h e p i a
Haute école du paysage, d'ingénierie
et d'architecture de Genève

Rue de la Prairie 4
CH-1202 Genève

Tél. +41 (0)22 546 24 00
Fax +41 (0)22 546 24 10

hepia@hesge.ch
www.hesge.ch/hepia

# VIRTUALISATION & STOCKAGE

## Travail de Bachelor
## Session 2013

**Professeur responsable** : LITZISTORF Gérald

**Diplômant** : DE OLIVEIRA Tiago

Filière ITI

Laboratoire de transmission de données

Genève, le 10 juillet 2013

Hes·SO GENÈVE
Haute Ecole Spécialisée
de Suisse occidentale

h e p i a
Haute école du paysage, d'ingénierie
et d'architecture de Genève

**Automne 2013**
**Session de bachelor**

# INGÉNIERIE DES TECHNOLOGIES DE L'INFORMATION

## ORIENTATION – COMMUNICATIONS, MULTIMÉDIA ET RÉSEAUX

## VIRTUALISATION & STOCKAGE

**Descriptif :**

Un réseau de stockage est généralement présent dans une infrastructure virtualisée. Il doit offrir souplesse (SAN / NAS), performance et sécurité (disponibilité).

L'objectif de la première partie de ce travail, proposé par SmartBee, consiste à définir les bonnes pratiques en matière de redondance du réseau de stockage (multipath, synchronisation, ...).

La deuxième partie doit analyser les avantages et inconvénients d'un réseau (commutateurs Ethernet) virtualisé.

**Travail demandé :**

Ce travail comprend les parties suivantes :

1) Proposer un réseau de stockage redondant composé de 2 équipements QNAP
   Les hyperviseurs sont basés sur Fedora 18
   Etudier les mécanismes de synchronisation entre QNAP
   Mettre en œuvre les mécanismes multipath
   Tous les équipements sont compatibles Ethernet 1 Gbit/s
   Les commutateurs Ethernet sont physiques
   Etablir divers scénarios réalistes
   Analyser les principaux risques de sécurité
   Mesurer les performances
   Si le temps le permet, remplacer les équipements QNAP par xxx

2) Etudier http://openvswitch.org/
   Quelles sont les fonctionnalités intéressantes pour une entreprise utilisant des hyperviseurs KVM ?
   Mettre en œuvre des scénarios pertinents
   Analyser les principaux risques de sécurité et l'impact sur les performances
   Si le temps le permet, automatiser la procédure d'installation

Sous réserve de modifications en cours du travail de Bachelor

Candidat :
**M. DE OLIVEIRA TIAGO**
Filière d'études : ITI
Orientation TIC

Professeur(s) responsable(s) :
Litzistorf Gérald

En collaboration avec : SmartBee
Travail de bachelor soumis à une convention de stage en entreprise : non
Travail de bachelor soumis à un contrat de confidentialité : non

Timbre de la direction

h e p i a

Haute école du paysage, d'ingénierie
et d'architecture de Genève

**Automne 2013**
**Session de bachelor**

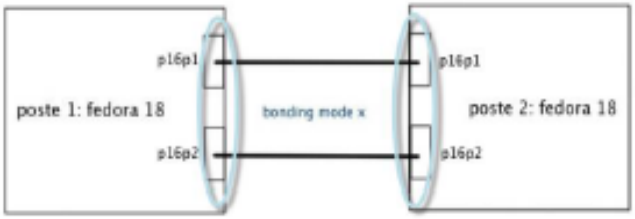
## Résumé : Virtualisation & Stockage

Un réseau de stockage SAN est généralement présent dans une infrastructure virtualisée. Il doit offrir souplesse (SAN /NAS), performances et sécurité (disponibilité).

Ce travail Bachelor proposé par la société SmartBee s'est déroulé durant 11 semaines. Il a consisté dans l'étude de solutions pouvant permettre d'améliorer les performances au niveau des liens Ethernet ainsi qu'en matière de redondance d'un réseau de stockage.

Mon mémoire se compose de deux chapitres :

### Chapitre I : Bonding sur Fedora 18

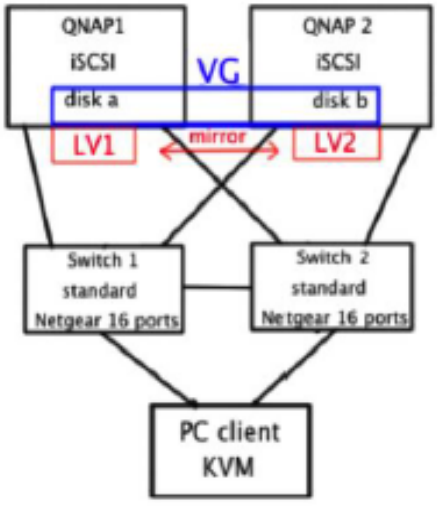



Le bonding est une méthode efficace permettant de mettre en commun plusieurs cartes Ethernet sous une même interface virtuelle possédant une adresse IP et une adresse MAC.
La configuration sous Linux s'effectue à l'aide d'un driver bonding se trouvant à l'origine dans le Kernel depuis la version 2.0.
Cependant, ce driver est très peu utilisé malgré des performances excellentes au niveau débit binaire.

Dans ce premier chapitre, j'ai pu analyser le fonctionnement en détail des modes 0 et 6 pouvant être configurés sans un switch évolué. Le mode 0 a montré qu'il était possible de mettre en place une redondance au niveau des liens Ethernet avec d'excellentes performances en débit binaire.

### Chapitre II : Virtualisation & Stockage

Dans ce deuxième chapitre, j'ai pu mettre en place un système redondant basé sur le bonding mode 0 avec iSCSI comme gestion des disques sur les 2 SANs du fabricant QNAP.




La redondance en cas de panne d'un des systèmes de stockage est assurée par du mirroring LVM sur Linux.

Pour terminer, j'ai aussi pu étudier le protocole NFS v.4 qui malheureusement ne permet pas la gestion en cas de Failover, ainsi que deux protocoles de réplication RSNC et RTRR.

Diplômant :
**M. De Oliveira Tiago**

Filière d'études : ITI
Ingénierie des Technologies de l'Information

Timbre de la direction

## Remerciements

Je tiens tout d'abord à remercier ma famille, en particulier mon épouse et ma fille pour leur soutient durant mes études et en particulier durant les 11 semaines de ce travail de diplôme.

Je remercie le corps enseignant de la filière de l'Ingénierie des Technologies de l'Information ayant encadré ma formation.

Monsieur Gérald Litzistorf pour m'avoir proposé ce projet de diplôme et m'avoir suivi tout au long de ce travail.

Monsieur Khaled Basbous qui a su m'apporter une aide précieuse en sa qualité d'assistant du Laboratoire de transmission de données.

Enfin, mes camarades de classe pour leur soutien.

## Table des matières

## Table des illustrations

### Chapitre I : Bonding sur Fedora 18



### Chapitre II : Virtualisation et Stockage

# Synthèse de ce travail de mémoire

## ❖ Chapitre I : Bonding sur Fedora 18

### *Introduction sur le bonding*

Le bonding est une méthode efficace permettant de mettre en commun plusieurs cartes Ethernet sous une même interface virtuelle possédant une adresse IP et une adresse MAC.

La configuration sous Linux s'effectue à l'aide d'un driver bonding se trouvant à l'origine dans le Kernel depuis la version 2.0.

Il est possible d'affiner cette configuration à l'aide de 2 méthodes de monitoring internes propres au bonding (miimon et arp_interval)[1]. Elles permettent toutes les deux la détection en cas de panne d'une des interfaces Ethernet physiques.

Le driver a besoin pour pouvoir fonctionner correctement (détecter les interfaces actives) qu'une seule de ces deux méthodes soit configurée.

Il existe 7 modes[2] de fonctionnement configurables pour le bonding. Chacun ayant des particularités distinctes au niveau de la gestion des entrées et des sorties pendant un échange de données entre périphériques d'un réseau.

Pour mon étude, j'ai choisi d'analyser plus en détail les modes 0 et 6. D'une part, car ils me paraissent les plus intéressants au niveau du fonctionnement et des performances. D'autre part, car ils peuvent être configurable en parallèle avec un switch standard comme le Netgear GS116E 16 ports 1Gbit/s mis à disposition dans le laboratoire.

Ainsi, j'ai procédé à des tests portant sur divers scénarios qui me semblent pertinents incluant les 2 modes. Le but étant de comprendre les mécanismes du bonding pour l'inclure dans l'objectif demandé dans le chapitre II de ce mémoire[3], à savoir la mise en place d'un système de stockage redondant hautement disponible.

### *Mode 0 Balance-round robin : Fonctionnement et résultats obtenus*

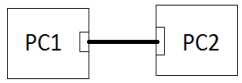
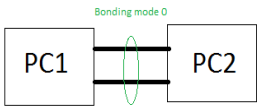
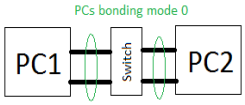
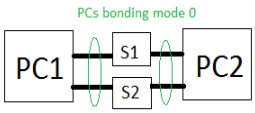
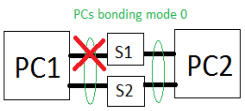
J'ai pu observer et décrire le fonctionnement[4] de ce mode en détail. L'interface virtuelle (bond0) est la seule à posséder une adresse IP. Toutes les interfaces possèdent la même adresse MAC, par défaut celle de la première carte Ethernet intégrée au bonding. Chose importante, cette adresse MAC ne changera jamais pour ce mode et ceci même en cas de coupure du

[1] Chapitre 1 Ajustement fin de la configuration bonding
[2] Annexe A.1 Caractéristiques des 7 différents modes de Bonding
[3] Chapitre II Virtualisation et Stockage : Schéma d'installation de départ
[4] Chapitre I Fonctionnement du mode 0 (balance – round robin)

lien concerné comme j'ai pu le tester avec des scénarios de coupure de lien. L'envoi de données s'effectue en séquentiel sur toutes les interfaces ainsi que la réception selon les tests effectués.

On peut voir ce mécanisme ainsi que celui du protocole ARP à l'aide du schéma de fonctionnement du mode 0 effectué à partir d'une capture des différentes interfaces Ethernet.

J'ai effectué des tests de fonctionnalité[5] pour ce mode dont voici les principaux résultats au niveau des performances.

| | | Débit utile théorique maximal [Mbit/s] | Débit utile mesuré [Mbit/s] | Utilisation débit utile [%] | Temps de transfert [s] | Débit utile théorique maximal [Mo/s] | Débit utile mesuré [Mo/s] | Utilisation débit utile [%] | Utilisation CPU PC1 [%] | Utilisation mémoire RAM PC1 [%] |
|---|---|---|---|---|---|---|---|---|---|---|
| Schéma | Test ¦ Méthode | Calcul | Iperf | Calcul | Iperf | Calcul | Moniteur Système F18 | Calcul | Moniteur Système F18 | Moniteur Système F18 |
|  | sans aucun mode de bonding | 971 | 950 | 97,84 | 8,5 | 121,36 | 120 | 98,88 | 22 | 22,4 |
|  | scénario 1 (2 liens Gbit/s fonctionnels) | 1942 | 1870 | 96,29 | 4,3 | 242,72 | 240 | 98,88 | 16 | 22,6 |
|  | Scénario 2 | Exige un switch évolué compatible EtherChannel pour fonctionner. | | | | | | | | |
|  | scénario 3 (2 liens Gbit/s fonctionnels) | 1942 | 1870 | 96,29 | 4,3 | 242,72 | 240 | 98,88 | 20 | 22,4 |
|  | scénario 3 (coupure de 1 lien) | 971 | 950 | 97,84 | 8,5 | 121,36 | 120 | 98,88 | 80 | 22,4 |

Les différents scénarios supportent la panne d'un lien en basculant automatiquement sur celui qui est encore fonctionnel.

### *Mode 6 Adaptive Load Balancing : Fonctionnement et résultats obtenus*

Le mode 6 permet l'équilibrage adaptatif de la charge en émission et en réception sur les interfaces Ethernet actives. J'ai pu décrire en détail le fonctionnement de ce mode à l'aide des mêmes scénarios tests utilisés pour le mode 0, ainsi que schématiser un échange de données. Lors de l'émission, c'est l'interface étant la moins chargée qui va être sélectionnée par le driver bonding pour effectuer l'envoi.

[5] Bonding mode 0 Scénario de test 1 : Deux PCs Fedora 18 avec bonding
Scénario test 2 : 2 PCs Fedora 18 et un Switch
Scénario test 3 : 2 PCs Fedora 18 et 2 Switchs
Tests de coupure de lien pour le mode 0

En réception c'est le même procédé d'équilibrage de charge. Le driver bonding intercepte les requêtes ARP dans un premier temps, puis il effectue une redistribution des flux en direction de l'interface la moins chargée en modifiant son adresse MAC.

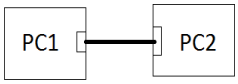
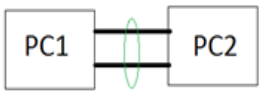
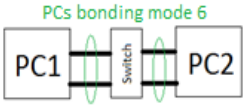
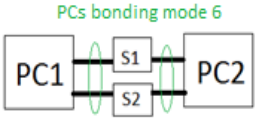
Ces constantes modifications d'adresses MAC font que ce mode ne peut fonctionner qu'avec la mise en commun obligatoire d'un switch.

L'interface virtuelle bond0 est la seule à avoir une adresse IP. Cette fois chaque interface physique possède sa propre adresse MAC et l'interface virtuelle bond 0 s'approprie celle de l'interface active par défaut.

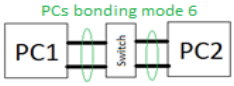
On remarque l'utilisation du protocole LLC [6] (Logical Link Control) sur chacune de ces interfaces. Chaque seconde, 3 trames LLC de 60 bytes sont envoyées par chaque interface avec comme source et destination sa propre MAC adresse.

La mise à jour du cache ARP s'effectué par des échanges de requêtes ARP entre le PC souhaitant transmettre et le PC qui se trouve en réception.

J'ai effectué des tests de fonctionnalité [7] pour ce mode dont voici les principaux résultats au niveau des performances.

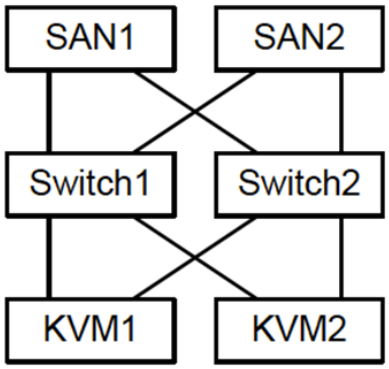
| Schéma | Test ¦ Méthode | Débit utile théorique maximal [Mbit/s] Calcul | Débit utile mesuré [Mbit/s] Iperf | Utilisation débit utile [%] Calcul | Temps de transfert [s] Iperf | Débit utile théorique maximal [Mo/s] Calcul | Débit utile mesuré [Mo/s] Moniteur Système F18 | Utilisation débit utile [%] Calcul | Utilisation CPU PC1 [%] Moniteur Système F18 | Utilisation mémoire RAM PC1 [%] Moniteur Système F18 |
|---|---|---|---|---|---|---|---|---|---|---|
|   | sans aucun mode de bonding | 971 | 950 | 97,84 | 8,5 | 121,36 | 120 | 98,88 | 22 | 22,4 |
|   | scénario 1 | Ce mode ne fonctionne pas avec ce scénario. Il faut obligatoirement que les liens soient connectés à un switch. | | | | | | | | |
|   | scénario 2 | 1942 | 943 | 48,56 | 8,5 | 121,36 | 120 | 98,88 | 30 | 22 |
|   | scénario 3 | Cela ne fonctionne pas car les deux liens bonding ne sont pas connectés au même switch. | | | | | | | | |

[6] Logical Link Control : http://www.javvin.com/protocolLLC.html
[7] Bonding mode 6 Scénario de test 1 : Deux PCs Fedora 18 avec bonding mode 6
Scénario test 2 : 2 PCs Fedora 18 et un Switch
Scénario test 3 : 2 PCs Fedora 18 et 2 Switchs

| Schéma | Test ¦ Méthode | Débit utile théorique maximal [Mbit/s] Calcul | Débit utile mesuré par connexion [Mbit/s] Iperf | Utilisation débit utile par connexion [%] Calcul | Débit utile total mesuré [Mbit/s] Iperf | Temps de transfert [s] Iperf | Débit utile théorique maximal [Mo/s] Calcul | Débit utile mesuré [Mo/s] Moniteur Système F18 | Utilisation débit utile [%] Calcul | Utilisation CPU PC1 [%] Moniteur Système F18 | Utilisation mémoire RAM PC1 [%] Moniteur Système F18 |
|---|---|---|---|---|---|---|---|---|---|---|---|
|   | scénario 2 (3 transferts simultannés de 1Go de PC1-->PC2) | 971 | 330 | 33,99 | 937 | 25,5 | 121,36 | 120 | 98,88 | 30 | 22 |

Les différents scénarios supportent la panne d'un lien en basculant automatiquement sur celui qui est encore fonctionnel.

## ❖ Chapitre II : Virtualisation et Stockage




Tout d'abord, j'ai effectué une étude théorique sur les architectures de stockage NAS et SAN dans le but d'avoir une vision globale sur le sujet[8].

Ensuite, l'étude de deux méthodes de stockage (iSCSI et NFS) [9] et de réplication (RTRR et RSYNC) [10] m'ont permis d'effectuer un choix [11] pour débuter la mise en œuvre d'un système redondant basé sur ce modèle.

J'ai débuté par une réalisation avec 2 SANs QNAP en bonding mode 0, basé sur NFS v.4 et avec une réplication RTRR[12].

Cette solution avec NFS v.4 est vite apparue comme non fonctionnelle car ce système de fichiers ne permet pas un basculement entre QNAPs, en cas de coupure ou de panne.

Mon choix s'est alors porté sur une solution fonctionnelle [13] avec iSCSI au lieu de NFS, puis la mise en place d'un Volume Groupe Linux composé des disques des 2 QNAPs. Enfin, deux Volumes Logiques en mirroring permettant d'effectuer une réplication des données ainsi qu'un basculement en cas de panne d'un des équipements.

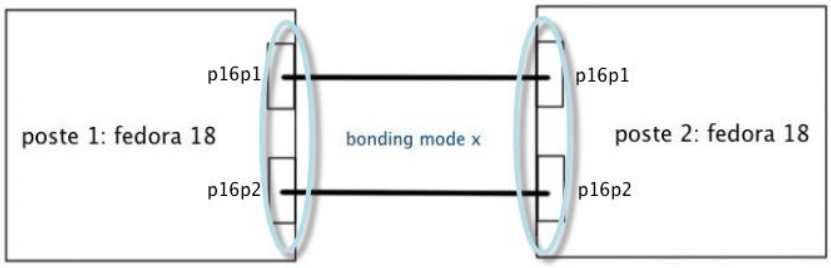
La solution finale [14] que je propose permet d'obtenir des performances en débit binaire de 1,87 Gbit/s grâce au bonding mode 0 sur les QNAPs et sur le PC client avec un hyperviseur KVM. Les switchs utilisés sont standards 1 Gbit/s et ont seulement besoin d'implémenter le protocole Spanning Tree.

Malgré des coupures de liens les machines virtuelles peuvent continuer de fonctionner sans pertes de données.

---

[8] Chapitre II Analyse Architectures NAS et
[9] Chapitre II Comparaison iSCSI et NFS
[10] Chapitre II Réplication RTRR et RSYNC
[11] Chapitre II Choix de la configuration
[12] Chapitre II Scénario 1 : 1 QNAP en NFS, 2 switchs et 1 PC client NFS
[13] Chapitre II Scénario 2 : 1 QNAPs en iSCSI, 2 switchs et 1 PC KVM
[14] Chapitre II Scénario 3 : 2 QNAPs en iSCSI Mirror, 2 switchs et 1 PC KVM

# Chapitre I
# Bonding sur Fedora 18



# Introduction

Le but dans ce chapitre est dans un premier temps d'étudier le document « Linux Ethernet Bonding Driver » [15] et plus particulièrement les 7 différents modes de Bonding qu'il est possible de configurer.

Par la suite, je mets en place une méthodologie de tests permettant d'observer, étudier et comprendre le fonctionnement des modes 0 [16] et 6 [17] sur système Linux Fedora 18 Gnome.

# Analyse

## La solution du Chanel Bonding

Il existe trois raisons principales qui poussent à mettre en place une agrégation de lien avec Bonding entre divers types de composants (commutateurs, serveurs, systèmes de stockage) :

- Disponibilité (fiabilité et redondance)
- Performance (gain au niveau débit binaire)
- Utilisation de cartes réseau non exploitées

Cependant, la redondance des cartes réseaux est souvent perçue comme inutile, voir couteuse. Pourtant, comme nous allons le voir, elle présente deux grands avantages. L'installation est simple et cela permet très rapidement de répondre aux besoins mentionnés à moindre coût.

## Qu'est-ce que le Chanel Bonding ?

Durant mes recherches, j'ai pu remarquer que de nombreux termes génériques existaient pour expliquer le même procédé. Ainsi, Chanel Bonding, Bonding, agrégation de liens (Link Aggregation), Link Bundling, Port Trunking ou encore NIC Teaming décrivent en réalité la même méthode.

Ces termes dépendent, soit du système d'exploitation (Linux, Windows, Mac), dans le cas de la mise en place sur un PC, soit du fabricant (Cisco, Netgear,...), dans le cas des commutateurs prenant en charge ce type de procédé.

Dans la figure suivante, on peut voir un exemple détaillant une configuration possible à l'aide de ce moyen entre différents types d'équipements.

[15] Linux Ethernet Bonding Driver : https://www.kernel.org/doc/Documentation/networking/bonding.txt
[16] Partie étude du mode 0 : Fonctionnement du mode 0 (balance – round robin)
[17] Partie étude du mode 6 : Fonctionnement du mode 6 (balance-alb)

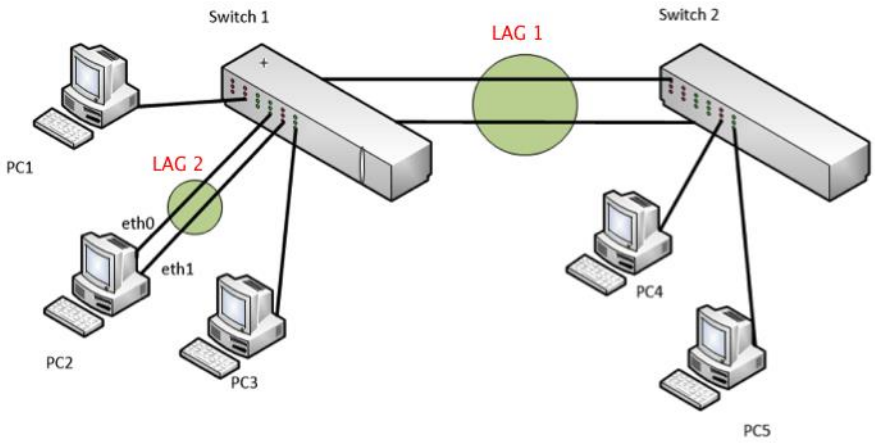



Figure 1: Exemple d'agrégation de liens

Dans cette illustration, on remarque 2 agrégations de liens (LAG) : entre deux switchs (LAG 1) et entre le PC2 avec un des switch (LAG 2).

Dans le LAG 1, on parlera communément de Port Trunking entre switchs lors de l'association de multiples ports pour créer un LAG. Il est assez commun sur ce type de liaison de trouver deux switchs évolués pouvant gérer le protocole IEEE 802.3ad.

Dans le LAG 2, on parlera de configuration Chanel Bonding ou Bonding sur le PC 2 sur Linux. Ce type de liaison est souvent entre un PC Linux et un switch standard, bien que l'un des modes possibles de bonding (mode 4) puisse gérer le protocole IEEE 802.3ad.

## Fonctionnement général du Chanel Bonding

Cette configuration utilisée sur un système Linux permet de mettre en commun plusieurs cartes Ethernet sous une même interface et de les faire agir comme une seule entité.

Ce procédé permet donc d'améliorer le débit binaire au-delà des limites d'un seul câble RJ45 et d'une seule interface grâce à la mise en commun de plusieurs liaisons simultanées. L'autre fonctionnalité est la tolérance aux pannes (network fault tolerance) qui est améliorée grâce à la redondance des liens.

On commence par crée une interface réseau virtuelle (MASTER) appelée bond0 à partir de n autres interfaces physiques (SLAVES) existantes sur le PC. L'interface bond0 assure la gestion au niveau adressage MAC et du trafic passant via les n interfaces physiques qui lui sont associées.

L'indication de l'adresse MAC qui doit être prise en compte lors d'un échange est transmise à l'aide de requêtes ARP.

Celles-ci permettent d'effectuer une mise à jour de la table ARP des différents équipements souhaitant communiquer.

ARP (Address Resolution Protocol) prend en charge la résolution d'adresse entre la couche liaison et la couche réseau. Ce protocole est important dans le bonding car il a pour rôle d'attribuer l'adresse MAC correspondant à une adresse IP à tout moment.

La figure suivante montre à l'aide d'un modèle en couches OSI l'implémentation du Bonding sous Linux.

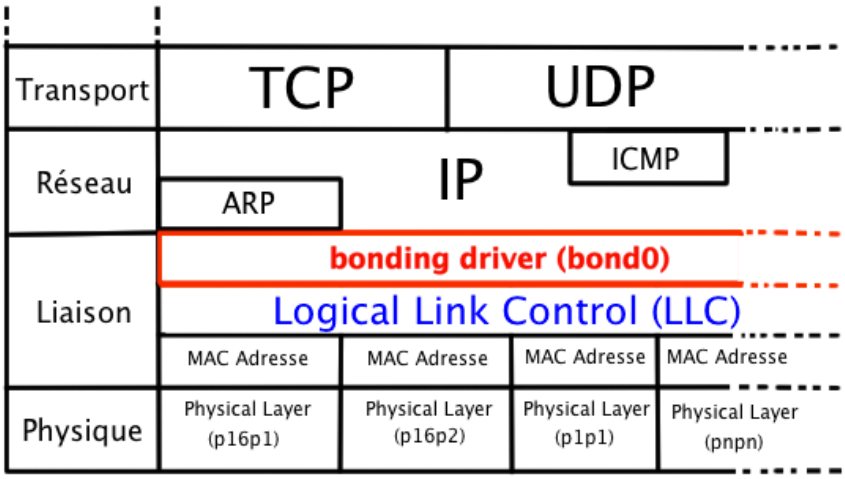



Figure 2: Modèle en couches Bonding

L'interface virtuelle bond0 se situe au niveau de la couche liaison. Elle est la seule à posséder une adresse IP et elle hérite par défaut de la MAC adresse de la première interface (p16p1). Le protocole LLC (Logical Link Control) intervient dans le mode 6 et il est expliqué en détail dans la partie «Fonctionnement du mode 6 (balance-alb)» de ce mémoire.

### Les différents modes

Il existe 7 modes de configuration possibles pour le Chanel Bonding[18], expliqués de façon plus ou moins précise dans les différents documents que j'ai pu étudier durant mes recherches.

Chaque mode permet de mettre en place une gestion des interfaces physiques par rapport à différents critères lors de la réception ou l'émission de données. Dans le tableau se trouvant en annexe[19], je propose une comparaison des différents modes par rapport à certaines particularités qui reviennent le plus souvent dans la documentation.

[18] Annexe : A.1 Caractéristiques des 7 différents modes de Bonding
[19] Annexe : A.2 Tableau comparatif des différents modes de bonding

### Les modes choisis pour mon étude

Suite à cette analyse théorique concernant le fonctionnement général du bonding, je décide d'étudier plus en détail les modes de bonding 0 et 6.

Principalement car ces deux modes peuvent être mis en place avec un switch standard implémentant au minimum le protocole Spanning Tree [20]. Cela représente un avantage important au niveau de la rapidité d'installation (pas besoin de connaissances spécifiques pour configurer un switch manageable) et du coût final pour une entreprise.

Le mode 0 est intéressant car il utilise, lors de l'émission, toutes les interfaces Ethernet physiques disponibles.

Ce mécanisme permettrait d'augmenter le débit binaire en utilisant les interfaces Gbit/s à disposition. Ainsi que de palier à une éventuelle panne au niveau des liens.

Le deuxième mode choisi est le mode 6. Le système de fonctionnement semble très abouti. En effet, ce mode propose une gestion sortante et entrante au niveau équilibrage de charge. De plus, à première vue, il serait le mieux adapté si on souhaite mettre en place une installation de type serveur avec de multiples connexions clients simultanées.

Le tableau suivant est un résumé, sous forme de comparaison, de mon choix d'étudier plus en détail les modes 0 et 6.

### Comparaison mode 0 et 6

| | Mode 0 | Mode 6 |
|---|---|---|
| **Avantages** | - débit binaire augmenté<br>- envoi séquentiel et cyclique sur deux interfaces<br>- une seule MAC adresse pour toutes les interfaces | - adaptative Load-Balancing en envoi et en réception<br>- gestion des interfaces selon les demandes et la charge |
| **Inconvénients** | - pas de Load-Balancing en réception | - 1 seule interface active au départ<br>- besoin d'un switch pour pouvoir fonctionner<br>- charge réseau augmentée dû à l'utilisation du protocole LLC toutes les secondes sur chaque interface active |

[20] Le protocole Spanning Tree : http://cisco.goffinet.org/s3/spanning_tree#.UdWrPjv0Evk

## Configuration Bonding

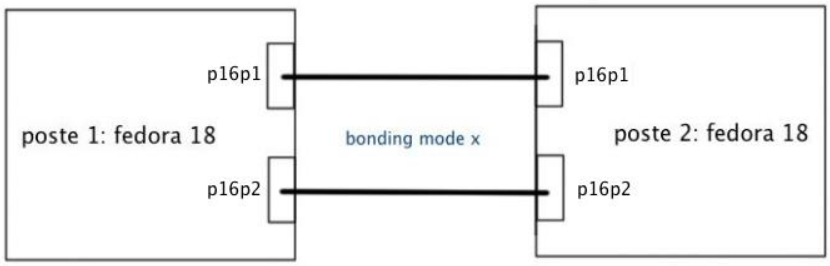



Figure 3: Schéma d'installation matériel

Deux PCs (A29 et A34) sous Fedora 18 avec chacun 1 carte Ethernet PCIe Intel®Pro/1000 PT Dual Port. La carte dispose de deux interfaces physiques Ethernet 1Gbit/s.

### ❖ Configuration Bonding sur les 2 postes

**-- Remarques importantes lors de la configuration**

Le gestionnaire de réseau NetworkManager ne permet pas, comme dans les versions antérieures de Fedora, de gérer et configurer l'agrégation de liens avec plusieurs cartes NIC. Si celui-ci n'est pas désactivé, l'installation ne se déroule pas correctement et ne fonctionne pas.
La solution est d'utiliser le service réseau et désactiver / désinstaller le NetworkManager. Les bonnes pratiques voudraient que l'on utilise le service réseau sur les serveurs et le NetworkManager sur les ordinateurs de bureau / ordinateurs portable. A l'état actuel on prévoit la gestion du bonding avec NetworkManager avec Fedora 20[21].

### ❖ *Configuration sous Fedora 18*

**-- Désactiver et désinstaller NetworkManager**

```
# chkconfig NetworkManager off
# service NetworkManager stop
# yum remove NetworkManager
```

**-- Activation du service réseau**

```
# chkconfig network on
# service network start
```

[21] NetworkManager and Bonding : https://fedoraproject.org/wiki/Features/NetworkManagerBonding

**-- Configuration de l'interface bond0**

```
# vi /etc/sysconfig/network-scripts/ifcfg-bond0
DEVICE="bond0"                          -- Nom logique de l'interface
ONBOOT="yes"                            -- Interface activée au démarrage
BOOTPROTO="none"                        -- Protocole utilisé au démarrage
IPADDR="192.168.0.2"
NETMASK="255.255.255.0"
GATEWAY="192.168.0.1"
USERCTL="no"                            -- Contrôle interface user non-root
BONDING_OPTS="mode=0 miimon=100"        -- Choix du mode et autres options
```

*miimon* : fréquence de la surveillance de liaison en ms.

**-- Configuration de l'interface p16p1**

```
# vi /etc/sysconfig/network-scripts/ifcfg-p16p1
DEVICE="p16p1"
ONBOOT="yes"
BOOTPROTO="none"
MASTER="bond0"
SLAVE="yes"
USERCTL="no"
```

**-- Configuration de l'interface p16p2**

```
# vi /etc/sysconfig/network-scripts/ifcfg-p16p2
DEVICE="p16p2"
ONBOOT="yes"
BOOTPROTO="none"
MASTER="bond0"
SLAVE="yes"
USERCTL="no"
```

**-- Création du fichier de configuration bonding.conf**

```
# vi /etc/modprobe.d/bonding.conf
alias bond0 bonding
```

**-- Vérifier si la configuration est correcte**

```
# cat /proc/net/bonding/bond0
```

**Remarque** : Lors d'un reboot, la configuration réseau peut être désactivée car le NetworkManager n'est plus disponible. La solution consiste à redémarrer le service réseau manuellement.

**-- Redémarrer le service réseau**

```
# service network restart
```

Lorsque l'on veut changer de mode, il faut arrêter le service réseau puis faire la modification dans l'interface bond0. Ensuite, redémarrer le service réseau ou rebooter le système.

### Explications sur mes choix lors de la configuration

J'ai effectué cette configuration du bonding après avoir étudié les paramètres pouvant être implémentés[22] au niveau de chaque interface. Mon choix s'est porté sur une configuration simple, cohérente et stable.

Certains de ces paramètres sont obligatoires au niveau de la configuration (MASTER, SLAVE, IPADDR, NETMASK, DEVICE et BONDING_OPTS).

Le paramètre BONDING_OPTS est fondamental pour que la configuration marche. Dans ce paramètre, il faut indiquer obligatoirement le mode choisi, ainsi que l'une des deux méthodes de contrôle des interfaces physiques (miimon ou arp_interval)[23].

Après avoir testé les valeurs 10, 100 et 1000 pour le paramètre miimon lors de tests de coupure de liens voici mes conclusions. La valeur 100 semble la plus adaptée pour deux raisons.

Tout d'abord les modes 0 et 6 que j'ai analysé travaillent avec les deux interfaces en simultané. Donc en cas de panne d'une carte Ethernet une détection toutes les 10 ms ne semble pas justifiée pour basculer le trafic.

A l'inverse, une valeur de détection de 1000 ms semble tout aussi mal adaptée car cela provoque des retransmissions de paquets inutiles lors d'un échange de données.

Les autres éléments de la configuration (ONBOOT, BOOTPROTO et USERCTL) restent secondaires et n'affectent pas le fonctionnent du driver bonding.

[22] Annexe : A.4 Descriptif des paramètres dans les interfaces
[23] Méthodes de monitoring (détection de liens)

## Ajustement fin de la configuration bonding

### Méthodes de monitoring (détection de liens)

Il est obligatoire de choisir une des deux méthodes de monitoring lors de la configuration de bond0 pour éviter une dégradation au niveau des performances si un lien venait à tomber.

J'ai pu constater que le driver bonding ne détectait pas la coupure d'un lien lorsqu'une de ses deux options n'était pas configurée. Il lui était ainsi impossible de donner la main à une des interfaces encore actives sans perte de données.

#### ❖ *miimon (monitoring des interfaces par mii-tool ou ethtool)*

```
-- Choix du mode et autres options
BONDING_OPTS="mode=0 miimon=100"        -- Paramètre miimon
```

Ce paramètre permet d'introduire une fréquence (ms) de scrutation des interfaces physiques à l'aide des outils mii-tool et ethtool[24] pour détecter si l'interface est active ou pas. La valeur par défaut de 100 ms est généralement conseillée. Selon moi elle permet une détection acceptable en cas de panne d'une interface. Cette valeur pourrait être ajustée selon le contexte et la définition que l'on peut avoir de « haute disponibilité  au niveau des liens ».

**Remarque** : La carte réseau doit supporter les registres MII et ETHTOOL. La valeur par défaut est 0 (option désactivée).

*Pour savoir si le pilote de la NIC supporte MII et ETHTOOL :*

```
# ethtool p16p1 | grep "Link detected:"
  Link detected: yes
```

#### ❖ *arp_interval et arp_ip_target (monitoring par arp)*

```
-- Choix du mode et autres options
BONDING_OPTS="mode=x arp_interval=xxx arp_ip_target=ip_sddress, ip_adress2"
```

Deuxième moyen de surveillance, cette fois à l'aide du protocole ARP. Cela évite l'utilisation de MII et ethtool.
Une requête ARP est envoyée par le driver bonding vers un PC qui est un arp_ip_target avec son ip_sddress.
Si aucune trame n'arrive pendant l'arp_interval (ms) indiqué, le driver transmet via cette interface une requête ARP à une autre IP définie dans arp_target (maximum 16 adresses IP). Si aucune réponse n'est obtenue, l'interface est désactivée.

[24] Annexe : A.5 Description des outils mii-tool et ethtool

**Remarque** : Compatible seulement avec les modes 0 et 2, les paramètres arp_interval et arp_ip_target doivent être signalés conjointement. La valeur par défaut est 0 (option désactivée).

- *Exemple de monitoring arp*

**-- Configuration de l'interface bond0**

```
# vi /etc/sysconfig/network-scripts/ifcfg-bond0
DEVICE="bond0"                          -- Nom logique de l'interface
ONBOOT="yes"                            -- Interface activée au démarrage
BOOTPROTO="none"                        -- Protocole utilisé au démarrage
IPADDR="192.168.0.2"
NETMASK="255.255.255.0"
GATEWAY="192.168.0.1"
USERCTL="no"                            -- Contrôle interface user non-root
BONDING_OPTS="mode=0 arp_interval=100 arp_ip_target=10.2.3.125"
```

Dans cet exemple, toutes les 100 ms, une requête ARP est envoyée vers l'adresse IP 10.2.3.125.

**-- Résultat avec une capture Wireshark** *(capture arp_interval-arp_ip_target)*

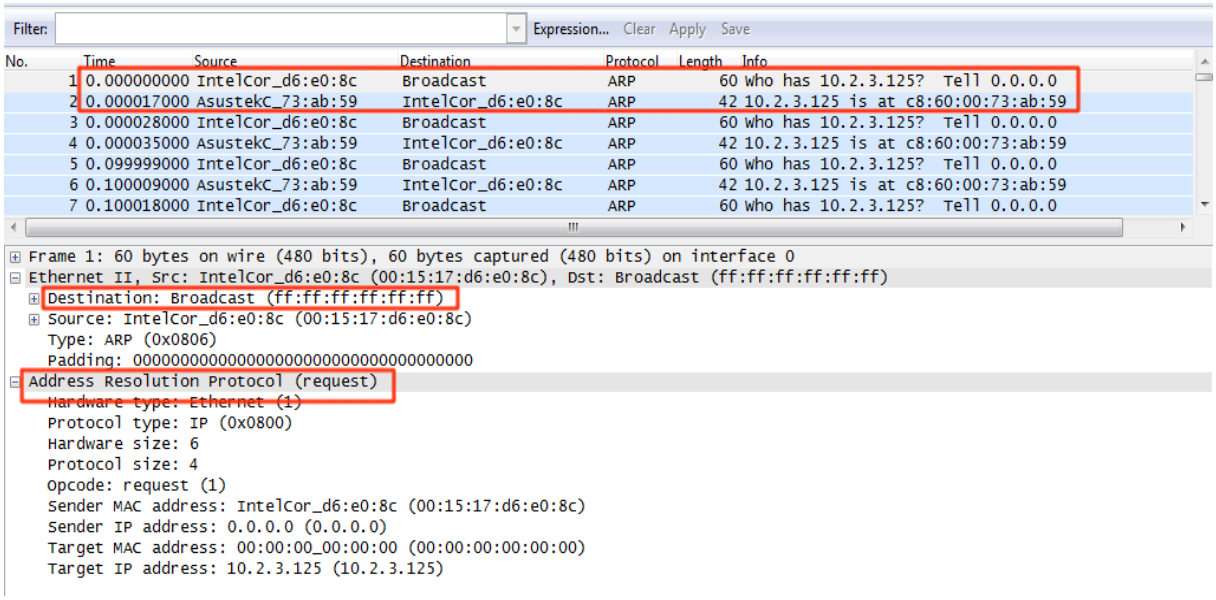



Figure 4: Capture Wireshark arp_interval

**Conclusion** : Cette capture permet d'observer qu'il y a un envoi par le PC configuré en bonding venant de l'interface bond0 (p16p1 + p16p2) d'une requête (arp request) toutes les 100 ms en broadcast demandant l'adresse MAC de 10.2.3.125.

### Options supplémentaires intéressantes

**-- updelay (0 par défaut)**

Temps de latence (ms) entre la découverte de la reconnexion d'une interface et de sa réutilisation par bond0.

**--downdelay (0 par défaut)**

Temps de latence (ms) entre la découverte de la déconnexion d'une interface et de sa désactivation de bond0.

**-- primary**

Cette option permet dans le cas du mode 6 de forcer le choix de la première interface par défaut au départ ou après une coupure de lien.

# *Etude du mode 0*
# *Balance – round robin*

# Réalisation

## Configuration mode 0

Afin d'effectuer les tests pour les divers scénarios qui vont suivre, je configure les deux PC clients Fedora 18 selon la marche à suivre exacte de la partie Configuration Bonding[25] se trouvant dans ce chapitre.

# Tests

## Objectif final

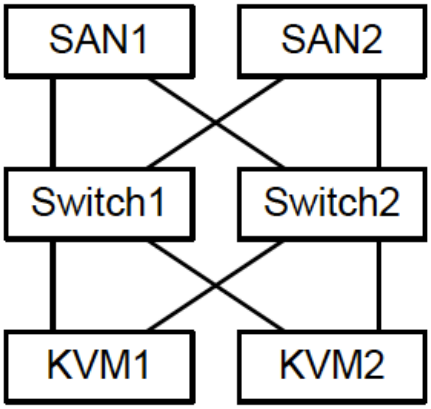



Figure 5: Schéma objectif final

L'objectif final de mon étude est de proposer un système redondant de stockage pour deux hyperviseurs KVM sur des Fedora 18.

Ce système sera composé de 2 équipements QNAP connectés sur deux switchs Ethernet 1 Gbit/s dans le but d'offrir un système hautement disponible. Les QNAPs TS-459 Pro II et TS-469 Pro II possèdent 2 interfaces Ethernet au maximum.

Cette partie est traitée en détail dans le chapitre II[26] de ce travail de mémoire. Les deux QNAPs étant basés sur un noyau Linux 2.6, je décide de commencer l'étude avec des PC configurés avec une distribution Fedora.

## Scénario de test 1 : Deux PCs Fedora 18 avec bonding

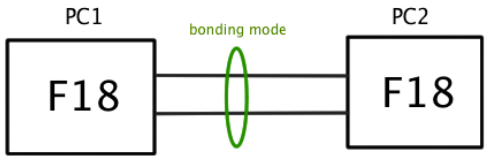



Figure 6: Schéma scénario test 1 mode 0

L'objectif dans cette partie est d'étudier le bonding en mode 0 au niveau du fonctionnement avec 2 interfaces physiques (Gbit/s) sur 2 PCs sur Fedora 18 Gnome.

Les deux PCs sont directement connectés entre eux.

[25] Chapitre I : Bonding sur Fedora 18 Réalisation - Configuration Bonding
[26] Chapitre II Virtualisation et Stockage : Scénario de départ


Figure 7: Schéma de principe mode 0

Ce mode doit avoir la même adresse MAC sur toutes les interfaces et une seule IP. Je commence donc par observer le fonctionnement du mode 0 au niveau des interfaces, puis dans un deuxième temps mesurer le débit binaire lors d'un transfert de fichier.

**-- Vérification si le mode 0 fonctionne :**

```
[root@localhost admin]# cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v3.7.1 (April 27, 2011)

Bonding Mode: load balancing (round-robin)
MII Status: up
MII Polling Interval (ms): 100          -- Fréquence de surveillance MII
Up Delay (ms): 0                        -- Délai en cas de reconnexion
Down Delay (ms): 0                      -- Délai en cas de déconnexion

Slave Interface: p16p1                  -- Le nom de l'interface
MII Status: up                          -- Etat de l'interface selon MII
Speed: 1000 Mbps                        -- Débit binaire du lien
Duplex: full                            -- Liaison full duplex
Link Failure Count: 0                   -- Compteur de lien coupé
Permanent HW addr: 00:15:17:d6:e0:8c    -- Adresse MAC réelle au niveau HW
Slave queue ID: 0
Slave Interface: p16p2
MII Status: up
Speed: 1000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 00:15:17:d6:e0:8d    -- Adresse MAC réelle au niveau HW
Slave queue ID: 0
```

**Conclusion :** On peut observer que le mode 0 est activé ainsi que les interfaces physiques p16p1 et p16p2 (MII Status : up).

### Tests avec la commande ping pour le scénario 1

**-- Etat des interfaces au départ (ifconfig PC1)**

```
[root@localhost admin]# ifconfig
bond0: flags=5187<UP,BROADCAST,RUNNING,MASTER,MULTICAST>  mtu 1500
        inet 192.168.0.2  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::215:17ff:fed6:e08c  prefixlen 64  scopeid 0x20<link>
        ether 00:15:17:d6:e0:8c  txqueuelen 0  (Ethernet)
        RX packets 7  bytes 676 (676.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 48  bytes 7545 (7.3 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

p16p1: flags=6211<UP,BROADCAST,RUNNING,SLAVE,MULTICAST>  mtu 1500
        ether 00:15:17:d6:e0:8c  txqueuelen 1000  (Ethernet)
        RX packets 3  bytes 268 (268.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 25  bytes 3919 (3.8 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 16  memory 0xfea80000-feaa0000
p16p2: flags=6211<UP,BROADCAST,RUNNING,SLAVE,MULTICAST>  mtu 1500
        ether 00:15:17:d6:e0:8c  txqueuelen 1000  (Ethernet)
        RX packets 4  bytes 408 (408.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 23  bytes 3626 (3.5 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 17  memory 0xfeae0000-feb00000
```

**Conclusion :** Une seule adresse IP et la même adresse MAC pour les différentes interfaces comme prévu.

**-- Ping du PC2 (192.168.0.4) depuis le PC1**

```
[root@localhost admin]# ping -c 4 192.168.0.4
PING 192.168.0.4 (192.168.0.4) 56(84) bytes of data.
64 bytes from 192.168.0.4: icmp seq=1 ttl=64 time=0.413 ms
64 bytes from 192.168.0.4: icmp seq=2 ttl=64 time=0.142 ms
64 bytes from 192.168.0.4: icmp seq=3 ttl=64 time=0.157 ms
64 bytes from 192.168.0.4: icmp_seq=4 ttl=64 time=0.168 ms

--- 192.168.0.4 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3000ms
rtt min/avg/max/mdev = 0.142/0.220/0.413/0.111 ms
```

**-- Résultat des interfaces après le ping sur le PC1**

```
[root@localhost admin]# ifconfig
bond0: flags=5187<UP,BROADCAST,RUNNING,MASTER,MULTICAST>  mtu 1500
        inet 192.168.0.2  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::215:17ff:fed6:e08c  prefixlen 64  scopeid 0x20<link>
        ether 00:15:17:d6:e0:8c  txqueuelen 0  (Ethernet)
        RX packets 12  bytes 1212 (1.1 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 53  bytes 8081 (7.8 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

p16p1: flags=6211<UP,BROADCAST,RUNNING,SLAVE,MULTICAST>  mtu 1500
        ether 00:15:17:d6:e0:8c  txqueuelen 1000  (Ethernet)
        RX packets 6  bytes 536 (536.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 28  bytes 4187 (4.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 16  memory 0xfea80000-feaa0000

p16p2: flags=6211<UP,BROADCAST,RUNNING,SLAVE,MULTICAST>  mtu 1500
        ether 00:15:17:d6:e0:8c  txqueuelen 1000  (Ethernet)
        RX packets 6  bytes 676 (676.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 25  bytes 3894 (3.8 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 17  memory 0xfeae0000-feb00000
```

**Conclusion :** On peut observer à l'aide d'une commande ping que la charge a été répartie lors de l'envoi entre les deux interfaces physiques. Le total de la charge est comptabilisé sur l'interface virtuelle bond0.

| | TX (trames avant) | TX (trames après) | RX (trames avant) | RX (trames après) |
|---|---|---|---|---|
| **bond0** | 48 | 53 (+5) | 7 | 12 (+5) |
| **p16p1** | 25 | 28 (+3) | 3 | 6 (+3) |
| **p16p2** | 23 | 25 (+2) | 4 | 6 (+2) |

**Remarque :** Le ping effectué correspond à exactement 4 trames en envoi et 4 en réception. Le ping terminé, une des interfaces (ici p16p1) effectue une requête ARP pour maintenir à jour le cache, ce qui correspond à 2 trames supplémentaires.

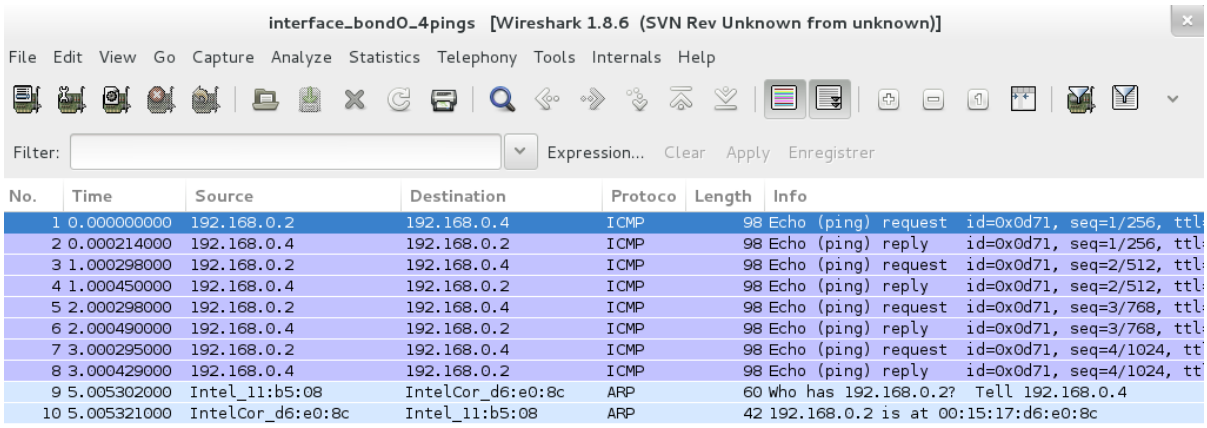
**-- Vue du cache ARP du PC1**

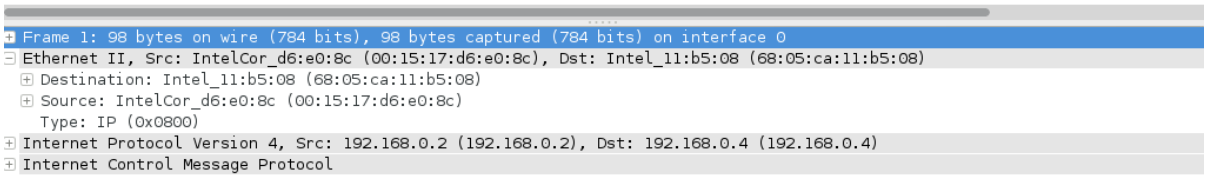
```
root@localhost admin]# arp -D
Address                  Hwtype  Hwaddress           Flags Mask      Iface
192.168.0.4              ether   68:05:ca:11:b5:08   C               bond0
```

Au niveau du cache ARP une seule adresse IP et une seule MAC adresse pour le PC2 configuré en mode 0 également.

**-- Acquisitions avec Wireshark** *(Scénario1/mode0/Ping)*

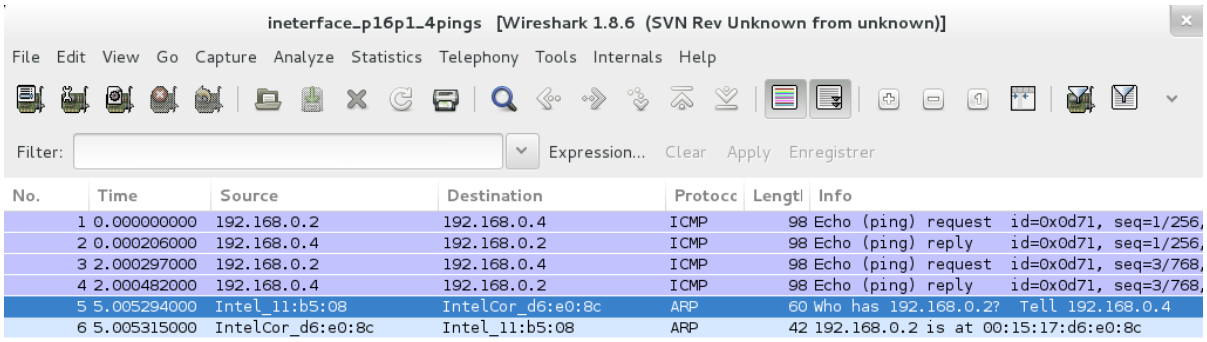
*Interface bond0 PC1*

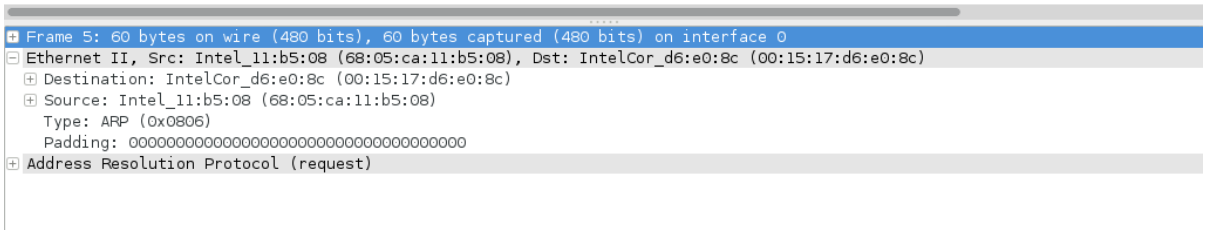


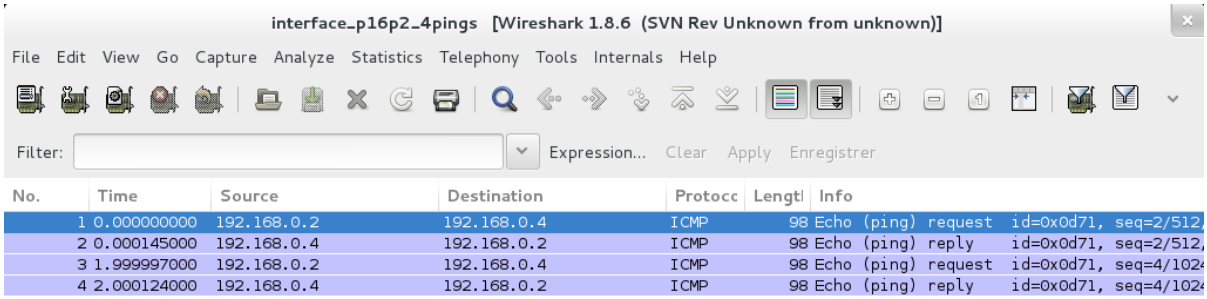

Une seule adresse MAC et IP pour toutes les interfaces du PC1 et du PC2. La requête est de type unicast car les deux PC sont connectés directement, donc les MAC adresses sont connues et se trouvent dans le cache ARP de chaque PC.

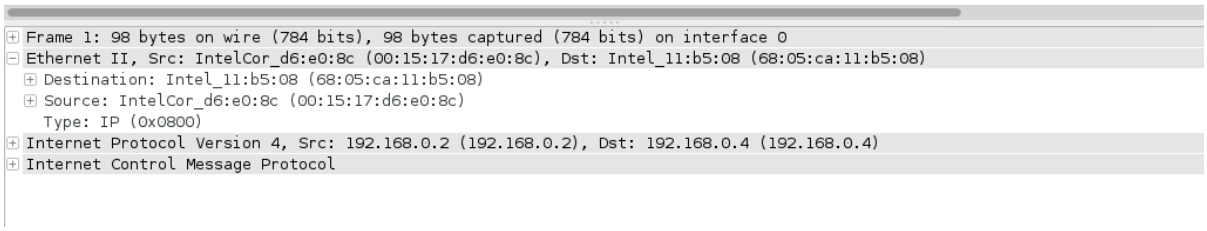
*Interface p16p1 PC1*



*Interface p16p2 PC2*




Cette acquisition m'a permis de constater qu'il n'y a pas de doublement au niveau des paquets lors de l'envoi d'un ping.

Au niveau de l'axe temporel, on peut voir les différents échanges ICMP entre les PCs lors de l'envoi du ping par PC1. Puis à la fin de l'échange, une requête arp est envoyée par le PC1 pour maintenir son cache à jour.

Le comportement est strictement le même sur les deux PCs comme j'ai pu le vérifier grâce aux acquisitions. Les données sont envoyées et reçues sur les deux interfaces de chaque PC de façon séquentielle.

### Tests avec la commande Iperf pour le scénario 1

Iperf permet de mesurer le débit binaire et la qualité d'un lien entre deux équipements en mode client-serveur.

Cette commande permet l'envoi de données au niveau TCP dans le but de connaître le débit utile au niveau socket.

De plus, on peut savoir exactement combien de données sont envoyées lors du transfert, ce qui va permettre de vérifier l'équilibrage de charge au niveau des liens en sortie.

**Remarque** : La commande Iperf travaille en mode client (PC1)-serveur (PC2). Il ne faut pas oublier d'ouvrir le port TCP 5001 au niveau du Firewall sur le PC1 et le PC2 pour que cet outil puisse fonctionner.

Voici la méthodologie employée et les résultats obtenus.

**-- Iperf en mode serveur sur le PC2**

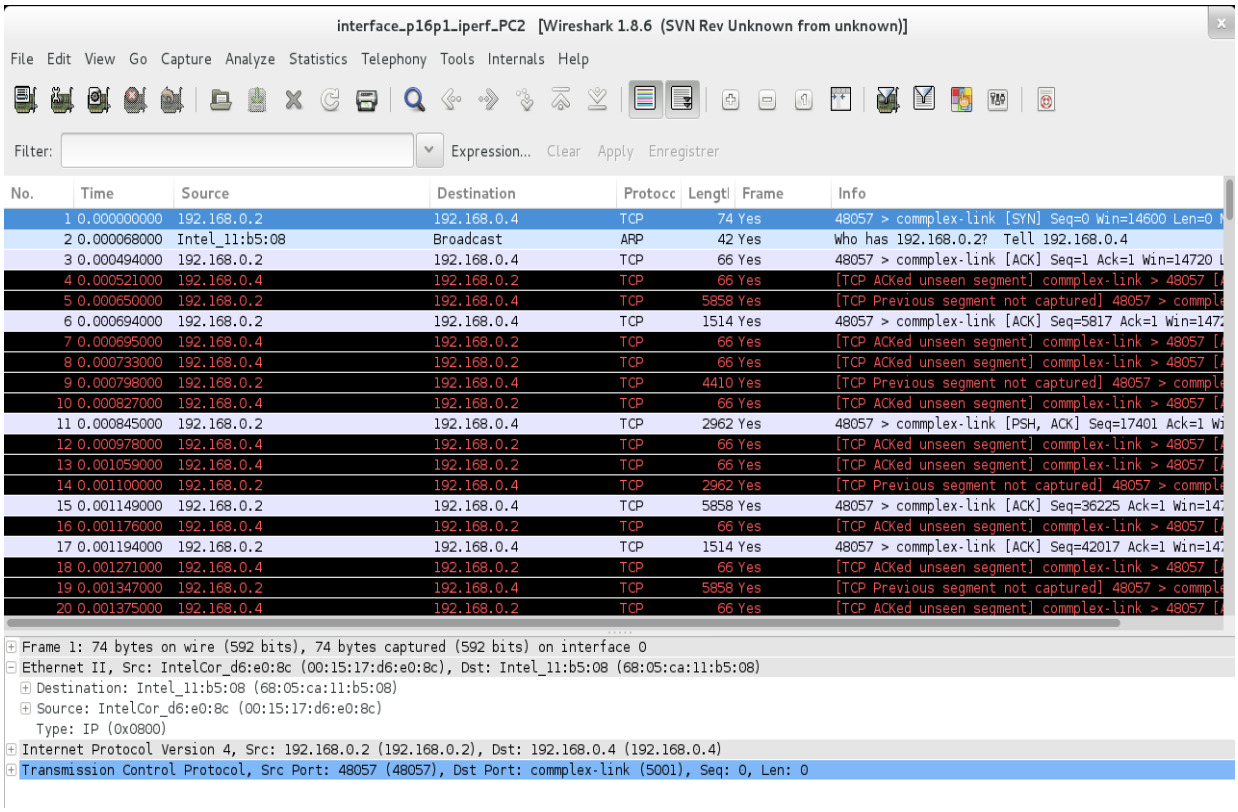
```
[root@localhost network-scripts]# Iperf -s
```

Wireshark sur le PC1 et sur le PC2 pour effectuer une acquisition lors de l'envoi sur toutes les interfaces (bond0, p16p1, p16p2).

**-- Iperf en mode client sur le PC1**

```
[root@localhost network-scripts]# Iperf -c 192.168.0.4
```

**-- Acquisitions avec Wireshark sur l'interface p16p1 sur le PC2 et l'option conversation** *(Scénario 1/mode0/Iperf)*



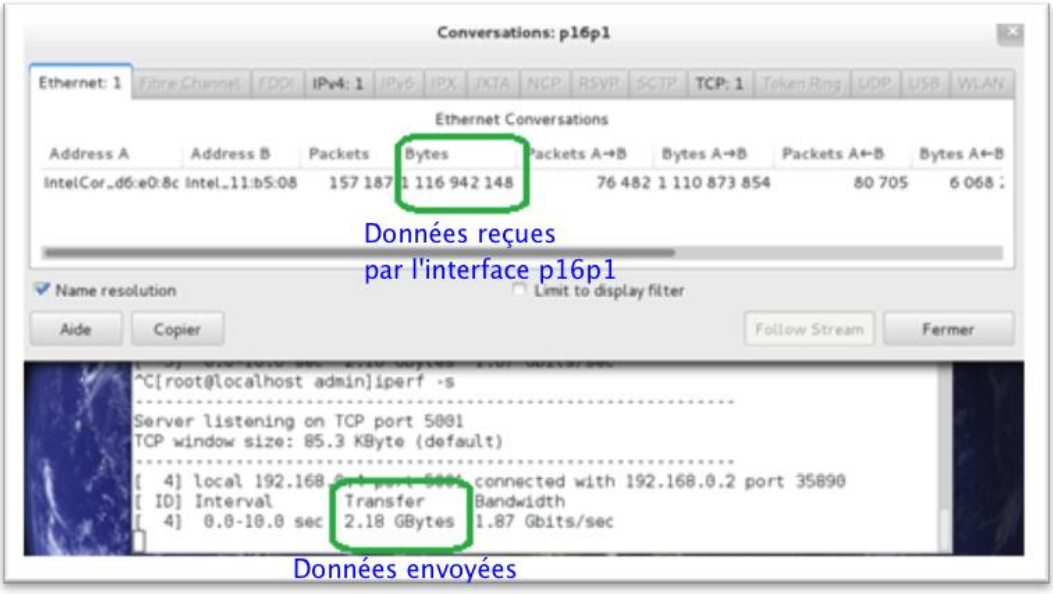



Figure 8: Acquisitions Wireshark mode 0 avec Iperf

**-- Résultat de la commande Iperf**

```
[root@localhost network-scripts]# Iperf -c 192.168.0.4
------------------------------------------------------------
Client connecting to 192.168.0.4, TCP port 5001
TCP window size: 22.9 KByte (default)
------------------------------------------------------------
[  3] local 192.168.0.2 port 35873 connected with 192.168.0.4 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  2.16 GBytes  1.87 Gbits/sec
```

Par défaut, Iperf lance un test d'une durée de 10 secondes. Durant cette période, cet outil envoie le maximum de charge utile possible vers le PC2.

Ici, les données utiles envoyées durant ce temps correspondent à 2,18 GBytes pour un débit binaire de 1,87 Gbit/s.

En parallèle, j'ai effectué une capture Wireshark. A l'aide de l'outil statistique de l'analyseur, je constate que 1,12 GBytes de données utiles sont envoyées sur l'interface p16p1 du PC1. Soit 50% de la charge utile transmise sur le réseau par Iperf. Les autres 50% sont transmis par l'interface p16p2 du PC1.

On peut ainsi conclure à l'aide de ce test que la charge est répartie lors d'un envoi de façon équitable, sur les deux liens, lors d'un transfert du PC1 vers le PC2. Le mode 0 utilise effectivement un mécanisme de Load Balancing lors d'un envoi de données.

### Test de débit pour scénario 1

Dans cette partie, je vais effectuer des tests dans le but de déterminer le débit binaire maximum pouvant être atteint avec le mode 0.

#### *Calcul théorique du débit utile maximum de 1 lien Gbit/s*

Avant de commencer les tests, il est judicieux de calculer théoriquement le débit utile maximum possible avec 1 lien Gbit/s.

**-- Description d'un scénario**

Imaginons l'envoi d'un fichier de 1Mo de données utiles ($2^{20}$ = 1048576 octets). L'envoi s'effectué avec IP et TCP sans aucune erreur. Je prends en considération que les données utiles qu'il est possible de transmettre (1500 octets au maximum).

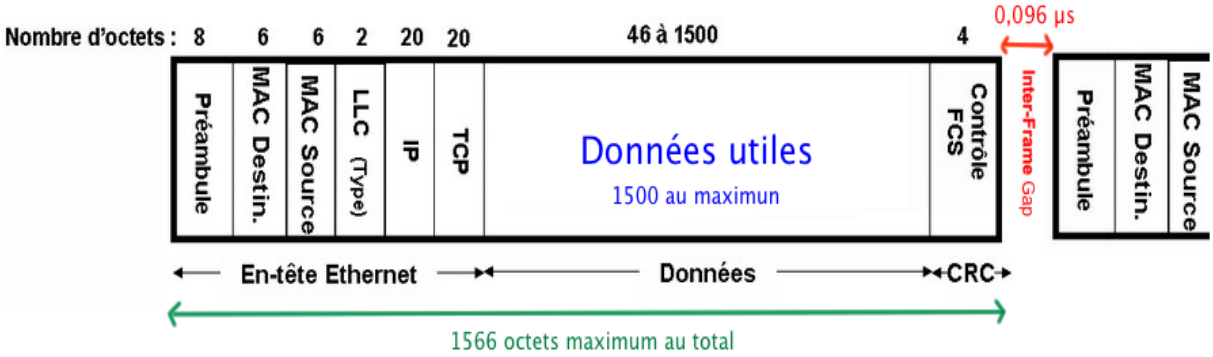



Figure 9: Format de 1 trame Ethernet avec IP-TCP

La partie overhead (Header Ethernet + CRC) n'est pas comptabilisée dans mon envoi. Enfin, il faudra tenir compte l'inter-Frame Gap, soit 0.096 µs pour un lien Gbit/s.

Il est évident que le fichier n'est pas envoyé en 1 fois avec 1 seule trame. Il est partagé en plusieurs trames lors de l'envoi.

**-- Calcul du nombre de trames de données utiles nécessaires pour envoyer 1Mo**

$$nb\ de\ trames = \frac{données\ utiles_{\ fichier}}{données\ utiles_{\ \max 1\ \text{trame}}} = \frac{1048576\ octets}{1500\ octets} \cong 700\ trames$$

Maintenant que j'ai le nombre de trames nécessaires pour envoyer le fichier de 1Mo, j'ai besoin de connaître le temps de transmission de 1 trame de 1566 octets.

**-- Temps pour transmettre 1 trame standard**

1Gbit/s = $2^{30}$ bit/s

$$t_{transmission\ 1\ trame} = \frac{données\ utiles\ _{fichier}}{1\ lien\ Gbit/s} = \frac{(1566\ octets\ \times 8)\ bit/s}{2^{30}\ bit/s} = 11{,}67\ \mu s$$

Ensuite, il est possible de calculer le débit utile maximum de 1 lien Gbit/s en tenant compte qu'il y aura 699 inter-frames gap de 0.096 µs.

**-- Calcul débit utile maximum de 1 lien Gbit/s**

$t_{réél\ de\ transmission} = nb\ de\ trames \cdot t_{transmission\ 1\ trame} + (nb\ _{inter-frames\ gap} \cdot t_{1\ inter-frame\ gap}) =$

$700 \cdot 11{,}67\ \mu s + (699 \cdot 0.096\ \mu s) = 8236{,}104\ \mu s \cong 8{,}24\ ms$

$$\frac{1\ Mo}{8{,}24 \cdot 10^{-3}\ secondes} = \frac{x\ Mo}{1\ seconde} \rightarrow x = 121{,}36\ Mo/s$$

***débit utile théorique maximun*** $_{1\ lien\ Gbit/s}$ **= 121,36 Mo/s = 971 Mbit/s**

### *Méthodologie de mesures avec Iperf et explications*

Je décide d'utiliser Jperf[27] qui est une interface graphique pour Iperf écrite en Java lors des mesures de débit. J'espère de cette façon rendre plus visuel et plus claire mes mesures.

**-- Iperf en mode serveur sur le PC2**

```
[root@localhost admin]# Iperf -s
```

**-- Iperf en mode client sur le PC1**

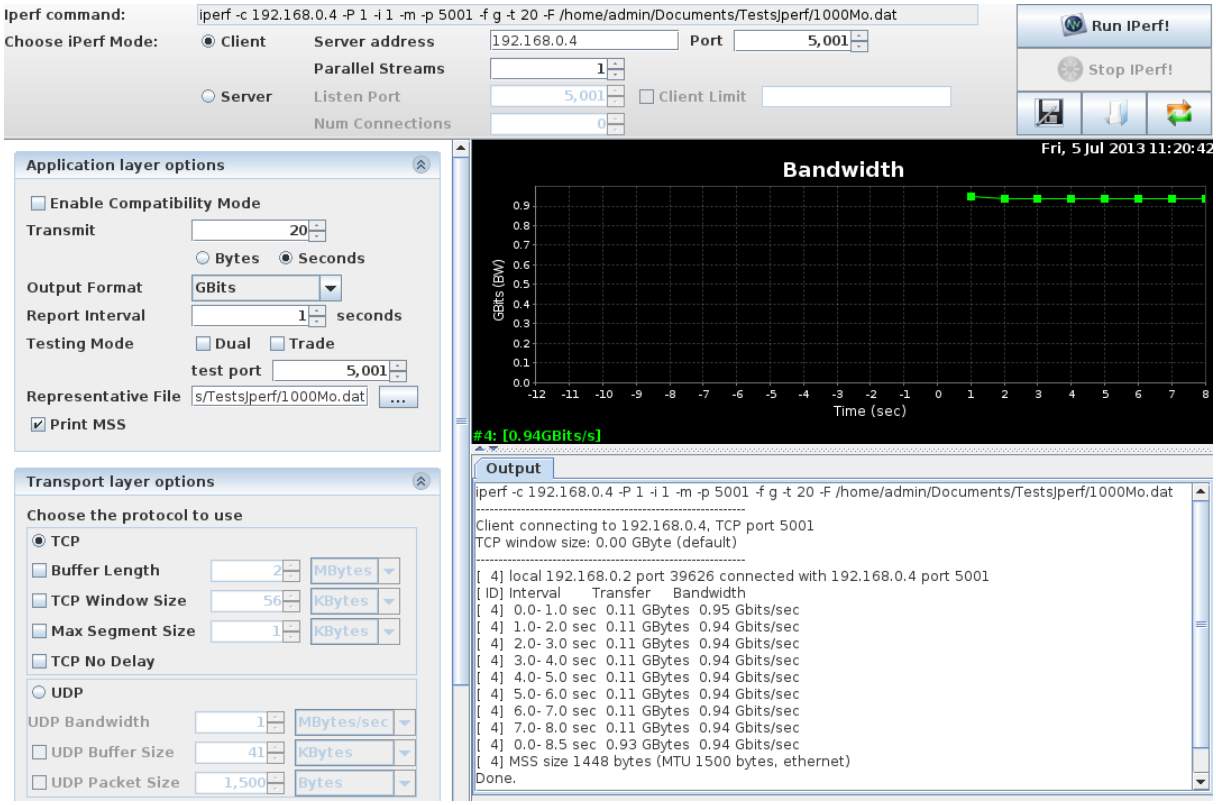
```
[root@localhost admin]# ./jperf.sh
```

La configuration au niveau de Jperf est assez simple à comprendre et visible au niveau des captures. De plus, la commande effectuée avec Iperf est clairement spécifiée.

La seule spécificité de ma configuration porte sur mon choix de transférer à chaque fois un fichier d'exactement 1Go. *(TestsJperf/config_jperf.jperf)*

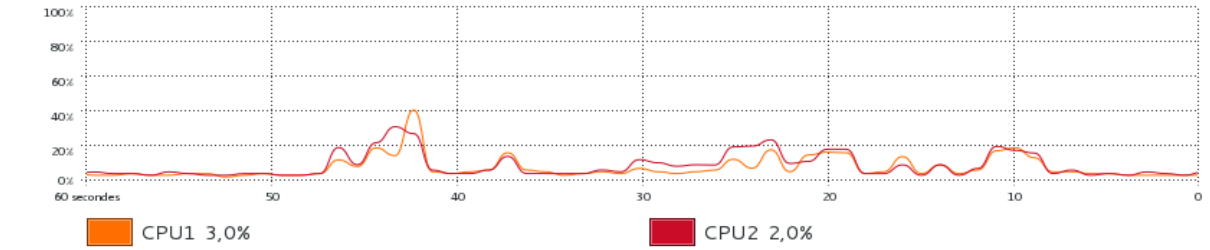
[27], Installations, configurations et démonstrations d'Iperf et Jperf : http://openmaniak.com/fr/iperf.php

## Résultats pour 1 lien Gbit/s sans bonding

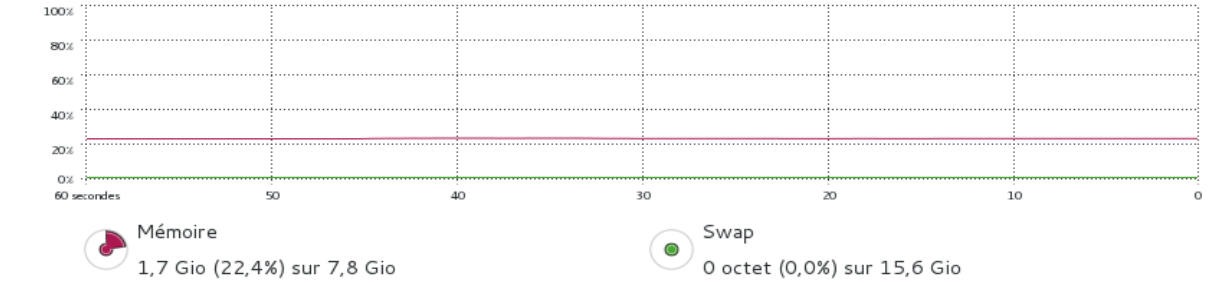



**-- Capture du moniteur de ressources Fedora sur PC1**

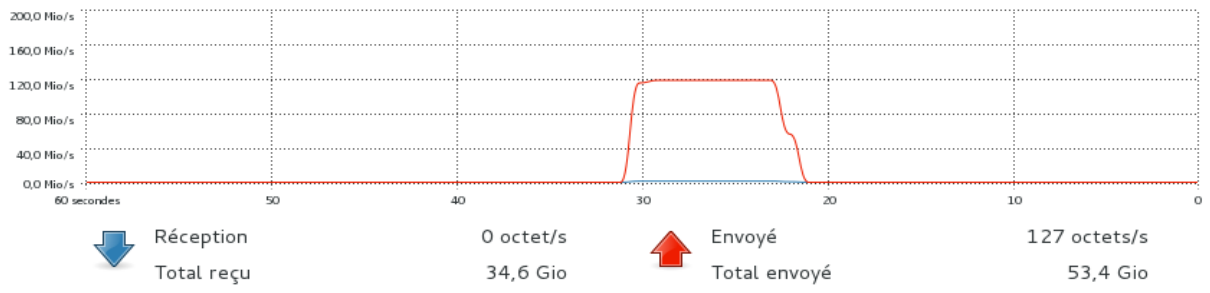
Historique d'utilisation du CPU




Historique d'utilisation de la mémoire physique et du fichier d'échange




Historique du trafic réseau



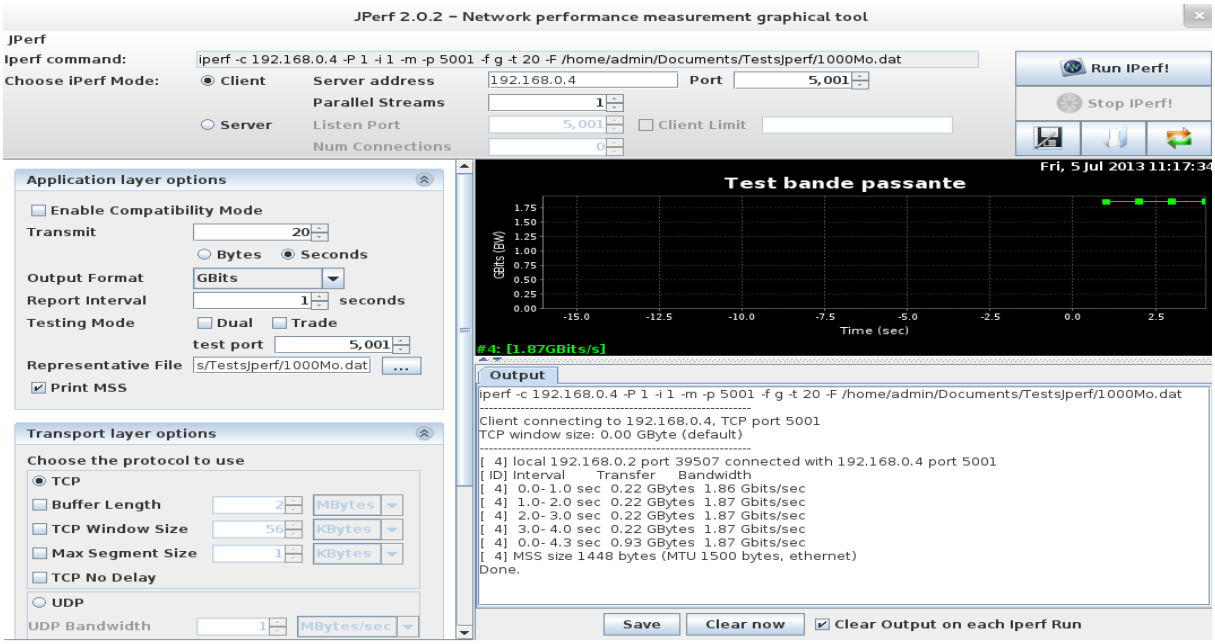
### *Tableau des résultats 1 lien Gbit/s sans bonding*

| Débit utile théorique maximal [Mbit/s] | Débit utile mesuré [Mbit/s] | Utilisation débit utile [%] | Temps de transfert [s] | Débit utile théorique maximal [Mo/s] | Débit utile mesuré [Mo/s] | Utilisation débit utile [%] | Utilisation CPU PC1 [%] | Utilisation mémoire RAM PC1 [%] |
|---|---|---|---|---|---|---|---|---|
| Calcul | Iperf | Calcul | Iperf | Calcul | Moniteur Système F18 | Calcul | Moniteur Système F18 | Moniteur Système F18 |
| 971 | 950 | 97,84 | 8,5 | 121,36 | 120 | 98,88 | 22 | 22,4 |

Figure 10: Tableau de résultats 1 lien Gbit/s sans bonding

On peut remarquer une utilisation faible au niveau du CPU et de la RAM sur le PC1 lors du transfert. Le débit binaire mesuré avec Iperf correspond à 950 Mbit/s.

## Résultats scénario 1 bonding mode 0




**-- Capture avec le moniteur système de Fedora sur le PC1**

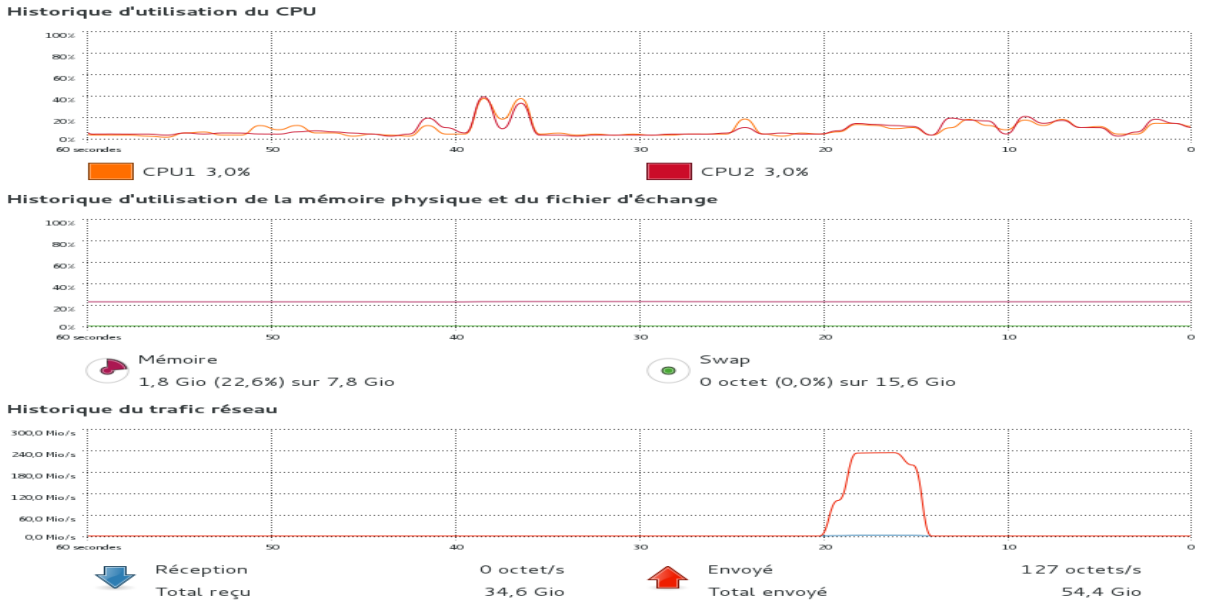


*Tableau des résultats pour le scénario 1 bonding mode 0*

| Débit utile théorique maximal [Mbit/s] | Débit utile mesuré [Mbit/s] | Utilisation débit utile [%] | Temps de transfert [s] | Débit utile théorique maximal [Mo/s] | Débit utile mesuré [Mo/s] | Utilisation débit utile [%] | Utilisation CPU PC1 [%] | Utilisation mémoire RAM PC1 [%] |
|---|---|---|---|---|---|---|---|---|
| Calcul | Iperf | Calcul | Iperf | Calcul | Moniteur Système F18 | Calcul | Moniteur Système F18 | Moniteur Système F18 |
| 1942 | 1870 | 96,29 | 4,3 | 242,72 | 240 | 98,88 | 16 | 22,6 |

L'utilisation au niveau CPU et RAM restent faibles. Le débit binaire correspond à 1,87 Gbit/s avec 2 liens Ethernet.

## Scénario test 2 : 2 PCs Fedora 18 et un Switch

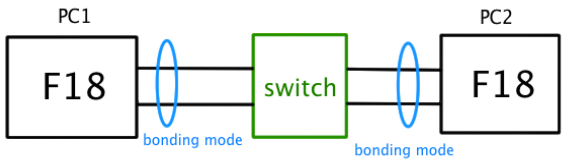



Figure 11: Schéma scénario test 2

L'objectif de ce scénario est d'étudier le mécanisme de fonctionnement du mode 0 configuré sur 2 PCs Fedora 18 avec 2 interfaces physiques (Gbit/s) reliées à un switch Netgear GS116E 16 ports Gbit/s.

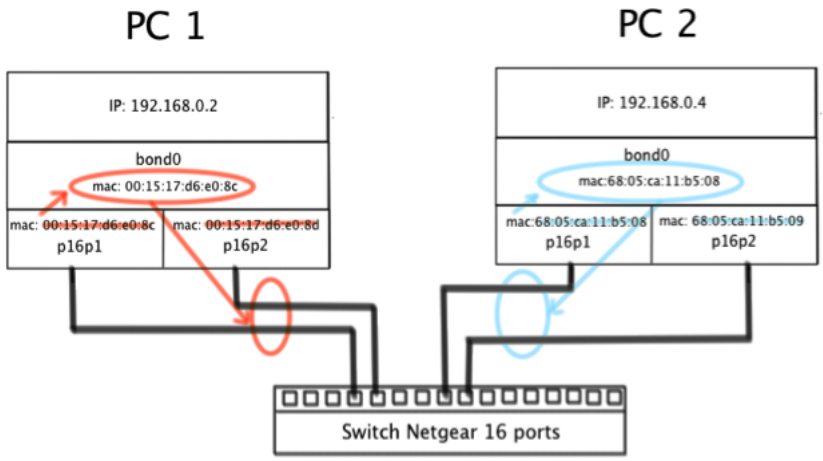


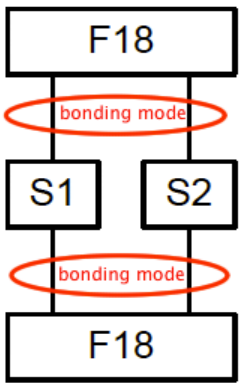

Figure 12: Schéma de principe mode 0 avec 1 switch

Je mets en place ce scénario avec un switch standard Netgear GS116E 16 ports Gbit/s. Cependant, lors des tests effectués avec Iperf, je m'aperçois d'une dégradation importante au niveau performances. Seulement 65% du débit binaire est utilisé avec les 2 liens Gbit/s.

Le mode 0 exige un switch évolué comme le Cisco SG-300-28 compatible EtherChannel pour fonctionner si on souhaite connecter tous les liens au même commutateur.

Je n'ai pas vu cette subtilité et j'ai longtemps cherché d'autres explications expliquant les dégradations de performances.

## Scénario test 3 : 2 PCs Fedora 18 et 2 Switchs




Dans ce nouveau scénario, je vais étudier l'effet sur le fonctionnement du mode 0 avec la mise en parallèle de deux switchs Netgear GS116E. Une seule interface physique Gbit/s est reliée à chacun des switchs.

Figure 13: Schéma scénario test 3

### Tests avec switchs Linux et TCPdump

Dans le but d'étudier les échanges ainsi que les mécanismes, je décide de remplacer le switch Netgear par un switch Linux.

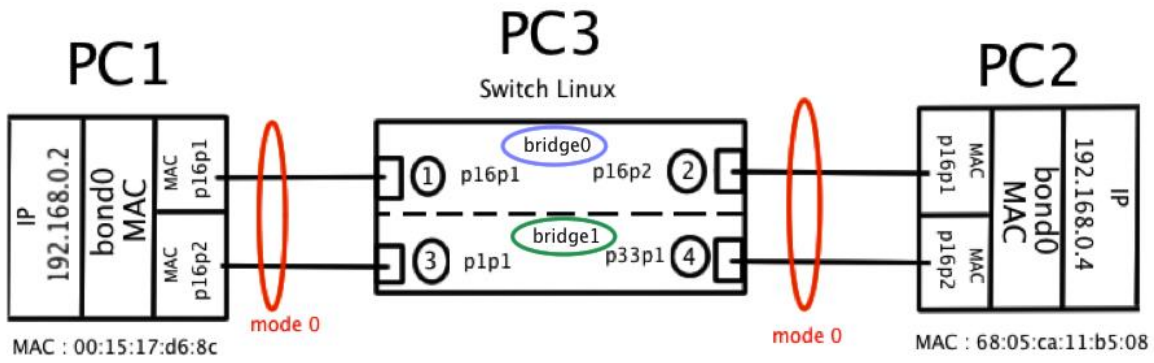



Figure 14: Schéma test mode 0 2 PCs et 2 switchs Linux

Pour obtenir deux switchs Linux sur le même PC, je procède à la configuration de 2 bridges[28] totalement indépendants ayant chacun deux cartes Ethernet comme on peut le voir sur le schéma.

[28] A.7 Configuration de deux bridges sur Fedora 18

**-- Résultat capture TCPdump**

| Interface | Sens | Temps | N° | Action |
|---|---|---|---|---|
| p16p1 | in | 15:58:07.815313 | 1 | B 00:15:17:d6:e0:8c (oui Unknown) ethertype ARP (0x0806), length 62: Request who-has 192.168.0.4 tell 192.168.0.2, length 46 |
| p16p2 | out | 15:58:07.815338 | 2 | Out 00:15:17:d6:e0:8c (oui Unknown) ethertype ARP (0x0806), length 62: Request who-has 192.168.0.4 tell 192.168.0.2, length 46 |
| p33p1 | in | 15:58:07.815458 | 3 | P 68:05:ca:11:b5:08 (oui Unknown) ethertype ARP (0x0806), length 62: Reply 192.168.0.4 is-at 68:05:ca:11:b5:08 (oui Unknown), length 46 |
| p1p1 | out | 15:58:07.815472 | 4 | Out 68:05:ca:11:b5:08 (oui Unknown) ethertype ARP (0x0806), length 62: Reply 192.168.0.4 is-at 68:05:ca:11:b5:08 (oui Unknown), length 46 |
| p1p1 | in | 15:58:07.815612 | 5 | P 00:15:17:d6:e0:8c (oui Unknown) ethertype IPv4 (0x0800), length 100: 192.168.0.2 > 192.168.0.4: ICMP echo request, id 2176, seq 1, length 64 |
| p33p1 | out | 15:58:07.815643 | 6 | Out 00:15:17:d6:e0:8c (oui Unknown) ethertype IPv4 (0x0800), length 100: 192.168.0.2 > 192.168.0.4: ICMP echo request, id 2176, seq 1, length 64 |
| p16p2 | in | 15:58:07.815825 | 7 | P 68:05:ca:11:b5:08 (oui Unknown) ethertype IPv4 (0x0800), length 100: 192.168.0.4 > 192.168.0.2: ICMP echo reply, id 2176, seq 1, length 64 |
| p16p1 | out | 15:58:07.815840 | 8 | Out 68:05:ca:11:b5:08 (oui Unknown) ethertype IPv4 (0x0800), length 100: 192.168.0.4 > 192.168.0.2: ICMP echo reply, id 2176, seq 1, length 64 |
| p33p1 | in | 15:58:12.825762 | 9 | P 68:05:ca:11:b5:08 (oui Unknown) ethertype ARP (0x0806), length 62: Request who-has 192.168.0.2 tell 192.168.0.4, length 46 |
| p1p1 | out | 15:58:12.825781 | 10 | Out 68:05:ca:11:b5:08 (oui Unknown) ethertype ARP (0x0806), length 62: Request who-has 192.168.0.2 tell 192.168.0.4, length 46 |
| p16p1 | in | 15:58:12.825945 | 11 | P 00:15:17:d6:e0:8c (oui Unknown) ethertype ARP (0x0806), length 62: Reply 192.168.0.2 is-at 00:15:17:d6:e0:8c (oui Unknown), length 46 |
| p16p2 | out | 15:58:12.825955 | 12 | Out 00:15:17:d6:e0:8c (oui Unknown) ethertype ARP (0x0806), length 62: Reply 192.168.0.2 is-at 00:15:17:d6:e0:8c (oui Unknown), length 46 |

-- Schéma de fonctionnement mode 0 dans le scénario 3 (monitoring miimon=100)

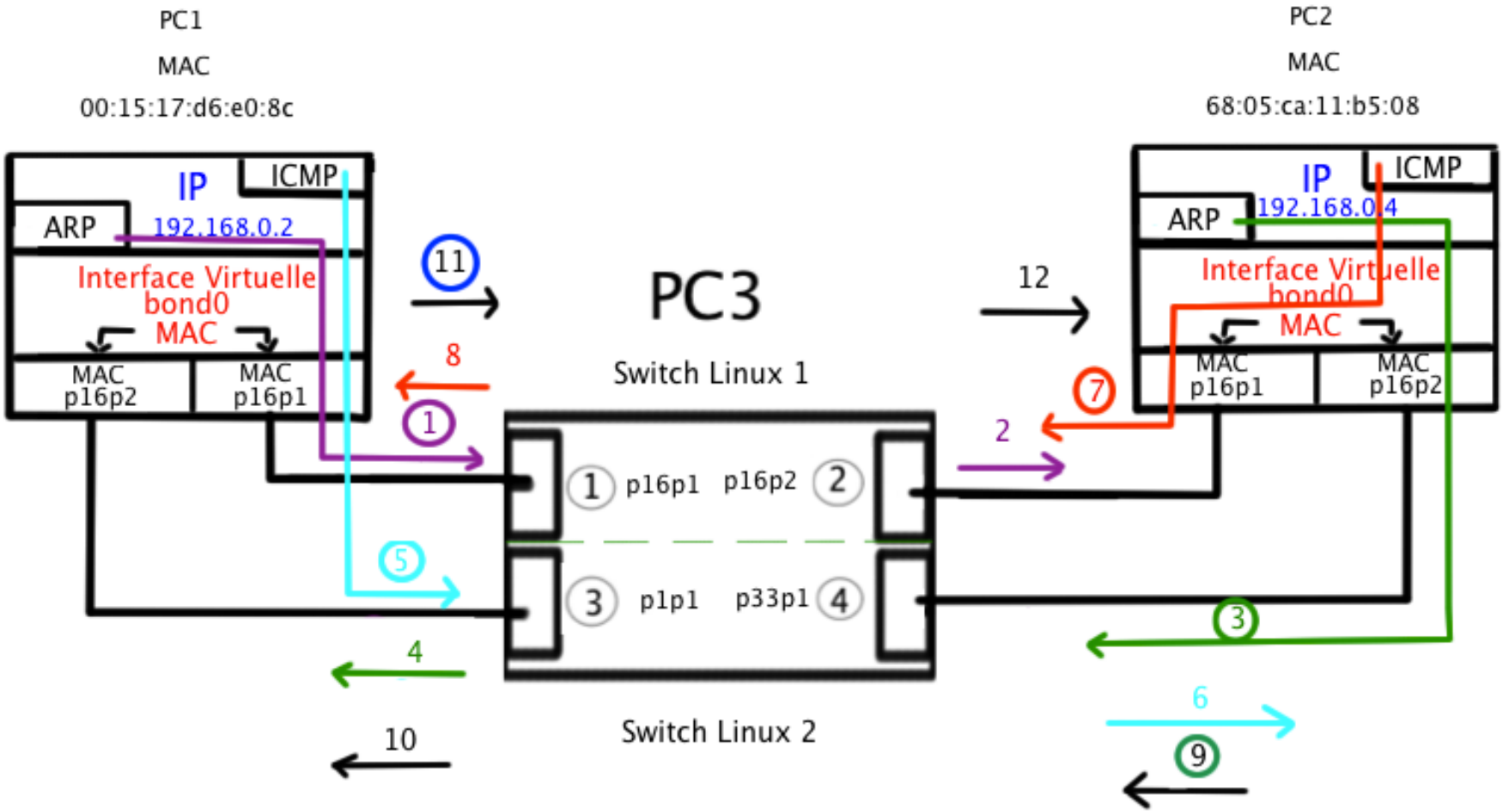



Figure 15: Schéma de fonctionnement mode 0 2 PCs et 2 switchs Linux

**Description des différentes étapes :**

**-- Partie ARP REQUEST PC1 --> PC2**

1. ARP Request du PC1 interface p16p1 choisie par le driver bonding pour l'envoi. MAC source 00:15:17:d6:e0:8c, MAC destination broadcast. Request who-has 192.168.0.4 tell 192.168.0.2.

2. Réception de l'ARP Request par l'interface p16p1 du PC2.

**-- Partie ARP REPLY PC1 <-- PC2**

3. Envoi ARP Reply depuis l'interface p16p2 du PC2. 192.168.0.4 is-at 68:05:ca:11:b5:08. Changement d'interface pour l'envoi.

4. Réception de l'ARP Reply par l'interface p16p2 de PC1.

**-- Partie ECHO REQUEST PC1 --> PC2**

5. ICMP Echo Request de PC1 vers PC2. Changement de l'interface de départ (point 1) vers p16p2 qui a reçue l'ARP Reply (point 4).

6. Réception ICMP Echo Request par l'interface p16p2 du PC2.

**-- Partie ICMP ECHO REPLY PC2 --> PC1**

7. Envoi ICMP Reply par interface du PC2 (p16p1). Changement de l'interface (point 3).

8. Réception ICMP Reply par l'interface qui a effectué le premier ARP Request sur le PC1 (p16p1).

**-- Nouvel ARP Request cette fois du PC2 --> PC1**

9. Envoi ARP Request depuis l'autre interface du PC2 (p16p2).

10. Réception ARP Request sur l'interface p16p2 du PC1, soit la dernière interface à être active.

-- Réponse ARP REPLY PC1 --> PC2

11. Réponse par l'autre interface du PC1 (p16p1) ARP Reply.

12. Réception ARP Reply sur l'interface p16p1 sur le PC2.

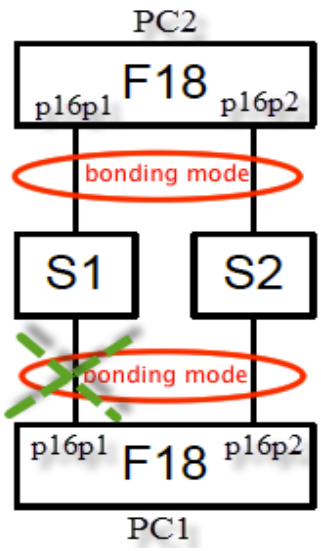
**Conclusion :** Le fonctionnement constaté à l'aide de la capture est typique du mécanisme pour le mode 0. On peut remarquer le changement d'interface lors de l'envoi de données sur les deux liens ainsi que la réception sur un lien. On observe aussi qu'à la fin de chaque échange, une nouvelle requête ARP est effectuée pour vérifier l'état de la liaison après 5 secondes.

**-- Vérification au niveau du cache du bridge**

```
[root@localhost admin]# brctl showmacs bridge0          -- Switch Linux 1
port no mac addr                is local?       ageing timer
  1     00:15:17:d6:e0:1c       yes                0.00
  2     00:15:17:d6:e0:1d       yes                0.00
  1     00:15:17:d6:e0:8c       no               185.73    -- MAC p16p1 PC1
  2     68:05:ca:11:b5:08       no               185.73    -- MAC p16p1 PC2
[root@localhost admin]# brctl showmacs bridge1          -- Switch Linux 2
port no mac addr                is local?       ageing timer
  1     00:15:17:d6:e0:8c       no               190.82    -- MAC p16p2 PC1
  2     68:05:ca:11:b5:08       no               190.82    -- MAC p16p2 PC2
  1     90:e2:ba:28:9b:6d       yes                0.00
  2     c8:60:00:73:ab:59       yes                0.00
```

Cette vue permet de faire la correspondance entre les ports et les adresses MAC de chaque interface. On remarque toujours qu'il y a une seule adresse MAC pour les deux interfaces de chaque PC.

### Tests de coupure de lien pour le mode 0




Dans cette partie, je vais étudier le comportement en cas de coupure d'un lien lors de la transmission d'un fichier déterminé alors que les PCs sont configurés en mode 0.

Je décide de prendre comme fichier test, une image .iso de Fedora 18. Ainsi, je vais pouvoir effectuer un contrôle d'intégrité du fichier après l'envoi par un checksum qui est disponible sur le site de Fedora[29].

Figure 16: Schéma coupure de lien mode 0

Je remets aussi en place les switch Netgear GS116E (S1 et S2) pour que ce scénario soit au plus près de la réalité.

**-- Envoi du fichier à l'aide de la commande scp vers PC2:**

```
[root@localhost test]# scp Fedora-18-x86_64-Live-Desktop.iso
admin@192.168.0.4:test
admin@192.168.0.4's password:
```

[29] Téléchargement image .iso et fichier checksum : http://www.fedora-fr.org/

**Remarque** : Pour pouvoir utiliser le transfert de fichiers avec la commande scp, il faut au préalable activer le service ssh dans le terminal (`# service sshd start`) pour l'ouverture des ports.

**Manipulation : J'enlève la prise RJ45 de l'interface p16p1.**

**-- Vérification de l'état des interfaces**

```
[root@localhost test]# cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v3.7.1 (April 27, 2011)

Bonding Mode: load balancing (round-robin)
MII Status: up
MII Polling Interval (ms): 100
Up Delay (ms): 0
Down Delay (ms): 0

Slave Interface: p16p1
MII Status: down                        -- L'interface n'est plus active
Speed: Unknown
Duplex: Unknown
Link Failure Count: 1                   -- Compteur de coupure de lien à 1
Permanent HW addr: 00:15:17:d6:e0:8c    -- L'adresse MAC par défaut reste
Slave queue ID: 0
Slave Interface: p16p2
MII Status: up
Speed: 1000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 00:15:17:d6:e0:8d
Slave queue ID: 0
```

L'interface p16p1 est détectée down après la coupure du lien et le compteur de coupure affiche 1.

**-- Vérification de la réception et de l'intégrité du fichier sur le PC2**

```
[root@localhost test]# ls
CHECKSUM  Fedora-18-x86_64-Live-Desktop.iso
[root@localhost test]# sha256sum -c CHECKSUM
Fedora-18-x86_64-Live-Desktop.iso: OK
```

Malgré la coupure du lien, le fichier a été transmis avec succès.

**-- Vérification au niveau du cache arp (avant-après coupure) sur PC2**

```
[root@localhost admin]# arp -D
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.0.2              ether   00:15:17:d6:e0:8c     C                     bond0
```

Le cache arp n'a pas changé avant-après coupure et après reconnexion du lien.

L'adresse MAC définie au départ dans bond0 (par défaut p16p1) ne change pas malgré la coupure du lien lui correspondant. Elle est indépendante et liée dans le mode 0 à l'interface virtuelle et non plus à l'interface physique.

**-- Analyse avec Wireshark**

L'analyse des échanges avec Wireshark est normale, tant au niveau IP que Mac Adresse, du côté envoi (PC1) comme du côté réception (PC2). Aucun signe de la perte de liaison! La gestion par bond0 de la coupure est totalement transparente. Cette information n'est pas transmise à travers le réseau.

**-- Diagramme des échanges arp-IP lors d'une coupure**

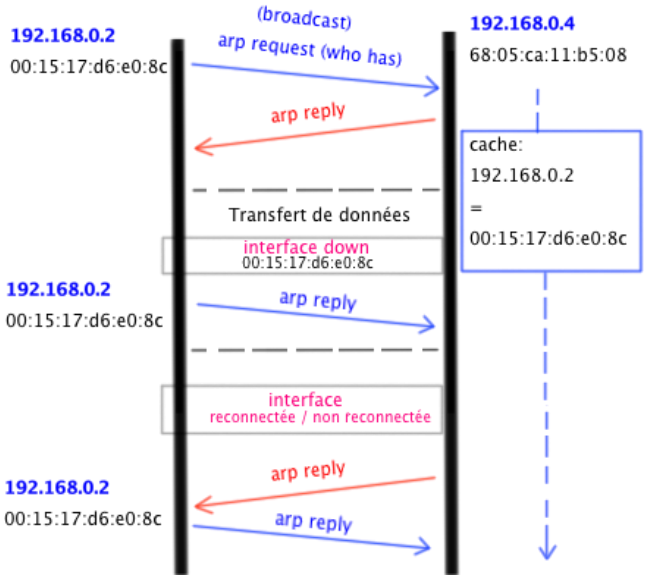
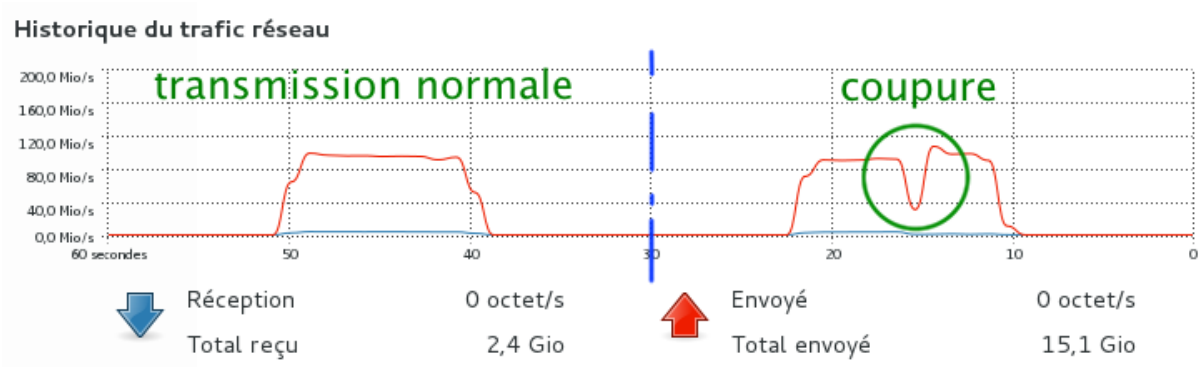
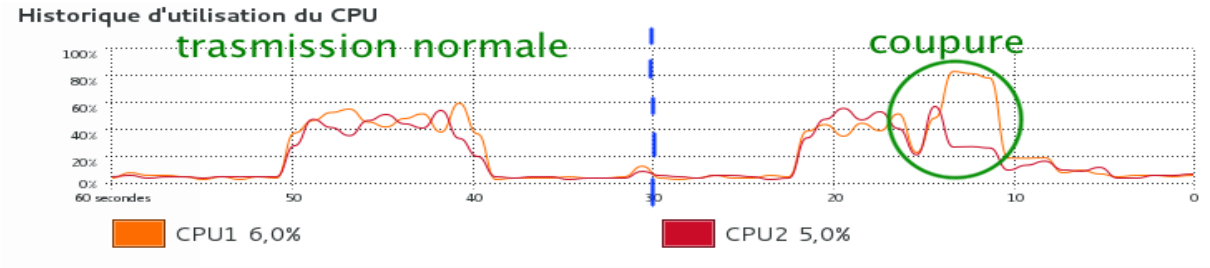



Figure 17: Diagramme d'échanges ARP -IP mode 0 lors d'une coupure

**-- Analyse avec le moniteur système de Fedora sur le PC1**




On observe très facilement le moment de déconnexion avec une baisse momentanée du débit. Cependant, il ne tombe pas à 0, car la deuxième carte fonctionne en simultané.




La charge au niveau CPU augmente au moment de la perte du lien. La gestion de l'incident par l'interface virtuelle bond0 utilise 80% des ressources CPU.

### Tests de débit binaire scénario 3

La méthodologie de mesures est la même que celle utilisée dans le scénario test 1 pour ce mode de bonding[30]. Les résultats sont semblables car le comportement est identique malgré la mise en parallèle des switchs.

#### *Résultats scénario 3 bonding mode 0*

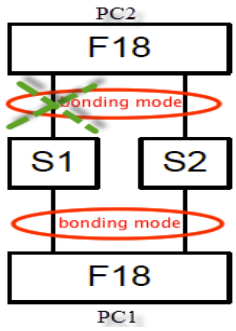
| Débit utile théorique maximal [Mbit/s] | Débit utile mesuré [Mbit/s] | Utilisation débit utile [%] | Temps de transfert [s] | Débit utile théorique maximal [Mo/s] | Débit utile mesuré [Mo/s] | Utilisation débit utile [%] | Utilisation CPU PC1 [%] | Utilisation mémoire RAM PC1 [%] |
|---|---|---|---|---|---|---|---|---|
| Calcul | Iperf | Calcul | Iperf | Calcul | Moniteur Système F18 | Calcul | Moniteur Système F18 | Moniteur Système F18 |
| 1942 | 1870 | 96,29 | 4,3 | 242,72 | 240 | 98,88 | 20 | 22,4 |

#### *Résultats lors de la coupure d'un lien sur PC1*

| Débit utile théorique maximal [Mbit/s] | Débit utile mesuré [Mbit/s] | Utilisation débit utile [%] | Temps de transfert [s] | Débit utile théorique maximal [Mo/s] | Débit utile mesuré [Mo/s] | Utilisation débit utile [%] | Utilisation CPU PC1 [%] | Utilisation mémoire RAM PC1 [%] |
|---|---|---|---|---|---|---|---|---|
| Calcul | Iperf | Calcul | Iperf | Calcul | Moniteur Système F18 | Calcul | Moniteur Système F18 | Moniteur Système F18 |
| 971 | 950 | 97,84 | 8,5 | 121,36 | 120 | 98,88 | 80 | 22,4 |

On remarque une augmentation au niveau des ressources CPU importante. Cependant, les autres données sont semblables à l'utilisation d'un seul lien Gbit/s.

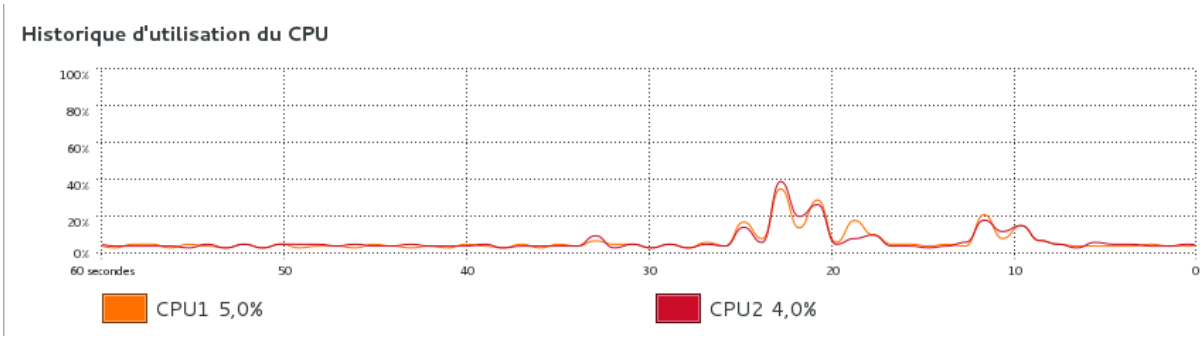
#### *Résultats lors de la coupure d'un lien sur PC2*

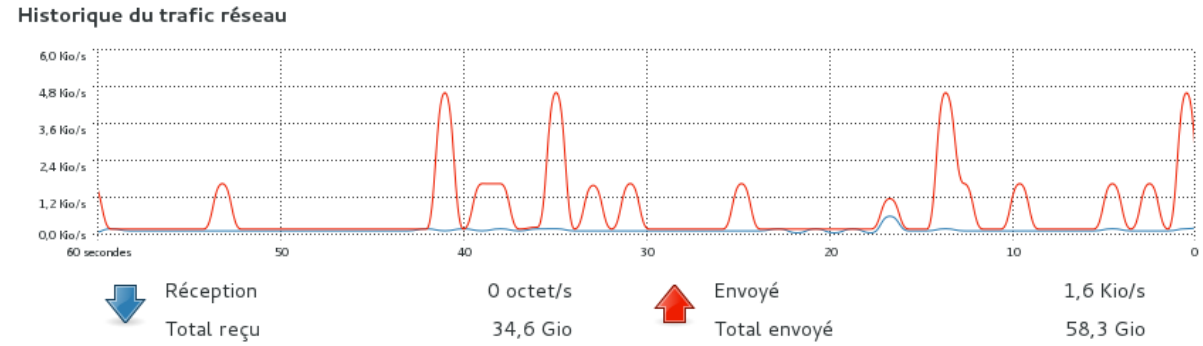



Une forte dégradation au niveau performances s'est produite. L'outil Iperf n'a pas réussi à effectuer une mesure convenable. Cependant, les acquisitions suivantes nous indiquent que la dégradation est causée par la retransmission au niveau des trames.

[30] Mode 0 Test de débit pour scénario 1

**-- Captures du moniteur système sur PC1 après la coupure du lien sur PC2**




Les ressources CPU utilisées sont de 40% et le débit binaire est médiocre.

Le PC1 en mode 0 effectue un envoi en séquentiel sur chacune de ses interfaces. Cependant, le lien sur le switch1 est coupé pour le PC2, donc les paquets n'aboutissent pas. Ils doivent être retransmis par l'autre interface du PC1.

Ce mécanisme est reproduit jusqu'à l'envoi de la totalité des données. La conséquence est une chute des performances.

Cette problématique permet de voir que la détection lors de la faille d'un des liens est seulement possible par le PC où se trouve la carte Ethernet concernée.

Cependant, on verra que l'on peut contourner ce problème en reliant les deux switchs entre eux par un lien. Cela permet d'effectuer une redirection en cas de panne sur un des liens des PCs configurés bonding mode 0.

Pour cela le switch standard doit au minimum implémenter le protocole Spanning Tree. Il permet la gestion des boucles réseau en bloquant ou activant certains ports du commutateur selon l'état des liens connectés au switch.

# Conclusion Finale Mode 0

## Fonctionnement du mode 0 (balance – round robin)

C'est le mode avec équilibrage de charge. Les données sont transmises séquentiellement de la première interface active à la dernière (de façon continue et cyclique) pour augmenter le nombre de paquets envoyés et donc améliorer le débit binaire.

Après avoir observé le mode 0 à l'aide de divers scénarios dans la partie Tests du mode 0 [31], je peux décrire son fonctionnement.

L'interface virtuelle bond0 (MASTER) est la seule à posséder une adresse IP. Les autres interfaces (SLAVES) faisant partie de la configuration n'ont pas d'IP.

Une seule adresse MAC pour toutes les interfaces faisant partie de la configuration (bond0, p16p1, p16p2). Au départ, la MAC adresse choisie par défaut est celle de la première interface physique (p16p1). Elle ne changera plus dans le mode 0 et ceci même en cas de coupure d'un lien comme j'ai pu le voir à l'aide des caches ARP.

On peut donc dire avec certitude que les adresses MACs des interfaces physiques sont modifiées, gérées et contrôlées par l'interface virtuelle bond0.

Cette spécificité d'une adresse MAC unique oblige à l'utilisation d'un switch évolué compatible EtherChannel [32] (comme le Cisco SG-300-28) lorsque l'on veut connecter tous les liens à un seul commutateur (voir scénario 2)[33].

Cependant, on peut utiliser des switchs standards en connectant un lien par commutateur (voir scénario 3)[34].

Au niveau des échanges, j'ai pu observer un mécanisme de load-balancing lors d'un envoi de données. La réception s'effectuée de la même sorte d'après les scénarios que j'ai pu mettre en œuvre.

Considérons le cas où l'on a configuré deux PCs avec le mode 0. Lors du premier envoi, avec le cache ARP vide, la première trame passe par le lien par défaut et est reçue par le lien par défaut sur l'autre PC.

---

[31] Partie Tests mode 0 : Scénario de test 1 : Deux PCs Fedora 18 avec bonding
[32] Explication sur l'agrégation de liens avec EtherChannel : http://fr.wikipedia.org/wiki/EtherChannel
[33] Partie Tests mode 0 : Scénario test 2 : 2 PCs Fedora 18 et un Switch
[34] Partie Tests mode 0 : Scénario test 3 : 2 PCs Fedora 18 et 2 Switchs

Si on effectue d’autres échanges directement, on sait quel est le lien qui va débuter ou recevoir la première trame à l’aide du cache, en affichant son résultat périodiquement.

On peut ainsi prévoir à la fin de l’échange quel lien va être pris en compte pour l’échange suivant.

Au niveau du débit binaire mesure, j’ai obtenu un maximum de 1.87 Gbit/s lors de l’utilisation de 2 liens Gbit/s.

# *Etude du mode 6*
# *Balance – Adaptive Load Balancing*

# Réalisation

## Configuration mode 6

Afin d'effectuer les tests pour les divers scénarios qui vont suivre, je configure les deux PCs clients Fedora 18 selon la marche à suivre de la partie Configuration Bonding [35] se trouvant dans ce chapitre.

La seule modification concerne la partie :

**-- Configuration de l'interface bond0**

```
# vi /etc/sysconfig/network-scripts/ifcfg-bond0
DEVICE="bond0"                          -- Nom logique de l'interface
ONBOOT="yes"                            -- Interface activée au démarrage
BOOTPROTO="none"                        -- Protocole utilisé au démarrage
IPADDR="192.168.0.2"
NETMASK="255.255.255.0"
GATEWAY="192.168.0.1"
USERCTL="no"                            -- Contrôle interface user non-root
BONDING_OPTS="mode=6 miimon=100"        -- Choix du mode et autres options
```

# Tests

## Scénario de test 1 : Deux PCs Fedora 18 avec bonding mode 6

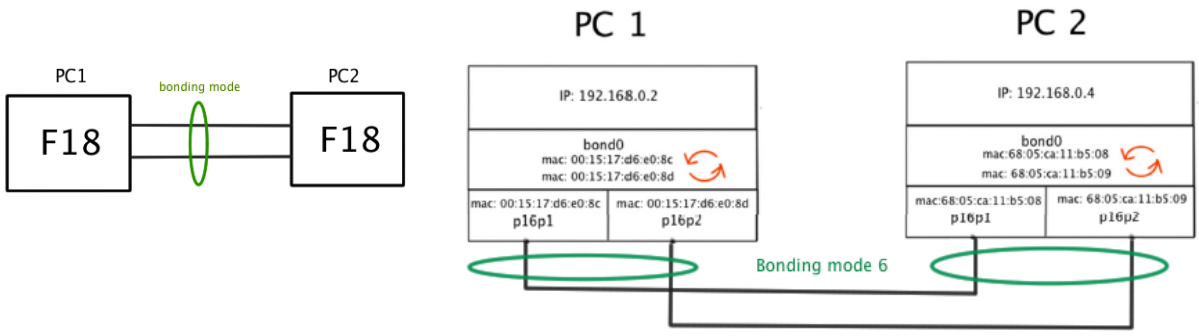



Figure 18: Schéma scénario test 1 mode 6

L'objectif dans cette partie est d'étudier le bonding en mode 6 au niveau fonctionnement avec 2 interfaces physiques (Gbit/s) sur 2 PCs sur Fedora 18 Gnome. Les deux PCs sont directement connectés entre eux.

[35] Chapitre I : Bonding sur Fedora 18 Réalisation - Configuration Bonding

**-- Vérification de la configuration du mode 6**

```
 [root@localhost admin]# cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v3.7.1 (April 27, 2011)

Bonding Mode: adaptive load balancing -- Mode 6
Primary Slave: None
Currently Active Slave: p16p1   -- Interface qui est active
MII Status: up
MII Polling Interval (ms): 100
Up Delay (ms): 0
Down Delay (ms): 0

Slave Interface: p16p1
MII Status: up
Speed: 1000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 68:05:ca:11:b5:08 -- Adresse MAC interface p16p1
Slave queue ID: 0

Slave Interface: p16p2
MII Status: up
Speed: 1000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 68:05:ca:11:b5:09 -- Adresse MAC interface p16p2
Slave queue ID: 0
```

**Conclusion :** Une seule interface active au départ p16p1.

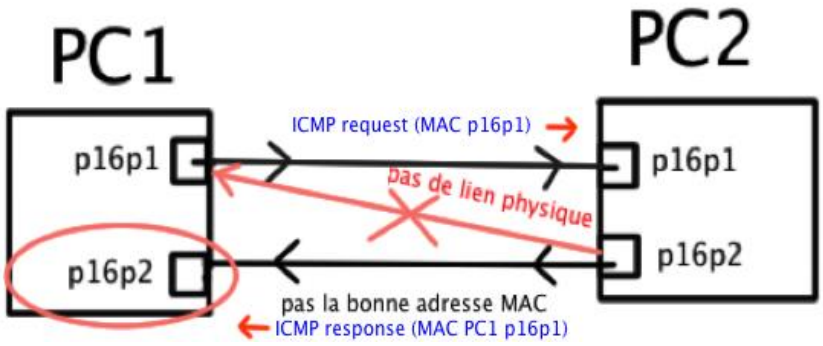
**-- Vérification de MAC adresse des interfaces**

```
[root@localhost admin]# ifconfig

bond0: flags=5187<UP,BROADCAST,RUNNING,MASTER,MULTICAST>  mtu 1500
        inet 192.168.0.4  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::6a05:caff:fe11:b508  prefixlen 64  scopeid 0x20<link>
        ether 68:05:ca:11:b5:08  txqueuelen 0  (Ethernet)
        RX packets 1  bytes 64 (64.0 B)
        RX errors 0  dropped 1  overruns 0  frame 0
        TX packets 187  bytes 14216 (13.8 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

p16p1: flags=6211<UP,BROADCAST,RUNNING,SLAVE,MULTICAST>  mtu 1500
        ether 68:05:ca:11:b5:08  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 112  bytes 9416 (9.1 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 16  memory 0xfea80000-feaa0000
```

```
p16p2: flags=6211<UP,BROADCAST,RUNNING,SLAVE,MULTICAST>  mtu 1500
        ether 68:05:ca:11:b5:09  txqueuelen 1000  (Ethernet)
        RX packets 1  bytes 64 (64.0 B)
        RX errors 0  dropped 1  overruns 0  frame 0
        TX packets 75  bytes 4800 (4.6 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Chaque interface a sa propre adresse MAC et une seule interface est active au départ. L'interface virtuelle bond0 prend la MAC adresse de l'interface active. C'est l'interface p16p1 (Currently Active Slave) avec l'adresse MAC lui correspondant qui répondra à la première demande d'un PC client. Si une deuxième demande est effectuée, l'interface p16p2 passera elle aussi en mode actif pour répondre à la sollicitation d'un autre PC client.

### Tests avec la commande ping pour le mode 6




L'interface Currently Active Slave sur les 2 PCs sont les mêmes, soit p16p1.

Lors d'un ping sur le PC1 par l'interface p16p1, l'interface qui est reliée physiquement sur le PC2 reçoit la demande (ICMP request). Cependant, le PC2 envoie la réponse (ICMP response) sur son autre interface. Le ping ne peut donc pas fonctionner, car l'interface P16p2 du PC2 répond à la mauvaise adresse MAC. De plus l'interface n'est pas relie physiquement à la bonne carte Ethernet pour pouvoir répondre à la demande ICMP request.

En conclusion, le mode 6 a besoin pour fonctionner d'un switch. Celui-ci va permettre d'avoir accès à tous les liens lors des changements d'interface en envoi et réception.

On va pouvoir observer avec plus de précision ce mécanisme dans le scénario de test 2 qui suit ou un switch est intercalé entre les deux PCs.

**-- Résultat du Ping** *(scénario 1/mode6/ping)*

```
[root@localhost admin]# ping 192.168.0.4
PING 192.168.0.4 (192.168.0.4) 56(84) bytes of data.
From 192.168.0.2 icmp_seq=1 Destination Host Unreachable
From 192.168.0.2 icmp_seq=2 Destination Host Unreachable
. . . . . .

--- 192.168.0.4 ping statistics ---
11 packets transmitted, 0 received, +8 errors, 100% packet loss, time
10001ms
```

## Scénario test 2 : 2 PCs Fedora 18 et un Switch

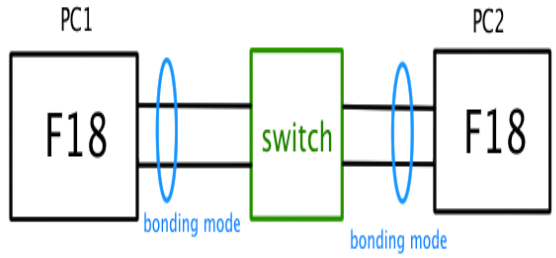



Figure 19: Schéma scénario test 2 mode 6

L'objectif de ce nouveau scénario est d'étudier le mécanisme d'équilibrage de charge et de débit binaire du mode 6.

La configuration est composée de 2 PCs Fedora 18 avec 2 interfaces physiques (Gbit/s) reliées à un switch Netgear GS116E 16 port.

Je vais également pouvoir schématiser le fonctionnement IP-Ethernet → ARP lors d'un échange.

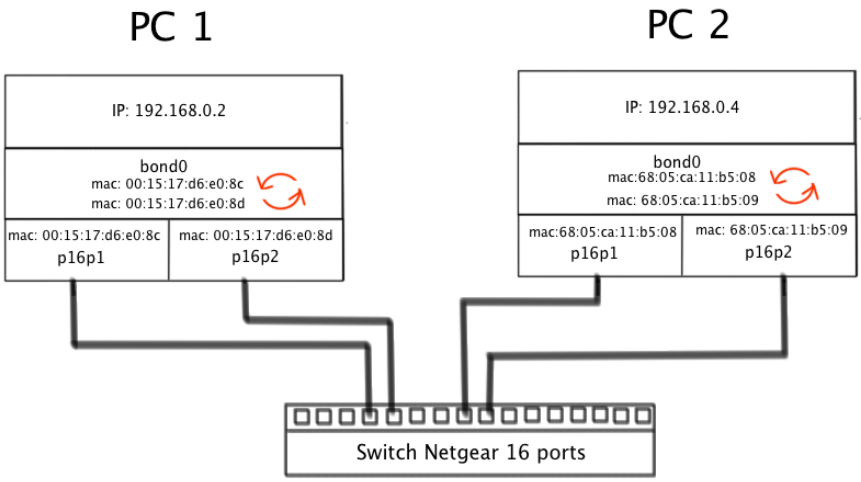


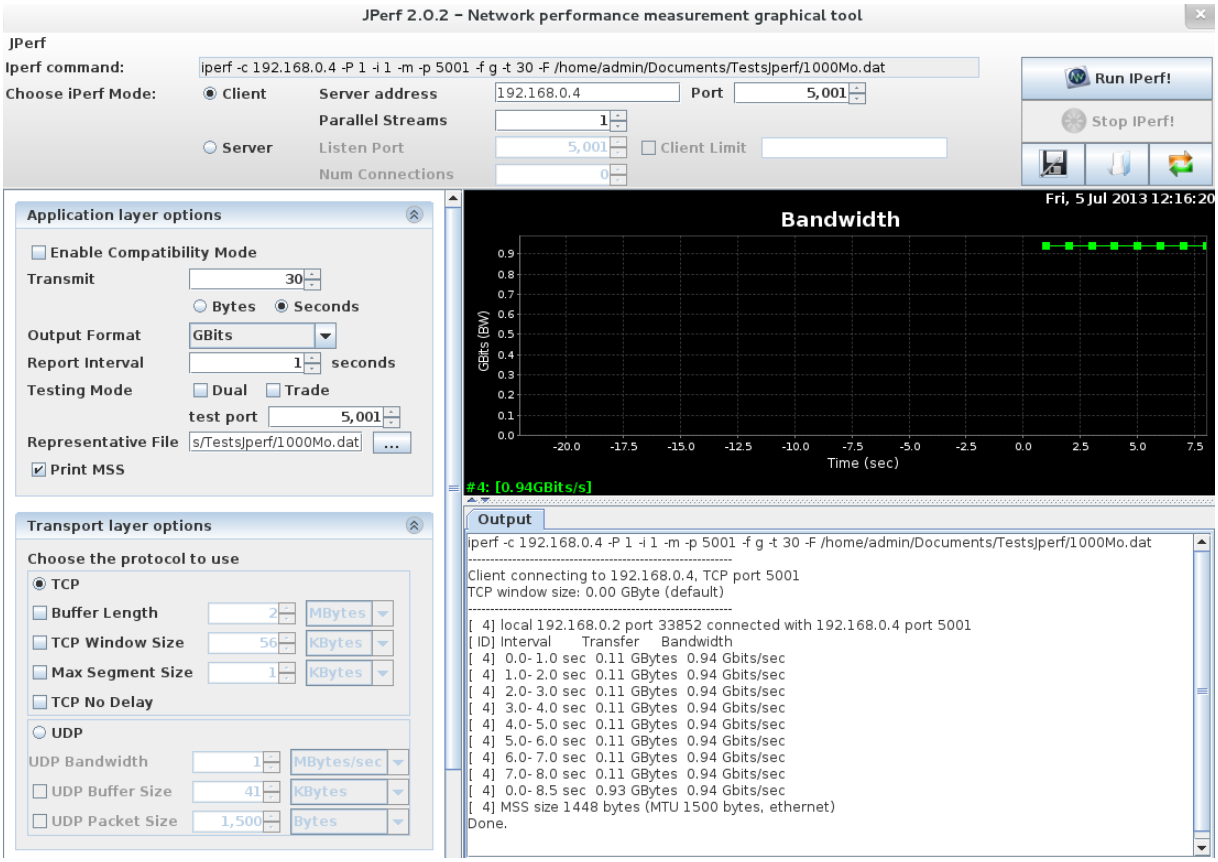

Figure 20: Schéma de principe mode 6

### Tests avec la commande Iperf

La méthodologie est exactement la même que celle utilisée dans la partie tests du mode 0[36].

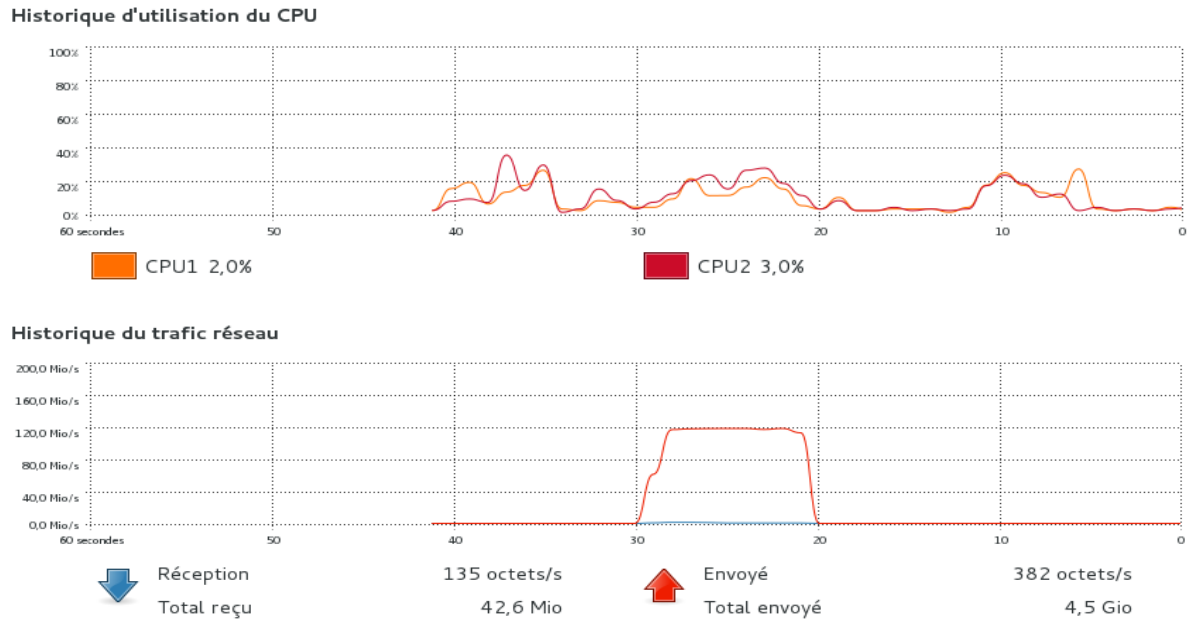
[36] Mode 0 Test de débit pour scénario 1

**-- Résultat avec Iperf lors de l'envoi d'un fichier de 1Go du PC1 client vers PC2 serveur**

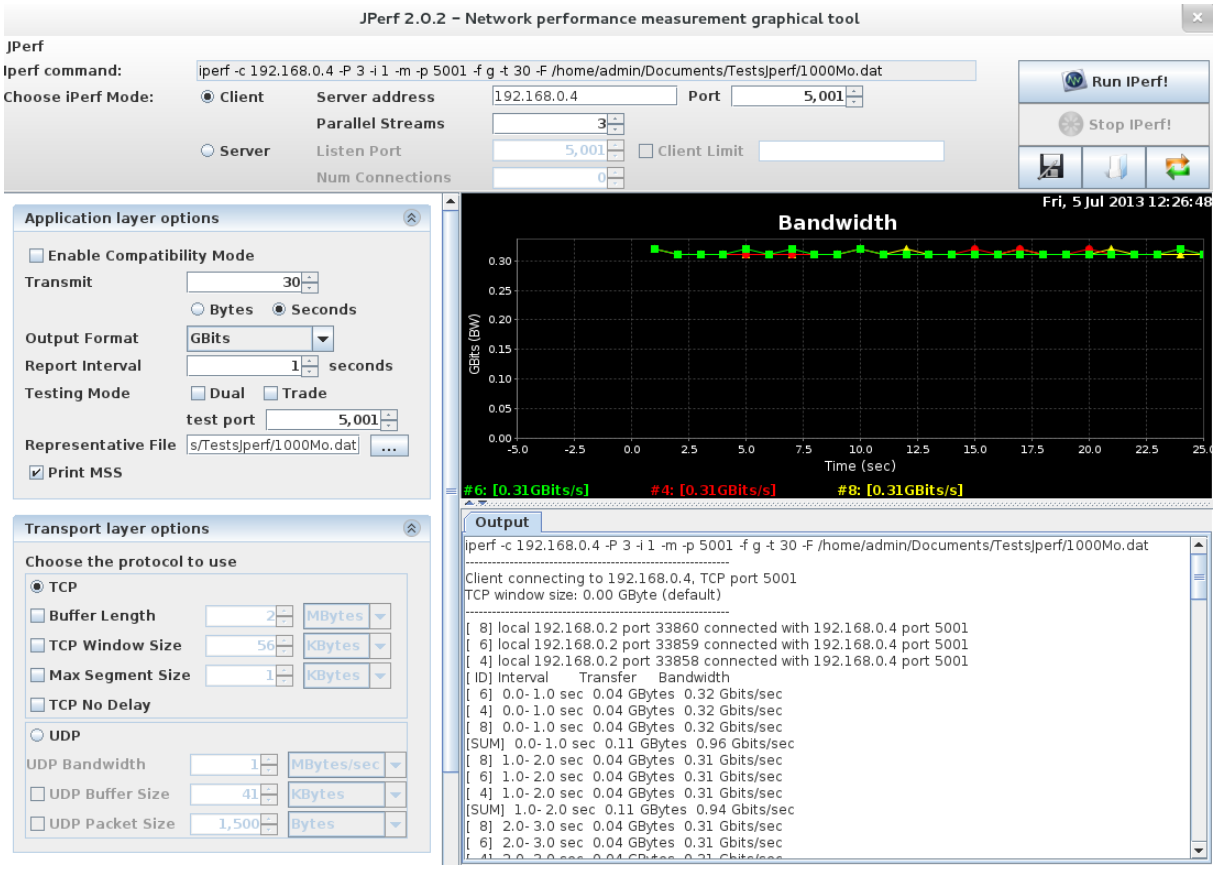



Le débit binaire correspond à une seule interface active lors d'une demande du PC1 client, soit 943 Mbits/s.

**-- Capture du moniteur de ressources Fedora sur PC1**

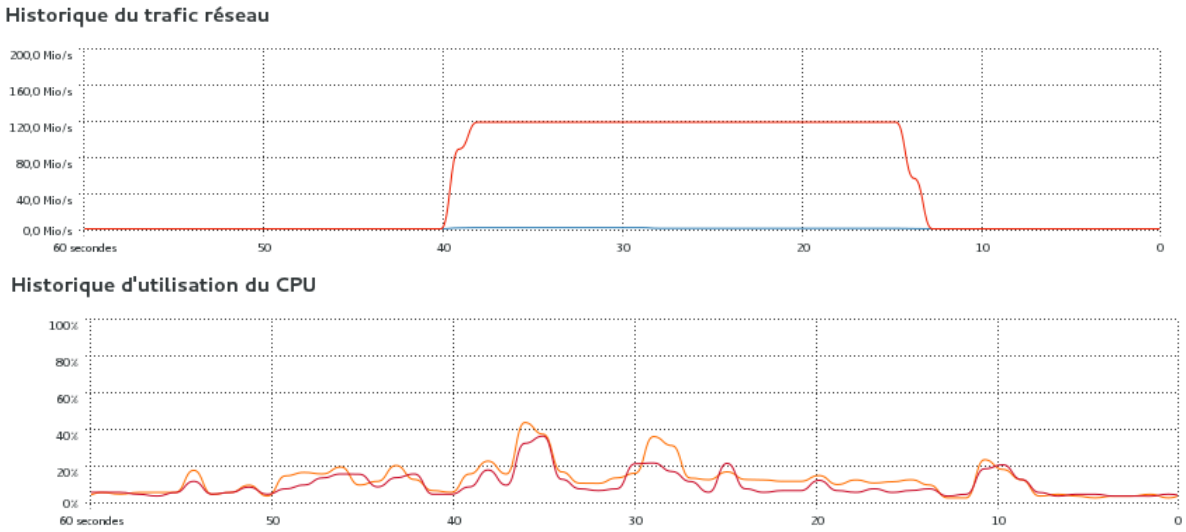


**-- Résultat avec Iperf lors de 3 transferts du même fichier (1Go) effectués simultanément sur le PC1 vers le PC2**




**Conclusion :** Dans le cas de 3 connections simultanées sur le même serveur, le débit binaire du lien Gbit/s est partagé entre les 3. L'interface virtuelle bond0 effectue la gestion, par le biais du mécanisme ARP, en attribuant quelle interface effectue l'échange avec quelle MAC adresse de destination.

**-- Capture du moniteur de ressources Fedora sur PC1**



On note une augmentation de l'utilisation CPU à 40% durant les transferts et l'utilisation du débit binaire dans sa totalité à 120 Mo/s mais sur une longueur de temps plus grande, soit 25,5 secondes. Le débit utile transmit est de 2.79 GBytes.

**-- Résultat avec Iperf lors de 2 transferts simultanés de 2 PCs (deux IP, deux MAC adresses différentes)**

```
[root@localhost admin]# iperf -c 192.168.0.4 -P 3 -i 1 -m -p 5001 -f g -t
30 -F /home/admin/Documents/TestsJperf/1000Mo.dat
------------------------------------------------------------

Client connecting to 192.168.0.4, TCP port 5001
TCP window size: 22.9 KByte (default)
------------------------------------------------------------
[  3] local 192.168.0.2 port 53424 connected with 192.168.0.4 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-8.5 sec  940 MBytes   952 Mbits/sec
```

```
[root@localhost admin]# iperf -c 192.168.0.4 -P 3 -i 1 -m -p 5001 -f g -t
30 -F /home/admin/Documents/TestsJperf/1000Mo.dat
------------------------------------------------------------

Client connecting to 192.168.0.5, TCP port 5001
TCP window size: 22.9 KByte (default)
------------------------------------------------------------
[  3] local 192.168.0.2 port 51169 connected with 192.168.0.5 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-8.5 sec  937 MBytes   953 Mbits/sec
```

**Conclusion :** Une interface dédiée par client en envoi et en réception à la fin de la négociation ARP.

### *Tableau des résultats pour le scénario 2 bonding mode 6*

**-- Mode de fonctionnement normal**

| Débit utile théorique maximal [Mbit/s] | Débit utile mesuré [Mbit/s] | Utilisation débit utile [%] | Temps de transfert [s] | Débit utile théorique maximal [Mo/s] | Débit utile mesuré [Mo/s] | Utilisation débit utile [%] | Utilisation CPU PC1 [%] | Utilisation mémoire RAM PC1 [%] |
|---|---|---|---|---|---|---|---|---|
| Calcul | Iperf | Calcul | Iperf | Calcul | Moniteur Système F18 | Calcul | Moniteur Système F18 | Moniteur Système F18 |
| 1942 | 943 | 48,56 | 8,5 | 121,36 | 120 | 98,88 | 30 | 22 |

**-- Lors de 3 transferts de 1 fichier de 1Go simultanés depuis le PC1 vers le PC2**

| Débit utile théorique maximal [Mbit/s] | Débit utile mesuré par connexion [Mbit/s] | Utilisation débit utile par connexion [%] | Débit utile total mesuré [Mbit/s] | Temps de transfert [s] | Débit utile théorique maximal [Mo/s] | Débit utile mesuré [Mo/s] | Utilisation débit utile [%] | Utilisation CPU PC1 [%] | Utilisation mémoire RAM PC1 [%] |
|---|---|---|---|---|---|---|---|---|---|
| Calcul | Iperf | Calcul | Iperf | Iperf | Calcul | Moniteur Système F18 | Calcul | Moniteur Système F18 | Moniteur Système F18 |
| 971 | 330 | 33,99 | 937 | 25,5 | 121,36 | 120 | 98,88 | 30 | 22 |

**-- Lors de 2 transferts de 1 fichier de 1Go depuis 2 PCs clients avec différentes en IP et MAC adresses**

Chaque client se voit attribuer un lien et les performances sont celles d'une liaison Gbit/s comme dans le mode normal.

### *Mise en place d'un Switch Linux*

Afin d'avoir plus de précision par rapport au fonctionnement ARP dans le mode 6, je décide à nouveau d'utiliser la fonction switch Linux comme dans le mode 0.

**Ainsi, je décide de mettre en place un bridge[37] sur le PC3 avec 4 interfaces Ethernet Gbit/s.**

**Ce bridge aura la même fonction qu'un switch à 4 ports.**

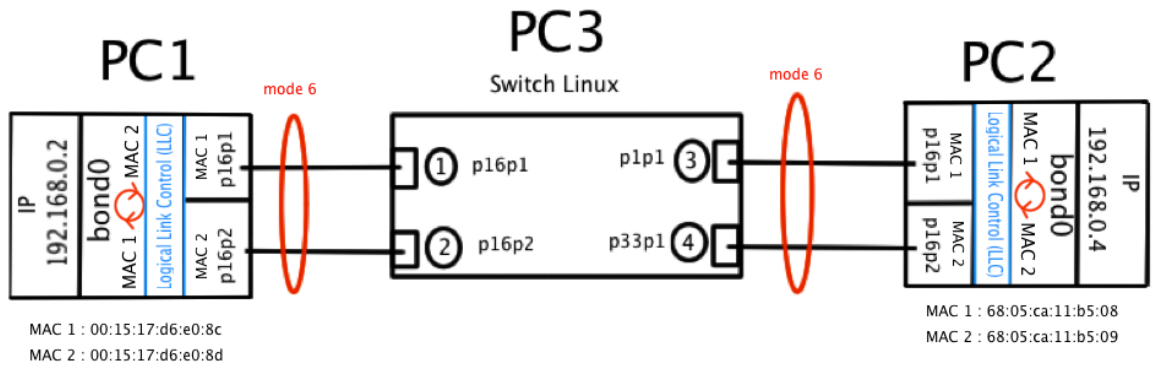



Figure 21: Schéma test mode 6 avec 2 PCs 1 switch Linux

### *Vérification de la table arp du bridge*

Dans ce mode, toutes les interfaces actives on leur propre adresse MAC. Dans cette partie, je vais vérifier au niveau du cache ARP du bridge (switch), le nombre d'interfaces visibles des PCs connectés, ainsi que la correspondance entre les adresses MACs.

[37] A.6 Configuration d'un bridge sur Fedora 18

**-- Commande à effectuer sur le bridge pour la table arp**

```
[root@localhost admin]# brtcl showmacs bridge0
```

**-- Résultat au départ sans aucun échange avant**

```
port no      mac addr    is local?  ageing timer

  1     00:15:17:d6:e0:1c   yes        0.00   -- Adresse MAC bridge (p16p1)
  2     00:15:17:d6:e0:1d   yes        0.00   -- Adresse MAC bridge (p16p2)
  1     00:15:17:d6:e0:8c   no         0.00   -- Adresse MAC PC1 (p16p1)
  2     00:15:17:d6:e0:8d   no         0.00   -- Adresse MAC PC1 (p16p2)
  3     68:05:ca:11:b5:08   no         0.00   -- Adresse MAC PC2 (p16p1)
  4     68:05:ca:11:b5:09   no         0.00   -- Adresse MAC PC2 (p16p2)
  3     90:e2:ba:28:9b:6d   yes        0.00   -- Adresse MAC bridge (p1p1)
  4     c8:60:00:73:ab:59   yes        0.00   -- Adresse MAC bridge (p33p1)
```

**Conclusion :** Chaque interface est visible individuellement par le bridge et chacune possède sa propre adresse MAC. Cette vue est permanente à cause de l'envoi sur chaque interface esclave de 3 trames LLC (Logical Link Control)[38] par seconde, comme on peut le voir dans cette capture.

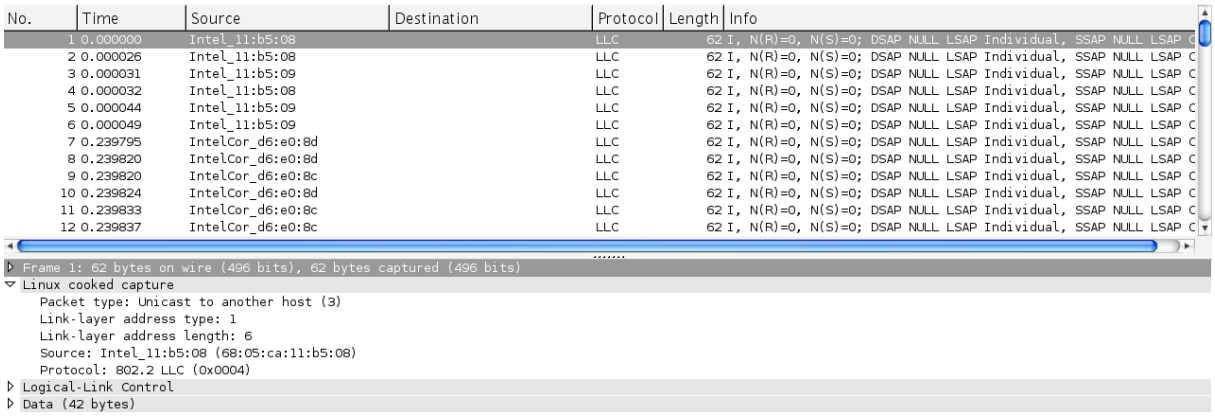



**Figure 22: Capture Wireshark protocole Logical Link Control (LLC)**

On constate que le protocole LLC permet de maintenir à jour le cache ARP au niveau du bridge (switch). Les 3 trames LLC, soit 60 bytes par trame, sont envoyées par le PC1 et le PC2 sur chacune de leurs interfaces physiques.

Si l'on observe plus attentivement les particularités de ses trames LLC, on voit que chaque trame utilise l'adresse MAC de l'interface esclave comme source et comme destination. Cependant, ces paquets LLC ne mettent pas à jour le cache ARP du PC en question car du point de vue de celui-ci, chaque

[38] Explication protocole LLC :
http://www.lapinbleu.ch/reseaux/ethernet/code/llc.htm
http://polymorphe.free.fr/cours/reseaux/lan/liaison8023.pdf

paquet est envoyé et reçu par chacune de ses interfaces physiques.

La mise à jour du cache ARP des PCs s'effectue par le biais des demandes de connexion, donc par l'envoi de requêtes ARP depuis un client ou depuis l'hôte qui souhaite transmettre des données.

### *Test avec TCPdump sur les différentes interfaces du switch Linux*

Dans le but de comprendre le mécanisme du mode 6, je procède à une acquisition sur toutes les interfaces du switch Linux. J'effectué comme pour le mode précédent un ping sur le PC1 vers le PC2.

Voici une mise en forme du résultat au niveau des échanges ARP et ICMP après avoir effectuée un filtrage sur le protocole LLC.

**-- Résultat de la capture TCPdump avec les interfaces concernées du bridge et direction de l'échange**

| Interface | Sens | Temps | n° | Action |
|---|---|---|---|---|
| p1p1 | o/i | 15:58:00.413611 | | P 68:05:ca:11:b5:08 (oui Unknown) 802.2, length 62: LLC, dsap Null (0x00) Individual, ssap Null (0x00) Command, ctrl 0x0000 : Information, send seq 0, rcv seq 0, Flags [Command], length 46 |
| p1p1 | o/i | 15 :58 :00.413622 | | P 68 :05 :ca :11 :b5 :08 (oui Unknown) 802.2, length 62 : LLC, dsap Null (0x00) Individual, ssap Null 0x00) Command, ctrl 0x0000 : Information, send seq 0, rcv seq 0, Flags [Command], length 46 |
| p1p1 | o/i | 15:58:00.413626 | | P 68:05:ca:11:b5:08 (oui Unknown) 802.2, length 62: LLC, dsap Null (0x00) Individual, ssap Null (0x00) Command, ctrl 0x0000: Information, send seq 0, rcv seq 0, Flags [Command], length 46 |
| | | ….. | | |
| p16p1 | in | 15:58:00.414883 | 1 | B 00:15:17:d6:e0:8c (oui Unknown) ethertype ARP (0x0806), length 62: Request who-has 192.168.0.4 tell 192.168.0.2, length 46 |
| p33p1 | out | 15:58:00.414897 | 2 | Out 00:15:17:d6:e0:8c (oui Unknown) ethertype ARP (0x0806), length 62: Request who-has 192.168.0.4 tell 192.168.0.2, length 46 |
| p1p1 | out | 15:58:00.414906 | 3 | Out 00:15:17:d6:e0:8c (oui Unknown) ethertype ARP (0x0806), length 62: Request who-has 192.168.0.4 tell 192.168.0.2, length 46 |
| p16p2 | out | 15:58:00.414913 | 4 | Out 00:15:17:d6:e0:8c (oui Unknown) ethertype ARP (0x0806), length 62: Request who-has 192.168.0.4 tell 192.168.0.2, length 46 |
| p33p1 | in | 15:58:00.414991 | 5 | P 68:05:ca:11:b5:09 (oui Unknown) ethertype ARP (0x0806), length 62: Reply 192.168.0.4 is-at 68:05:ca:11:b5:09 (oui Unknown), length 46 |
| p16p1 | out | 15:58:00.414998 | 6 | Out 68:05:ca:11:b5:09 (oui Unknown) ethertype ARP (0x0806), length 62: Reply 192.168.0.4 is-at 68:05:ca:11:b5:09 (oui Unknown), length 46 |
| p16p1 | in | 15:58:00.415130 | 7 | P 00:15:17:d6:e0:8c (oui Unknown) ethertype IPv4 (0x0800), length 100: 192.168.0.2 > 192.168.0.4: ICMP echo request, id 2095, seq 1, length 64 |
| p16p2 | out | 15:58:00.415136 | 8 | Out 00:15:17:d6:e0:8c (oui Unknown) ethertype IPv4 (0x0800), length 100: 192.168.0.2 > 192.168.0.4: ICMP echo request, id 2095, seq 1, length 64 |
| p1p1 | in | 15:58:00.415298 | 9 | P 68:05:ca:11:b5:08 (oui Unknown) ethertype IPv4 (0x0800), length 100: 192.168.0.4 > 192.168.0.2: ICMP echo reply, id 2095, seq 1, length 64 |
| p16p1 | out | 15:58:00.415304 | 10 | Out 68:05:ca:11:b5:08 (oui Unknown) ethertype IPv4 (0x0800), length 100: 192.168.0.4 > 192.168.0.2: ICMP echo reply, id 2095, seq 1, length 64 |
| p16p1 | o/i | 15:58:00.556854 | 11 | P 00:15:17:d6:e0:8c (oui Unknown) 802.2, length 62: LLC, dsap Null (0x00) Individual, ssap Null (0x00) Command, ctrl 0x0000: Information, send seq 0, rcv seq 0, Flags [Command], length 46 |

| | | | | |
|---|---|---|---|---|
| | | ……. | | |
| p16p2 | in | 15:58:02.456852 | 12 | P 00:15:17:d6:e0:8d (oui Unknown) ethertype ARP (0x0806), length 62: Reply 192.168.0.2 is-at 00:15:17:d6:e0:8d (oui Unknown), length 46 |
| p33p1 | out | 15:58:02.456868 | | Out 00:15:17:d6:e0:8d (oui Unknown) ethertype ARP (0x0806), length 62: Reply 192.168.0.2 is-at 00:15:17:d6:e0:8d (oui Unknown), length 46 |
| p16p2 | in | 15:58:02.456876 | | P 00:15:17:d6:e0:8d (oui Unknown) ethertype ARP (0x0806), length 62: Reply 192.168.0.2 is-at 00:15:17:d6:e0:8d (oui Unknown), length 46 |
| p33p1 | out | 15:58:02.456881 | | Out 00:15:17:d6:e0:8d (oui Unknown) ethertype ARP (0x0806), length 62: Reply 192.168.0.2 is-at 00:15:17:d6:e0:8d (oui Unknown), length 46 |
| p33p1 | in | 15:58:02.513553 | | P 68:05:ca:11:b5:09 (oui Unknown) ethertype ARP (0x0806), length 62: Reply 192.168.0.4 is-at 68:05:ca:11:b5:09 (oui Unknown), length 46 |
| p16p2 | out | 15:58:02.513566 | | Out 68:05:ca:11:b5:09 (oui Unknown) ethertype ARP (0x0806), length 62: Reply 192.168.0.4 is-at 68:05:ca:11:b5:09 (oui Unknown), length 46 |
| p33p1 | in | 15:58:02.513574 | | P 68:05:ca:11:b5:09 (oui Unknown) ethertype ARP (0x0806), length 62: Reply 192.168.0.4 is-at 68:05:ca:11:b5:09 (oui Unknown), length 46 |
| p16p2 | out | 15:58:02.513577 | | Out 68:05:ca:11:b5:09 (oui Unknown) ethertype ARP (0x0806), length 62: Reply 192.168.0.4 is-at 68:05:ca:11:b5:09 (oui Unknown), length 46 |
| | | ..... | | |
| p16p1 | o/i | 15:58:06.556876 | | P 00:15:17:d6:e0:8c (oui Unknown) 802.2, length 62: LLC, dsap Null (0x00) Individual, ssap Null (0x00) Command, ctrl 0x0000: Information, send seq 0, rcv seq 0, Flags [Command], length 46 |

**-- Schéma de fonctionnement mode 6 par rapport à la capture TCPdump** *(Scénario 2/mode6)*

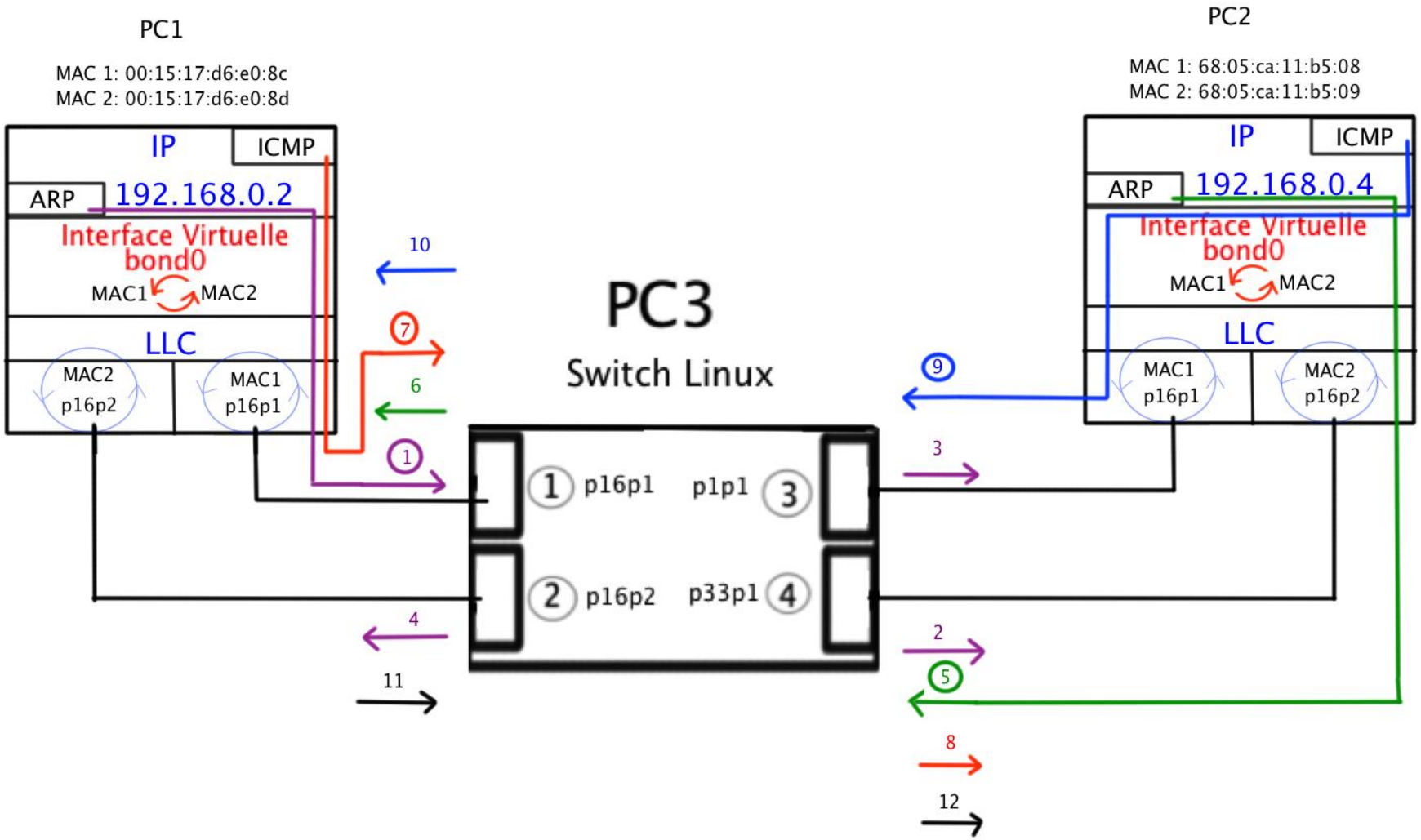



Figure 23: Schéma de fonctionnement mode 6

**Description des différentes étapes :**

**-- Requêtes LLC de toutes les interfaces (PC1 et PC2) vers elles-mêmes durant tout l'échange**

**-- Partie ARP REQUEST PC1 --> PC2**

1. ARP Request venant du PC1 interface p16p1. MAC source 00:15:17:d6:e0:8c, MAC destination broadcast. Request who-has 192.168.0.4 tell 192.168.0.2.

2. 3. 4 Broadcast de l'ARP Request sur toutes les interfaces du switch Linux.

**-- Partie ARP REPLY PC1 <-- PC2**

5. Envoi ARP Reply depuis l'interface p16p2 du PC2. 192.168.0.4 is-at 68:05:ca:11:b5:09.

6. Réception de l'ARP Reply par l'interface qui a effectué la demande sur le PC1 (p16p1).

**-- Partie ICMP ECHO REQUEST PC1 --> PC2**

7. ICMP Echo Request de PC1 vers PC2. Depuis la même interface (p16p1).

8. Réception ICMP Echo Request par l'interface p16p2 du PC2 qui a répondu à l'ARP reply précédemment.

**-- Partie ICMP ECHO REPLY PC2 --> PC1**

9. Envoi ICMP Reply par l'autre interface du PC2 (p16p1).

10. Réception ICMP Reply par l'interface qui a effectué le request sut le PC1 (p16p1).

**-- Après 0.040 ms un nouveau ARP REPLY**

Des interfaces p16p2 du PC1 et P16p33 du PC2. Mise à jour du cache ARP.

**Conclusion** : C'est un Load Balancing effectué par rapport aux interfaces grâce aux réponses ARP renvoyées par le PC2, serveur. Voici le fonctionnement observé :

Le PC1 client fait un broadcast sur le réseau et demande qui est l'adresse IP du PC2 fonctionnant en mode 6.

Ce broadcast remonte de p16p1 ou p16p2 vers bond0 (l'interface virtuelle).

Bond0 répondra à cette requête (who has) après avoir vérifié l'utilisation des cartes réseau.

La carte qui sera la moins utilisée sera choisie et renverra la réponse au broadcast et donc son adresse MAC.

Si aucune carte n'est utilisée à ce moment, la carte choisie sera la « Currently Active Slave » (par défaut la p16p1).

Le client se connectera à la carte réseau la moins utilisée en utilisant l'adresse MAC envoyée par le PC2.

Par la suite, un autre PC pourrait se voir répondre par la deuxième carte réseau physique (qui sera moins utilisée que la précédente). Ce deuxième PC se connectera alors via celle-ci pour transmettre. Les deux cartes seront alors activées en même temps, mais de façon indépendante.

Ce mécanisme est valable tant en envoi qu'en réception.

### *Tests de coupure de lien pour le mode 6*

Dans cette partie, je vais observer le comportement en cas de coupure d'un lien lors de la transmission d'un fichier déterminé (image ISO de Fedora 18). Je décide de remettre le switch Netgear GS116E 16 ports pour que ce scénario soit au plus près de la réalité.

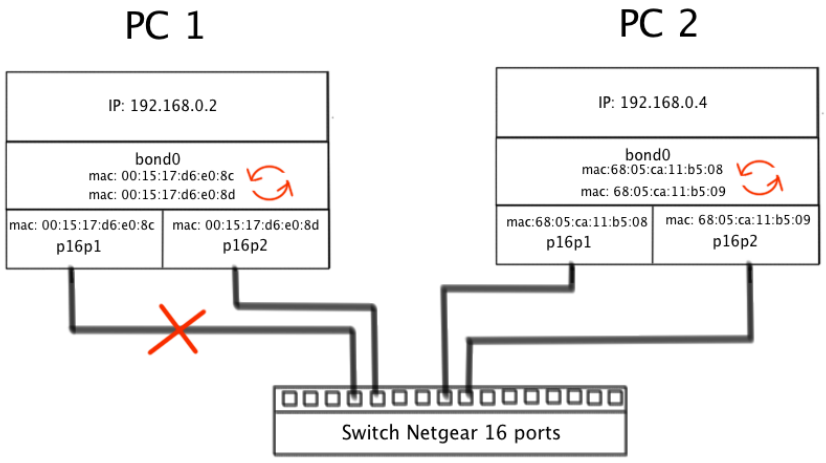



Figure 24: Coupure de lien mode 6

**-- Envoi du fichier à l'aide de la commande scp vers PC2:**

```
[root@localhost test]# scp Fedora-18-x86_64-Live-Desktop.iso
admin@192.168.0.4:test
```

**Manipulation : J'enlève la prise RJ45 de l'interface p16p1.**

**-- Vérification de l'état des interfaces**

```
[root@localhost test]# cat /proc/net/bonding/bond0

Ethernet Channel Bonding Driver: v3.7.1 (April 27, 2011)

Bonding Mode: adaptive load balancing          -- le mode 6
Primary Slave: None
Currently Active Slave: p16p2                  -- Changement d'interface
MII Status: up                                    active par défaut
MII Polling Interval (ms): 100
Up Delay (ms): 0
Down Delay (ms): 0

Slave Interface: p16p1
MII Status: down                               -- Le lien par défaut est
Speed: Unknown                                    tombé (1er lien)
Duplex: Unknown
Link Failure Count: 1                          -- Compteur de coupure lien
Permanent HW addr: 00:15:17:d6:e0:8c           -- MAC adresse 1er lien
Slave queue ID: 0

Slave Interface: p16p2                         -- 2ème lien
MII Status: up
Speed: 1000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 00:15:17:d6:e0:8d           -- MAC adresse 2ème lien
Slave queue ID: 0
```

**Conclusion :** L'interface active a changée après que le lien de départ par défaut soit déconnecté.

Le fichier a été transmis avec succès vers le PC2 et après avoir effectué un checksum, tout est correct au niveau de son intégrité.

**-- Vérification au niveau du cache arp (avant-après coupure)**

```
root@localhost admin]# arp -D

Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.0.2              ether   00:15:17:d6:e0:8c     C                   bond0
```

**Conclusion** : Malgré la coupure il n'y a pas de changements au niveau de l'adresse MAC utilisée. Gestion effectuée par bond0.

**-- Vérification au niveau du cache arp (après reconnexion)**

```
root@localhost admin]# arp -D

Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.0.2              ether   00:15:17:d6:e0:8d   C                     bond0
```

Après reconnexion des 2 liens, c'est l'interface p16p2 qui est active par défaut, le compteur Link Failure Count est maintenant à 1 pour l'interface p16p1.

**-- Capture Wireshark coupure de lien**

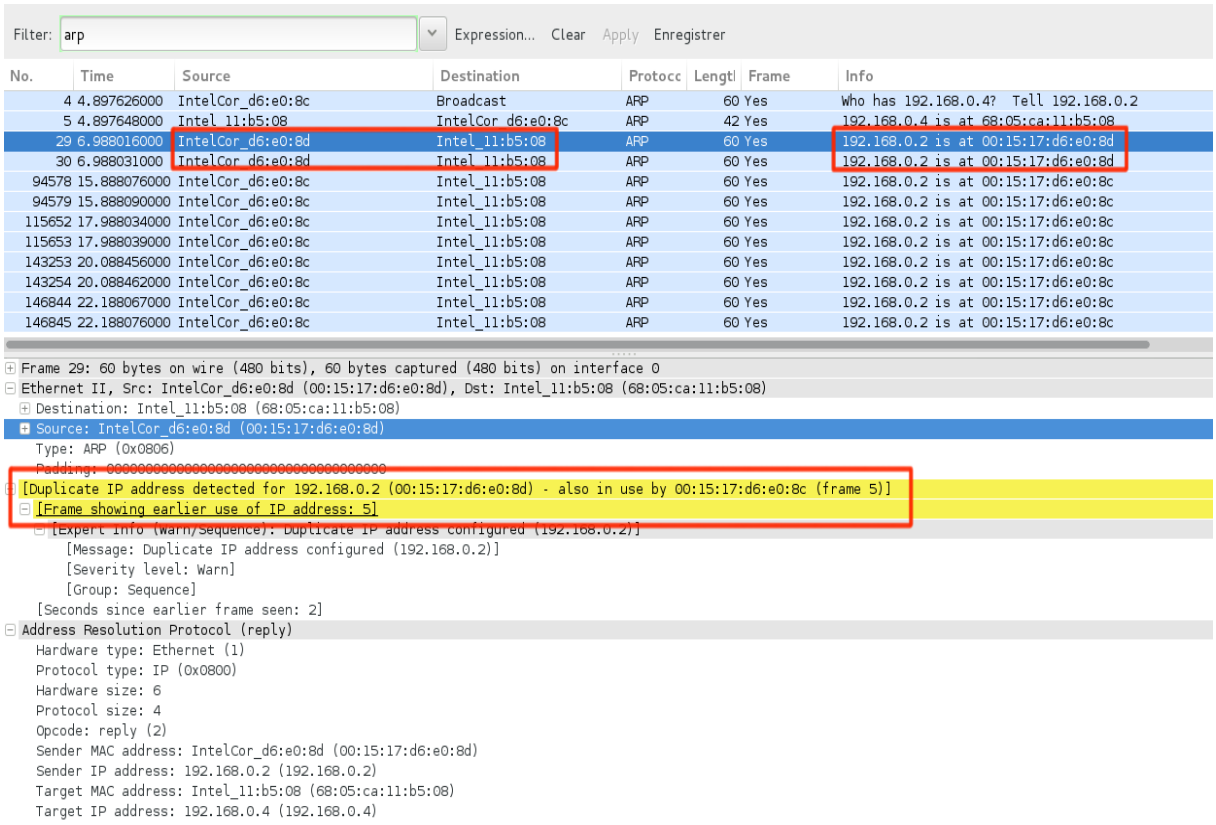



Figure 25: Capture Wireshark coupure de lien mode 6

**Conclusion** : On observe avec l'analyseur que l'interface p16p2 (MAC 00:15:17:d6:e0:8d) a pris le relais après la coupure de lien. Cependant, c'est la MAC adresse de la liaison qui a été déconnectée qui est gardée pour le reste de la transmission.

Cela est possible car les MACs adresses sont en réalité gérées au niveau de l'interface virtuelle bond0 qui est capable de choisir celle qui lui convient selon les besoins.

**-- Diagramme des échanges arp pour le mode 6**

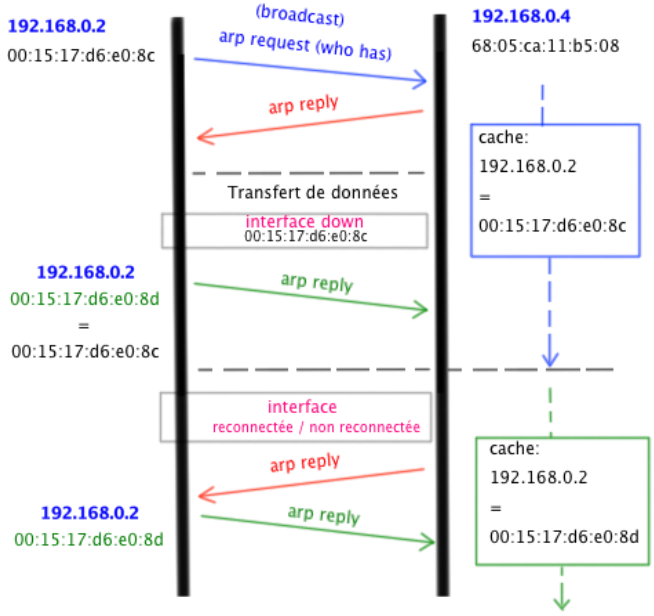
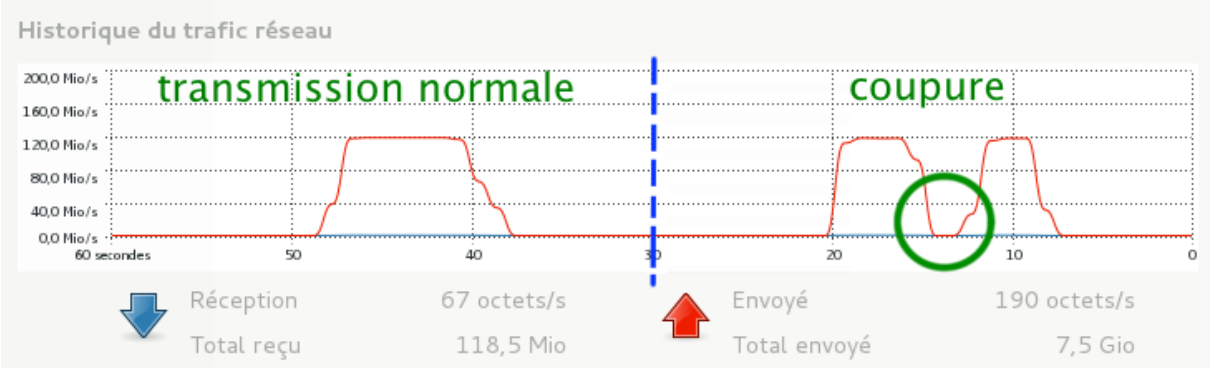
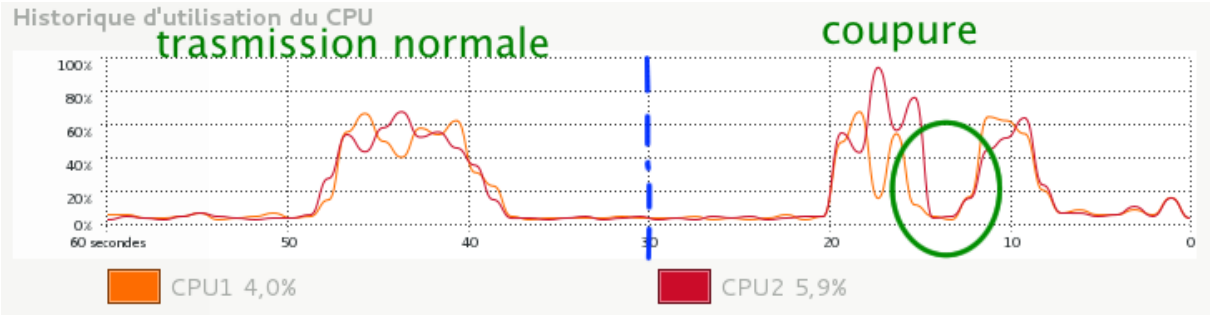



Figure 26: Diagramme ARP – IP mode 6

**-- Analyse avec le moniteur système de Fedora**

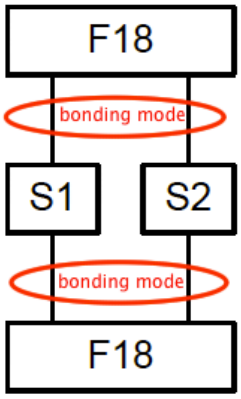



Au moment de la coupure d'un des liens, on observe qu'il y a un basculement sur la deuxième interface car le débit tombe à 0.




Au niveau de la charge CPU on note une baisse au moment de la perte du lien, puis cela remonte lors du basculement sur la deuxième interface.

## Scénario test 3 : 2 PCs Fedora 18 et 2 Switchs




Dans ce scénario, je vais étudier le fonctionnement du mode 6 lors de la mise en parallèle de deux switchs Netgear GS116E.

Une seule interface physique Gbit/s est reliée à chacun des switchs.

Ce nouveau scénario va me permettre de savoir si cette configuration est possible.

Figure 27: Schéma scénario test 3

Dès le départ avec une simple commande ping du PC1 vers le PC2, je me rends compte que cela n'est pas fonctionnel.

La problématique est la même que dans le scénario 1[39] testé pour ce mode. Pour fonctionner, le mode 6 a besoin que tous les liens soient connectés à un switch qui gère les différents changements d'interface.

# Conclusion Finale Mode 6

## Fonctionnement du mode 6 (balance-alb)

Ce mode permet l'équilibrage de charge auto-adaptatif en émission et en réception. Le trafic en sortie est adapté à la charge des interfaces actives.

L'interface ayant le moins de charge sur le moment va être choisie pour l'envoi. Ce mode permet également d'équilibrer la charge en entrée en interceptant les requêtes ARP et en les redistribuant pour diriger le flux vers l'interface la moins chargée. C'est l'interface virtuelle bond0 qui intercepte les réponses ARP et change l'adresse MAC par celle de l'une des interfaces.

A la fin de la configuration, l'interface bond0 est la seule à avoir une adresse IP. Chaque interface possède sa propre adresse MAC. Mais une seule interface est active au départ en cas d'envoi ou de réception.

C'est la première interface qui est active par défaut (p16p1) et son adresse MAC est prise aussi par bond0.

Ce mode utilise le protocole LLC[40] (Logical Link Control) sur chacune de ces interfaces actives. Chaque seconde 3 trames LLC

[39] Tests mode 6 Scénario de test 1 : Deux PCs Fedora 18 avec bonding
[40] Logical Link Control : http://www.javvin.com/protocolLLC.html

de 60 bytes sont envoyées par chaque interface active avec comme MAC source et MAC destination sa propre adresse. Ceci permet d'indiquer quelles interfaces sont connectées au switch et quelles adresses MAC elles emploient.

Il est important de remarquer que le mode 6 ne peut pas fonctionner sans un switch. En effet, les interfaces interagissent entre elles lors des échanges et le switch permet de diriger le flux lors des changements.

Le fonctionnement de ce mode est proche de celui d'un serveur client. Les connexions entre PCs sont établies à l'aide d'une requête ARP avant le début de la transmission.

Avant chaque échange, une requête ARP est envoyée en broadcast par le client. Cette requête est interceptée par le driver bonding qui répond en transmettant l'adresse MAC de l'interface la moins utilisée.

Ce lien entre les deux interfaces du PC1 et du PC2 ne change pas jusqu'à la fin de l'entière transmission des données par les deux PCs.

Par la suite, si une nouvelle demande est effectuée par un autre PC, la deuxième interface est mise à disposition par le driver bonding. Les deux cartes réseau sont dans ce cas actives en même temps mais de façon indépendante.

Ce fonctionnement peut être décrit d'après les résultats des tests effectués avec divers scénarios dans la partie tests qui suit.

# Chapitre II
# Virtualisation et Stockage

# Enoncé

Dans ce chapitre, mon objectif consiste dans la mise en place d'un système redondant basé sur 2 SANs QNAP, 2 switchs Netgear modèle GS116E et 1 hyperviseur KVM sur Fedora 18.

Pour cela, je vais étudier les solutions les plus adaptées pouvant être utilisées au niveau réplication (RTRR ou RSYNC), agrégation de liens (bonding mode 0 ou 6) et gestion des fichiers (NFS ou iSCSI) et des disques sur les 2 QNAPs.

Mon système doit également être aussi capable de pouvoir basculer en toute transparence, en cas de coupure de liens et lors d'une panne d'un ou plusieurs équipements.

# Analyse

## Architectures NAS et SAN

Le NAS (Network Attached Storage) ainsi que le SAN (Storage Area Network) sont des solutions de stockage permettant la centralisation des données et l'accès rapide aux fichiers.

Le SAN forme un réseau de stockage fiable et dédié, basé sur la Fibre Channel, qui offre une grande souplesse tant en distance qu'en connectivité.

Cette architecture est intéressante pour des applications demandant un stockage spécifique, comme les bases de données qui doivent effectuer des mises à jour fréquentes en temps réel.

Les serveurs de stockage réseau NAS sont eux connectés directement à une infrastructure réseau (LAN) existante. Cette architecture permet le partage d'un même fichier entre de multiples serveurs et clients se trouvant sur le même réseau.

Les NAS sont adaptés pour le stockage de données lors de l'utilisation d'applications comme la CAO, le développement logiciel, l'hébergement web ou email.

Les implémentations de type NAS et SAN ne s'excluent pas mutuellement. Au contraire, elles sont souvent des solutions complémentaires permettant d'offrir de bonnes performances au niveau stockage pour une entreprise.

### Principales différences entre SAN et NAS

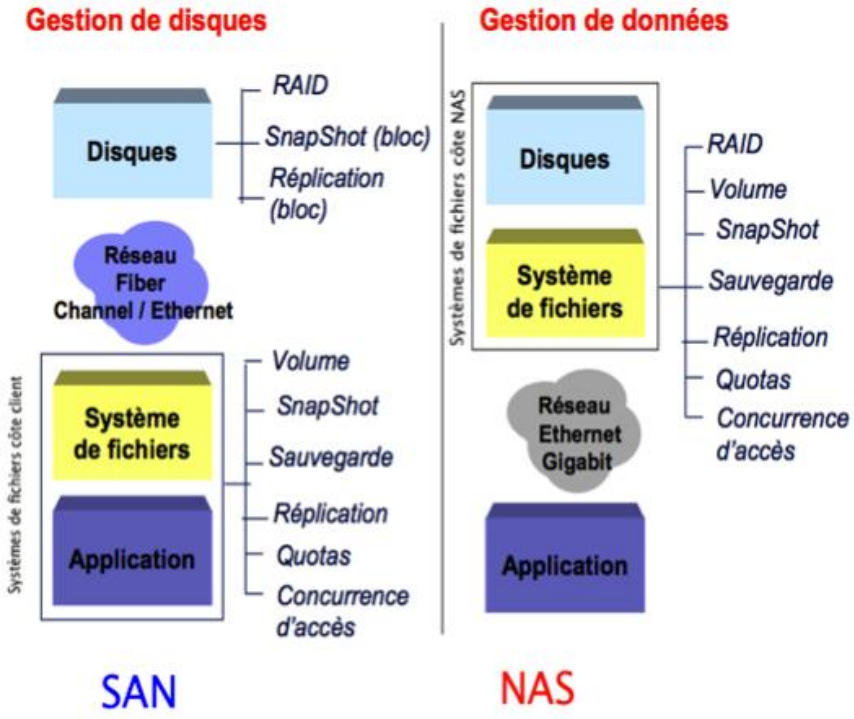



Figure 28: Topologie des architectures SAN et NAS

Dans cette figure, on remarque que dans les configurations NAS, le système de fichiers réside dans le NAS lui-même. A l'inverse, dans les configurations SAN, le système de fichiers réside sur le client qui possède la portion de stockage partitionné alloué sur le SAN.

Le tableau suivant permet de voir les principales caractéristiques de ces 2 architectures de stockage.

| | NAS | SAN |
|---|---|---|
| **Installation** | Mise en place rapide et facile sur un réseau LAN existant | Déploiement souvent complexe demandant un nouveau réseau |
| **Disponibilité** | En permanence, travail au niveau fichiers (NFS) | En permanence, travail au niveau blocs (iSCSI) |
| **Système de fichiers** | Du côté NAS | Du côté client |
| **Fiabilité** | Architecture composée de systèmes redondants avec possibilité de réplication locale | Architecture haute disponibilité conçue contre les risques de panne partielle ou totale |
| **Capacité & Scalabilité** | On peut atteindre plusieurs téraoctets de données, possibilité de mise en commun d'autres dispositifs de stockage | |

| | | |
|---|---|---|
| **Sécurité** | Par mot de passe, accès client avec partage de fichiers | Système isolé avec aide à la protection, partage et mouvement de données |
| **Coût** | Très abordable | Elevé |
| **Administration** | Centralisé, sur un poste administrateur en mode GUI ou CLI | |
| **Compatibilité** | Support des applications client NFS et SMB. Facilité de migration. | Support des applications serveur avec haut niveau de performances SCSI, iSCSI |
| **Protocole** | Ethernet | Fibre Channel, Infiband |
| **Débit binaire** | Dépendant du réseau local | Dépendant de la Fibre Channel (environ 100Mb/s en transferts de données réels) |

Figure 29: Tableau comparatif NAS et SAN

## Comparaison iSCSI et NFS

Quand on parle de gestion de disques par IP, on a le choix entre deux méthodes permettant d'administrer les entrées et les sorties disque, que cela soit pour du simple stockage ou pour de la virtualisation. C'est évidemment du NFS et de l'iSCSI.

Le but de ce paragraphe n'est pas d'expliquer en détail le fonctionnement de NFS[41] et de iSCSI, mais de mettre en avant les principales caractéristiques en effectuant une comparaison[42] rapide entre ces deux méthodes.

La finalité étant de pouvoir faire un choix entre les deux pour implémenter un des protocoles dans le système de stockage redondant qui m'est demandé d'étudier.

### Le protocole NFS

NFS (Network File System) a été inventé par Sun Microsystems. Ce protocole s'est imposé comme un standard par sa simplicité de mise en place et d'emploi.

Le mécanisme NFS est robuste car il repose sur une représentation standard des objets avec le protocole XDR.

[41] Travail de semestre Système de Stockage QNAP: http://www.tdeig.ch/linux/DeOliveira_EPS.pdf
[42] A Performance Comparison of NFS and iSCSI for IP-Networked Storage : http://lass.cs.umass.edu/papers/pdf/FAST04.pdf

Puis sur le mécanisme d'appels de procédures distantes implémenté par le protocole RPC (Remote Procedure Call).

Les accès au disque s'effectuent par un système de fichiers NFS en lecture et en écriture. Le File System est côté serveur. Il est accessible via le partage réseau configuré côté client.

Au niveau performances, NFS est très performant pour le transfert de fichiers de petite taille.

La dernière version NFS 4.1 est encore en développement, mais elle annonce des nouveautés comme la gestion du Failover au niveau des liens, grâce à la possibilité de configurer plusieurs serveurs NFS à la suite.

Cependant, pour le moment cette fonction ne semble pas encore implémentée et aucune documentation ne relate d'une possible mise en œuvre actuellement.

### Le protocole iSCSI

La deuxième possibilité de gestion des disques est la méthode iSCSI. C'est un protocole qui fonctionne au niveau blocs en encapsulant les commandes SCSI qui interagissent directement avec les disques durs. Comme pour NFS, c'est un protocole très utilisé quand on souhaite travailler avec un disque distant en utilisant toutes les possibilités mises à disposition au niveau du débit binaire sur le réseau.

Le File System est côté client et les accès aux disques en écriture et en lecture se font par blocs. Pour mettre en place ce type de stockage, il faut configurer[43] un iSCSI Target sur le serveur et un iSCSI Initiator sur le client.

### Schémas de comparaison NFS et iSCSI

Les deux schémas suivants permettent d'avoir une description globale du fonctionnement des protocoles NFS et iSCSI.

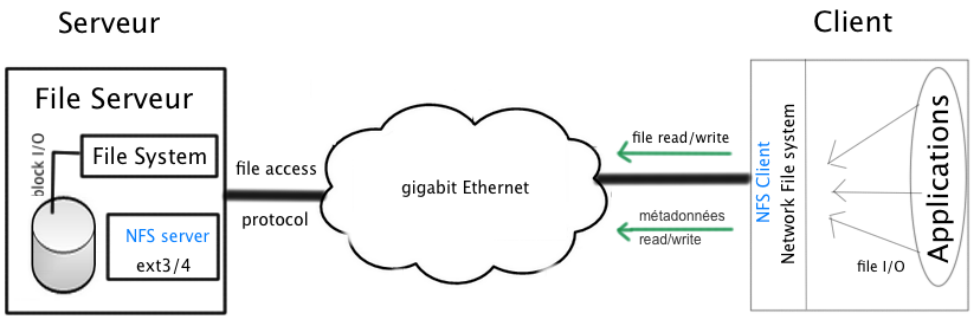



Figure 30: Schéma de fonctionnalité NFS

[43] Annexe : A.3 Configuration client iSCSI Fedora 18

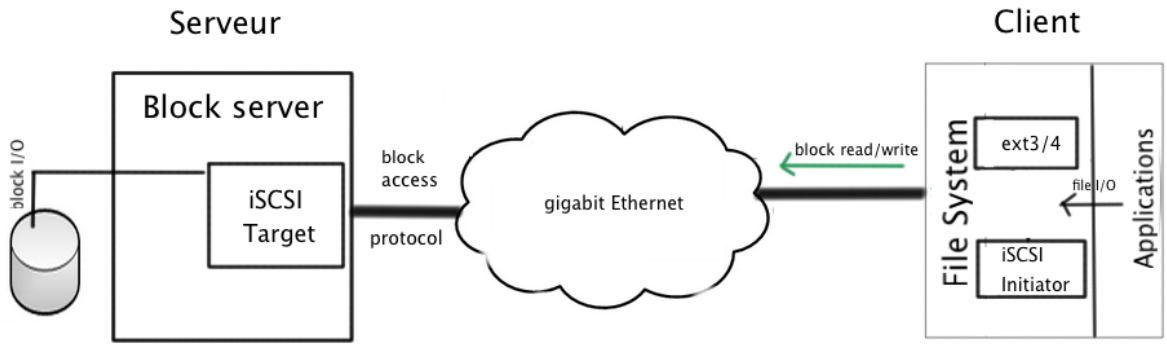



Figure 31: Schéma de fonctionnalité iSCSI

## Réplication RTRR et RSYNC

La réplication entre deux systèmes de stockage est un processus qui consiste à reproduire à l'identique les données du premier équipement sur le deuxième. Ceci permet de sauvegarder l'intégrité des données en cas de panne matériel, mais aussi de fournir de la haute disponibilité par cette redondance matérielle.

Sur QNAP, RTRR et RSYNC sont deux protocoles de transport de données pouvant être configurés pour effectuer la réplication entre QNAPS ou QNAP et un autre équipement de stockage.

### RTRR (Real Time Remote Replication)

La réplication à distance en temps réel (RTRR) est une fonction permettant une copie des données en temps réel ou planifiée entre des SANs locaux (LAN) ou distants (WAN), un serveur FTP ou un disque externe.

La duplication des données s'effectué en mode synchrone et par fichiers. Il est évidemment possible d'effectuer un filtre sur les éléments que l'on souhaite sauvegarder.

En mode temps réel, toutes modifications sur la source seront immédiatement répliquées dans le dossier cible, alors que dans le mode planification, le dossier source sera reproduit selon une planification choisie.

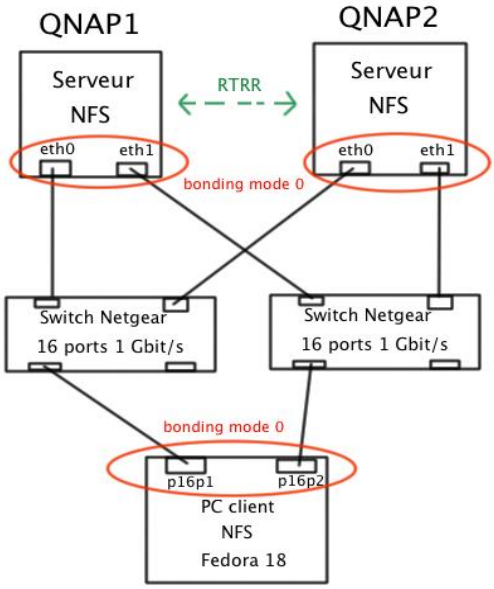
### RSYNC (Remote Synchronization) [44]

Ce protocole permet de dupliquer les fichiers sur un dossier local ou distant, sur un autre serveur RSYNC. La fonction de réplication à distance permet d'accomplir facilement cette tâche par l'envoi par bloc des données.

[44] Description RSYNC : http://man.developpez.com/man1/rsync.1.php/

On peut immédiatement effectuer la réplication ou la programmer pour qu'elle se fasse de façon périodique ou à un moment spécifique de la journée. Cela permet de réduire l'utilisation au niveau débit binaire du réseau ainsi que le temps nécessaire à cette tâche. Tous les fichiers sont comprimés avant d'être transférés sur le réseau. Cependant, cette fonction ne permet pas d'effectuer une réplication en temps réel comme RTRR.

## Choix de la configuration




Après l'étude théorique des diverses possibilités pouvant être configurées sur les 2 QNAPs, je décide de mettre en place les solutions suivantes:

- *configuration du mode 0 pour le bonding*

Ma décision est prise après l'étude du bonding pour les modes 0 et 6 dans le chapitre I de ce mémoire.

Tout d'abord, car le mode 6 ne fonctionne pas pour le scénario final proposé. Ensuite, l'envoi séquentiel sur les 2 liens permet l'augmentation du débit binaire.

Enfin, l'utilisation de la même MAC Adresse pour les 2 interfaces permet, selon moi, une meilleure gestion en cas de panne matériel.

- configuration d'un partage de systèmes de fichiers NFS

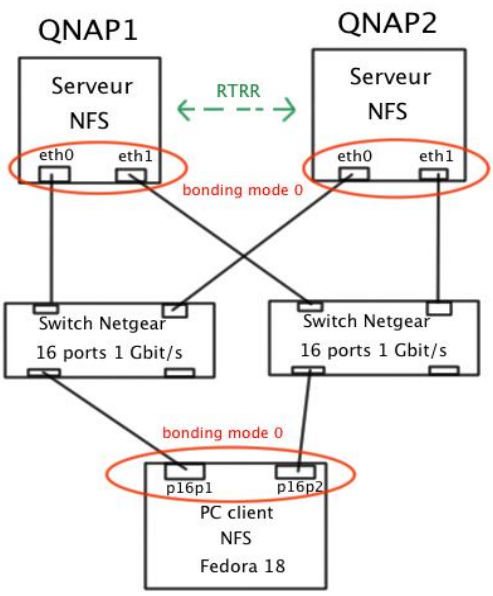
J'ai fait le choix de commencer mon étude en mettant en place un système de partage NFS, car sa mise en place est très simple, tant au niveau client qu'au niveau serveur. De plus, au niveau du cahier des charges cela semble un choix possible.

Cela va me permettre aussi de voir quels problèmes cela peut poser avec NFS en cas de Failover et surtout si c'est pris en charge.

- *configuration du protocole de réplication RTRR*

En vue de l'étude théorique, le protocole RTRR semble le choix le plus judicieux car il permet une copie en temps réel en cas de changements sur l'un des serveurs QNAP. Par contre, ce protocole est propriétaire, donc je ne pourrais pas décrire son fonctionnement en détail.

## Schéma d'installation de départ




Cette installation est un point de départ dans l'étude d'une proposition de système de stockage redondant.

Cela va me permettre de vérifier le bon fonctionnement matériel de ma solution finale étape par étape et avec les mécanismes que j'ai choisi de mettre en place[45] (bonding mode 0, NFS, RTRR).

Au final, le dispositif sera composé de 2 SANs QNAP, de 2 switchs Netgear GS116E et 2 hyperviseurs KVM, comme demandé dans le cahier des charges.

Figure 32: Schéma d'installation de départ

### Matériel à disposition

#### *Hardware PCs*

- Carte mère ASUS P5G41T-M LX
- Processeur Core2Duo 3 Ghz, 6Mb de cache, FSB de 1333Mhz
- Disque dur de 320GB en S-ATA
- RAM (2x4GB en DDR800)
- 1 carte Ethernet PCIe Intel®Pro/1000 PT Dual Port

#### *SANs*

- QNAP TS-459 Pro II[46]
- QNAP TS-469 Pro (mêmes spécifications que le TS-459)

#### *Commutateurs*

- Switchs Netgear GS116E 16 ports x 1 Gbit/s

#### *Distributions software*

- PCs Fedora 18 Gnome
- Hyperviseur KVM
- QNAPs avec un système Linux 2.6

[45] Chapitre II Virtualisation et Stockage : Choix de la configuration
[46] Annexe : A.8 Spécifications des systèmes QNAP

# Configuration QNAPs

## Installation de base best practices

Le début de l'installation de base sur les QNAPs a fait l'étude d'un travail de semestre antérieur[47]. En cas de besoin, il faut s'aider de la marche à suivre se trouvant dans ce document ou utiliser le manuel en ligne du fabriquant[48]. A la fin de cette installation, chaque QNAP doit avoir au minimum une adresse IP fixe.

## Activer système de fichiers NFS

L'activation se fait dans ***Accueil >> Service réseau >> Service NFS.*** Il suffit de cocher la case correspondante à NFS.

## Créer dossiers avec droits NFS

Je crée deux répertoires intitulés « dosssier1 » et « dossier2 » à l'aide du Web File Manager[49] QNAP. Ensuite, je configure la gestion des droits d'accès NFS dans ***Accueil >> Gestion des droits d'accès >> Dossier de partage***[50] sur les dossiers en question.

## Activer le bonding mode 0

L'activation se fait dans l'administration QNAP sous ***Accueil >> Administration du système >> Paramètres de réseau***


Figure 33: Activation bonding sur QNAP

Dans ***Paramètres*** de la partie Port Trunking, sélectionner dans la nouvelle fenêtre le type d'agrégation (Balance-rr) mode 0 ainsi que les deux interfaces disponibles sur le QNAP.

[47] Travail de semestre Système de Stockage QNAP: http://www.tdeig.ch/linux/DeOliveira_EPS.pdf
[48] QNAP Turbo NAS Manuel de l'utilisateur : http://docs.qnap.com/nas/fr/index.html?home.htm
[49] Utilisation du Web File Manager : http://docs.qnap.com/nas/fr/index.html?home.htm
[50] Gestion droits NFS QNAP : http://docs.qnap.com/nas/fr/index.html?home.htm


Une fois terminée la partie encadrée en bleu sur l'image nous montre une vue affichant les interfaces en mode bonding, l'adresse IP utilisée, l'adresse MAC, le débit binaire des liens ainsi que le MTU [51] (Maximum Transmission Unit).

### *Configuration des interfaces des 2 QNAPs*

| Device | IP | MAC Adresse | DHCP | Interface en mode 0 | Débit du lien |
|---|---|---|---|---|---|
| **QNAP1** | 192.168.0.10 | 00:08:9B:CE:07:E3 | NON | Ethernet 1+2 | 1000 Mbps |
| **QNAP2** | 192.168.0.20 | 00:08:9B:D4:17:15 | | | |

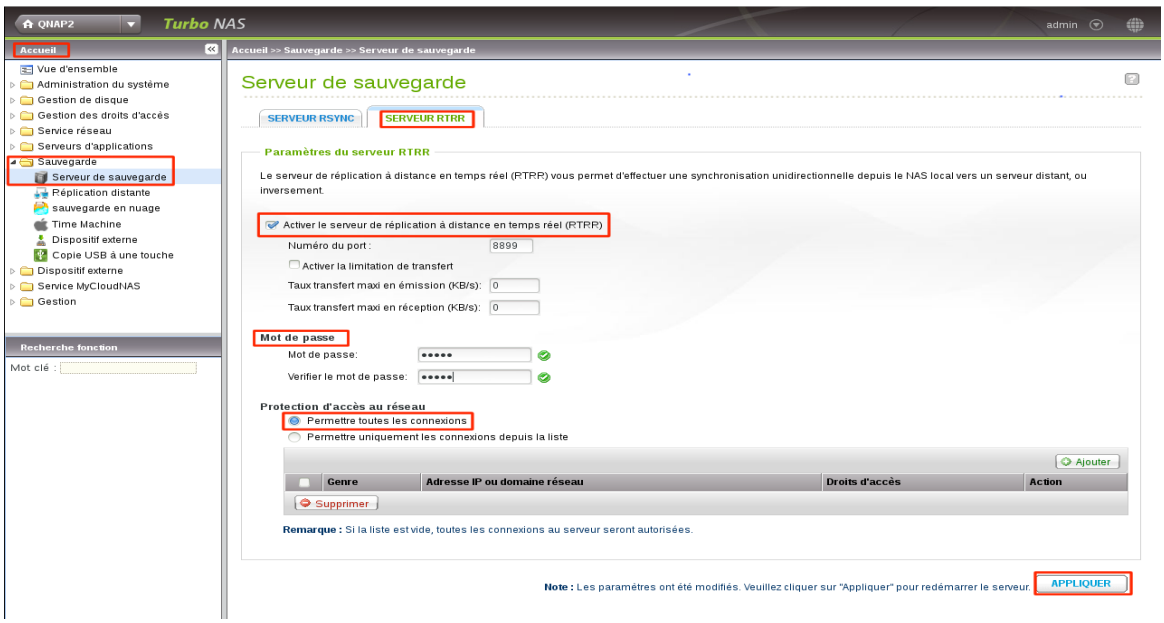
## Configuration de la réplication RTRR

Cette configuration doit être activée **sur les deux QNAPs** afin d'obtenir une réplication des données en temps réel lors d'une modification sur l'un des équipements.

### *Activer serveur RTRR*

Aller dans ***Accueil >> Sauvegarde >> Serveur de sauvegarde*** et suivre les instructions dans la partie Serveur RTRR.

Plusieurs options sont possibles dans le but d'affiner la configuration, comme le choix d'un numéro de port spécifique pour les échanges de réplication ou encore de permettre uniquement les connexions depuis les équipements spécifiées dans une liste.




[51] Explication « Maximum Transmission Unit » : http://fr.wikipedia.org/wiki/Maximum_Transmission_Unit

### *Activer client RTRR*

Aller dans ***Accueil >> Sauvegarde >> Réplication distante*** et sélectionner RTTR. Suivre les instructions pour la configuration sur le site QNAP [52] . Dans le cas de ma configuration, effectuez les choix suivants dans les étapes:

- 2/11 : Dossier local vers dossier distant
- 3/11 : introduire adresse IP du QNAP sur lequel on souhaite effectuer la réplication (192.168.0.10 puis 192.168.0.20) et 1e type de serveur RTRR. Le test de liaison effectué à cette étape m'indique 102,721 MB/s comme débit binaire.
- 4/11 et 5/11 : faire correspondre la paire de dossiers à répliquer Dossier1 >> Dossier 1 puis Dossier2 >> Dossier2
- 6/11 : choisir l'option Real-Time

Une fois la configuration finie, un simple test d'envoi de fichier à l'aide du Web File Manager sur l'un des QNAPs permet d'observer instantanément sa copie sur le deuxième QNAP.

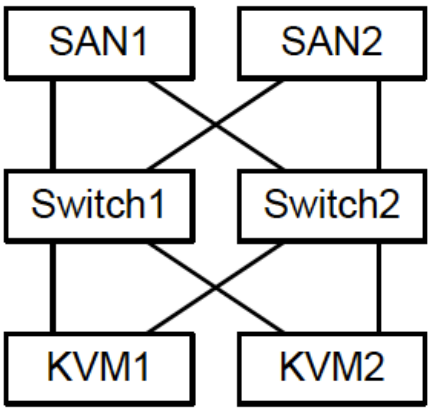
## Configuration PCs

### Partage NFS sur PC client[53]

```
-- Création d'un dossier contenant le partage NFS
root@localhost admin]# mkdir mnt/dossierNFS

-- Monter le dossier à chaque démarrage en éditant le fichier /etc/fstab
root@localhost admin]# vi /etc/fstab
192.168.0.10:/dossier1 /mnt/dossierNFS nfs auto,user,rw 0 0

-- Monter toutes les partitions contenues dans fstab
root@localhost admin]# mount –a

-- Lister les informations de montage d'un serveur NFS
root@localhost admin]# shownount –c 192.168.0.10
```

[52] Configuration Client RTRR : http://www.qnap.com/index.php?lang=fr&sn=2877
[53] Documentation NFS Fedora : http://doc.fedora-fr.org/wiki/Partage_de_disques_en_r%C3%A9seau_avec_NFS

## Scénario de départ




**SAN1** : QNAP-459 avec Bonding mode 0

**SAN2** : QNAP-469 avec Bonding mode 0

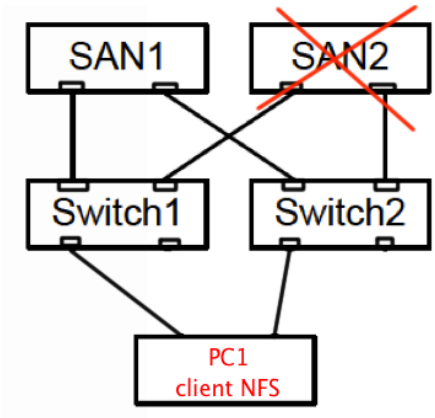
**Switch1 et 2** : Netgear 16 ports Gbit/s

**KVM1** : Fedora 18 Gnome avec Bonding mode 0

**KVM2** : Même configuration que KVM1. Pas utilisé au départ

L'objectif des divers scénarios qui vont suivre est d'observer s'il est possible de supporter une panne du SAN1. La panne est simulée en débranchant les 2 interfaces réseau.

## Scénario 1 : 1 QNAP en NFS, 2 switchs et 1 PC client NFS




Dans ce scénario, le client possède dossier de partage NFS et ne connaît que SAN1 (Serveur NFS QNAP). Tous les liens sont en bonding mode 0.

Figure 34: Scénario 1 avec 1 QNAP et un client NFS

L'objectif de ce premier scénario est d'étudier le mécanisme de réplication synchrone en temps réel basé sur RTTR entre les deux QNAPs. J'effectuerais aussi quelques tests de performance en écriture sur le serveur NFS.

**-- Test de fonctionnalité avec un ping PC1 --> QNAP1**

```
[root@localhost admin]# ping -c 1 192.168.0.10
PING 192.168.0.10 (192.168.0.10) 56(84) bytes of data.
64 bytes from 192.168.0.10: icmp_seq=1 ttl=64 time=0.129 ms
--- 192.168.0.10 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.129/0.129/0.129/0.000 ms
```

**-- Avec traceroute PC1 --> PC2**

```
[root@localhost admin]# traceroute 192.168.0.10
traceroute to 192.168.0.10 (192.168.0.10), 30 hops max, 60 byte packets
 1  192.168.0.10 (192.168.0.10)  0.176 ms  0.155 ms  0.128 ms
```

Le dispositif est fonctionnel et le temps de réponse moyen est acceptable dans ce contexte avec un serveur QNAP configuré en NFS et un PC client.

### Test de performances

Pour les différents tests de performance, je me suis basé sur le laboratoire Storage[54].

**Test 1 :** Dans ce premier test, je vais effectuer un transfert de fichier depuis le poste client NFS sur le QNAP. Le fichier à envoyer est créé avec l'outil dd[55] et envoyé à l'aide de SCP[56] (Secure Copy Protocol) qui permet de transférer des fichiers entre des systèmes via le protocole sécurisé SSH.

**-- Outil dd et scp pour envoi d'un fichier de 100Mo**

```
-- Effacer les différents caches sur PC Linux
[root@localhost admin]# echo 3 > /proc/sys/vm/drop_caches[root@localhost

-- Création d'un fichier de 100Mo et envoi par SCP
admin]# dd if=/dev/zero of=testNFS bs=100M count=1; scp testNFS
admin@192.168.0.10:/share/Dossier1;
1+0 enregistrements lus
1+0 enregistrements écrits
104857600 octets (105 MB) copiés, 0,448072 s, 234 MB/s
admin@192.168.0.10's password:
testNFS                                       100%  100MB  16.7MB/s   00:06
```

**-- Outil dd et scp pour envoi d'un fichier de 1Go**

```
-- Effacer le cache sur le PC Linux
[root@localhost admin]# echo 3 > /proc/sys/vm/drop_caches[root@localhost

-- Création d'un fichier de 1Go et envoi par SCP
admin]# dd if=/dev/zero of=testNFS bs=1G count=1; scp testNFS
admin@192.168.0.10:/share/Dossier1;
1+0 enregistrements lus
1+0 enregistrements écrits
1073741824 octets (1,1 GB) copiés, 16,6743 s, 64,4 MB/s
admin@192.168.0.10's password:
testNFS                                       100% 1024MB  17.4MB/s   00:59
```

[54] Laboratoire tdeig : Lab_Storage.pdf
[55] Fonctionnement de l'outil dd sur Linux : http://doc.ubuntu-fr.org/dd
[56] Fonctionnement SCP : http://linux-attitude.fr/post/scp-d-une-machine-a-une-autre

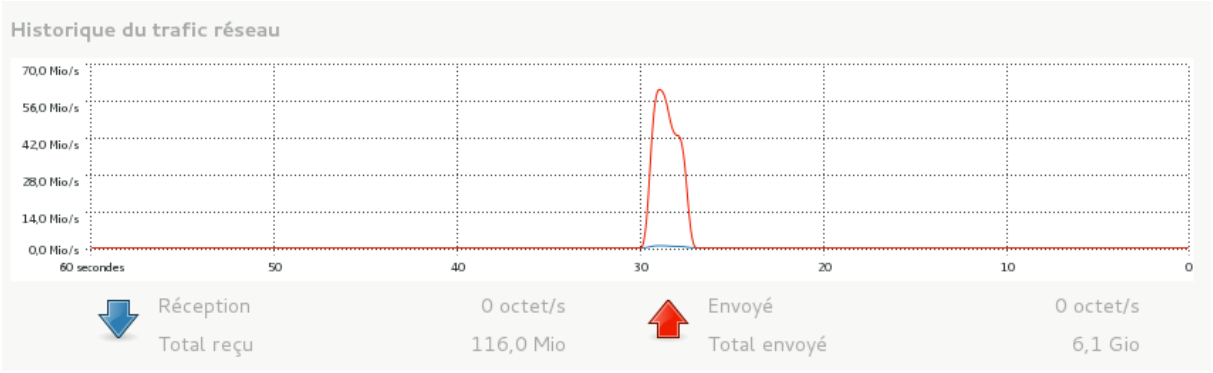
**Conclusion :** Lors de l'envoi de deux fichiers de tailles différentes à l'aide de SCP, on constate que le débit binaire obtenu (17 MB/s) est presque équivalent.

Les temps de transferts sont acceptables en vue des tailles des fichiers. Il faut compter 1 minute pour transférer un fichier de 1 Go sur le QNAP via la structure réseau mise en place dans ce scénario avec NFS.

**Test 2 :** Ce deuxième test consiste dans un premier temps, à la création d'un fichier comme précédemment à l'aide de l'outil dd. Puis cette fois, on va copier à l'aide de la commande PV le fichier dans le dossier de partage NFS se trouvant dans le PC client. Le transfert va s'effectuer sur le QNAP à l'aide de ce dossier de partage et je vais observer au niveau de l'outil réseau le débit binaire.

**-- Commande PV sur dossier NFS et envoi de 100Mo sur QNAP1**

```
-- Création du fichier de 100Mo
root@localhost admin]# dd if=/dev/zero of=f100M bs=1k count=100000
100000+0 enregistrements lus
100000+0 enregistrements écrits
102400000 octets (102 MB) copiés, 0,370656 s, 276 MB/s
-- On vide les caches Linux
[root@localhost admin]# sync; sysctl -w vm.drop_caches=3
vm.drop_caches = 3
-- Copie du fichier dans le dossier de partage du client NFS
[root@localhost admin]# pv f100M > /mnt/dossierNFS/f100M
97,7MiO 0:00:01 [62,6MiB/s] [================================>] 100%
```

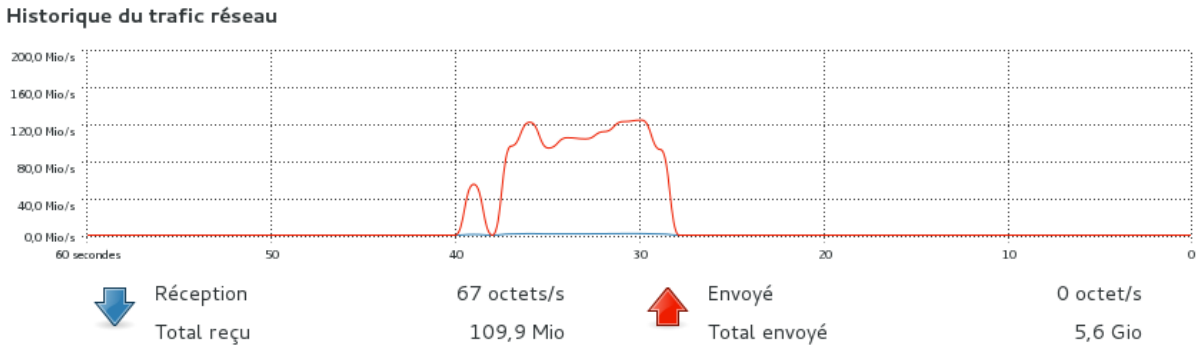



**Conclusion :** Les performances lors de l'envoi d'un fichier de taille moyenne (100 Mo) à l'aide du dossier de partage NFS sur le poste client vers le serveur QNAP s'approxime de 60 Mo/s en quelques secondes. Ayant étudié les performances du protocole NFS lors de mon travail de semestre [57], je sais que cela correspond au débit utile. En effet, l'overhead rajouté par ce protocole lors d'un échange est de 0,02% au maximum.

[57] Travail de semestre Système de Stockage QNAP: http://www.tdeig.ch/linux/DeOliveira_EPS.pdf

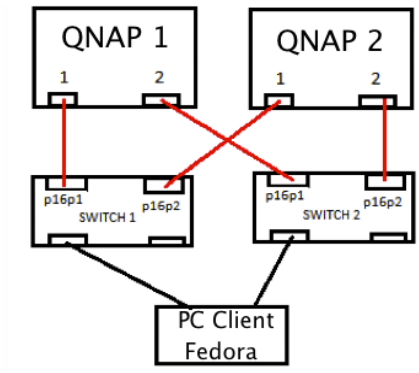
**-- Commande PV sur dossier NFS et envoi de 1Go sur QNAP1**

```
-- Création du fichier de 100Mo
root@localhost admin]# dd if=/dev/zero of=f1G bs=1k count=1000000
1000000+0 enregistrements lus
1000000+0 enregistrements écrits
1024000000 octets (1,0 GB) copiés, 17,5794 s, 58,3 MB/s
-- On vide les caches Linux
[root@localhost admin]# sync; sysctl -w vm.drop_caches=3
vm.drop_caches = 3
-- Copie du fichier dans le dossier de partage du client NFS
[root@localhost admin]# pv f1G > /mnt/dossierNFS/f1G
 976MiO 0:00:22 [44,1MiB/s] [================================>] 100%
```




Le même test avec un fichier de plus grande taille (1 Go) laisse entrevoir des performances avoisinant les 120 Mo/s.

## Observation du mécanisme de réplication RTRR entre QNAPs




Dans le but de pouvoir effectuer une capture avec TCPdump lors des différents échanges de réplication entre QNAPs, je change les switchs Netgear par des switchs Linux.

Les captures qui ont été effectuées sur les interfaces p16p1 et p16p2 des 2 switchs. Interfaces des QNAPs et du PC client sont en bonding mode 0.

**-- Analyse des captures de RTRR** *(Scénario 1/Captures RTRR)*

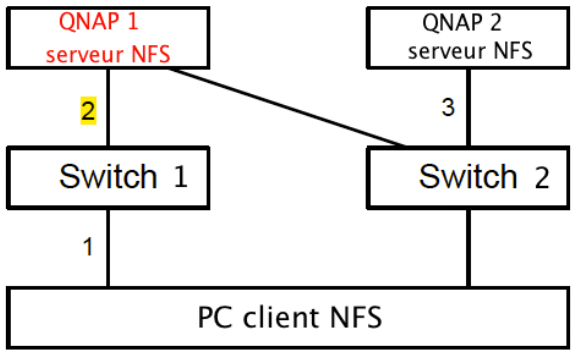
Le mécanisme de réplication étant propriétaire, l'analyseur n'est pas en mesure de fournir des informations détaillées sur le fonctionnement de RTRR.

Cependant, j'ai pu constater deux choses intéressantes. Tout d'abord, le mécanisme de transfert des données entre QNAPs ne commence que lorsque le fichier est intégralement transféré par le client sur le QNAP1.

Ce n'est qu'à ce moment, qu'il est transféré sur le QNAP 2. Il n'y a aucun échange entre les équipements auparavant.

Pour terminer, les différents échanges sont effectués par TCP /IP, ce qui permet d'assurer la fiabilité des données transférées. Le protocole de réplication RTRR est, selon moi, le meilleur choix pour effectuer une duplication de données entre QNAPs en vue de l'analyse effectuée.

### Détection de coupure d'un lien sur le QNAP 1 et PC client NFS




Le but, dans cette partie du scénario est de tester s'il est possible de détecter la coupure du lien 2.

Configuration NFS simple avec 1 Client NFS en bonding mode 0, 2 switchs et 1 Serveur NFS QNAP en bonding mode 0.

#### *Sur le serveur NFS QNAP*

**-- A l'aide de l'interface d'administration**

On peut observer l'état des liens en temps réel dans les onglets ***Gestion/Informations Système/Statut du port*** ou au niveau des logs système dans ***Administration du système/Journaux du système***. Ce moyen est très basique, mais reste une possibilité de contrôle direct.

**-- Sur la façade du QNAP[58]**

En cas de coupure d'un ou plusieurs liens sur les QNAPs, un message d'erreur s'affiche sur la devanture de l'équipement en continu en indiquant quelle NIC est indisponible.

#### *Sur le PC client NFS*

**-- Avec une connexion en ssh sur le QNAP**

On se connectant directement sur le QNAP avec un terminal en ssh on peut avoir accès au statu des interfaces à l'aide de la commande Linux pour le bonding.

```
-- Connexion en ssh depuis le terminal Fedora
root@localhost admin]# ssh admin@192.168.0.10

-- Dans le terminal Linus de QNAP
[/] cd ..
[/] ls
[/] # cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v3.7.1 (April 27, 2011)
```

[58] Annexe A.9 Détection visuelle d'une coupure de lien ou de réseau sur la façade du QNAP

```
Bonding Mode: load balancing (round-robin) -- mode de bonding utilisé
MII Status: up
MII Polling Interval (ms): 100
Up Delay (ms): 0
Down Delay (ms): 0
Slave Interface: eth0                    -- Interface physique QNAP eth0
MII Status: down                         -- Le lien est coupé
Speed: Unknown
Duplex: Unknown
Link Failure Count: 1
Permanent HW addr: 00:08:9b:ce:07:e3     -- Adresse MAC du lien eth0
Slave queue ID: 0

Slave Interface: eth1                    -- Interface physique QNAP eth1
MII Status: up
Speed: 1000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 00:08:9b:ce:07:e4     -- Adresse MAC du lien eth1
Slave queue ID: 0
```

**-- Envoi d'un ping du PC1 vers le QNAP**

Cette méthode reste la plus intéressante car elle permet d'avoir très rapidement l'état des interfaces. Cependant, elle n'indique pas exactement quel lien a subi une coupure.

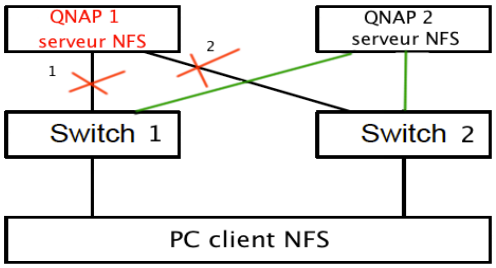
```
-- Ping du PC1 vers QNAP1
[root@localhost dossierNFS]# ping -c4 192.168.0.10
PING 192.168.0.10 (192.168.0.10) 56(84) bytes of data.
64 bytes from 192.168.0.10: icmp_seq=2 ttl=64 time=0.236 ms
64 bytes from 192.168.0.10: icmp_seq=4 ttl=64 time=0.264 ms

--- 192.168.0.10 ping statistics ---
4 packets transmitted, 2 received, 50% packet loss, time 2999ms
rtt min/avg/max/mdev = 0.236/0.250/0.264/0.014 ms
```

On remarque lors de l'envoi de 4 ping que 50% des paquets n'ont pas été reçus et sont perdus. On peut en déduire que un des liens est tombé car on se trouve en bonding mode 0.

Une solution serait un script de vérification de liens sur le PC client comme dans cet exemple :

```
#!/bin/bash
ping -c4 - q 192.168.0.10
TEST=$?
# Effectuer un test pour determiner si QNAP 1 est vivant
if [ $TEST -ne 1 ] then
        echo "QNAP 1 OK"
else
        echo "QNAP 1 est DOWN" fi
```

### Détection de coupure des 2 liens sur le QNAP 1 et PC client NFS




Le but de ce scénario est de vérifier s'il est possible de détecter et de basculer de façon automatique sur le QNAP 2 en cas de coupure des deux liens sur le QNAP 1. Je vais vérifier par des tests deux solutions possibles au niveau théorique.

#### *Solution 1 : Configuration multi serveurs avec NFS v.4*

Cette solution consiste à mettre en place une nouvelle fonctionnalité implémentée dans cette nouvelle monture de NFS v.4[59]. Différents documents sous-entendent qu'il est possible de gérer le Failover d'un serveur NFS[60] et que le client puisse basculer sur un autre serveur.

Je procède à la configuration[61] qui ressemble beaucoup à celle de NFS v.3 et je me rends compte dans mes recherches qu'il n'existe pas de documents mentionnant une marche à suivre précise. Le problème principal réside dans l'impossibilité de spécifier de multiples adresses de serveurs pour le basculement en cas de coupure ou panne du premier.

La conclusion est sans appel, malgré les différentes tentatives de configuration, il est impossible dans l'état actuel de développement de la version NFS 4 de mettre en place un tel procédé. Cela reste pour le moment, une méthode théorique. Je décide de trouver une autre solution.

#### *Solution 2 : Un script de détection de coupure de liens avec basculement*

Dans cette solution, je vais tenter à l'aide d'un script bash de détecter automatiquement le Failover du QNAP 1 dans un premier temps.

Puis si cela est vrai, le dossier de partage NFS de QNAP 1 ne répondra plus sur le PC client. Il devra alors être démonté et remplacé par un nouveau dossier de partage appartenant à QNAP 2. Le but final est de procédé à un basculement automatisé.

Mes essais avec NFS v3.0, ont rencontré des problèmes liés au mécanisme de NFS_LOCKS qui empêchent tout démontage du dossier

---

[59] RFC NFS v.4 :
http://www.citi.umich.edu/projects/nfsv4/rfc/draft-ietf-nfsv4-minorversion1-02.txt
Documentation Oracle versions NFS :
http://docs.oracle.com/cd/E24843_01/html/E22298/rfsintro-27.html#scrolltoc

[60] Basculement côté client en NFS v.4.1 :
http://docs.oracle.com/cd/E24843_01/html/E22298/rfsrefer-45.html#rfsrefer-51

[61] Montage de serveurs et client en NFS 4 :
http://www.sysadminfr.org/fr/articles/debian%20nfsv4

NFS après que QNAP 1 ne réponde plus. Avec NFS v.3, il n'est pas possible d'effectuer de basculement avec cette méthode.

La solution consiste à utiliser NFS v.4. Dans cette version, ce mécanisme de blocage n'est plus utilisé au même titre que d'autres services devenus obsolètes.

NFS v.4 n'étant pas activé par défaut sur les QNAPs, il faut dans un premier temps activer cette fonction à l'aide d'un terminal ssh.

**-- Activer NFS v.4.0 sur les deux QNAPs**

```
-- Connexion en ssh depuis le terminal Fedora
root@localhost admin]# ssh admin@192.168.0.10
-- Dans le terminal Linus de QNAP
[/] cd ..
[/] ls
[/] # /bin/setcfg NFS Enable_V4 TRUE
[/] # /etc/init.d/nfs restart
```

Puis, je commence par monter le partage de QNAP 1 sur le PC client NFS. C'est le partage qui doit être accessible en temps normal. QNAP 2 n'est pas monté donc pas connu du PC client NFS.

```
-- Monter dossier de partage sur PC client NFS
root@localhost admin]# mount –t nfs4 192.168.0.10:/Dossier1 /mnt/NFS
```

A ce stade, j'ai accès aux documents se trouvant sur QNAP 1 à travers le dossier NFS sur le PC client.

**Manipulation :** Je déconnecte les liens de QNAP 1 pour simuler une panne. Et je lance le script.

**-- Script de détection de coupure de lien et basculement**

```
#!/bin/bash
ping -c4 192.168.0.10
TEST=$?
echo $?
#Effectuer un test pour savoir si QNAP1 est connecte
if [ $TEST -ne 1 ]
   then
        echo "QNAP 1 marche"
   else
        echo "QNAP 1 ne marche pas"
        umount -f /mnt/NFS
        echo "Demontage de QNAP 1 termine... "
        mount -t nfs4 192.168.0.20:/Dossier1 /mnt/NFS
        echo "Montage du dossier NFS de QNAP 2 termine... "
fi
```

Je constate au niveau du terminal et en accédant au fichier de partage NFS que je suis sur QNAP 2.

Le script a permis de basculer d'un système de stockage NFS à un autre lors de la coupure de liens sur le premier.

**-- Résultat au niveau du terminal du PC client NFS**

```
[root@localhost /]# ./scriptTest.sh
-- Ping sur QNAP 1 ne répond pas
PING 192.168.0.10 (192.168.0.10) 56(84) bytes of data.
--- 192.168.0.10 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 2999ms
0  -- Sortie du résultat test (impossible d'effectuer le ping)
QNAP 1 ne marche pas
Demontage de QNAP 1 termine...
Montage du dossier NFS de QNAP 2 termine...
-- Dossiers se trouvant sur QNAP 2 accessibles
[root@localhost /]# ls /mnt/NFS
f1G  test  UbuntuServer.raw  UbuntuServer.raw.gz  UbuntuSrv.raw
```

La durée de basculement est supérieure à 20 secondes et provoque des ralentissements.

**-- Envoi d'un fichier et comportement lors d'une coupure**

Un plus grand problème survient lorsque j'essaie de transférer un fichier sur QNAP 1 et qu'une coupure intervient au même moment.

Le résultat de ce scénario est que le script ne permet pas de basculer sur QNAP 2 et provoque des erreurs.

La raison est que le démontage du dossier de partage NFS est impossible car il est déjà occupé (*erreur : umount.nfs4 : /mnt/NFS : device is busy*) par la tâche de transfert. Le système se bloque, empêchant toute manipulation au niveau du terminal.

La solution serait de tuer le processus qui effectue le transfert, mais cela implique la perte d'intégrité au niveau des données et que celles-ci ne puissent définitivement pas être transférées sur QNAP 2 après la coupure de façon automatisée. Cette solution n'est pas acceptable.

### Comportement lors de la coupure de 2 liens sur le QNAP1 NFS et PC client KVM

Pour terminer, j'ai effectué des tests avec un hyperviseur KVM sur le PC client qui se sont révélés non fonctionnels avec le même scénario.

Après avoir configuré l'hyperviseur KVM sur le PC client à l'aide de la marche à suivre best practices du laboratoire KVM[62], je mets en route une machine virtuelle (VM) dont l'image se trouve sur le serveur de stockage NFS QNAP1.

J'effectue une coupure des liens de QNAP1. Puis, je procède à un basculement manuel sur QNAP 2.

Le résultat est que malgré que la VM utilise la mémoire RAM pour rester active durant la coupure du lien, elle est incapable de continuer à fonctionner après le basculement sur QNAP 2. Je suis obligé de redémarrer la VM et je perds par conséquent les données se trouvant en RAM.

Pour moi, les limites de l'utilisation de NFS sont atteintes pour ce scénario en démontrant l'impossibilité de travailler avec la virtualisation.

### Conclusion générale pour le scénario 1

L'utilisation de NFS comme système de fichiers réseau dans un contexte de haute disponibilité avec des QNAPs n'est pas fonctionnelle avec des clients KVM.

Tout d'abord, parce que le protocole NFS ne permet pas dans la version actuelle (NFS v.4) de gérer le Failover au niveau des liens et des serveurs de stockage.

Le basculement à l'aide d'un script est possible, cependant l'intégrité des données n'est pas garantie lorsqu'une panne surgit en simultané pendant un transfert de données.

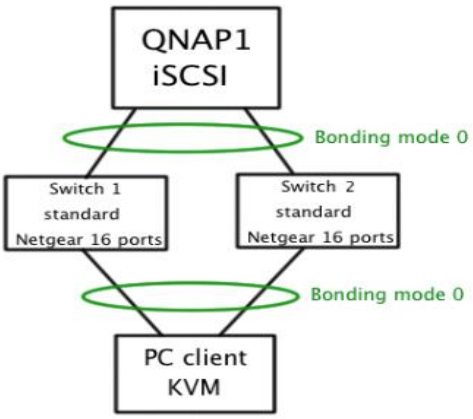
Le temps de basculement par ce moyen étant trop élevé (20 secondes), il empêche tout déploiement possible avec un hyperviseur. Enfin, il n'est pas possible de reprendre la main sur la machine virtuelle après le basculement sans devoir la redémarrer et par conséquent perdre les informations stockées en mémoire RAM.

Au mieux, ce scénario peut permettre une gestion simple de sauvegarde de fichiers sur deux serveurs QNAPs et de basculer manuellement en cas de panne du premier serveur.

En vue de ces résultats, je décide de changer de système de fichiers de stockage et d'employer iSCSI qui semble offrir plus d'avantages dans ce type de scénario.

[62] Laboratoire www.tdeig.ch : Lab_KVM

## Scénario 2 : 1 QNAPs en iSCSI, 2 switchs et 1 PC KVM




Le QNAP est configuré comme serveur iSCSI, 2 switchs GS116E avec ports 1Gbit/s et le PC client est un hyperviseur KVM tournant sur un système d'exploitation Fedora 18.

Je vais tester si cette solution est fonctionnelle.

Figure 35: Scénario 2 chapitre II

### Mise en place du serveur QNAP iSCSI et PC client iSCSI

Je commence par mettre en place la configuration de stockage iSCSI sur QNAP1[63] avec un LUN de 10 Go. Le choix de la taille du disque est purement un choix personnel n'ayant aucune influence sur la configuration.

Ce choix est à adapter selon les besoins, pour ma part cette taille est suffisante pour stocker l'image d'une machine virtuelle Ubuntu Serveur 12.04.1 LTS que je vais utiliser pour mes tests.

J'accède au disque iSCSI Target QNAP1 sur le PC client (KVM) en effectuant la configuration iSCSI Initiator[64], puis je termine par le formatage du disque en ext4 (choix personnel) depuis le PC client. Ensuite, j'effectue le montage et la copie de l'image de la VM Ubuntu Serveur en ligne de commandes.

Ces manipulations sont effectuées depuis le PC client. Je pars du principe que l'hyperviseur KVM est préinstallé sur le PC.

**-- Formatage du disque avec un système de fichier ext4**

```
root@localhost /]# mkfs -t ext4 /dev/sdb
```

**-- Montage du disque iSCSI sur le PC Client**

```
root@localhost /]# mkdir /mnt/DisqueVM  -- Création d'un nouveau dossier
-- sdb correspond au disque iSCSI (LUN de 10Go) sur QNAP1
root@localhost /]# mount -t ext4 /dev/sdb /mnt/DisqueVM
```

[63] Configuration QNAP cible iSCSI : http://www.qnap.com/index.php?lang=fr&sn=3503
[64] Annexe A.3 Configuration client iSCSI Fedora 18
Lien permettant de comprendre la configuration iSCSI par étapes : http://www.annoratuto.com/tutoriels/adminsitration-systeme/gnu-linux/red-hat-entreprise/24-la-technologie-san-avec-tgt-iscsi.html

**-- Copie de l'image de la VM sur le disque iSCSI**

```
root@localhost /]# cp /home/admin/Téléchargements/UbuntuServer.img
                   /mnt/DisqueVM/UbuntuServer.img
```

La copie de l'image de la VM Ubuntu d'une taille de 8,6 Go prend 1 minute.

Je termine mon installation par la configuration de la machine virtuelle[65] (Serveur Ubuntu) à l'aide du gestionnaire KVM en tenant compte du fait que l'image de la VM se trouve sur QNAP1.

### Performances obtenues

Pour utiliser l'outil Iperf lors de mes tests, je procède à son installation à l'aide du package IPKG[66] (Itsy Package Management System) sur le terminal Linux de QNAP par le biais d'une connexion ssh.

Voici les résultats obtenus sur le Linux QNAP, lors de l'envoi d'une charge utile de 954 MBytes par Iperf du PC client vers QNAP1.

**-- Commande effectuée sur le terminal du PC client**

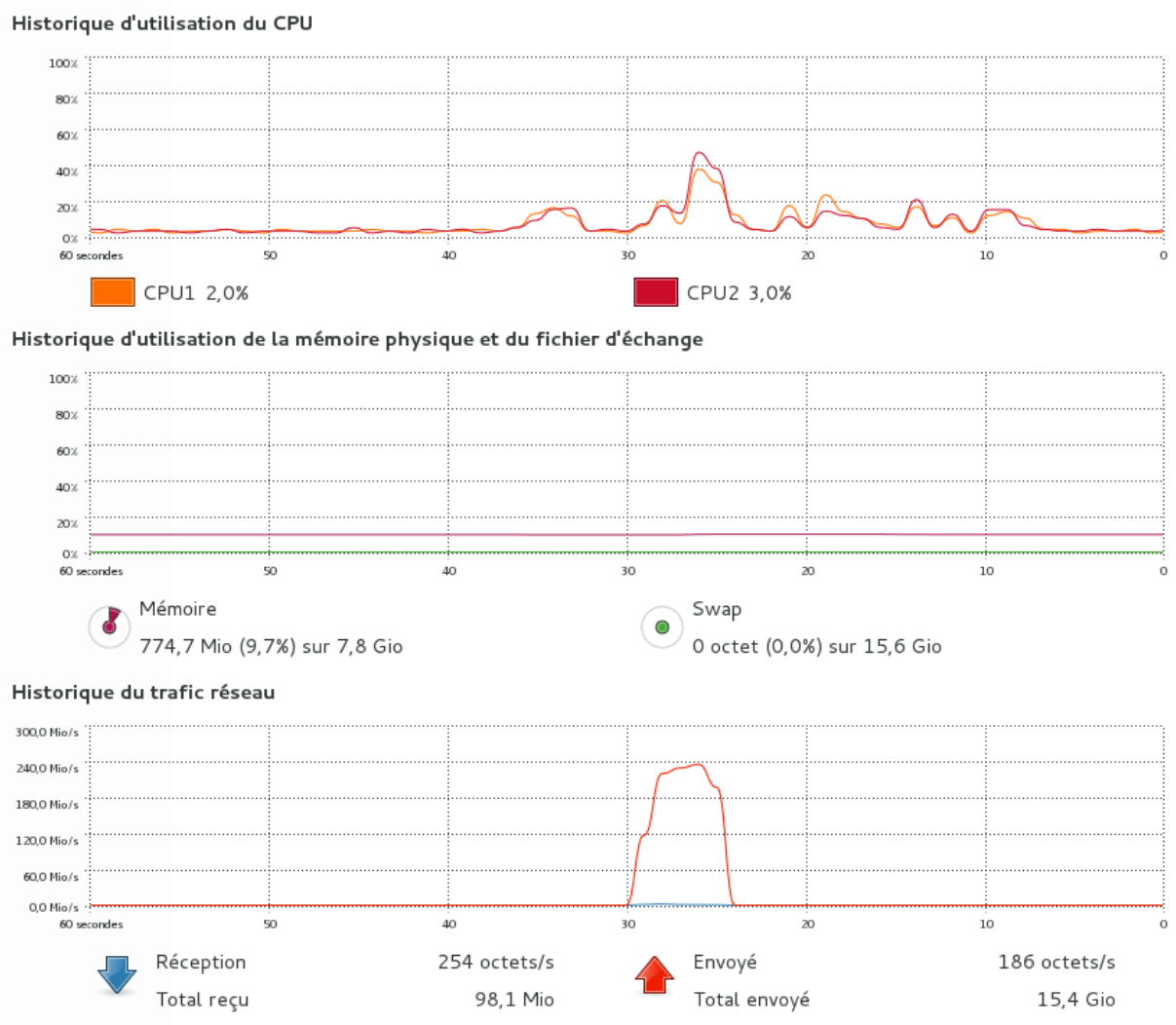
```
[root@localhost /]# iperf -n 1000000000 -t 4 -c 192.168.0.10
```

**-- Résultat de la commande sur le Linux QNAP**

```
 [~] # iperf -s
[  4] local 192.168.0.10 port 5001 connected with 192.168.0.2 port 34364
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-5.0 sec  954 MBytes  1.84 Gbits/sec

[  5] local 192.168.0.10 port 5001 connected with 192.168.0.2 port 34366
[ ID] Interval       Transfer     Bandwidth
[  5]  0.0-4.4.0 sec  954 MBytes  1.83 Gbits/sec

[  4] local 192.168.0.10 port 5001 connected with 192.168.0.2 port 34368
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-4.4 sec  954 MBytes  1.83 Gbits/sec

[  5] local 192.168.0.10 port 5001 connected with 192.168.0.2 port 34372
[ ID] Interval       Transfer     Bandwidth
[  5]  0.0-4.4 sec  954 MBytes  1.84 Gbits/sec
```

[65] Laboratoire www.tdeig.ch : Lab_KVM
[66] Installation Iperf sur le Linux de QNAP : http://wiki.makeitfit.ch/IPKG (Qnap Linux)

**-- Charge CPU et RAM côté client**



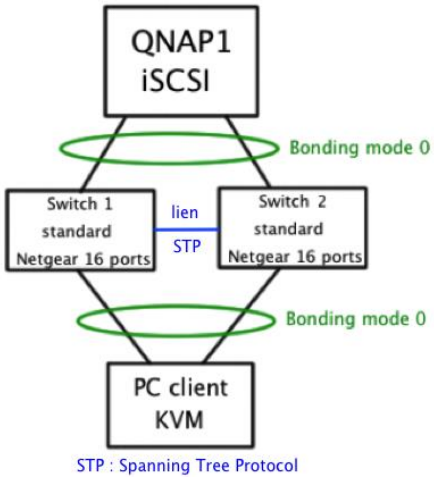

On remarque tout d'abord que le CPU est utilisé à 50% de ses capacités durant les 5 secondes du transfert par Iperf de 954 MBytes de charge utile. Cet outil nous indique le débit binaire de notre réseau soit 1,84 Gbit/s.

### Comportement lors de la coupure de 1 lien sur le QNAP1 (panne partielle)

En cas de perte d'un lien, les performances sont très dégradées à cause de la retransmission de paquets. C'est exactement la même problématique rencontrée lors des tests d'un scénario du mode 0[67].

[67] Chapitre I – Tests mode 0 Scénario 3 : Résultats lors de la coupure d'un lien sur PC2

### Solution pour palier au problème de pertes de performance




La solution consiste à relier les deux switchs entre eux avec un lien standard. Les switchs implémentant à la base le protocole Spanning Tree[68] vont éviter les boucles dans les réseaux tant qu'il n'y a pas de coupure au niveau des liens du PC client KVM et du QNAP 1.

Si une coupure venait à se produire, le switch1 ou 2 redirige le flux vers l'autre switch à l'aide de la MAC adresse et les performances deviennent celles d'un lien Gbit/s soit 950Mbit/s.

### Conclusion générale pour le scénario 2

Les performances avec ce scénario sont assurées avec iSCSI pour le bon fonctionnement d'un client KVM. Il n'y a plus de dégradation en effectuant une redondance supplémentaire au niveau des switchs. De plus, il est possible de supporter au maximum deux pannes de liens en même temps. Une sur le PC client KVM et l'autre sur le QNAP1.

Cependant, ce scénario ne permet pas d'effectuer une redondance des données avec deux QNAPs et aucun basculement n'est possible.

## Scénario 3 : 2 QNAPs en iSCSI Mirror, 2 switchs et 1 PC KVM

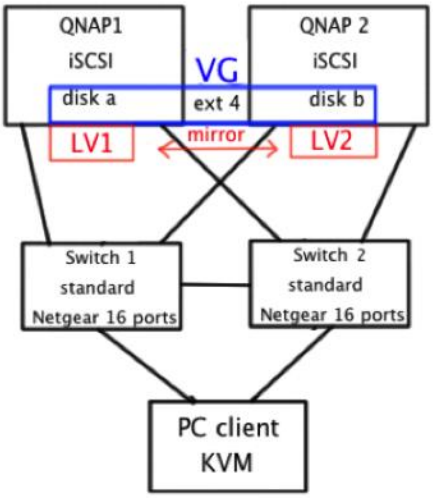



Figure 36: Scénario 3 Chapitre II

Les QNAPs sont configurés comme serveurs iSCSI, 2 switchs GS116E avec ports 1Gbit/s reliés entre eux. Le PC client est un hyperviseur KVM tournant sur un système d'exploitation Fedora 18. Le PC Client et les QNAPs sont en bonding mode 0.

Dans ce dernier scénario, je vais tester qu'il est possible d'effectuer un basculement entre les 2 QNAPs en cas de panne. Ce changement ne doit pas affecter le comportement d'une VM (Ubuntu Serveur 12.04.1 LTS) fonctionnant en même temps sur hyperviseur.

[68] Excellente explication avec le même scénario du protocole Spanning Tree : http://reussirsonccna.fr/le-spanning-tree-et-ses-3-problemes/

### Description de la solution

Les diverses actions sont effectuées dans le terminal du PC client KVM.

La solution consiste dans la création d'un Volume Group (VG) composé des 2 disques iSCSI (disk a 10Go et disk b 10Go) de chaque QNAP. Ensuite, je crée à partir de ce VG 2 Volumes Logiques (LV) de même taille (9Go). En utilisant une commande Linux, je vais effectuer un mirroring des 2 LV[69].

Pour finir, je vais formater mon disque iSCSI de 9Go en ext4 (choix personnel), le monter et y transférer ma machine virtuelle Ubuntu Serveur 12.04.1 LTS que je pourrais par la suite configurer et démarrer sur l'hyperviseur KVM du PC.

### *Mise en place du scénario 3*

**-- Etape 1 : Configuration iSCSI sur QNAP1, QNAP2 et PC KVM**

Je commence par configurer les serveurs iSCSI[70] QNAP1 et QNAP2 avec un LUN de 10 Go chacun à l'aide de la marche à suivre du fabriquant.

Au niveau du PC client KVM, j'effectue la configuration iSCSI Initiator selon l'annexe A.3[71] de ce mémoire.

```
-- Vérification des deux disques iSCSI
[root@localhost admin]# cat /proc/partitions
major minor  #blocks  name

   8       16   10485760 sdb          -- Disque iSCSI QNAP1
   8       32   10485760 sdc          -- Disque iSCSI QNAP2
```

**-- Etape 2 : Création du VG et du LV Mirror**

```
-- Création du volume groupe avec les 2 disques
[root@localhost admin]# vgcreate vgKVM /dev/sdb /dev/sdc
-- Commande permettant de vérifier si le volume group a été crée
[root@localhost admin]# vgdisplay

-- Création des volumes logiques et mirroring
[root@localhost admin]# lvcreate -L 9G -n lvKVM1 -m 1 vgKVM
-L = taille fixe, -n = name du LV, -m 1 = mirror des deux côtes

-- Commande permettant de voir la progression en pourcentage
[root@localhost admin]# lvs
[root@localhost admin]# lvs -a -o +devices
```

Il faut attendre que la réplication se termine entre les 2 Volumes Logiques créés. Compter 5 minutes pour 9Go.

---

[69] Redhat, Creating Mirrored Volumes : https://access.redhat.com/site/documentation/en-US/Red Hat Enterprise Linux/6/html/Logical Volume Manager Administration/mirror create.html
[70] Configuration QNAP cible iSCSI : http://www.qnap.com/index.php?lang=fr&sn=3503
[71] Annexe A.3 Configuration client iSCSI Fedora 18

```
-- Commande permettant de voir ou est stocké chaque LV après le mirroring
[root@localhost admin]# lsblk
NAME                              MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sdb                                 8:16   0    10G  0 disk
└─vgKVM-lvKVM1_mimage_1 (dm-8)    253:8    0     9G  0 lvm
  └─vgKVM-lvKVM1 (dm-9)           253:9    0     9G  0 lvm
sdc                                 8:32   0    10G  0 disk
├─vgKVM-lvKVM1_mlog (dm-6)        253:6    0     4M  0 lvm
│ └─vgKVM-lvKVM1 (dm-9)           253:9    0     9G  0 lvm
└─vgKVM-lvKVM1_mimage_0 (dm-7)    253:7    0     9G  0 lvm
  └─vgKVM-lvKVM1 (dm-9)           253:9    0     9G  0 lvm
```

**-- Etape 3 : Formatage, montage du LV puis transfert de la VM**

```
-- Formater /dev/vgKVM/lvKVM1
[root@localhost admin]# mkfs -t ext4 /dev/vgKVM/lvKVM1

-- Montage du volume
[root@localhost admin]# mkdir /mnt/DiskVM
[root@localhost admin]# mount -t ext4 /dev/vgKVM/lvKVM1 /mnt/DiskVM

-- Copie de l'image Ubuntu Serveur
[root@localhost admin]# cp /home/admin/Téléchargements/UbuntuServer.raw
                       /mnt/DiskVM
```

**-- Etape 4 : Démarrage sur l'hyperviseur de la VM**

L'image de VM transférée, il ne me reste plus qu'à la configurer sur l'hyperviseur KVM à l'aide du document du laboratoire KVM[72].

### *Tests lors de la coupure des 2 liens de QNAP1*

Pour ce test, la VM Ubuntu Serveur est démarrée sur le PC client KVM. Je coupe les 2 liens reliant le QNAP1 aux switchs.

La VM continue de marcher sans aucun problème. Je peux vérifier la coupure à l'aide de la commande *lsblk*.

```
[root@localhost admin]# lsblk
NAME                    MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sdb                       8:16   0    10G  0 disk
sdc                       8:32   0    10G  0 disk
└─vgKVM-lvKVM1 (dm-9)    253:9    0     9G  0 lvm
```

Le disque sdb qui correspond au QNAP1 est présent mais son statut a changé. Il n'est plus relié en mirroring à sdc.

Je crée un nouveau dossier dans la VM puis je la redémarre. Elle fonctionne correctement et le dossier est présent ce qui veut dire qu'il n'y a pas eu de perte de données lors du redémarrage.

[72] Laboratoire www.tdeig.ch : Lab_KVM.pdf

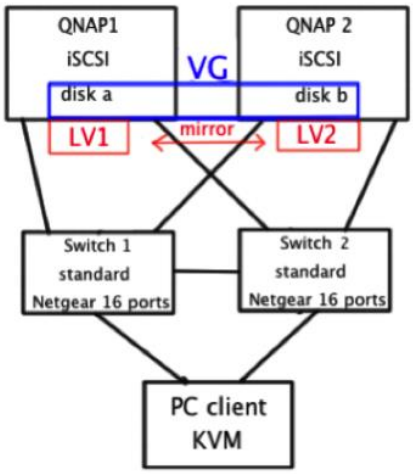
### *Récupération du LV et du mirroring*

Je reconnecte les 2 liens sur le QNAP1 selon le schéma de ce scénario. Puis récupère le volume sdb en relançant le mirroring entre les deux LVs.

```
-- Commande permettant de savoir si les LV sont actives dans le VG
[root@localhost admin]# vgchange -aay vgKVM

-- Rédemarrer le miroring après une coupure
[root@localhost admin]# lvconvert -m 1 /dev/vgKVM/lvKVM1
```

Le VG est relancé en quelques minutes (affichage d'un pourcentage). On peut vérifier la remise en ordre et le redémarrage du mirroring avec la commande `#lsblk`.

## Solution finale sans aucun système de fichiers disque




Le scénario est exactement le même ainsi que la configuration des composants et le mode de bonding utilisé. Cependant, ici je vais tester une solution sans fichiers système au niveau des volumes.

Cette solution permet d'avoir de meilleures performances au niveau des machines virtuelles.

**Figure 37: Scénario 3 Solution Finale**

### *Mise en place de la solution finale*

La mie en place est strictement la même que dans la partie Mise en place du scénario 3 d'avant. Cependant, ici il n'est plus nécessaire d'effectuer l'étape 3 concernant le formatage et le montage.

Je vais directement transférer mon image de la machine virtuelle UbuntuServer.raw bloc par bloc sur le volume disque à l'aide de la commande dd Linux.

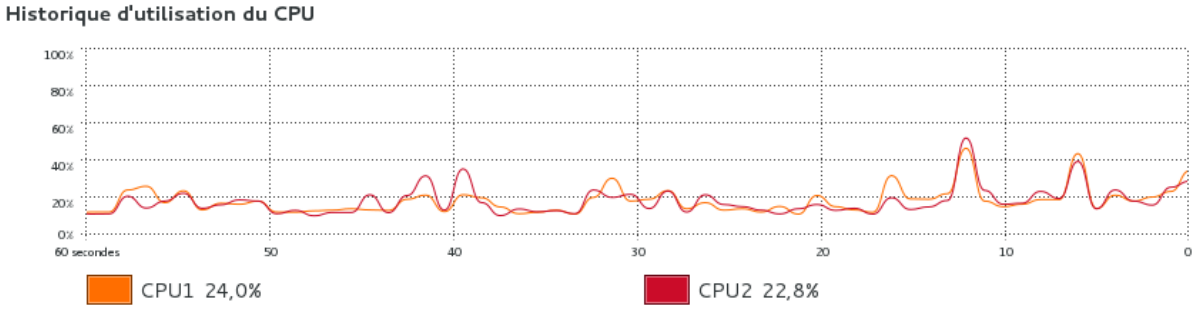
```
-- Copie des données UbuntuServer.raw dans lvKVM1
[root@localhost admin]# dd if=/home/admin/Téléchargements/UbuntuServer.raw
                        > of=/dev/vgKVM/lvKVM1
```

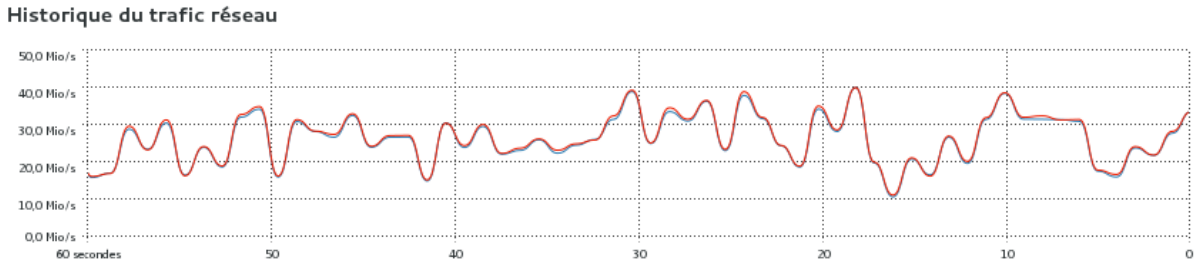
Comptez 6 minutes pour transférer 7,3 Go correspondant à l'image de la VM.

Pour configurer la VM au niveau de l'hyperviseur KVM, il faut chercher l'image dans le répertoire /dev/vgKVM/lvKVM1.

### *Tests effectués durant la reconstruction du mirroring après une panne de QNAP1*

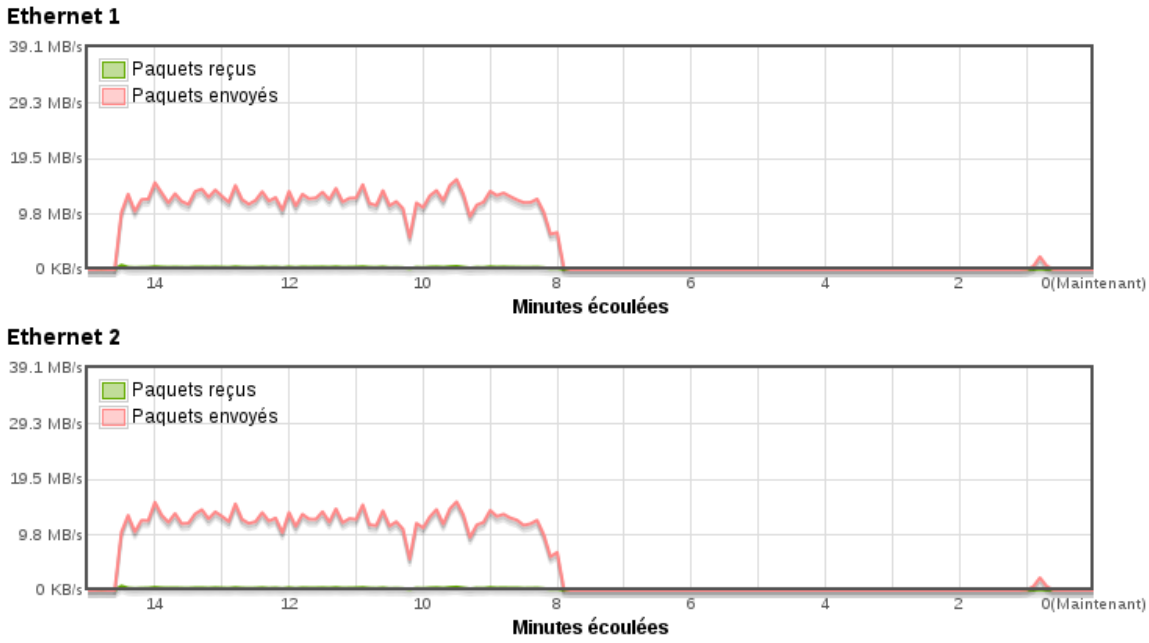
**-- Moniteur ressources système Fedora 18**




Utilisation des ressources CPU de 50% et un trafic réseau avoisinant les 40 Mo/s durant la reconstruction.

**-- Moniteur de ressources système QNAP1**




On peut observer sur cette capture le trafic engendré sur le deux interfaces Ethernet de QNAP1.

### Conclusion générale sur le scénario 3

Ce scénario est selon moi la proposition la plus intéressante pour arriver à remplir le cahier des charges qui m'a été proposé.

Il permet d'obtenir des performances de 1,87 Gbit/s avec la configuration bonding mode 0 sur les PCs client KVM ainsi que sur les QNAPs. Les pannes au niveau des liens sont supportées ainsi que le basculement entre les systèmes de stockage QNAPs.

Cette mise en œuvre permet la reprise et la gestion des machines virtuelles sans perte de données lors d'une panne.

Pour finir, les disques sont gérés sans un système de fichiers, ce qui apporte en théorie de meilleures performances au niveau des machines virtuelles.

# Difficultés rencontrées

## Difficultés sur le chapitre I : Bonding sur Fedora 18

- ***La prise en main sur un système Linux Fedora 18***

N'ayant jamais travaillé sur un système Linux Fedora 18, j'ai dû, dans un premier temps, m'habituer à certaines différences par rapport à un Linux Ubuntu, par exemple.

Pour prendre un exemple concret, le nom des interfaces logiques sont nommées de façon peu explicite (p16p1, p16p2, p1p1, p33p1) au lieu de eth0, eth1…

Cela peut paraître anodin comme problème, cependant dans mon cas, j'ai trouvé cela déstabilisant au départ quand je cherchais à effectuer une simple configuration ou un test. Pourtant, avec le temps on s'habitue et on ne fait plus réellement attention à ces différences d'écriture.

Par contre, je ne suis pas certain que dans le cas du Bonding, ce soit le système Linux le plus approprié au niveau production. C'est mon constat après utilisation et d'après mes recherches sur ce sujet.

- ***Lors de la configuration du Bonding sur Fedora 18***

La mise en place du Bonding sur les PCs (A29 et A34) avec la distribution Fedora 18 ne fut pas aisée. Il n'existe pas une méthode explicitement proposée pour cette version à l'heure actuelle.

Lors de mes recherches, je me suis inspiré de la méthode proposé pour la version 16 de Fedora en effectuant des changements lors de mes nombreux essais et des diverses méthodes proposées dans différents forums traitant de cette problématique.

J'ai été surpris de m'apercevoir que le problème principal empêchant l'installation était le service sensé la faciliter, le NetworkManager [73] . Ceci n'était pas indiqué sur la documentation en ligne sur le Bonding sur Fedora.org.

Une fois le service NetworkManager désactivé et remplacé par le service Network cela a facilité la mise en place de la configuration. Beaucoup de temps de perdu pour une information manquante dans la documentation.

[73] https://fedoraproject.org/wiki/Features/NetworkManagerBonding

- ***Difficultés générales lors de l'étude des modes de bonding***

L'étude des modes 0 et 6 ne fut pas simple. J'ai consacré énormément de temps, presque 7 semaines, pour arriver à comprendre le fonctionnement de ces deux modes.

Les raisons sont diverses. Tout d'abord, le manque de documentation technique précise sur le sujet. En effet, on retrouve toujours les mêmes documents retranscrits, sans une réelle valeur pédagogique et ne permettant d'avoir qu'une vague idée sur le fonctionnement général.

Pour exemple, la documentation du bonding sur Fedora est anecdotique, peut claire et voir incorrecte si on regarde la méthode de configuration qui est proposée. Il a donc fallu orienter mes recherches sur des forums de discussion pour chercher des réponses aux différents problèmes rencontrés dans cette étude.

La multitude de possibilités au niveau scénarios et le choix d'une méthodologie de tests fut compliquée et difficile à mettre en place. En effet, il n'est pas aisé d'expliquer les mécanismes sans devoir effectuer des acquisitions qui prennent énormément de tempos au niveau configuration.

Une fois les acquisitions obtenues, il a fallu analyser et schématiser le fonctionnement des deux modes de la façon la plus claire avec un niveau de détail élevé, comme on me l'a suggéré.

Les diverses acquisitions étant effectuées sur toutes les interfaces physiques des différents équipements, il ne fut pas simple, ni trivial, de démêler les différents échanges simultanés entre 8 interfaces Ethernet pour expliquer les différentes requêtes ARP et ICMP.

- **Avec le scénario 2 du mode 0[74]**

L'étude de ce scénario a posé des problèmes car tout laissait croire que le mode 0 n'exigeait pas de switch évolué. C'est vrai, mais pas pour ce scénario précis ! La dégradation au niveau des performances m'a longtemps laissé envisager à un problème matériel ou de configuration pour deux raisons.

Tout d'abord, les premiers tests effectués avec un switch Linux étaient cohérents au niveau fonctionnalité. Enfin, la dégradation au niveau du débit binaire était supérieure à un lien Gbit/s donc potentiellement possible.

J'ai pu me rendre compte de la subtilité de ce mode après avoir pu effectuer des tests avec un switch évolué (Cisco SG-300-28), prenant en charge la technologie d'agrégation de

[74] Chapitre Bonding sur Fedora 18 Scénario test 2 : 2 PCs Fedora 18 et un Switch

liens EtherChannel et en relisant plus en détail le chapitre 5 du document « Linux Ethernet Bonding  Driver »[75].

## Difficultés sur le chapitre II : Virtualisation et Stockage

- **Dans le scénario 1 avec NFS**

Les principaux problèmes rencontrés dans ce chapitre sont dus aux NFS_LOCKS qui empêchent tout démontage du dossier NFS après que QNAP 1 ne réponde plus.

J'ai cherché à trouver une solution avec NFS v.4, cependant à l'heure actuelle le mécanisme permettant d'effectuer un basculement n'est pas implémenté et reste théorique.

- **Dans le scénario 3**

Dans ce scénario, j'ai eu certains soucis par rapport aux commandes à utiliser ainsi que les étapes à effectuer pour pouvoir mettre en place les différents volumes. J'ai effectué des tests pas à pas pour aboutir à la bonne configuration.

La dernière difficulté fut de trouver comment redémarrer le mirroring des volumes logiques après une coupure. La documentation ne stipulant pas directement la solution cela m'a pris du temps avant de trouver.

[75] Linux Ethernet Bonding Driver : https://www.kernel.org/doc/Documentation/networking/bonding.txt

# Liens & références

- **Documentation Fedora en ligne**

  *Méthodologie à suivre pour la configuration des interfaces pour mettre en place le Bonding :*

  http://docs.fedoraproject.org/en-US/Fedora/16/html/System_Administrators_Guide/s2-networkscripts-interfaces-chan.html

- *Livre* « **Understanding Linux Network Internal** » *par Christian Benvenuti, Chapitre 8 : Device Registration and Initialization, p.166 à 169*

  *Comprendre le fonctionnement des outils MII-tool et Ethtool en détail :*

  http://books.google.ch/books?id=yy7tihZLgGYC&pg=PT192&lpg=PT192&dq=MII+or+ETHTOOL+ioctls&source=bl&ots=NDblgiAjh-&sig=nTtB7LagQbqwVQ3NxWJDWdBEWDU&hl=fr&sa=X&ei=l7qkUcmbPIflOqGqgMgI&ved=0CDQQ6AEwAQ

- **Documentation QNAP**

  Mode d'emploi QNAP Turbo NAS Manuel de l'utilisateur

  http://docs.qnap.com/nas/fr/index.html?home.htm

- **Documentation sur NFS**

  *RFC NFS v.4*

  http://www.citi.umich.edu/projects/nfsv4/rfc/draft-ietf-nfsv4-minorversion1-02.txt

  *Documentation Oracle versions NFS*

  http://docs.oracle.com/cd/E24843_01/html/E22298/rfsintro-27.html#scrolltoc

  *Basculement côté client en NFS v.4.1*

  http://docs.oracle.com/cd/E24843_01/html/E22298/rfsrefer-45.html#rfsrefer-51

  *Montage de serveurs et client en NFS 4*

  http://www.sysadminfr.org/fr/articles/debian%20nfsv4

- **Documentation VG et LV**

  *Redhat, Creating Mirrored Volumes*

  https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Logical_Volume_Manager_Administration/mirror_create.html

# Conclusion technique

## Chapitre I : Bonding sur Fedora 18

Après avoir étudié et mis en œuvre une configuration avec le driver bonding sur un système Linux Fedora 18, je peux apporter certaines conclusions.

Ce mécanisme de Channel bonding sur Linux est assez méconnu. Cependant, il mériterait d'être plus souvent utilisé et déployé car il permet d'obtenir des résultats très intéressants au niveau performances et gestion du débit binaire.

Le bonding mode 0 permet l'utilisation de plusieurs cartes Ethernet en simultané. La gestion au niveau des coupures des liens est possible et les performances en débit binaire peuvent aller jusqu'à 1,86 Gbit/s avec 2 cartes réseau.

Le mode 6 du bonding est plus adapté à des configurations du type serveur avec plusieurs connections clients en simultané. En effet, la gestion de la bande passante est adaptée par rapport à la charge instantanée.

Pour terminer, il n'est pas obligatoire d'investir sur un switch évolué pour pouvoir mettre en œuvre des configurations avec ces 2 modes. C'est un avantage non négligeable lors de la mie en place d'un système de stockage redondant basé sur le mode 0.

## Chapitre II : Virtualisation et Stockage

Dans ce chapitre, j'ai pu mettre en place un système redondant basé sur le bonding mode 0, iSCSI comme gestion de disques sur 2 SANs QNAP et du mirroring avec LVM pour permettre le basculement en cas de panne d'un des équipements.

J'ai aussi pu étudier le protocole NFS v.4 qui malheureusement ne permet pas la gestion en cas de Failover, ainsi que deux protocoles de réplication RSNC et RTRR.

Malgré que RTRR soit un protocole propriétaire de chez QNAP, il semble très adapté lorsqu'on souhaite effectuer une réplication entre QNAPs pour du stockage simple de fichiers.

Pour terminer, j'ai pu utiliser le système de gestion de volumes LVM sur Linux. Cet outil permet de modifier dynamiquement la taille des disques, sans mettre en péril les données. Parmi les fonctions les plus intéressantes, le mirroring de volumes logiques est une méthode très pratique.

# Conclusions personnelles

## Chapitre I : Bonding sur Fedora 18

Cette étude m'a permis, tout d'abord, de travailler sur un système Linux Fedora 18 pour la première fois. J'ai trouvé intéressant d'avoir pu utiliser une autre distribution que celle que j'utilise habituellement.

J'ai part la même occasion pu compléter et améliorer mes connaissances sur Linux, après un temps d'adaptation à la distribution Fedora.

Ainsi, j'ai pu configurer, tester et comprendre les mécanismes du bonding pour les modes 0 et 6.

L'étude fut difficile et longue tant le sujet est vaste et peu fourni au niveau de la documentation. Cependant, j'ai pris un certain plaisir dans cette partie recherche pour pouvoir expliquer le plus précisément les deux modes.

Personnellement, je ne pensais pas effectuer une étude aussi poussée sur le bonding. Cependant, je dois avouer que le sujet est fort intéressant.

A la fin de ce chapitre, je suis convaincu que le bonding est une solution très performante permettant une gestion à faible coût lors de la mise en place d'un système de stockage.

## Chapitre II : Virtualisation et Stockage

Ce chapitre m'a permis d'étudier différentes méthodes théoriques permettant d'obtenir au final un système de stockage redondant fonctionnel.

J'ai pu étudier et approfondir mes connaissances à propos des protocoles NFS et iSCSI au niveau pratique.

Cette étude m'a aussi appris à avancer de façon plus méthodique pour aboutir à une solution fonctionnelle.

Les méthodes de gestion des volumes avec LVM m'ont permis d'observer les possibilités d'utilisation étendues d'un système Linux dont je n'avais pas connaissance. Cela m'a été très utile pour compléter ma formation sur Linux.

Pour finir, j'ai pu travailler avec du matériel de qualité professionnelle tant au niveau des switchs que des 2 QNAPs que j'avais à disposition.

# A Annexes

## A.1 Caractéristiques des 7 différents modes de Bonding

***balance-rr or 0 (équilibrage de charge)***

Mode par défaut. Envoi des paquets en mode séquentiel de la première interface à la dernière. Permet d'effectuer de l'équilibrage de charge et de la tolérance aux pannes. Si une carte tombe en panne, elle ne sera plus utilisée mais les autres continueront l'activité. Débit binaire augmenté (somme des cartes Ethernet).

***active-backup or 1 (sauvegarde active)***

Une seule interface active à la fois. L'autre devient active, si et seulement si la première tombe en panne. Mode offrant une tolérance aux pannes. Le débit binaire correspond à celui possible par l'interface active.

***balance-xor or 2 (balance XOR)***

Transmission basée sur une politique de hachage. Une seule interface est employée à l'envoi vers la même adresse MAC. Transferts en parallèle avec cette méthode. Le choix de l'interface suit la règle : (Adresse MAC de la source XOR Adresse MAC de la destination) modulo nombre d'interfaces. Ce mode permet l'équilibrage de charge et de la tolérance aux pannes.

***broadcast or 3 (broadcast)***

Mode de transmission en broadcast vers toutes les interfaces actives. Mode offrant une tolérance aux pannes. Charge réseau importante.

***802.3ad or 4***

Utilisation de la spécification IEEE 802.3ad Dynamic Link Aggregation, élargissement dynamique au niveau du débit binaire transmis. Utilisation de toutes les interfaces avec création de groupes d'agrégation qui partagent le même débit binaire et les mêmes spécifications. Demande des prérequis au niveau des switchs.

***balance-tlb or 5 (traffic load balancing)***

Transmission avec équilibrage de charge adaptatif. Pas besoin de prérequis par rapport aux switchs. Trafic en envoi distribué en fonction de la charge de chaque interface calculé

selon son débit. En cas de pannes, l'autre esclave prend l'adresse MAC de l'esclave en panne automatiquement.

***balance-alb or 6 (adaptive load balancig)***

Transmission avec équilibrage de charge adaptatif. Mode étendu du mode 5 qui inclut le load balancing aussi en réception. La réception lors de l'équilibrage de charge est obtenue par négociation ARP. Le module intercepte les réponses ARP et change l'adresse MAC par celle d'une de ses interfaces actives. Pas besoin de prérequis par rapport aux switchs.

## A.2 Tableau comparatif des différents modes de bonding

Voici l'explication des spécificités se trouvant dans la première colonne du tableau qui suit :

- *Tolérance aux pannes :* ce mode est capable de détecter et de gérer une ou plusieurs coupures de liens sans pertes de données.

- *Equilibrage de charge (load-balancing) :* ce mode est capable repartir la charge sur les différents liens actifs lors d'une transmission entrante ou/et sortante.

- *Prérequis switch évolué :* si on connecte un PC configuré en bonding sur un switch, ce mode ne fonctionnera pas sur un switch standard mais aura besoin d'un commutateur gérant le protocole IEEE 802.3ad.

- *n interfaces employées :* nombre d'interfaces physiques pouvant être mises en place lors de la configuration.

- *n interfaces actives bonding :* nombre d'interfaces qui sont activés dans ce mode dans une situation standard.

- *Le Débit binaire est augmenté* : ce mode permet d'augmenter le débit binaire lors de l'utilisation de plusieurs liens.

- *Charge réseau :* ce mode fait augmenter la charge réseau de façon importante par son fonctionnement. L'envoi de trames est effectué en broadcast sur toutes les interfaces.

- *Avantage :* Principal avantage de ce mode par rapport aux autres.

- *Inconvénient :* Le principal désavantage par rapport aux autres modes.

**Tableau comparatif des différents modes de Bonding**

| Spécif. \ Modes | 0 *balance-rr* | 1 *active-backup* | 2 *balance-xor* | 3 Broadcast | 4 *802.3ad* | 5 *traffic load balancing* | 6 *adaptive load balancig* |
|---|---|---|---|---|---|---|---|
| **Tolérance aux pannes** | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| **Equilibrage de charge (Load Balancing)** | ✓ sortant[1] | — | ✓ sortant[1] | — | — | ✓ sortant[1] | ✓ sortant[1] et entrant[2] |
| **Prérequis switchs évolué** | — | — | — | — | ✓ | — | — |
| **n interfaces employées** | 2..n | 2 | 2..n | 2..n | 2..n | 2..n | 2..n |
| **n interfaces actives bonding** | 2..n | 1 | 1 | 2..n | 2..n | 1 au départ | 1 au départ |
| **Le débit binaire est augmenté** | ✓ | — | — | — | ✓ | ✓[3] | ✓[3] |
| **Charge réseau** | — | — | — | ↑ | — | — | — |
| **Avantage** | Débit binaire augmenté | Redondance avec basculement | Transferts parallélisés | Données transmises sur toutes les interfaces | Elargissement dynamique de la bande pasante | Equilibrage de charge pour le trafic sortant | Equilibrage de charge en envoi et en réception |
| **Inconvénient** | Réception sur une seule interface | 1 seul lien actif à la fois | 1 interface affectée à l'envoi vers une même adresse MAC | Réseau chargé | Besoin d'un switch évolué | Pas d'équilibrage de charge pour le trafic entrant | A besoin d'un switch pour fonctionner |

[1] Sortant depuis l'équipement configuré avec le bonding
[2] Entrant vers l'équipement configuré avec le bonding
[3] La débit binaire est augmenté à partir de 2 interfaces actives. Par exemple dans le cas d'une deuxième connexion.

## A.3 Configuration client iSCSI Fedora 18

**-- Installer iSCSI tools et lancer le service**

```
# yum -y install iscsi-initiator-utils
# service iscsi start
```

**-- Modifier le fichier pour les droits d'accès au serveur iSCSI (configuration facultative)**

```
# vi /etc/iscsi/iscsid.conf
```

**-- Découvrir le serveur target**

```
# iscsiadm -m discovery -t sendtargets -p IP_Serveur_Target
# chkconfig iscsi on
# chkconfig iscsid on
```

**-- Vérifier le statut après découverte**

```
# iscsiadm -m node -o show
```

**-- Effectuer le login au près du serveur**

```
# iscsiadm -m node --login
```

**-- Vérifier l'établissement de la session**

```
 # iscsiadm -m session -o show
```

**-- Vérifier les partitions**

```
# cat /proc/partitions
```

## A.4 Descriptif des paramètres dans les interfaces

Ci-contre les paramètres possibles configurables au niveau des interfaces lors de la mise en place du bonding[76].

**DEVICE = *<nom>***

où ***<nom>*** est le nom logique de l'interface Ethernet

**ONBOOT = *<answer***

où ***<answer>*** est l'un des éléments suivants:

- **yes** – L'interface doit être activée au démarrage.
- **no** – L'interface ne doit pas être activée au démarrage.

**USERCTL = *<answer>***

où ***<answer>*** est l'un des éléments suivants:

- yes - les utilisateurs non-root sont autorisés à contrôler cette interface.
- no - les utilisateurs non-root ne sont pas autorisés à contrôler cette interface.

**MASTER = *<bond-interface>***

où ***<bond-interface>*** est l'interface de liaison de chaîne à laquelle l'interface Ethernet est liée.

Utilisée avec le paramètre SLAVE en complément.

**SLAVE = *<bond-interface>***

où ***<bond-interface>*** est l'un des suivants:

- yes - Cet appareil est contrôlé par l'interface de liaison de canaux spécifiée dans le MASTER.
- no – Ce dispositif n'est pas contrôlé par l'interface de liaison de canaux spécifié dans le MASTER

Utilisée avec le paramètre MASTER en complément.

[76] http://www.centos.org/docs/5/html/Deployment_Guide-en-US/s1-networkscripts-interfaces.html

**BOOTPROTO = *<protocole>***

où ***<protocole>*** est l'un des suivants:

- none - Aucun protocole de démarrage ne doit être utilisé.
- Bootp – Le protocole BOOTP doit être utilisé.
- dhcp - Le protocole DHCP doit être utilisé.

**IPADDR = *<adresse>***

où ***<adresse>*** est l'adresse IP.

**NETMASK = *<masque>***

où ***<masque>*** est la valeur du masque de réseau.

**BONDING_OPTS = <options>**

où <options> est une ou plusieurs paramètres suivants :

**mode**

(0 à 6) définit quel mode va être utilisé.

**miimon**

Définit la fréquence des requêtes MII link monitoring en millisecondes pour permettre de déterminer si le lien est actif. Il est conseillé de positionner cette valeur à 100. 0 est la valeur par défaut.

**arp_interval**

Définit le délai en millisecondes entre chaque requête monitor ARP compatible avec le mode 0 ou 2. Si la valeur est à 0 alors ARP monitoring est désactivé. 0 est la valeur par défaut.

**primary**

Option utilisable pour les modes actif-passif. Favorise une interface dans un agrégat. Si celle-ci venait à redevenir active, elle prend la main sur les autres.

**updelay**

Définit le temps en millisecondes pour qu'une interface soit détectée comme active. 0 est la valeur par défaut.

## A.5 Description des outils mii-tool et ethtool

Mii-tool (issue du package net-tools) comme ethtool sont historiquement deux packages sur Linux permettant la gestion et la configuration des cartes réseau en ligne de commande.

Le choix de l'utilisation de l'un ou l'autre de ses outils dépend de l'ancienneté de la carte que l'on souhaite configurer.

L'outil mii-tool (Media Independent Interface) a été déprécié en faveur d'ethtool. En effet, il ne prenait pas en charge les cartes Gigabit. Cependant, une mise à jour de l'outil permet actuellement de l'utiliser avec les cartes de nouvelle génération.

Dans le cas du bonding, le paramètre miimon se trouvant dans les options de configuration de bond0 va faire appel à ces outils.

Ils sont les seuls à pouvoir accéder aux registres MII et ETHTOOL se trouvant sur les cartes réseau. Le paramètre miimon, permet ainsi de contrôler selon une fréquence (ms) déterminée si la carte, donc le lien, est toujours actif.

Si la carte réseau ne supporte pas les registres MII et ETHTOOL, on ne pourra pas détecter la déconnexion du lien.

La solution est alors de sonder la carte à l'aide de l'envoi de requêtes ARP.

Les différents liens proposés en bas de page[77] permettent de comprendre au besoin le fonctionnent de ses deux outils de façon détaillée.

Je conseille les pages 166 à 169 du livre « **Understanding Linux Network Internal** » *de Christian Benvenuti, Chapitre 8 : Device Registration and Initialization* qui donnent explique en détail MII et Ethtool :

http://books.google.ch/books?id=yy7tihZLgGYC&pg=PT192&lpg=PT192&dq=MII+or+ETHTOOL+ioctls&source=bl&ots=NDblgiAjh-&sig=nTtB7LagQbqwVQ3NxWJDWdBEWDU&hl=fr&sa=X&ei=l7qkUcmbPIflOqGqgMgI&ved=0CDQQ6AEwAQ

[77] Manipulation de paramètres de carte réseau avec mii-tool et ethtool
http://www.k-tux.com/manipulation-de-parametres-de-carte-reseaux-mii-tool-ethtool
Determine Ethernet link with mii-tool et Ethtool
http://blog.bottomlessinc.com/2011/02/using-ethtool-to-determine-ethernet-link-speed-not-mii-tool/

## A.6 Configuration d'un bridge sur Fedora 18[78]

```
-- Pour savoir si le bridge-utilities fonctionne correctement
# brctl

-- Créer un bridge device qui s'appelle bridge0
# brctl addbr bridge0

-- N'attribuer aucune IP aux 4 interfaces
# ifconfig p16p1 0.0.0.0
# ifconfig p16p2 0.0.0.0
# ifconfig p1p1 0.0.0.0
# ifconfig p33p1 0.0.0.0

-- Ajouter les interfaces au bridge
# brctl addif bridge0 p16p1
# brctl addif bridge0 p16p2
# brctl addif bridge0 p1p1
# brctl addif bridge0 p33p1

-- Activer le bridge
# ifconfig bridge0 up

-- Attribuer une adresse IP au bridge si besoin
# ifconfig bridge0 xxx.xxx.xxx.xxx netmask xxx.xxx.xxx.xxx

-- Enlever le filtre pour laisser tout le trafic
# cd /proc/sys/net/bridge
# ls
 bridge-nf-call-arptables  bridge-nf-call-iptables
 bridge-nf-call-ip6tables  bridge-nf-filter-vlan-tagged
# for f in bridge-nf-*; do echo 0 > $f; done

-- Vérifier le fonctionnement du bridge et des interfaces
# brctl show bridge0

-- Vérifier le fonctionnement au niveau MAC adresses
# brctl showmacs bridge0
```

**Remarque :** Il est fortement conseillé de désactiver ou de désinstaller NetworkManager pour des questions de conflit réseau.

[78] The Linux Foundation – Bridge
http://www.linuxfoundation.org/collaborate/workgroups/networking/bridge
Configuring a Bridge in Linux
http://www.6test.edu.cn/~lujx/linux_networking/0131777203_ch12lev1sec3.html

## A.7 Configuration de deux bridges sur Fedora 18

```
-- Créer un bridge device qui s'appelle bridge0
# brctl addbr bridge0
-- Créer un bridge device qui s'appelle bridge1
# brctl addbr bridge1
-- N'attribuer aucune IP aux 4 interfaces
# ifconfig p16p1 0.0.0.0
# ifconfig p16p2 0.0.0.0
# ifconfig p1p1 0.0.0.0
# ifconfig p33p1 0.0.0.0
-- Ajouter les interfaces sur chaque bridge
# brctl addif bridge0 p16p1
# brctl addif bridge0 p16p2
# brctl addif bridge1 p1p1
# brctl addif bridge1 p33p1
-- Activer le bridge0
# ifconfig bridge0 up
-- Activer le bridge1
# ifconfig bridge1 up
-- Attribuer une adresse IP au bridge si besoin
# ifconfig bridgeX xxx.xxx.xxx.xxx netmask xxx.xxx.xxx.xxx
-- Enlever le filtre pour laisser tout le trafic
# cd /proc/sys/net/bridge
# ls
 bridge-nf-call-arptables  bridge-nf-call-iptables
 bridge-nf-call-ip6tables  bridge-nf-filter-vlan-tagged
# for f in bridge-nf-*; do echo 0 > $f; done
```

## A.8 Spécifications des systèmes QNAP

### *QNAP TS-459 Pro II (SAN – NAS)*

- Système de stockage 4 baies
  - Intel Atom 1.8 GHz Dual Core, 3 GB RAM
  - Noyau Linux 2.6, 2 NIC 1 Gbit/s
- Disque WD6000HLHX
  - Capacité = 600 GB
  - Interface SATA 6 Gbit/s
  - Vitesse de rotation = 10'000 rpm
  - Cache de 32 MB
  - Data Transfer Rate = 145 MB/s
  - MTBF = 1.4 mio heures
  - 600'000 cycles arrêt-démarrage
  - Consommation (W) = 4.3 (repos) = 6.4 (RW)
  - Température de fonctionnement = 5-55°C
  - Prix = 210.- en mai 2012


## A.9 Détection visuelle d'une coupure de lien ou de réseau sur la façade du QNAP


Indication que le lien Ethernet 1 est coupé du réseau.


Indication que les 2 liens Ethernet du QNAP sont coupés du réseau.

## A.10 Users names et logins utilisés sur les équipements

**PCs Linux Fedora 18**

USER NAME : admin

LOGIN : admin

**QNAP1 (XX= 10) et QNAP2 (XX=20)**

USER NAME : admin

LOGIN : admin

Connexions en ssh dans terminal : ssh admin@192.168.0.xx

LOGIN : admin

**Machine Virtuelle Ubuntu Serveur 12.04.1 LTS**

USER NAME : labotd

LOGIN : labolabo