.

Exemple :

```
# cat /etc/rc2.d/S25conf-ethernet.ksh
…..
route add -net 10.40.192.0/18 10.43.208.1
route add -net 10.30.192.0/18 10.43.208.1
route add -net 10.42.192.0/18 10.43.208.1
route add -net 10.43.192.0/18 10.43.208.1
route add -host 164.17.42.135 10.43.208.1
route add -host 164.17.42.136 10.43.208.1
route add -host 164.17.51.107 10.43.208.1
route add -host 164.17.41.47 10.43.208.1
route add -host 164.17.46.41 10.43.208.1
```

Notes : a) Ici, ci-avant, le « modificateur » « /18 » spécifie le masque de sous réseau. C'est le nombre de 1 qui sert de filtre (vu plus haut).
Par exemple, on rajoute des routes statiques sur la patte adm, pour joindre tel ou tel sous-réseau. Sinon, on va (par défaut) sur le router par défaut. Exemple :
Route add –net 10.43.19.0/18 10.43.208.1 avec :
-net : pour ajouter un réseau, « 10.43.208.1 », adresse d'un router d'administration différent du router par défaut. Pour ce réseau « 10.43.19.0/18 », on passe par tel ou tel router. (ici « /18 » est le nombre de « 1 » qui sert de filtre).
b) On appelle cette notation « /nn » la notation CIDR (Classless Inter-Domain Routing) « /18 », « /21 », « /24 » …
c) Pour retirer une route : exemple : route delete –net 10.43.192.0/18 10.43.208.1

Quelques exemples:

Nombre de 1	Valeur en binaires	Valeur en hexadécimal	Masque de sous réseau
18	11111111 11111111 11	FF FF C- --	255.255.192.0
21	11111111 11111111 11111	FF FF F8 --	255.255.248.0
24	11111111 11111111 11111111	FF FF FF --	255.255.255.0

Attention: Dans le cas on l'on désire supprimer une route. Nous utilisons la commande route avec l'option delete. Il faut bien spécifier tous les champs comme à la création.

Exemple:
route add -net 10.40.192.0/18 10.43.208.1

route delete -net 10.40.192.0 10.43.208.1 ne fonctionnera pas
route delete -net 10.40.192.0/18 10.43.208.1 fonctionnera.

6.2 Quelques commandes et précisions:

[su0245@root:/etc/rc2.d] netstat -nrv

```
    IRE Table: IPv4
Destination      Mask                Gateway             Device   Mxfrg   Rtt     Ref     Flg     Out     In/Fwd
------------------- --------------- ------------------- ------   -----   -----   ---     ---     -----   ------
164.17.51.107    255.255.255.255     10.43.208.1                  1500*      0    1       UGH       0         0
164.17.46.41     255.255.255.255     10.43.208.1                  1500*      0    1       UGH       0         0
164.17.41.47     255.255.255.255     10.43.208.1                  1500*      0    1       UGH       0         0
164.17.42.136    255.255.255.255     10.43.208.1                  1500*      0    1       UGH       0         0
164.17.42.135    255.255.255.255     10.43.208.1                  1500*      0    1       UGH       0         0
10.43.74.0       255.255.254.0       10.43.74.19     ce0:1        1500*      0    1       U       382         0
10.43.74.0       255.255.254.0       10.43.74.19     ce0          1500*      0    1       U         0         0
10.43.74.0       255.255.254.0       10.43.74.19     ce4          1500*      0    1       U       373         0
10.43.208.0      255.255.240.0       10.43.213.10    ce1:1        1500*      0    1       U      1463         0
10.43.208.0      255.255.240.0       10.43.213.10    ce1          1500*      0    1       U         0         0
10.43.208.0      255.255.240.0       10.43.213.10    ce5          1500*      0    1       U       377         0
10.43.144.0      255.255.240.0       10.43.149.6     ce8          1500*      0    1       U        19         0
10.42.192.0      255.255.192.0       10.43.208.1                  1500*      0    1       UG     1097         0
10.43.192.0      255.255.192.0       10.43.208.1                  1500*      0    1       UG        0         0
10.40.192.0      255.255.192.0       10.43.208.1                  1500*      0    1       UG       10         0
10.30.192.0      255.255.192.0       10.43.208.1                  1500*      0    1       UG        0         0
224.0.0.0        240.0.0.0           10.43.213.10    ce1:1        1500*      0    1       U         0         0
Default          0.0.0.0             10.43.74.1                   1500*      0    1       UG      397         0
127.0.0.1        255.255.255.255     127.0.0.1       lo0          8232*      0    7       UH     2363         0
```

Ref: Nombre de routes actuelles qui partagent la même adresse dans la couche Interface réseau(Ethernet).

Flags État de la route :
U l'interface est active (up)
H la destination est un hôte (pas un réseau)
G la destination est une passerelle (route indirecte)

D redirection ICMP
UG : Gateway UP.
UH : host UP.
UGH : ?

Note : avec « netstat –rnv », on voit les interface « plombées ».

/usr/sbin/snoop:
Vous utilisez « snoop » pour capturer des paquets du réseau et afficher leur contenu. L'affichage s'effectue dès la réception du paquet ou en différé, après avoir enregistré les paquets dans un fichier. Dans le cas d'un réseau chargé, l'affichage des paquets prend du temps et il est possible d'en perdre ; l'enregistrement dans un fichier est plus efficace.
Le « snoop » permet de déterminer les chemins réseaux physiques ouverts. On saura par où l'on peut installer le serveur, par le réseau, par « jumpstart » (avec la commande « boot net … »).

traceroute:
L'utilitaire traceroute permet de savoir si les paquets empruntent effectivement la route prévue lorsqu'on cherche à joindre une destination.
Avec la commande traceroute, on sait par quelle patte on sort (sa log l'indique, par exemple qfe8 …).
Exemple : traceroute 10.169.11.18

nslookup : permet de savoir si l'alias (le nom) ou l'adresse IP est résolu au niveau du DNS.

La commande /sbin/ifconfig:
La commande ifconfig, employée par l'administrateur système, configure tous les paramètres de l'interface réseau. Elle est utilisée dans la phase de démarrage du système par le script /etc/rcS.d/S30network.sh (Solaris 8) pour définir l'adresse réseau de chaque interface physique du système. Elle sert à nouveau dans le script etc/rc2.d/S72inetsvc, pour réinitialiser les interfaces configurées par NIS/NIS+. ifconfig peut servir à redéfinir un paramètre IP d'une interface. L'argument plumb de la commande ifconfig ouvre le périphérique associé à l'interface physique et met en place les streams nécessaires à TCP/IP pour utiliser ce périphérique. L'interface apparaît alors dans la liste donnée par ifconfig -a. L'argument unplumb détruit les streams et ferme le périphérique. L'interface n'apparaît plus dans la liste donnée par ifconfig -a après qu'elle a été enlevée à l'aide de la commande ifconfig unplumb.

Forcer en full duplex dès L'OBP au boot net:
Boot net: speed=100,duplex=full,link_clock=master,
exemple:
obp> boot net:speed=100,duplex=full - install

7. ANNEXES

7.1 Annexe A: le fichier S68net-tune

```
#!/sbin/sh
# /etc/rc2.d/S68net-tune /etc/init.d/net-tune
# Copyright (c) 2000 by Sun Microsystems, Inc.
# All rights reserved.
#
#ident  "@(#)net-tune      2.1.0     060825 SMI ES-PTS-NET/Lor"
#
##############################################################################
# This script is not needed for most installations !                         #
#                                                                            #
# Newer driver versions provide an example driver.conf file, e.g.e1000g.conf #
# below /kernel/drv or /platform/*/kernel/drv . Use this file when existing. #
##############################################################################
#
# This script MAY be needed for rare special customizations. In this case,
# copy the script and edit the following example code to suit your needs.
#
# Depending on your changes, this script overwrites the Sun[TM] recommended
# default values (To use the default values, you do not need this script).
# In particular, Sun's[TM] recommendation is to leave Ethernet Auto-negotiation
# ON at both link partners (this is the default). See Product Documentation and
# SunSolve Infodocs for proper use first (e.g. Infodocs 17416, 23041, 41665,
# 70282). Most parameters must be set on both sides to have effect
# (e.g check also switch and IP partner settings).
# To install:
# 1) cp S68net-tune to /etc/rc2.d
# 2) perform edits on the sript as required
# 3) chmod 744 /etc/rc2.d/S68net-tune
# 4) chown root:sys /etc/rc2.d/S68net-tune
# 5) ln /etc/rc2.d/S68net-tune /etc/init.d/net-tune

PATH=/usr/bin:/usr/sbin

case "$1" in
     'start')
    echo "Implementing Solaris Network Tuning."

# hme-Interfaces
# hme0
    #ndd -set /dev/hme instance 0
    #ndd -set /dev/hme adv_100T4_cap 0
    #ndd -set /dev/hme adv_100fdx_cap 1
    #ndd -set /dev/hme adv_100hdx_cap 0
    #ndd -set /dev/hme adv_10fdx_cap 0
    #ndd -set /dev/hme adv_10hdx_cap 0
    #ndd -set /dev/hme adv_autoneg_cap 0
# hme1
    #ndd -set /dev/hme instance 1
    #ndd -set /dev/hme adv_100T4_cap 0
    #ndd -set /dev/hme adv_100fdx_cap 1
    #ndd -set /dev/hme adv_100hdx_cap 0
    #ndd -set /dev/hme adv_10fdx_cap 0
    #ndd -set /dev/hme adv_10hdx_cap 0
    #ndd -set /dev/hme adv_autoneg_cap 0

# eri-Interfaces
# eri0
        #ndd -set /dev/eri instance 0
        #ndd -set /dev/eri adv_100T4_cap 0
        #ndd -set /dev/eri adv_100fdx_cap 1
        #ndd -set /dev/eri adv_100hdx_cap 0
        #ndd -set /dev/eri adv_10fdx_cap 0
        #ndd -set /dev/eri adv_10hdx_cap 0
```

```
        #ndd -set /dev/eri adv_autoneg_cap 0

# qfe-Interfaces
# qfe0
    #ndd -set /dev/qfe instance 0
    #ndd -set /dev/qfe adv_100T4_cap 0
    #ndd -set /dev/qfe adv_100fdx_cap 1
    #ndd -set /dev/qfe adv_100hdx_cap 0
    #ndd -set /dev/qfe adv_10fdx_cap 0
    #ndd -set /dev/qfe adv_10hdx_cap 0
    #ndd -set /dev/qfe adv_autoneg_cap 0
# qfe1
    #ndd -set /dev/qfe instance 1
    #ndd -set /dev/qfe adv_100T4_cap 0
    #ndd -set /dev/qfe adv_100fdx_cap 1
    #ndd -set /dev/qfe adv_100hdx_cap 0
    #ndd -set /dev/qfe adv_10fdx_cap 0
    #ndd -set /dev/qfe adv_10hdx_cap 0
    #ndd -set /dev/qfe adv_autoneg_cap 0

# dmfe-Interfaces (Netra X1)
# Use this script OR dmfe.conf from this folder.
# dmfe0
    #ndd -set /dev/dmfe0 adv_100fdx_cap 1
    #ndd -set /dev/dmfe0 adv_100hdx_cap 0
    #ndd -set /dev/dmfe0 adv_10fdx_cap 0
    #ndd -set /dev/dmfe0 adv_10hdx_cap 0
    #ndd -set /dev/dmfe0 adv_autoneg_cap 0
# dmfe1
    #ndd -set /dev/dmfe1 adv_100fdx_cap 1
    #ndd -set /dev/dmfe1 adv_100hdx_cap 0
    #ndd -set /dev/dmfe1 adv_10fdx_cap 0
    #ndd -set /dev/dmfe1 adv_10hdx_cap 0
    #ndd -set /dev/dmfe1 adv_autoneg_cap 0

# vge-Interfaces (Gigabit 1.x)
# vge0
    #ndd -set /dev/vge instance 0
    #ndd -set /dev/vge fdr_filter 1
    #ndd -set /dev/vge link_negotiation 0
# vge1
    #ndd -set /dev/vge instance 1
    #ndd -set /dev/vge fdr_filter 1
    #ndd -set /dev/vge link_negotiation 0

# ge-Interfaces (Gigabit 2.x and 3.x)
# example: forced 1000 Mbit/s, 802.3x Flow Control send and receive
# check if your NIC is 802.3x capable (ndd parameter pause_cap, asm_dir_cap)
# ge0
    #ndd -set /dev/ge instance 0
    #ndd -set /dev/ge adv_1000fdx_cap 1
    #ndd -set /dev/ge adv_1000hdx_cap 0
        #ndd -set /dev/ge adv_pauseTX 1
    #ndd -set /dev/ge adv_pauseRX 1
    #ndd -set /dev/ge adv_1000autoneg_cap 0
# ge1
    #ndd -set /dev/ge instance 1
    #ndd -set /dev/ge adv_1000fdx_cap 1
    #ndd -set /dev/ge adv_1000hdx_cap 0
        #ndd -set /dev/ge adv_pauseTX 1
    #ndd -set /dev/ge adv_pauseRX 1
    #ndd -set /dev/ge adv_1000autoneg_cap 0


# ce-Interfaces (GigaSwift UTP/Fiber)
# PDE encourages the use of ce.conf for boot persistant configurations.
# Pls. use the ce.conf file in this folder instead of this script.


# bge-Interfaces (SF-V210, SF-V240, SB-1500, ... on-board only)
# Use this script OR bge.conf from this folder.
```

```
# example: forced 1000 Mbit/s, 802.3x Flow Control send and receive
# bge0
        # Speed/Mode values, set the desired to 1, all other to 0:
    #ndd -set /dev/bge0 adv_1000fdx_cap 1
    #ndd -set /dev/bge0 adv_1000hdx_cap 0
    #ndd -set /dev/bge0 adv_100fdx_cap 0
    #ndd -set /dev/bge0 adv_100hdx_cap 0
    #ndd -set /dev/bge0 adv_10fdx_cap 0
    #ndd -set /dev/bge0 adv_10hdx_cap 0
        #ndd -set /dev/bge0 adv_asym_pause_cap 0
        #ndd -set /dev/bge0 adv_pause_cap 1
    #ndd -set /dev/bge0 adv_autoneg_cap 0
# bge1
        # Speed/Mode values, set the desired to 1, all other to 0:
    #ndd -set /dev/bge1 adv_1000fdx_cap 1
    #ndd -set /dev/bge1 adv_1000hdx_cap 0
    #ndd -set /dev/bge1 adv_100fdx_cap 0
    #ndd -set /dev/bge1 adv_100hdx_cap 0
    #ndd -set /dev/bge1 adv_10fdx_cap 0
    #ndd -set /dev/bge1 adv_10hdx_cap 0
        #ndd -set /dev/bge1 adv_asym_pause_cap 0
        #ndd -set /dev/bge1 adv_pause_cap 1
    #ndd -set /dev/bge1 adv_autoneg_cap 0

# switch MTU path discovery off
    #ndd -set /dev/ip ip_path_mtu_discovery 0
# TCP/UDP buffer tunings
    #ndd -set /dev/tcp tcp_xmit_hiwat 65536
    #ndd -set /dev/tcp tcp_recv_hiwat 65536
    #ndd -set /dev/udp udp_xmit_hiwat 65536
    #ndd -set /dev/udp udp_recv_hiwat 65536
# TCP congestion window size, for Solaris <= 2.5, only
    #ndd -set /dev/tcp tcp_cwnd_max 65535
# Change TCP maximum retransmission interval, for Solaris <= 7, only
        #ndd -set /dev/tcp tcp_rexmit_interval_max 60000
# Change number of TCP slow start initial packets to work arround
# TCP slow start behaviour. For Solaris <= 7, only
        #ndd -set /dev/tcp tcp_slow_start_initial 2

         ;;

     'stop')
         echo "No kernel parameters changed."
         ;;

        *)
         echo "Usage: $0 {start|stop}"
         ;;

  esac
  exit 0
```

7.2 Annexe B: le fichier ce.conf

```
# ce.conf
# GigaSwift (ce) driver configuration file
# Copyright (c) 2000 by Sun Microsystems, Inc.
# All rights reserved.
#
#ident  "@(#)ce.conf      1.0.5     041110 SMI ES-PTS-NET/Lor"
#
###############################################################################
# This file is not needed for most installations !
#
# It MAY be needed for rare special customizations. In this case,
# copy the file and edit the following example to suit your needs.
###############################################################################
#
```

```
# Depending on your changes, this file overwrites the Sun[TM] recommended
# default values (To use the default values, you do not need this file).
# In particular, Sun's[TM] recommendation is to leave Ethernet Auto-negotiation
# ON at both link partners (this is the default). See Product Documentation and
# SunSolve Infodocs 41665, 72033 for proper use first.
#
# Most parameters must be set on both sides to have effect
# (e.g check also switch and remote partner(s) settings).
#
# To install:
# 1) On SPARC systems, copy ce.conf to /platform/sun4u/kernel/drv/ce.conf
#    On x86 Opteron systems, copy ce.conf to /kernel/drv/ce.conf
# 2) edit file to your needs. Default is to do nothing.
# 3) reboot the system
#
# For settings per interface, name, parent and unit-address
# must be specified. Obtain the hardware path from /etc/driver_aliases and
# /etc/path_to_inst as shown below (unit-address is NOT the instance !) :
#
# % grep ce /etc/driver_aliases
# ce "pci108e,abba"
#           |                      % grep ce /etc/path_to_inst
#           |                      "/pci@21c,700000/pci@1/network@0" 0 "ce"
#           |                              |                       |
#           V                              V                       V
# name="pci108e,abba" parent="/pci@21c,700000/pci@1" unit-address="0"
#
# after this, append the settings you want.
# Repeat for each interface to be configured. All other will be default.
# Don't forget the ";" after each section. Hardware path examples:
#
# name="pci108e,abba" parent="/pci@21c,700000/pci@1" unit-address="0"
# (GigaSwift NIC in Sun Fire 15000 hsPCI Board IO 16 PCI Slot 1)
#
# name="pci100b,35" parent="/pci@9,600000/pci@2/pci@0" unit-address="1"
# (2nd port of QGE NIC in Sun Fire V880 PCI Slot 7)
#
# name="pci108e,abba" parent="/pci@9,700000" unit-address="2"
# name="pci108e,abba" parent="/pci@9,600000" unit-address="1"
# (V480R onboard interfaces)
#
# For global settings, no hardware path needs to
# be specified, the settings are used for all ce interfaces.
###############################################################################
#
# Select hardware path (not needed if all interfaces set the same)
#name="" parent="" unit-address=""
#
# example: forced 1000 Mbit/s, 802.3x Flow Control send and receive
# Speed/Mode values, set the desired to 1, all other to 0:
#adv_1000fdx_cap=1
#adv_1000hdx_cap=0
#adv_100fdx_cap=0
#adv_100hdx_cap=0
#adv_10fdx_cap=0
#adv_10hdx_cap=0
#
# Enable Ethernet Flow Control
#adv_asmpause_cap=0
#adv_pause_cap=1
#
# Clock Master values, change in forced 1000 Mbit/s back-to-back config only:
#master_cfg_enable=1 # enable usage of master_cfg_value (ce >1.118)
#master_cfg_value=0 # one link partner 1, the other 0 (ce >1.118)
#
# Advertize values above per Auto-negotiation.
# Should be set to 1 in most cases, 0 disables Auto-negotiation
#adv_autoneg_cap=0
#
#
# Enable Jumbo Frames
```

```
#accept-jumbo=1
;
```

Exemple :

```
# ce.conf
# GigaSwift (ce) driver configuration file
# Copyright (c) 2000 by Sun Microsystems, Inc.
# All rights reserved.
#
#ident  "@(#)ce.conf      1.0.5     041110 SMI ES-PTS-NET/Lor"
#
###############################################################################
# This file is not needed for most installations !
#
# It MAY be needed for rare special customizations. In this case,
# copy the file and edit the following example to suit your needs.
###############################################################################
# a mettre dans /platform/sun4u/kernel/drv/ce.conf
#
# Depending on your changes, this file overwrites the Sun[TM] recommended
# default values (To use the default values, you do not need this file).
# In particular, Sun's[TM] recommendation is to leave Ethernet Auto-negotiation
# ON at both link partners (this is the default). See Product Documentation and
# SunSolve Infodocs 41665, 72033 for proper use first.
#
# Most parameters must be set on both sides to have effect
# (e.g check also switch and remote partner(s) settings).
#
# To install:
# 1) On SPARC systems, copy ce.conf to /platform/sun4u/kernel/drv/ce.conf
#    On x86 Opteron systems, copy ce.conf to /kernel/drv/ce.conf
# 2) edit file to your needs. Default is to do nothing.
# 3) reboot the system
#
# For settings per interface, name, parent and unit-address
# must be specified. Obtain the hardware path from /etc/driver_aliases and
# /etc/path_to_inst as shown below (unit-address is NOT the instance !) :
#
# % grep ce /etc/driver_aliases
# ce "pci108e,abba"
#           |                       % grep ce /etc/path_to_inst
#           |                       "/pci@21c,700000/pci@1/network@0" 0 "ce"
#           |                              |                      |
#           V                              V                      V
# name="pci108e,abba" parent="/pci@21c,700000/pci@1" unit-address="0"
#
# after this, append the settings you want.
# Repeat for each interface to be configured. All other will be default.
# Don't forget the ";" after each section. Hardware path examples:
#
# name="pci108e,abba" parent="/pci@21c,700000/pci@1" unit-address="0"
# (GigaSwift NIC in Sun Fire 15000 hsPCI Board IO 16 PCI Slot 1)
#
# name="pci100b,35" parent="/pci@9,600000/pci@2/pci@0" unit-address="1"
# (2nd port of QGE NIC in Sun Fire V880 PCI Slot 7)
#
# name="pci108e,abba" parent="/pci@9,700000" unit-address="2"
# name="pci108e,abba" parent="/pci@9,600000" unit-address="1"
# (V480R onboard interfaces)
#
# For global settings, no hardware path needs to
# be specified, the settings are used for all ce interfaces.
###############################################################################
#
# Select hardware path (not needed if all interfaces set the same)
#name="" parent="" unit-address=""
#
# example: forced 1000 Mbit/s, 802.3x Flow Control send and receive
# Speed/Mode values, set the desired to 1, all other to 0:
```

```
# patte ce0 CLI 1Go
name="pci108e,abba" parent="/pci@8,700000/pci@2" unit-address="0"
adv_1000fdx_cap=1
adv_1000hdx_cap=0
adv_100fdx_cap=0
adv_100hdx_cap=0
adv_10fdx_cap=0
adv_10hdx_cap=0
;
# patte ce1 ADM 100 FD
name="pci108e,abba" parent="/pci@8,700000/pci@2" unit-address="1"
adv_1000fdx_cap=0
adv_1000hdx_cap=0
adv_100fdx_cap=1
adv_100hdx_cap=0
adv_10fdx_cap=0
adv_10hdx_cap=0
;
# patte ce8 SVG 100 FD
name="pci108e,abba" parent="/pci@9,700000" unit-address="2"
adv_1000fdx_cap=0
adv_1000hdx_cap=0
adv_100fdx_cap=1
adv_100hdx_cap=0
adv_10fdx_cap=0
adv_10hdx_cap=0
#
# Enable Ethernet Flow Control
#adv_asmpause_cap=0
#adv_pause_cap=1
#
# Clock Master values, change in forced 1000 Mbit/s back-to-back config only:
#master_cfg_enable=1 # enable usage of master_cfg_value (ce >1.118)
#master_cfg_value=0 # one link partner 1, the other 0 (ce >1.118)
#
# Advertize values above per Auto-negotiation.
# Should be set to 1 in most cases, 0 disables Auto-negotiation
adv_autoneg_cap=0
#
#accept-jumbo=1
;
```

7.3 Annexe C: le fichier bge.conf

```
#
# Copyright 2004 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
#ident  "@(#)bge.conf   1.7     04/09/23 SMI"
#
# /platform/sun4u/kernel/drv/bge.conf file for the BGE driver, for
# Broadcom 579x Gigabit Ethernet devices
#
# All the properties below can be set globally (i.e. for all instances
# of BGE), or on a per-instance basis.  See driver.conf(4) for details
# of the syntax of global and per-instance properties.
#
# Properties specified in this file take effect when the driver is first
# loaded, typically just after system boot.  Changes to the file will
# therefore not take effect until the next reboot, but will be permanent
# thereafter.
#
# Some of the driver's parameters can also be changed using ndd(1m).
# Changes made with ndd apply only to a specific instance (e.g. bge1).
# They take effect immediately, but are lost if the driver is unloaded.
#
```

```
#
# The autonegotiation feature can be controlled by the boolean properties
# listed below.
#
# Firstly, 'adv_autoneg_cap' controls whether autonegotiation is enabled.
#
# If autonegotiation is turned OFF ("forced mode"), the remaining 'adv_*'
# speed/duplex properties force selection of a specific mode, namely,
# the first mode found to be enabled, in highest-to-lowest speed order
# (thus, if adv_1000fdx_cap=1, all other values will be ignored; to force
# 10/hdx mode, *all* the faster modes must be explicitly disabled).
#
# BEWARE - it's very easy to end up with a non-working link using forced
# mode.  There's NO validation that the link partner actually supports
# the mode that this device has been forced into.  In some cases, this
# will prevent the link coming up; in others, the link status will show
# 'up' (electrical connection made) but data transfer will not work at
# all, or will work poorly (low throughput, high collision rates, etc).
#
# Note that many switches *require* autonegotiation in order to operate
# at 1000Mbps or in full-duplex mode or with flow control.  In other words,
# the only combinations that are likely to work with autonegotiation off
# are 100Mbps/half-duplex and 10Mbps/half-duplex, unless the peer has also
# been manually forced to some other (matching) combination.
#
# With autonegotiation ON (the default and preferred mode), the 'adv_*'
# properties control which capabilities are advertised to the partner.
# The default is to advertise all the capabilities that the hardware
# supports; thus, the properties below serve only to limit the advertised
# capabilities to restricted subset -- it is not possible to advertise a
# capability that the hardware does not support.
#
# The autonegotiation process will then automagically select the fastest
# speed/duplex mode and greatest degree of flow control supported by both
# partners.
#
# If the local device is set to autonegotiate, but the link partner can't
# or doesn't autonegotiate, the correct speed will be determined anyway,
# and HALF-DUPLEX mode will be selected, as mandated by the IEEE802.3
# standard.  This will yield the correct result if the partner is in fact
# incapable of autonegotiating: it must be a half-duplex device, because
# the only devices that don't support autonegotation are half-duplex (the
# standard says that all full-duplex-capable devices must also support
# autonegotiation).
#
# However, this choice will NOT be correct if the peer is actually capable
# of autonegotiation and full-duplex operation, and has been manually set
# to "forced full-duplex without autonegotiation" (a mode not recommended
# by the IEEE standard).  The link will appear to work, but the duplex
# mismatch will result in packet loss and spurious "late collisions".  In
# such cases, the preferred solution is to enable autonegotiation by the
# peer.  Failing that, autonegotiation by the BGE device can be disabled,
# and forced mode used to match the peer's forced settings as above.
#
# adv_autoneg_cap       = 1;
# adv_1000fdx_cap       = 1;
# adv_1000hdx_cap       = 1;
# adv_100T4_cap         = 0;
# adv_100fdx_cap        = 1;
# adv_100hdx_cap        = 1;
# adv_10fdx_cap         = 1;
# adv_10hdx_cap         = 1;
# adv_asym_pause_cap    = 0;
# adv_pause_cap         = 1;
#
# All of these parameters can can also be queried and modified at run-time
# by use of the ndd(1m) command.
#
#
#
```

```
# OBP's device driver exports methods to set the link speed explicity,
# which then pass the information to the Solaris driver through the
# 'transfer-speed' property.  It therefore SHOULDN'T be set here, but
# is documented for completeness.  If the 'transfer-speed' property is
# set to 10, 100, or 1000, the link will be set to the selected speed,
# and autonegotiation ENABLED but restricted to the specified speed.
# The correct duplex setting will be determined by autonegotiation.
#
# This property, if set, overrides and alters the settings of the adv_*
# parameters corresponding to the properties above.
#
# transfer-speed        = 1000;
#
##############################################################################
#
# As a third alternative, the following two properties can be set to
# force the link speed/duplex setting instead.  Doing so will override
# and alter the settings of the adv_* parameters corresponding to the
# properties above, and take precedence over all other means of setting
# the speed/duplex at boot time.
#
# Autonegotiation will be DISABLED if EITHER of these properties is set,
# therefore BOTH properties should be set explicity if either one is.
# 'speed' may be set to 10, 100 or 1000, while 'full-duplex' may be 0 or 1.
#
# See the warning above about the potential for misconfiguration when
# autonegotiation is disabled.  Defining these properties could leave your
# system configured so that the network will not work at all after reboot,
# requiring manual intervention and further reboots to recover!
#
# speed                 = 100;
# full-duplex           = 0;
#
##############################################################################
#
# The property below represents the list of subsystem vendor/device pairs
# with which driver operation is supported.  This list will be updated and
# extended as new subsystems are validated ...
#
##############################################################################
#
# Example for settings per interface (here: bge0 and bge1 on-board a v20z)
#
#name="bge" parent="/pci@0,0/pci1022,7450@a" unit-address="2"
#adv_autoneg_cap=0 adv_1000fdx_cap=0 adv_1000hdx_cap=0 adv_100fdx_cap=1
#adv_100hdx_cap=0 adv_10fdx_cap=0 adv_10hdx_cap=0;
#
#name="bge" parent="/pci@0,0/pci1022,7450@a" unit-address="2,1"
#adv_autoneg_cap=1 adv_1000fdx_cap=1 adv_1000hdx_cap=0 adv_100fdx_cap=0
#adv_100hdx_cap=0 adv_10fdx_cap=0 adv_10hdx_cap=0;

bge-known-subsystems    = 0x108e1647,
                          0x108e1648,
                          0x108e16a7,
                          0x108e16a8,
                          0x17c20010,
                          0x17341013,
                          0x101402a6,
                          0x10f12885,
                          0x17c20020,
                          0x10b71006,
                          0x10280109,
                          0x1028865d,
                          0x0e11005a,
                          0x103c12bc;
```

Exemple :

```
##############################################################################
#
```

```
# Copyright 2005 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
#ident  "@(#)bge.conf   1.8     05/09/01 SMI"
#
# Driver.conf file for the BGE driver, for
# Broadcom 579x Gigabit Ethernet devices
#
# All the properties below can be set globally (i.e. for all instances
# of BGE), or on a per-instance basis.  See driver.conf(4) for details
# of the syntax of global and per-instance properties.
#
# Properties specified in this file take effect when the driver is first
# loaded, typically just after system boot.  Changes to the file will
# therefore not take effect until the next reboot, but will be permanent
# thereafter.
#
# Some of the driver's parameters can also be changed using ndd(1m).
# Changes made with ndd apply only to a specific instance (e.g. bge1).
# They take effect immediately, but are lost if the driver is unloaded.
#
###############################################################################
#
# The autonegotiation feature can be controlled by the boolean properties
# listed below.
#
# Firstly, 'adv_autoneg_cap' controls whether autonegotiation is enabled.
#
# If autonegotiation is turned OFF ("forced mode"), the remaining 'adv_*'
# speed/duplex properties force selection of a specific mode, namely,
# the first mode found to be enabled, in highest-to-lowest speed order
# (thus, if adv_1000fdx_cap=1, all other values will be ignored; to force
# 10/hdx mode, *all* the faster modes must be explicitly disabled).
#
# BEWARE - it's very easy to end up with a non-working link using forced
# mode.  There's NO validation that the link partner actually supports
# the mode that this device has been forced into.  In some cases, this
# will prevent the link coming up; in others, the link status will show
# 'up' (electrical connection made) but data transfer will not work at
# all, or will work poorly (low throughput, high collision rates, etc).
#
# Note that many switches *require* autonegotiation in order to operate
# at 1000Mbps or in full-duplex mode or with flow control.  In other words,
# the only combinations that are likely to work with autonegotiation off
# are 100Mbps/half-duplex and 10Mbps/half-duplex, unless the peer has also
# been manually forced to some other (matching) combination.
#
# With autonegotiation ON (the default and preferred mode), the 'adv_*'
# properties control which capabilities are advertised to the partner.
# The default is to advertise all the capabilities that the hardware
# supports; thus, the properties below serve only to limit the advertised
# capabilities to restricted subset -- it is not possible to advertise a
# capability that the hardware does not support.
#
# The autonegotiation process will then automagically select the fastest
# speed/duplex mode and greatest degree of flow control supported by both
# partners.
#
# If the local device is set to autonegotiate, but the link partner can't
# or doesn't autonegotiate, the correct speed will be determined anyway,
# and HALF-DUPLEX mode will be selected, as mandated by the IEEE802.3
# standard.  This will yield the correct result if the partner is in fact
# incapable of autonegotiating: it must be a half-duplex device, because
# the only devices that don't support autonegotation are half-duplex (the
# standard says that all full-duplex-capable devices must also support
# autonegotiation).
#
# However, this choice will NOT be correct if the peer is actually capable
# of autonegotiation and full-duplex operation, and has been manually set
# to "forced full-duplex without autonegotiation" (a mode not recommended
# by the IEEE standard).  The link will appear to work, but the duplex
```

```
# mismatch will result in packet loss and spurious "late collisions".  In
# such cases, the preferred solution is to enable autonegotiation by the
# peer.  Failing that, autonegotiation by the BGE device can be disabled,
# and forced mode used to match the peer's forced settings as above.
#

# Exemple :
# name="pci108e,abba" parent="/pci@21c,700000/pci@1" unit-address="0"
# adv_autoneg_cap=0 adv_1000fdx_cap=0 adv_1000hdx_cap=0
# adv_100fdx_cap=1 adv_100hdx_cap=0 adv_100T4_cap=0 adv_10fdx_cap=0
# adv_10hdx_cap=0;

# interface bge0 100 FD  SVG-ADM
name="bge" parent="/pci@1f,700000" unit-address="2"
adv_autoneg_cap=0
adv_1000fdx_cap=0
adv_1000hdx_cap=0
adv_100fdx_cap=1
adv_100hdx_cap=0
adv_100T4_cap=0
adv_10fdx_cap=0
adv_10hdx_cap=0;

# interface bge1 1 Go CLI
name="bge" parent="/pci@1f,700000" unit-address="2,1"
adv_autoneg_cap=0
adv_1000fdx_cap=1
adv_1000hdx_cap=0
adv_100fdx_cap=0
adv_100hdx_cap=0
adv_100T4_cap=0
adv_10fdx_cap=0
adv_10hdx_cap=0;

# All of these parameters can can also be queried and modified at run-time
# by use of the ndd(1m) command.
#
##############################################################################
#
# OBP's device driver exports methods to set the link speed explicity,
# which then pass the information to the Solaris driver through the
# 'transfer-speed' property.  It therefore SHOULDN'T be set here, but
# is documented for completeness.  If the 'transfer-speed' property is
# set to 10, 100, or 1000, the link will be set to the selected speed,
# and autonegotiation ENABLED but restricted to the specified speed.
# The correct duplex setting will be determined by autonegotiation.
#
# This property, if set, overrides and alters the settings of the adv_*
# parameters corresponding to the properties above.
#
# transfer-speed        = 1000;
#
##############################################################################
#
# As a third alternative, the following two properties can be set to
# force the link speed/duplex setting instead.  Doing so will override
# and alter the settings of the adv_* parameters corresponding to the
# properties above, and take precedence over all other means of setting
# the speed/duplex at boot time.
#
# Autonegotiation will be DISABLED if EITHER of these properties is set,
# therefore BOTH properties should be set explicity if either one is.
# 'speed' may be set to 10, 100 or 1000, while 'full-duplex' may be 0 or 1.
#
# See the warning above about the potential for misconfiguration when
# autonegotiation is disabled.  Defining these properties could leave your
# system configured so that the network will not work at all after reboot,
# requiring manual intervention and further reboots to recover!
#
# speed                 = 100;
# full-duplex           = 0;
```

```
#
###############################################################################
#
# The property below represents the list of subsystem vendor/device pairs
# with which driver operation is supported.  This list will be updated and
# extended as new subsystems are validated ...
#
bge-known-subsystems    = 0x108e1647,
                          0x108e1648,
                          0x108e16a7,
                          0x108e16a8,
                          0x17c20010,
                          0x17341013,
                          0x101402a6,
                          0x10f12885,
                          0x17c20020,
                          0x10b71006,
                          0x10280109,
                          0x1028865d,
                          0x0e11005a,
                          0x103c12bc;
###############################################################################
#
# The properties below represents the number of receive and send ring used.
# For BCM5705, BCM5782, etc, there are only 1 receive ring and 1 send ring.
# Otherwise, there can be up to 16 receive rings and 4 send rings.
#
bge-rx-rings            = 1;
bge-tx-rings            = 1;
```

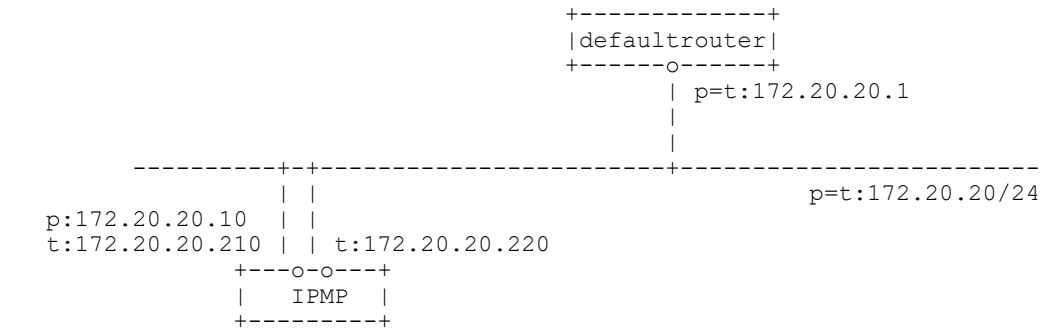
7.4 Annexe D: IPMP

```
Contents:

1.  Production and test interfaces in the same IP subnet
    1.1  With defaultrouter
    1.2  Without defaultrouter
    1.3  With dedicated hosts acting as test targets with "host-routes"
    1.4  Configuration example for 1.1, 1.2 and 1.3

2.  Production and test interfaces in different IP subnets but the same physical
network
    2.1  With defaultrouter in production subnet and test subnet
    2.2  With defaultrouter in production subnet but without defaultrouter in test
subnet
    2.3  With dedicated hosts acting as test targets with "host-routes"
    2.4  Configuration example for 2.1, 2.2 and 2.3
    2.5  Configuration example for 2.2 if you use IPMP on the test subnet

Good to know:
-------------
The operation of IP Multipathing (in.mpathd) depends on the routing
configuration. Therefore in.mpathd always refers to the routing-table
(IRE-cache) to distinguish which test partner(s) are going to be used.
Test partners are required for deciding if the interface is working properly.

in.mpathd by default chooses the defaultrouter as single test-target
for probing. If no defaultrouter exists for the test-interface ip address,
arbitrary hosts on the link are detected by sending out "all hosts"
multicast packets (224.0.0.1) on the wire to detect its test-partners.
An "all routers" multicasts (224.0.0.2) will never be sent! The first five
hosts that are responding to the echo packets are chosen as targets
for probing. In such a non-defaultrouter environment, the in.mpathd
always tries to find five probe targets via an "all hosts" multicast.

The in.mpathd detects failures and repairs by sending out 'icmp-echo'
probes (like pinging) from all interfaces that are part of the IPMP group.
If there are no replies to five consecutive probes, the interface is
considered to have failed and performs a failover of the network access
```

to another interface in the IPMP group. The probing rate depends on the failure detection time which is defined in /etc/default/mpathd. By default, failure detection time is 10 seconds. Thus the five probes will be sent within the failure detection time.

1. Production and test interfaces in the same IP subnet

1.1 With defaultrouter

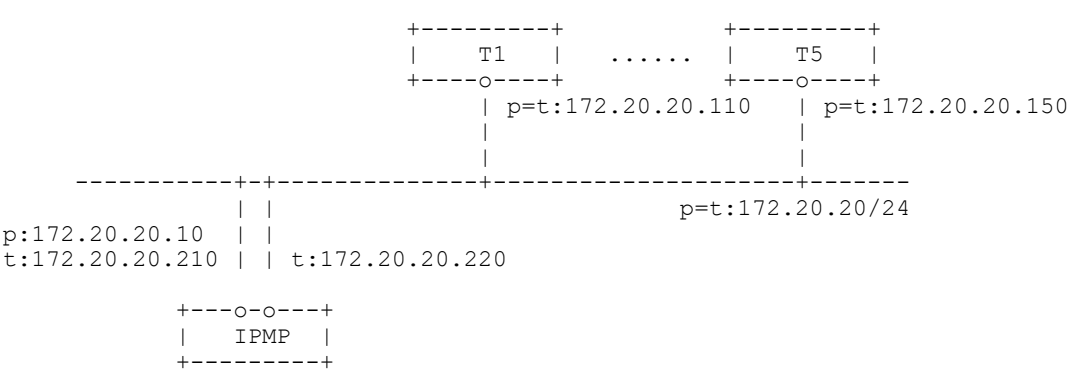


IPMP only use the defaultrouter as probe target. Each test interface of the IPMP group send ICMP requests only to the defaultrouter. To get the configuration, IPMP looks to the routing table and is independent from /etc/defaultrouter file. NO "all hosts" multicast (224.0.0.1) will be sent.

Advantages:
- Easiest configuration for IPMP.

Disadvantages:
- When the defaultrouter is down IPMP failover does not work anymore. The in.mpathd does NOT send out multicasts to get other probe targets, therefore all interfaces in the IPMP group get the state "failed". You can ignore this bug/feature when you have a defaultrouter which is 100% online! Please look to RFE 4431511 and 4489960 for further information.
- If you have a lot of IPMP groups, the defaultrouter has to reply to a lot of ICMP requests. Take care of defaultrouter. Do not overload the defaultrouter.
- The defaultrouter has to reliably answer ICMP echo requests. (e.g. firewalls sometimes do not)

1.2 Without defaultrouter



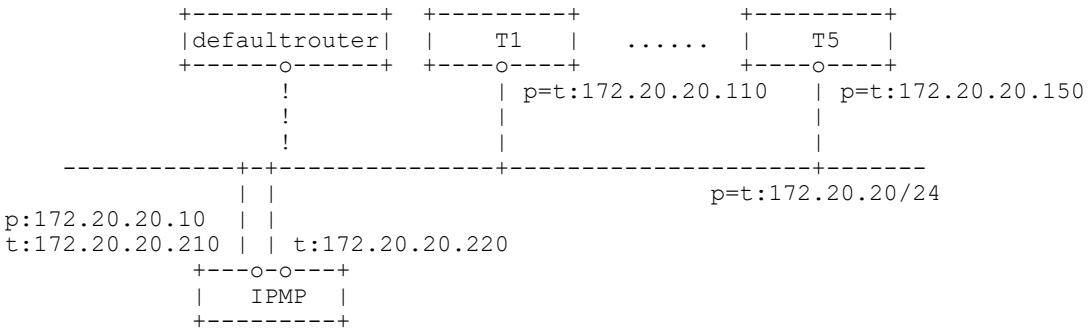
IPMP dynamically determines five arbitrary hosts on the link via "all

hosts" multicast address (224.0.0.1). At the least, you need one probe target that IPMP will work. Beware that one probe target is not reliable enough. If there are less than five targets available the in.mpathd sent out the "all hosts" multicasts to get a complete list of five probe targets.

Advantages:
- easiest configuration for IPMP in a subnet without a defaultrouter.
- is very reliable due to the five targets.

Disadvantages:
- a subnet without an defaultrouter is very rare.

1.3 With dedicated hosts acting as test targets with "host-routes"



Some "host routes" will be defined with a startscript in /etc/rc2.d/S70staticroutes. (The script is attached to this document.) When IPMP refer to the routing table it will choose the first five defined "host routes" as probe targets. This is due to the fact that normally the "host routes" are before the defaultrouter in the routing table. If you have less than five "host routes" also the defaultrouter (when available) will be used as probe target as well.

Example:
a) Configuration with host1, host2 ... hostN (with N=5 or N>5), defaultrouter :
==> The first five hosts (host1 ... host5) will be defined as target.

b) Configuration with less than 5 hosts : for instance, host1, host2, defaultrouter :
==> The three systems (host1, host2, defaultrouter) will be defined as target.

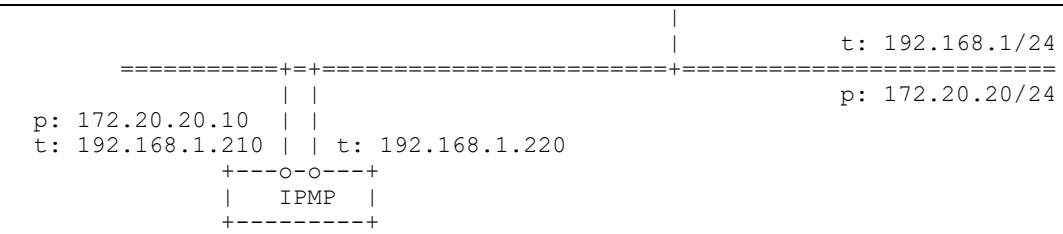
Also in this case the in.mpathd tries to get five probe targets all the time from the routing table. Remember in this configuration the in.mpathd does NOT send "all hosts" multicasts!

Advantages:
- The defaultrouter is not important for the IPMP configuration because if the defaultrouter is not available you have still some "host routes" for probing.
- IPMP is always high available due to independency to the defaultrouter

Disadvantages:
- More administrative work to do.
- Due to static configuration you should check that some of the probe targets are always available.

1.4 Configuration examples for 1.1, 1.2 and 1.3

```
-----------------------------------------------

/etc/hosts
----------
127.0.0.1        localhost
172.20.20.10     host10       loghost
172.20.20.210    host10-test-qfe0
172.20.20.220    host10-test-qfe4

/etc/hostname.qfe0
------------------
host10 netmask + broadcast + group ipmp0 up \
addif host10-test-qfe0 deprecated -failover netmask + broadcast + up

/etc/hostname.qfe4
------------------
host10-test-qfe4 deprecated -failover netmask + broadcast + group ipmp0 up

ifconfig output:
----------------
qfe0: flags=9040843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
        inet 172.20.20.10 netmask ffffff00 broadcast 172.20.20.255
        groupname ipmp0
        ether 8:0:20:e8:88:dc
qfe0:1:
flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER>
mtu 1500 index 3
        inet 172.20.20.210 netmask ffffff00 broadcast 172.20.20.255
qfe4:
flags=9040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER>
mtu 1500 index 4
        inet 172.20.20.220 netmask ffffff00 broadcast 172.20.20.255
        groupname ipmp0
        ether 8:0:20:e8:89:34


Note: The example describe the setup of an active-active IPMP configuration
      which is best practices. But you can also configure an active-standby
      configuration. You simple have to add the "standby" flag to the
      /etc/hostname.qfe4 file. More Details available in Infodoc 79624.


2.  Production and test interfaces in different IP subnets
==========================================================

If you have not enough additional ip-addresses on hand for setting up
IPMP, you can configure the ipmp-test-interfaces in a different
ip-network than your production network (e.g. 192.168., 10. ..). But you
must make sure that there are enough test-partners (also in the new
test-network) who are responding to the ipmp-test-interfaces. You may
also configure a defaultrouter in the new test-network in case you have
an existing 100.1% reliable test-partner which should act as a single
test-partner. In such a configuration in.mpathd will only use its
test-subnet IP addresses as source address for outgoing probe packets.

Note: The in.mpathd only looks to the test subnet. Therefore if you
      have no IP addresses available in the test subnet the IPMP
      group will fail although if the production subnet is available.


2.1 With defaultrouter in production subnet and test subnet
-----------------------------------------------------------

                                      +-------------+
                                      |defaultrouter|
                                      +------o------+
                                             | p: 172.20.20.1
                                             | t: 192.168.1.1
```

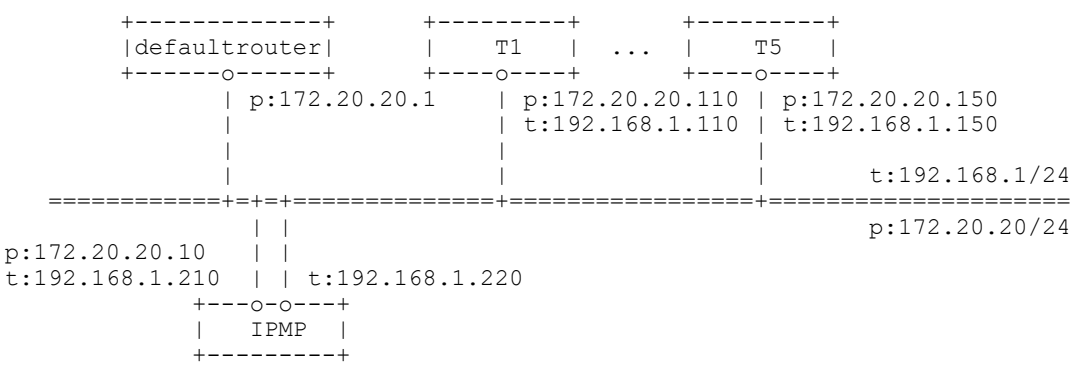


IPMP only use the defaultrouter as probe target. Each test interface of the IPMP group send ICMP requests only to the defaultrouter. To get the configuration IPMP looks to the routing table and is independent from /etc/defaultrouter file. NO "all hosts" multicast (224.0.0.1) will be sent.

Advantages:
- Test interfaces don't need IP addresses of the production subnet

Disadvantages:
- The interface of the defaultrouter has to reside in both the production AND test subnet.
- Exceptional configuration of defaultrouter.
- All which are mentioned in section 1.1

2.2 With defaultrouter in production subnet net but without defaultrouter in test subnet



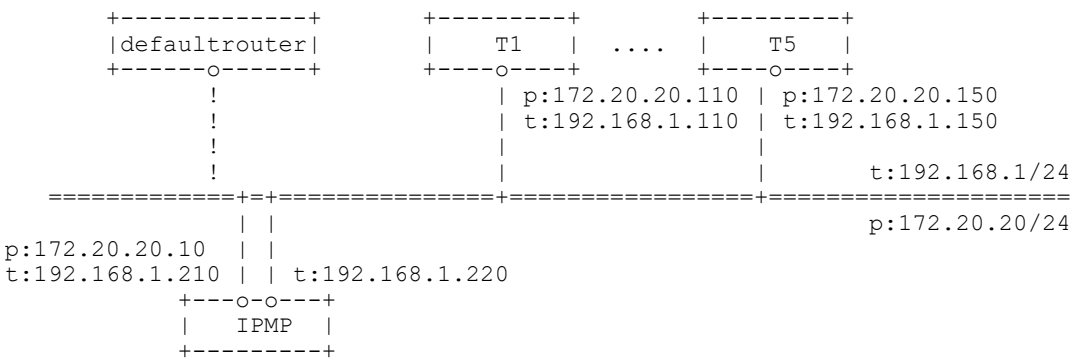
IPMP dynamically determines five arbitrary hosts in the test subnet via "all hosts" multicast address (224.0.0.1). At least you need one probe target that IPMP will work. Beware that one probe target is not reliable enough. If there are less than five targets available the in.mpathd sent out the "all hosts" multicasts to get a complete list of five probe targets.

Advantages:
- easiest configuration for IPMP if you have too less IP addresses available in the production subnet.
- is very reliable due to the five targets.

Disadvantages:
- the probe targets must be available before you can setup the IPMP host.
- more administrative work because you have to setup the probe targets as an additional interface in the test subnet.
  Recommendation: Setup an additional logical network interface on target host. (e.g. add a new interface to /etc/hostname.qfe0 with the 'addif' option)
  If your test partners are on systems which run also IPMP you must add an logical network interface NOT flagged deprecated and NOT flagged nofailover. An address associated with this interface will then be used as source address by responding properly to "all hosts" multicast packets which are used for the automatic IPMP test partner detection.

Beware that Solaris 8 does NOT require the additional logical network interface in the test subnet on the target host. So, in case of an upgrade from Solaris 8 to Solaris 9 or higher you have to change your configuration. Please refer to 2.5 for a detailed example.

2.3 with dedicated hosts acting as test targets with "host-routes"



Some "host routes" will be defined in the test subnet with a startscript in /etc/rc2.d/S70ipmp.targets (The script is attached to this document.). When IPMP refer to the routing table it will choose the first five defined "host routes" as probe targets in the test subnet. This is due to the fact that normally the "host routes" are before the defaultrouter in the routing table. If you have less than five "host routes" also the defaultrouter (when available in the test subnet) will be used as probe target as well.

Example:
- please look to the example of section 1.3

Advantages:
- test interfaces don't need IP addresses of the production subnet
- all which are mentioned in section 1.3

Disadvantages:
- all which are mentioned in section 1.3
- all which are mentioned in section 2.2

2.4 Configuration examples for 2.1, 2.2 and 2.3

/etc/hosts

```
127.0.0.1        localhost
172.20.20.10     host10       loghost
192.168.1.210    host10-test-qfe0
192.168.1.220    host10-test-qfe4
```

/etc/hostname.qfe0

```
host10 netmask + broadcast + group ipmp0 up \
addif host10-test-qfe0 deprecated -failover netmask + broadcast + up
```

/etc/hostname.qfe4

```
host10-test-qfe4 deprecated -failover netmask + broadcast + group ipmp0 up
```

ifconfig output:

```
qfe0: flags=9040843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
        inet 172.20.20.10 netmask ffffff00 broadcast 172.20.20.255
```

```
        groupname ipmp0
        ether 8:0:20:e8:88:dc
qfe0:1:
flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER>
mtu 1500 index 3
        inet 192.168.1.210 netmask ffffff00 broadcast 192.168.1.255
qfe4:
flags=9040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER>
mtu 1500 index 4
        inet 192.168.1.220 netmask ffffff00 broadcast 192.168.1.255
        groupname ipmp0
        ether 8:0:20:e8:89:34


Note: The example describe the setup of an active-active IPMP configuration
      which is best practices. But you can also configure an active-standby
      configuration. You simple have to add the "standby" flag to the
      /etc/hostname.qfe4 file. More Details available in Infodoc 79624.



2.5  Configuration example for 2.2 if you use IPMP on the test subnet
---------------------------------------------------------------------
/etc/hosts
----------
127.0.0.1       localhost
# IPMP active IP address:
172.20.20.10    host10       loghost
# IPMP test interface
192.168.1.210   host10-test-ce0
# additional active interface in the 'test subnet'
# to be able to respond to probe packets from other
# IPMP setups (See 2.2). Will also failover.
192.168.1.211   host10-prod-ce0
# IPMP test interface
192.168.1.220   host10-test-ce1

/etc/hostname.ce0
-----------------
172.20.20.10 netmask + broadcast + group ipmp0 up \
addif 192.168.1.211 netmask + broadcast + up \
addif 192.168.1.210 netmask + broadcast + deprecated -failover up

/etc/hostname.ce1
-----------------
192.168.1.220 netmask + broadcast + group ipmp0 deprecated -failover up

ifconfig output:
----------------
ce0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 172.20.20.10 netmask ffffff00 broadcast 172.20.20.255
        groupname ipmp0
        ether 8:0:20:e2:da:d5
ce0:1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.1.211 netmask ffffff00 broadcast 192.168.1.255
ce0:2: flags=9040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER>
mtu 1500 index 2
        inet 192.168.1.210 netmask ffffff00 broadcast 192.168.1.255
ce1: flags=9040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER>
mtu 1500 index 3
        inet 192.168.1.220 netmask ffffff00 broadcast 192.168.1.255
        groupname ipmp0
        ether 8:0:20:e2:da:d6



A.
----------- Begin of start script /etc/init.d/ipmp.targets --------------

#!/sbin/sh
# /etc/rc2.d/S70ipmp.targets /etc/init.d/ipmp.targets
```

```
# Copyright (c) 2005 by Sun Microsystems, Inc.
# All rights reserved.
#
#ident  "@(#)ipmp.targets      1.0.0
#
# Edit the following IPMP test  TARGETS to suit your needs.
# To install:
# 1) cp ipmp.targets /etc/init.d
# 2) perform edits on the script as required (e.g: add TARGETS)
# 3) chmod 744 /etc/init.d/ipmp.targets
# 4) chown root:sys /etc/init.d/ipmp.targets
# 5) ln /etc/init.d/ipmp.targets /etc/rc2.d/S70ipmp.targets
#
TARGETS="192.168.85.117 192.168.85.127 192.168.85.137"

case "$1" in
        'start')
            /usr/bin/echo "Adding static routes for use as IPMP targets"
                for target in $TARGETS; do
          /usr/sbin/route add -host $target $target
                done
                  ;;
        'stop')
              /usr/bin/echo "Removing static routes for use as IPMP targets"
                 for target in $TARGETS; do
                /usr/sbin/route delete -host $target $target
                 done
                  ;;
  esac
---------- End of start script /etc/init.d/ipmp.targets --------------
```

7.5 Annexe D : exemple de fichier « conf-ethernet.ksh »

```
# Maj par B. Lisan, le 23/11/2007

# A) Intialisation des vitesses de chaque patte
#    ========================================
# Note : selon les preconisation SUN pour Solaris 10,
# la partie initialisation des vitesses doit etre realise plutot au niveau
# du fichier 'ce.conf' situe dans le repertoire '/platform/sun4u/kernel/drv'
# cette partie A) n'est conservee que pour des raisons historiques & d'usage SFR

# ce0 (cli) en 1000FD

ndd -set /dev/ce instance 0
ndd -set /dev/ce adv_autoneg_cap 1
ndd -set /dev/ce adv_1000fdx_cap 1
ndd -set /dev/ce adv_1000hdx_cap 0
ndd -set /dev/ce adv_100T4_cap 0
ndd -set /dev/ce adv_100fdx_cap 0
ndd -set /dev/ce adv_100hdx_cap 0
ndd -set /dev/ce adv_10fdx_cap 0
ndd -set /dev/ce adv_10hdx_cap 0

# ce1 (adm) et ce8 (svg) en 100 FD

for i in 1 8
do
   ndd -set /dev/ce instance $i
   ndd -set /dev/ce adv_autoneg_cap 0
   ndd -set /dev/ce adv_1000fdx_cap 0
   ndd -set /dev/ce adv_1000hdx_cap 0
   ndd -set /dev/ce adv_100fdx_cap 1
   ndd -set /dev/ce adv_100hdx_cap 0
   ndd -set /dev/ce adv_10fdx_cap 0
   ndd -set /dev/ce adv_10hdx_cap 0
done
```

```
# B) intialisation des routes statiques
#    ==================================
# format : route add [ -host | -net ] adresse_IP -netmask Masque adresse_IP_Gateway
Metric
# Maj par B. Lisan, le 23/11/2007 (en fonction du PAIP HR3 pour la gateway adm
10.43.208.1)

# Administration Campus NET
route add -net 10.40.192.0 -netmask 255.255.192.0 10.42.208.1 1
# Administration ACH1 NET
route add -net 10.30.192.0 -netmask 255.255.192.0 10.42.208.1 1
# Administration ACH2 NET
route add -net 10.42.192.0 -netmask 255.255.192.0 10.42.208.1 1
# Administration ACH3 NET
route add -net 10.43.192.0 -netmask 255.255.192.0 10.42.208.1 1
# Etrust        HOST : TOME
route add -host 164.17.41.47 -netmask 255.255.255.255 10.42.208.1 1

# Acces CLI APP ACH2 NET
#route add -net 10.42.64.0 -netmask 255.255.224.0 10.42.90.1 1
"conf-ethernet.ksh" 55 lines, 1987 characters
[su0321@root:/etc/init.d] cat conf-ethernet.ksh
# Maj par B. Lisan, le 23/11/2007

# A) Intialisation des vitesses de chaque patte
#    ========================================
# Note : selon les preconisation SUN pour Solaris 10,
# la partie initialisation des vitesses doit etre realise plutot au niveau
# du fichier 'ce.conf' situe dans le repertoire '/platform/sun4u/kernel/drv'
# cette partie A) n'est conservee que pour des raisons historiques & d'usage SFR

# ce0 (cli) en 1000FD

ndd -set /dev/ce instance 0
ndd -set /dev/ce adv_autoneg_cap 1
ndd -set /dev/ce adv_1000fdx_cap 1
ndd -set /dev/ce adv_1000hdx_cap 0
ndd -set /dev/ce adv_100T4_cap 0
ndd -set /dev/ce adv_100fdx_cap 0
ndd -set /dev/ce adv_100hdx_cap 0
ndd -set /dev/ce adv_10fdx_cap 0
ndd -set /dev/ce adv_10hdx_cap 0

# ce1 (adm) et ce8 (svg) en 100 FD

for i in 1 8
do
   ndd -set /dev/ce instance $i
   ndd -set /dev/ce adv_autoneg_cap 0
   ndd -set /dev/ce adv_1000fdx_cap 0
   ndd -set /dev/ce adv_1000hdx_cap 0
   ndd -set /dev/ce adv_100fdx_cap 1
   ndd -set /dev/ce adv_100hdx_cap 0
   ndd -set /dev/ce adv_10fdx_cap 0
   ndd -set /dev/ce adv_10hdx_cap 0
done

# B) intialisation des routes statiques
#    ==================================
# format : route add [ -host | -net ] adresse_IP -netmask Masque adresse_IP_Gateway
Metric
# Maj par B. Lisan, le 23/11/2007 (en fonction du PAIP HR3 pour la gateway adm
10.43.208.1)

# Administration Campus NET
route add -net 10.40.192.0 -netmask 255.255.192.0 10.42.208.1 1
# Administration ACH1 NET
route add -net 10.30.192.0 -netmask 255.255.192.0 10.42.208.1 1
# Administration ACH2 NET
route add -net 10.42.192.0 -netmask 255.255.192.0 10.42.208.1 1
# Administration ACH3 NET
```

```
route add -net 10.43.192.0 -netmask 255.255.192.0 10.42.208.1 1
# Etrust        HOST : TOME
route add -host 164.17.41.47 -netmask 255.255.255.255 10.42.208.1 1

# Acces CLI APP ACH2 NET
#route add -net 10.42.64.0 -netmask 255.255.224.0 10.42.90.1 1
# Acces CLI APP ACH3 NET
#route add -net 10.43.64.0 -netmask 255.255.224.0 10.42.90.1 1
```

7.6 Annexe E : syntaxe de quelques commandes réseau

```
ifconfig bge0 down
ifconfig bge0 unplumb
ifconfig bge0 plumb
ifconfig bge0 10.84.196.47 netmask 255.255.248.0 broadcast +
ifconfig bge0 up

ifconfig -a plumb : on lance toutes les pattes.

ifconfig –m bge0 : il est transformé en « mega-device », en half, full-duplex
( ?).
ndd /dev/bge0 \? : Interrogation des options du ndd pour ce type de carte …
(ici une bge).

route add -host 10.43.145.13 10.43.198.1
route delete -host 10.43.140.250 10.43.198.1

[su0570@root:/var/tmp]
route add -net 10.43.160.0 -netmask 255.255.255.224 10.43.140.250
route add -net 10.43.192.0/18 10.43.198.1
route add default 10.43.27.1
```

7.7 Annexe F : Autres definitions

Routeur filtrant n. m. : Élément de coupe-feu constitué d'un routeur dont la fonction est de permettre ou de refuser aux utilisateurs l'accès à un réseau privé, en fonction de critères prédéterminés contenus dans l'en-tête des paquets IP qu'ils expédient.
Note(s) : Le routeur filtrant examine les adresses IP de l'expéditeur et du destinataire, les numéros de port utilisés et le type de paquet expédié (TCP, UDP). Il intervient au niveau de la couche réseau du modèle OSI.
Bien qu'il puisse être utilisé seul comme coupe-feu de première ligne, le routeur filtrant est généralement utilisé en association avec d'autres dispositifs de sécurité pour former un coupe-feu plus robuste.
Synonyme(s) :, routeur-filtre n. m., routeur de filtrage n. m.
Anglais : screening router
Synonyme(s) : filtering router, packet filter, (access control list, ACL).

7.8 CIDR - Classless Inter-Domain Routing Guide

- RFC 1123

- /n -> n=number of bits in mask

Prefix	Netmask	Binary	Number of Networks
/1	128.0.0.0	10000000000000000000000000000000	128 Class A domains
/2	192.0.0.0	11000000000000000000000000000000	64 Class A domains
/3	224.0.0.0	11100000000000000000000000000000	32 Class A domains
/4	240.0.0.0	11110000000000000000000000000000	16 Class A domains
/5	248.0.0.0	11111000000000000000000000000000	8 Class A domains
/6	252.0.0.0	11111100000000000000000000000000	4 Class A domains
/7	254.0.0.0	11111110000000000000000000000000	2 Class A domains
/8	255.0.0.0	11111111000000000000000000000000	1 Class A domain
/9	255.128.0.0	11111111100000000000000000000000	128 Class B domains
/10	255.192.0.0	11111111110000000000000000000000	64 Class B domains
/11	255.224.0.0	11111111111000000000000000000000	32 Class B domains
/12	255.240.0.0	11111111111100000000000000000000	16 Class B domains
/13	255.248.0.0	11111111111110000000000000000000	8 Class B domains
/14	255.252.0.0	11111111111111000000000000000000	4 Class B domains
/15	255.254.0.0	11111111111111100000000000000000	2 Class B domains
/16	255.255.0.0	11111111111111110000000000000000	1 Class B domain
/17	255.255.128.0	11111111111111111000000000000000	128 Class C domains
/18	255.255.192.0	11111111111111111100000000000000	64 Class C domains
/19	255.255.224.0	11111111111111111110000000000000	32 Class C domains
/20	255.255.240.0	11111111111111111111000000000000	16 Class C domains
/21	255.255.248.0	11111111111111111111100000000000	8 Class C domains
/22	255.255.252.0	11111111111111111111110000000000	4 Class C domains
/23	255.255.254.0	11111111111111111111111000000000	2 Class C domains
/24	255.255.255.0	11111111111111111111111100000000	1 Class C domain
/25	255.255.255.128	11111111111111111111111110000000	128 hosts
/26	255.255.255.192	11111111111111111111111111000000	64 hosts
/27	255.255.255.224	11111111111111111111111111100000	32 hosts
/28	255.255.255.240	11111111111111111111111111110000	16 hosts
/29	255.255.255.248	11111111111111111111111111111000	8 hosts
/30	255.255.255.252	11111111111111111111111111111100	4 hosts
/31	255.255.255.254	11111111111111111111111111111110	2 hosts
/32	255.255.255.255	11111111111111111111111111111111	1 host

Sources :

http://en.wikipedia.org/wiki/Classless_Inter-Domain_Routing

7.9 Mécanisme de masquage de l'adresse IP par le masque

Voici le résultat de faire un AND sur tous les bits entre l'adresse ip et le netmasks:

```
10000010 01100100 01010100 01100100 IP address
11111111 11111111 11100000 00000000 Subnet Mask
-------- -------- -------- --------
10000010 01100100 01001011 10011011 Network starting IP address
```

Faisons la conversion en décimal, du résultat, afin qu'on puisse le lire plus facilement (inversion du processus de conversion du décimal en binaire):

```
10000010 01100100 01001011 10011011 = 130 100 75 155
```

Source : http://www.tech-unity.com/forums/printthread.php?t=40

7.10 Conversions des Netmasks

Conversion des netmasks en format bitmask, décimal, hexadécimal et binaire :

Si vous n'avez jamais eu besoin de savoir ce qu'est un masque de réseau et à quoi il ressemble exprimé dans un autre format, cette table des équivalents devrait vous aider. Il contient les IPv4 netmasks communs exprimés dans quatre formats différents.

Bitmask (Bits)	Dotted Decimal	Hexadecimal	Binary
/0	0.0.0.0	0x00000000	00000000 00000000 00000000 00000000
/1	128.0.0.0	0x80000000	10000000 00000000 00000000 00000000
/2	192.0.0.0	0xc0000000	11000000 00000000 00000000 00000000
/3	224.0.0.0	0xe0000000	11100000 00000000 00000000 00000000
/4	240.0.0.0	0xf0000000	11110000 00000000 00000000 00000000
/5	248.0.0.0	0xf8000000	11111000 00000000 00000000 00000000
/6	252.0.0.0	0xfc000000	11111100 00000000 00000000 00000000
/7	254.0.0.0	0xfe000000	11111110 00000000 00000000 00000000
/8	255.0.0.0	0xff000000	11111111 00000000 00000000 00000000
/9	255.128.0.0	0xff800000	11111111 10000000 00000000 00000000
/10	255.192.0.0	0xffc00000	11111111 11000000 00000000 00000000
/11	255.224.0.0	0xffe00000	11111111 11100000 00000000 00000000
/12	255.240.0.0	0xfff00000	11111111 11110000 00000000 00000000
/13	255.248.0.0	0xfff80000	11111111 11111000 00000000 00000000
/14	255.252.0.0	0xfffc0000	11111111 11111100 00000000 00000000
/15	255.254.0.0	0xfffe0000	11111111 11111110 00000000 00000000
/16	255.255.0.0	0xffff0000	11111111 11111111 00000000 00000000
/17	255.255.128.0	0xffff8000	11111111 11111111 10000000 00000000
/18	255.255.192.0	0xffffc000	11111111 11111111 11000000 00000000
/19	255.255.224.0	0xffffe000	11111111 11111111 11100000 00000000
/20	255.255.240.0	0xfffff000	11111111 11111111 11110000 00000000
/21	255.255.248.0	0xfffff800	11111111 11111111 11111000 00000000
/22	255.255.252.0	0xfffffc00	11111111 11111111 11111100 00000000
/23	255.255.254.0	0xfffffe00	11111111 11111111 11111110 00000000
/24	255.255.255.0	0xffffff00	11111111 11111111 11111111 00000000
/25	255.255.255.128	0xffffff80	11111111 11111111 11111111 10000000
/26	255.255.255.192	0xffffffc0	11111111 11111111 11111111 11000000
/27	255.255.255.224	0xffffffe0	11111111 11111111 11111111 11100000
/28	255.255.255.240	0xfffffff0	11111111 11111111 11111111 11110000
/29	255.255.255.248	0xfffffff8	11111111 11111111 11111111 11111000
/30	255.255.255.252	0xfffffffc	11111111 11111111 11111111 11111100
/31	255.255.255.254	0xfffffffe	11111111 11111111 11111111 11111110
/32	255.255.255.255	0xffffffff	11111111 11111111 11111111 11111111

Source : http://www.pawprint.net/designresources/netmask-converter.php

7.11 Réseau Masques / Netmasks pour les nuls

La notation masques de réseau est un groupe d'adresses IP. Prenons une analogie familière. Mon téléphone aux États-Unis était 510-595-3830. 510 est le code régional qui comprenait un grand nombre de numéros de téléphone à San Francisco, Oakland, Emeryville domaine. De même, les adresses IP sont également regroupés par leur réseau préfixe.
Les Netmasks sont importants pour la répartition en sous- réseaux, pour les individus ou organisations. Ils sont également utilisés pour identifier les réseaux d'acheminement.
Chaque (IPv4) adresse IP de 4 à 8 bits entiers dans la forme; abcd So an IPv4 IP address contains 32 bits. Ainsi, une IPv4 adresse IP contient 32 morceaux. Un masque de réseau indique le nombre de bits qui sont fixés pour toutes les adresses IP au sein du réseau.
Prenons un réseau qui contient les adresses IP entre 72.31.32.0 - 72.31.32.255. Dans ce réseau, les 24 premiers bits (des 3 premières 8 bits entiers) sont constants pour toutes les adresses IP. Ainsi, le masque de réseau pour ce réseau seront désignés comme **/ 24.**
Il ya une notation longue des masques réseau qui est utilisé dans Windows et Linux également de préciser le masque de sous-réseau de votre propre ordinateur. Ici, au lieu de l'écriture constante du nombre de bits dans l'adresse IP au format décimal, vous devez le spécifier dans le format d'adresse IP. Ainsi / 24 devient 255.255.255.0. Les trois premiers 255 représente les 24 bits de l'adresse.
Chaque réseau spécifié par le masque de réseau peut contenir un nombre maximum d'adresses IP des ordinateurs ou des dispositifs porteurs (comme des routeurs), limité par le masque de réseau. Par exemple, un / 24 réseau peut contenir jusqu'à 2^{8-1} (255) ordinateurs. Une des adresses (la plus élevée) est réservée à l'adresse de diffusion du réseau (l'adresse de broadcast). Le numéro 8 est obtenue en soustrayant 24 à 32.
Consultez le tableau ci-dessous pour les masques de réseau à la fois dans les formes et le nombre maximum de machines pour la référence rapide:

Forme courte	Forme complète	Nombre maximum de machines	Commentaires
/8	/255.0.0.0	16 777 215	Classe A adresse
/16	/255.255.0.0	65 535	Classe B adresse
/17	/255.255.128.0	32 767	
/18	/255.255.192.0	16 383	
/19	/255.255.224.0	8 191	
/20	/255.255.240.0	4 095	
/21	/255.255.248.0	2 047	
/22	/255.255.252.0	1 023	
/23	/255.255.254.0	511	
/24	/255.255.255.0	255	Classe C adresse
/25	/255.255.255.128	127	
/26	/255.255.255.192	63	
/27	/255.255.255.224	31	
/28	/255.255.255.240	15	
/29	/255.255.255.248	7	
/30	/255.255.255.252	3	

Source : http://blog.taragana.com/index.php/archive/network-masks-netmasks-for-dummies/fr/

8. BIBLIOGRAPHIE

[1] IPv6: théorie et pratique, de Gisèle Cizault, O'Reilly, 2005.
[2] http://fr.wikipedia.org/wiki/Sous-r%C3%A9seau
[3] http://en.wikipedia.org/wiki/Subnetwork

9. ANNEXE : DEFINITIONS

1) Routage :

Le **routage** désigne le mécanisme par lequel les données d'un équipement expéditeur sont acheminées jusqu'à leur destinataire en examinant les informations situées au niveau 3 du modèle OSI (IP par exemple), même si aucun des deux ne connaît le chemin complet que les données devront suivre.

2) Nœud de réseau :

On distingue deux types de nœuds du réseau :

- les hôtes qui ne peuvent être qu'expéditeur ou destinataire d'un message.
- les routeurs qui transmettent des messages entre au moins deux réseaux ou sous-réseaux d'une même autorité. Lorsqu'un tel élément de routage est utilisé entre deux réseaux dépendant d'autorités différentes (par exemple entre un réseau local et internet), on utilise alors un élément plus évolué : une passerelle.

2) Table de routage :

La table de routage est composée de la liste des réseaux connus, chacun de ces réseaux étant associé à un ou plusieurs routeurs voisins qui va acheminer les paquets vers cette destination.

Si la table routage est entrée manuellement par l'administrateur, on parle de **routage statique** (viable pour de petits réseaux).

Si le routeur construit lui-même la table de routage en fonctions des informations qu'il reçoit (par l'intermédiaire de protocoles de routage), on parle de **routage dynamique.**

Avec un **routage statique**, une station ne peut atteindre que ce qu'on lui indique avec les commandes routes. Ceci est réciproque. En utilisant de manière intelligente le routage statique sur une station, on peut ainsi protéger cette station.

Dans le **routage dynamique**, régulièrement, les routeurs diffusent des informations indiquant quels réseaux on peut atteindre par eux. Les machines et les routeurs enregistrent ces informations et mettent à jour leur table de routage.
En conséquence, il y a plusieurs langages pour échanger ces informations et différents types d'informations sont échangées, selon plusieurs protocoles de routage (Routage dynamique ou AS, Routing Information Protocol ou RIP ...).

La table de routage est une table de correspondance entre l'adresse de la machine visée et le noeud suivant auquel le routeur doit délivrer le message. En réalité il suffit que le message soit délivré sur le réseau qui contient la machine, il n'est donc pas

nécessaire de stocker l'adresse IP complète de la machine: seul l'identificateur du réseau de l'adresse IP (c'est-à-dire l'ID réseau) a besoin d'être stocké.
La table de routage est donc un tableau contenant des paires d'adresses :

Ainsi grâce à cette table, le routeur, connaissant l'adresse du destinataire encapsulée dans le message, va être capable de savoir sur quelle interface envoyer le message (cela revient à savoir quelle carte réseau utiliser), et à quel routeur, directement accessible sur le réseau auquel cette carte est connectée, remettre le datagramme.

Ce mécanisme consistant à ne connaître que l'adresse du prochain maillon menant à la destination est appelé *routage par sauts successifs* (en anglais *next-hop routing*).

Cependant, il se peut que le destinataire appartienne à un réseau non référencé dans la table de routage. Dans ce cas, le routeur utilise un **routeur par défaut** (appelé aussi *passerelle par défaut*).

Voici, de façon simplifiée, ce à quoi pourrait ressembler, une table de routage :

Le message est ainsi remis de routeur en routeur par sauts successifs, jusqu'à ce que le destinataire appartienne à un réseau directement connecté à un routeur. Celui-ci remet alors directement le message à la machine visée...
Dans le cas du routage statique, c'est l'administrateur qui met à jour la table de routage.

Dans le cas du routage dynamique, par contre, un protocole appelé **protocole de routage** permet la mise à jour automatique de la table afin qu'elle contienne à tout moment la route optimale.

3) Routes :

Il existe trois types de routes dans la table de routage:

- les routes correspondant à des réseaux directement connectés : pour ces réseaux, le routeur peut acheminer le paquet directement à la destination finale en faisant appel au protocole de niveau 2 (Ethernet par exemple).
- les routes statiques: configurée *en dur* sur le routeur par l'administrateur du réseau,

- les routes dynamiques, apprises d'un protocole de routage dynamique dont le rôle est de diffuser les informations concernant les réseaux disponibles.

4) Routeurs :

Les routeurs sont les dispositifs permettant de "choisir" le chemin que les datagrammes vont emprunter pour arriver à destination.
Il s'agit de machines ayant plusieurs cartes réseau dont chacune est reliée à un réseau différent. Ainsi, dans la configuration la plus simple, le routeur n'a qu'à "regarder" sur quel réseau se trouve un ordinateur pour lui faire parvenir les datagrammes en provenance de l'expéditeur.

Sources :
http://www.commentcamarche.net/internet/routage.php3
www.urec.cnrs.fr/IMG/pdf/93.04.cours.TCP_IP.2.pdf

10. EXEMPLE DE CONFIGURATION IPMP

```
su0528svg
[su0528@root:/etc] cat hostname.bge1
su0528-cli1 netmask + broadcast + group cli deprecated -failover up
addif su0528cli netmask + broadcast + failover up
[su0528@root:/etc] cat hostname.bge2
su0528-prv1 netmask + broadcast + group prv deprecated -failover up
addif su0528priv netmask + broadcast + failover up
[su0528@root:/etc] cat hostname.ce0
su0528-cli2 netmask + broadcast + group cli deprecated -failover standby up
[su0528@root:/etc] cat hostname.ce1
su0528-prv2 netmask + broadcast + group prv deprecated -failover standby up
[su0528@root:/etc] cat defaultrouter
10.169.32.1
[su0528@root:/etc] cat hosts
#Begin regle
# Règle de nommage des hosts :
#    ab0000xxxy.tld :
#       a : [s]erveur, adresse [v]irtuelle, machine v[i]rtuelle, [c]hassis, [p]oste de
travail
#       b : [w]icrosoft, [u]nix, [a]ppliance, [o]S/390
#       0000 : N° Incrément
#       xxx : Interface [cli]ent, [adm]inistration, [svg] sauvegarde, [csl] console
#       y : N° d'incrément gérant les adresses IP multiples
#       tld : Nom de domaine (.prod .pack, .rece, .tras, .deve, .form)
#End regle
#
# Internet host table
#
127.0.0.1       localhost
10.169.48.184   su0528svg                                       # bge0 (1000 FDX)
10.169.48.8     catalpasvg

# IPMP
10.169.32.253   su0528-cli1                                     # adresse physique CLI
(bge1) (1000 FDX)
10.169.58.13    su0528-prv1                                     # adresse physique PRV
Interconnect (bge2) (1000 FDX)
10.169.33.35    su0528-cli2                                     # adresse physique
double-attachement CLI (ce0) (1000 FDX)
10.169.58.14    su0528-prv2                                     # adresse physique
double-attachement PRV Interconnect (ce1) (1000 FDX)
10.169.33.36    su0528  su0528cli       loghost                 # adresse virtuelle
CLI (bge1:1)
10.169.58.15    su0528priv                                      # adresse virtuelle
PRV Interconnect (bge2:1)
[su0528@root:/etc] cat netmasks
#
# The netmasks file associates Internet Protocol (IP) address
# masks with IP network numbers.
#
#       network-number  netmask
#
# The term network-number refers to a number obtained from the Internet Network
# Information Center.  Currently this number is restricted to being a class
# A, B, or C network number.  In the future we should be able to support
# arbitrary network numbers per the Classless Internet Domain Routing
# guidelines.
#
# Both the network-number and the netmasks are specified in
# "decimal dot" notation, e.g:
#
#               128.32.0.0 255.255.255.0
#
10.169.32.0     255.255.254.0   # bge1 et ce0 (cli)
10.169.48.0     255.255.254.0   # bge0 (svg)
10.169.58.0     255.255.255.224 # bge2 et ce1 (prive interconnect)
```

```
[su0528@root:/etc] ifconfig -a
lo0: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
bge0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 10.169.48.184 netmask fffffe00 broadcast 10.169.49.255
        ether 0:3:ba:9f:36:65
bge1: flags=9040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER> mtu
1500 index 3
        inet 10.169.32.253 netmask fffffe00 broadcast 10.169.33.255
        groupname cli
        ether 0:3:ba:9f:36:66
bge1:1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
        inet 10.169.33.36 netmask fffffe00 broadcast 10.169.33.255
bge2: flags=9040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER> mtu
1500 index 4
        inet 10.169.58.13 netmask ffffffe0 broadcast 10.169.58.31
        groupname prv
        ether 0:3:ba:9f:36:67
bge2:1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 4
        inet 10.169.58.15 netmask ffffffe0 broadcast 10.169.58.31
ce0:
flags=69040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER,STANDBY,INACT
IVE> mtu 1500 index 5
        inet 10.169.33.35 netmask fffffe00 broadcast 10.169.33.255
        groupname cli
        ether 0:3:ba:db:63:4b
ce1:
flags=69040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER,STANDBY,INACT
IVE> mtu 1500 index 6
        inet 10.169.58.14 netmask ffffffe0 broadcast 10.169.58.31
        groupname prv
        ether 0:3:ba:db:63:4c
[su0528@root:/etc] cd /platform/sun4u/kernel/drv/
[su0528@root:/platform/sun4u/kernel/drv]
[su0528@root:/platform/sun4u/kernel/drv] cat bge.conf
#
# Copyright 2005 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
#ident  "@(#)bge.conf   1.8     05/09/01 SMI"
#
# Driver.conf file for the BGE driver, for
# Broadcom 579x Gigabit Ethernet devices
#
# All the properties below can be set globally (i.e. for all instances
# of BGE), or on a per-instance basis.  See driver.conf(4) for details
# of the syntax of global and per-instance properties.
#
# Properties specified in this file take effect when the driver is first
# loaded, typically just after system boot.  Changes to the file will
# therefore not take effect until the next reboot, but will be permanent
# thereafter.
#
# Some of the driver's parameters can also be changed using ndd(1m).
# Changes made with ndd apply only to a specific instance (e.g. bge1).
# They take effect immediately, but are lost if the driver is unloaded.
#
#
# The autonegotiation feature can be controlled by the boolean properties
# listed below.
#
# Firstly, 'adv_autoneg_cap' controls whether autonegotiation is enabled.
#
# If autonegotiation is turned OFF ("forced mode"), the remaining 'adv_*'
# speed/duplex properties force selection of a specific mode, namely,
# the first mode found to be enabled, in highest-to-lowest speed order
# (thus, if adv_1000fdx_cap=1, all other values will be ignored; to force
# 10/hdx mode, *all* the faster modes must be explicitly disabled).
#
```

```
# BEWARE - it's very easy to end up with a non-working link using forced
# mode.  There's NO validation that the link partner actually supports
# the mode that this device has been forced into.  In some cases, this
# will prevent the link coming up; in others, the link status will show
# 'up' (electrical connection made) but data transfer will not work at
# all, or will work poorly (low throughput, high collision rates, etc).
#
# Note that many switches *require* autonegotiation in order to operate
# at 1000Mbps or in full-duplex mode or with flow control.  In other words,
# the only combinations that are likely to work with autonegotiation off
# are 100Mbps/half-duplex and 10Mbps/half-duplex, unless the peer has also
# been manually forced to some other (matching) combination.
#
# With autonegotiation ON (the default and preferred mode), the 'adv_*'
# properties control which capabilities are advertised to the partner.
# The default is to advertise all the capabilities that the hardware
# supports; thus, the properties below serve only to limit the advertised
# capabilities to restricted subset -- it is not possible to advertise a
# capability that the hardware does not support.
#
# The autonegotiation process will then automagically select the fastest
# speed/duplex mode and greatest degree of flow control supported by both
# partners.
#
# If the local device is set to autonegotiate, but the link partner can't
# or doesn't autonegotiate, the correct speed will be determined anyway,
# and HALF-DUPLEX mode will be selected, as mandated by the IEEE802.3
# standard.  This will yield the correct result if the partner is in fact
# incapable of autonegotiating: it must be a half-duplex device, because
# the only devices that don't support autonegotation are half-duplex (the
# standard says that all full-duplex-capable devices must also support
# autonegotiation).
#
# However, this choice will NOT be correct if the peer is actually capable
# of autonegotiation and full-duplex operation, and has been manually set
# to "forced full-duplex without autonegotiation" (a mode not recommended
# by the IEEE standard).  The link will appear to work, but the duplex
# mismatch will result in packet loss and spurious "late collisions".  In
# such cases, the preferred solution is to enable autonegotiation by the
# peer.  Failing that, autonegotiation by the BGE device can be disabled,
# and forced mode used to match the peer's forced settings as above.
#
# adv_autoneg_cap       = 1;
# adv_1000fdx_cap       = 1;
# adv_1000hdx_cap       = 1;
# adv_100T4_cap         = 0;
# adv_100fdx_cap        = 1;
# adv_100hdx_cap        = 1;
# adv_10fdx_cap         = 1;
# adv_10hdx_cap         = 1;
# adv_asym_pause_cap    = 0;
# adv_pause_cap         = 1;
#
# All of these parameters can can also be queried and modified at run-time
# by use of the ndd(1m) command.
#
# bge0 (svg)
name="bge" parent="/pci@1f,700000" unit-address="2" adv_autoneg_cap=1
adv_1000fdx_cap=1 adv_1000hdx_cap=0 adv_100fdx_cap=0 adv_100hdx_cap=0 adv_10fdx_cap=0
adv_10hdx_cap=0;
# bge1 (cli1)
name="bge" parent="/pci@1f,700000" unit-address="2,1" adv_autoneg_cap=1
adv_1000fdx_cap=1 adv_1000hdx_cap=0 adv_100fdx_cap=0 adv_100hdx_cap=0 adv_10fdx_cap=0
adv_10hdx_cap=0;
# bge2 (prv1)
name="bge" parent="/pci@1d,700000" unit-address="2" adv_autoneg_cap=1
adv_1000fdx_cap=1 adv_1000hdx_cap=0 adv_100fdx_cap=0 adv_100hdx_cap=0 adv_10fdx_cap=0
adv_10hdx_cap=0;
##################################################################
#
# OBP's device driver exports methods to set the link speed explicity,
```

```
# which then pass the information to the Solaris driver through the
# 'transfer-speed' property.  It therefore SHOULDN'T be set here, but
# is documented for completeness.  If the 'transfer-speed' property is
# set to 10, 100, or 1000, the link will be set to the selected speed,
# and autonegotiation ENABLED but restricted to the specified speed.
# The correct duplex setting will be determined by autonegotiation.
#
# This property, if set, overrides and alters the settings of the adv_*
# parameters corresponding to the properties above.
#
# transfer-speed        = 1000;
#
###############################################################################
#
# As a third alternative, the following two properties can be set to
# force the link speed/duplex setting instead.  Doing so will override
# and alter the settings of the adv_* parameters corresponding to the
# properties above, and take precedence over all other means of setting
# the speed/duplex at boot time.
#
# Autonegotiation will be DISABLED if EITHER of these properties is set,
# therefore BOTH properties should be set explicity if either one is.
# 'speed' may be set to 10, 100 or 1000, while 'full-duplex' may be 0 or 1.
#
# See the warning above about the potential for misconfiguration when
# autonegotiation is disabled.  Defining these properties could leave your
# system configured so that the network will not work at all after reboot,
# requiring manual intervention and further reboots to recover!
#
# speed                 = 100;
# full-duplex           = 0;
#
###############################################################################
#
# The property below represents the list of subsystem vendor/device pairs
# with which driver operation is supported.  This list will be updated and
# extended as new subsystems are validated ...
#
bge-known-subsystems    = 0x108e1647,
                          0x108e1648,
                          0x108e16a7,
                          0x108e16a8,
                          0x17c20010,
                          0x17341013,
                          0x101402a6,
                          0x10f12885,
                          0x17c20020,
                          0x10b71006,
                          0x10280109,
                          0x1028865d,
                          0x0e11005a,
                          0x103c12bc;
###############################################################################
#
# The properties below represents the number of receive and send ring used.
# For BCM5705, BCM5782, etc, there are only 1 receive ring and 1 send ring.
# Otherwise, there can be up to 16 receive rings and 4 send rings.
#
bge-rx-rings            = 1;
bge-tx-rings            = 1;

[su0528@root:/platform/sun4u/kernel/drv] cat ce.conf
# ce0 (cli2)
name="ce" parent="/pci@1d,700000/pci@1" unit-address="0" adv_autoneg_cap=1
adv_1000fdx_cap=1 adv_1000hdx_cap=0 adv_100fdx_cap=0 adv_100hdx_cap=0 adv_10fdx_cap=0
adv_10hdx_cap=0;
# ce1 (prv2)
name="ce" parent="/pci@1d,700000/pci@1" unit-address="1" adv_autoneg_cap=1
adv_1000fdx_cap=1 adv_1000hdx_cap=0 adv_100fdx_cap=0 adv_100hdx_cap=0 adv_10fdx_cap=0
adv_10hdx_cap=0;
[su0528@root:/platform/sun4u/kernel/drv] cd /etc/init.d
```

```
[su0528@root:/etc/init.d] cat conf-ethernet.ksh
cat: cannot open conf-ethernet.ksh
[su0528@root:/etc/init.d] cat ../rc2.d/S25conf-ethernet.ksh
cat: cannot open ../rc2.d/S25conf-ethernet.ksh
[su0528@root:/etc/init.d] netstat -rnv

IRE Table: IPv4
  Destination             Mask           Gateway          Device Mxfrg  Rtt  Ref Flg
Out  In/Fwd
-------------------- --------------- -------------------- ------ ----- ----- --- --- -
---- ------
10.169.58.0          255.255.255.224 10.169.58.15         bge2:1  1500*    0   1 U
157     0
10.169.58.0          255.255.255.224 10.169.58.15         ce1     1500*    0   1 U
53      0
10.169.58.0          255.255.255.224 10.169.58.15         bge2    1500*    0   1 U
0     0
10.169.32.0          255.255.254.0   10.169.33.36         bge1:1  1500*    0   1 U
56      0
10.169.32.0          255.255.254.0   10.169.33.36         ce0     1500*    0   1 U
26      0
10.169.32.0          255.255.254.0   10.169.33.36         bge1    1500*    0   1 U
0     0
10.169.48.0          255.255.254.0   10.169.48.184        bge0    1500*    0   1 U
87183     0
224.0.0.0            240.0.0.0       10.169.33.36         bge1:1  1500*    0   1 U
0     0
default              0.0.0.0         10.169.32.1                  1500*    0   1 UG
138     0
127.0.0.1            255.255.255.255 127.0.0.1            lo0     8232*    0   7 UH
339     0
[su0528@root:/etc/init.d]
```

11. RÉCAPITULATIF DE COMMANDES RESEAU

Exemples :

```
ifconfig bge0 down
ifconfig bge0 plumb
ifconfig bge0 10.84.196.47 netmask 255.255.248.0 broadcast +
ifconfig bge0 up

ifconfig bge0 unplumb
```

ifconfig –m bge0 : il est transformé en « mega-device », en half, full-duplex (?).
ndd /dev/bge0 \? : interrogation des options du ndd pour ce type de carte … (ici une bge).

```
route add -host 10.43.145.13 10.43.198.1
route delete -host 10.43.140.250 10.43.198.1
route add -net 10.43.160.0 -netmask 255.255.255.224 10.43.140.250
route add -net 10.43.192.0/18 10.43.198.1
route add default 10.43.27.1
```

Exemples de commandes pour le test d'une config IPMP (test de bascule d'une patte à l'autre …)

```
inetconv -i
inetconv
fg
inetconv -i /etc/inet/inetd.conf
if_mpadm -d bge1
if_mpadm -r bge1
tail /var/adm/messages
dladm show-dev
ifconfig -a | grep UP
```

tail /var/adm/messages (ou "less /var/adm/messages » ou « grep mpath /var/adm/messages »).

```
ntpq -p
```