Quelques informations sur la configuration réseau sous Solaris

Version 1.0

Auteur: Richard Luce

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.

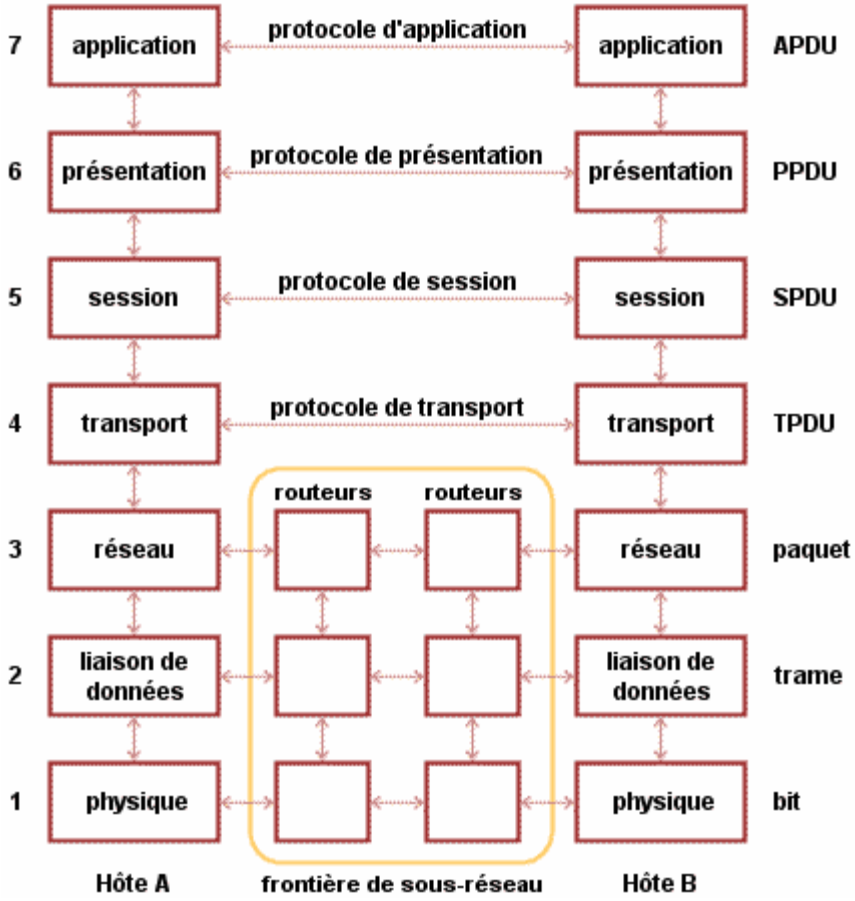
On verra par exemple :

- 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. LA COMMANDE NDD:

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.

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 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.

3.1 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

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

Positionner l'instance

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

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)
```

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.

- On positionne les vitesses dans le fichier « /etc/rc2.d/S25conf-ethernet.ksh »

Ce fichier est un lien sur le fichier « /etc/init.d/conf-ethernet.ksh »

Attention: Pour les interfaces « ce », « bge » et « dmfe » il est recommandé 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 »). Il existe une troisième méthode qui consiste à positionner les vitesses dans « /etc/system ». Cette dernière n'est pas recommandée. Configurer ces fichiers, obliger de rebooter le serveur.

3.2 Le fichier /platform/sun4u/kernel/drv/ce.conf

ou /platform/sun4u/kernel/drv/dmfe.conf
Ou /platform/sun4u/kernel/drv/bge.conf

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

Exemple:

```
#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 positionnés pour tous les ports utilisant ce driver.

Exemple1: Comment faire si l'on souhaite par exemple positionner le ce0 en 100 full duplex et le ce1 en 1000 full duplex avec autoneg?
Il faut introduire des paramètres dans le fichier de configuration.

```
    # grep '"ce"' /etc/path_to_inst
"/pci@9,700000/network@2" 0 "ce"
"/pci@9,600000/network@1" 1 "ce"

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;
```

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:

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.

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

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
```

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.

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.

Note : 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.

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 (redondance de cartes ou switch) nous basculerons sur l'autre interface sans perturbation de service.

- S'assurer que le paramètre local-mac-address? est positionné à true.

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

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.

 # 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.

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.

/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 autre 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

Le /18 spécifie le masque de sous réseau. C'est le nombre de 1 qui sert de filtre (vu plus haut)

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

/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.

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
;
```

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;
```

7.4 Annexe D: IPMP

Contents:

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.

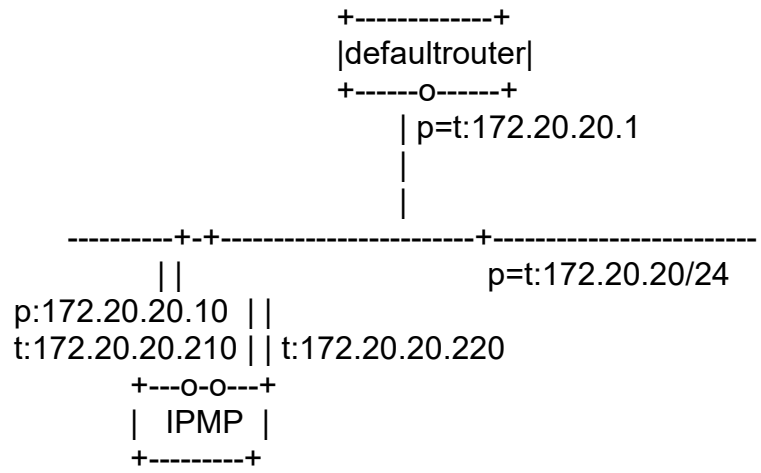
in.mpathd by default chooses the defaultrouter as single test-target for probing. If no defaultrouter exists for the test-interface ip address, arbitrary hosts on the link are detected by sending out "all hosts" multicast packets (224.0.0.1) on the wire to detect its test-partners. An "all routers" multicasts (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

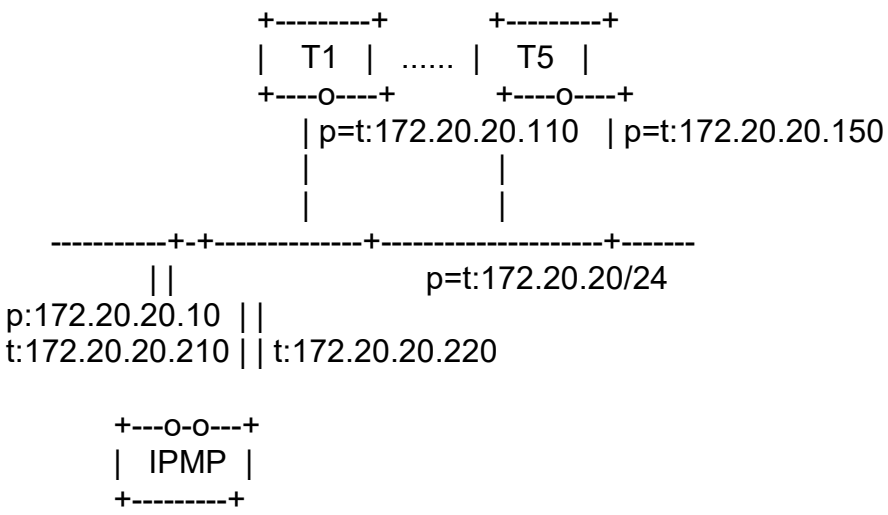


IPMP only use the defaultrouter as probe target. Each test interface of the IPMP group send ICMP requests only to the defaultrouter. To get the configuration, IPMP looks to the routing table and is independent from /etc/defaultrouter file. NO "all hosts" multicast (224.0.0.1) will be sent.

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

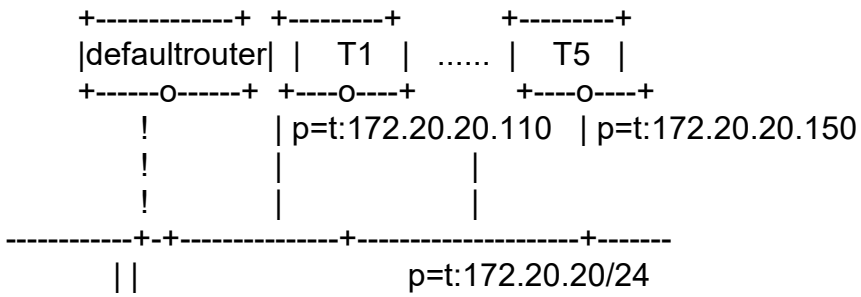


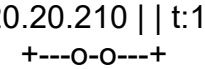
IPMP dynamically determines five arbitrary hosts on the link via "all hosts" multicast address (224.0.0.1). At the least, you need one probe target that IPMP will work. Beware that one probe target is not reliable enough. If there are less than five targets available the in.mpathd sent out the "all hosts" 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,NOFAI
LOVER>
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,NOFAI
LOVER>
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.

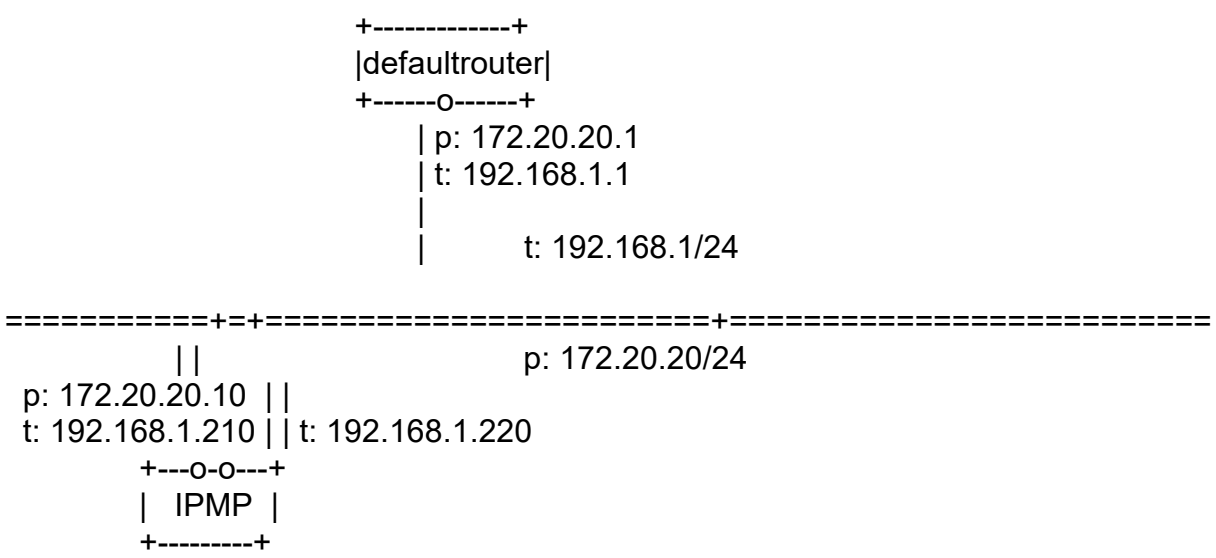
2. Production and test interfaces in different IP subnets

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



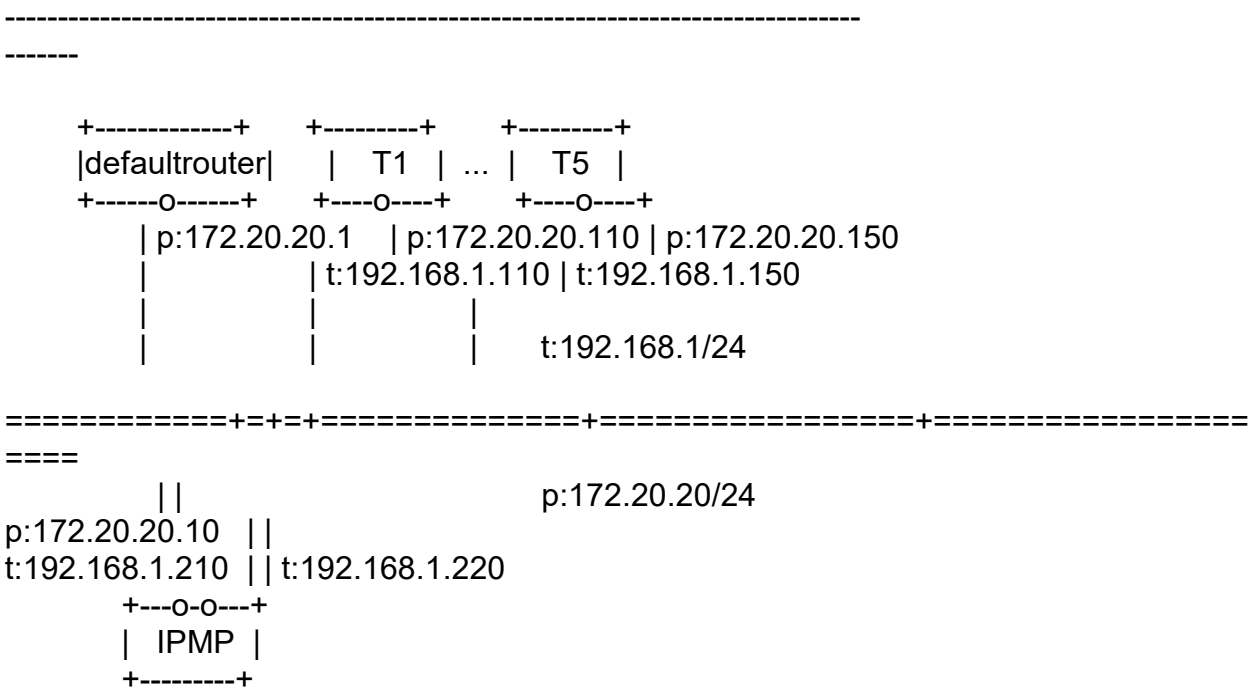
IPMP only use the defaultrouter as probe target. Each test interface of the IPMP group send ICMP requests only to the defaultrouter. To get the configuration IPMP looks to the routing table and is independent from /etc/defaultrouter file. NO "all hosts" multicast (224.0.0.1) will be sent.

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

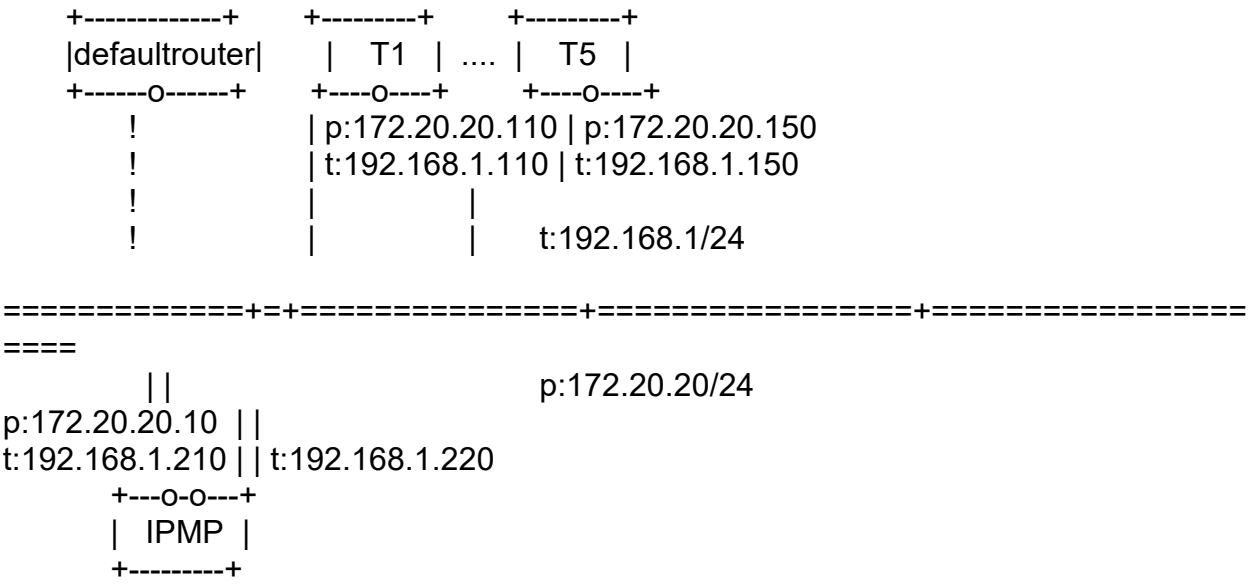


IPMP dynamically determines five arbitrary hosts in the test subnet via "all hosts" multicast address (224.0.0.1). At least you need one probe target that IPMP will work. Beware that one probe target is not reliable enough. If there are less than five targets available the in.mpathd sent out the "all 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,NOFAI
LOVER>
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,NOFAI
LOVER>
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 ----------
```