Agrégation de liens Ethernet

Bonding sur Linux et
Etherchannel sur switch Cisco

Documentation version 1.0 créé le 19 avril 2005

Dernière mise à jour le 21 avril 2005

Auteurs :

Haugeard Laurent laurent@cri74.org

Table des matières

1 – La problématique

Il s'agit de faire en sorte que notre serveur de backup dispose d'une capacité supérieure à 1Gbps pour sa connexion Ethernet. Le serveur possède 2 interfaces Ethernet gigabits intégrées à la carte mère, il est raccordé sur un switch Cisco 2970 doté de 24 ports cuivres 10/100/1000

Sur le switch Cisco nous groupons les ports Ethernet en utilisant la technologie « Etherchannel ». La configuration correspondante sur le serveur Linux s'appuie le module « bonding ».

Ce document rend compte des points suivants:

- présentation des technologies
- description de la configuration
- vérification du comportement sur rupture de lien en pleine charge
- vérification de la répartition de charge

Des courbes du trafic mésuré sur les ports du switch Cisco 2970 sont réalisées avec Cricket. Nous dirons quelques mots sur la configuration de Cricket.

2 – La technologie et les composants

2.1 – Etherchannel

Le mécanisme mis en oeuvre pour grouper des ports sur les switchs Cisco est **Etherchannel** (nommé trunk chez d'autres constructeurs).

Il s'agit de créer une interface logique (PortChannelx, x étant le numéro du port) associée à plusieurs ports physiques. Etherchannel se charge de:

- faire du load balancing en répartissant le trafic sur les ports physiques.
- Faire du failover en renvoyant immédiatement le trafic sur un autre port du groupement si un port physique « tombe ».

Etherchannel permet aussi d'appliquer de façon globale des configurations sur tous les ports d'un même groupement

Pour qu'un groupement de ports fonctionne, il faut que tous les ports aient les mêmes caractéristiques (vitesse, full ou hallf duplex), et ce, en local sur le switch mais aussi sur l'équipement auquel le switch est raccordé. De plus, il va de soit que tous les ports d'un groupement appartiennent au même VLAN ou sont configurés 802.1q.

Etherchannel est une technologie qui fonctionne en autonome sur le switch. Deux équipements raccordés entre eux doivent avoir une configuration parfaitement symétrique sous peine de dysfonctionnement (attention aux boucles sur le réseau Ethernet).

En complément de Etherchannel, afin de rendre plus transparente la mise en place des mécanismes de groupement de ports, des protocoles existent pour que les équipements dialoguent entre eux afin de décider si un port peut faire partie d'un groupement. Ces protocoles sont les suivants:

- LACP (normalisé IEEE 802.3ad)
- PAgP (propriétaire Cisco)

2.2 – Bonding

Coté linux, Etherchannel s'appelle **bondind** et se matérialise sous la forme d'un module disponible en standard dans le noyau. Comme pour le switch, on regroupe plusieurs interfaces physiques (ethx) sur une interface logique (nommée bondx, x est le numéro de l'nterface). Les solutions offertes par le bonding sont nombreuses:

Solution	*Description*
Round Robin	Groupement de ports pour load balancing et Failover: Le trafic sortant de l'interface est émis alternativement sur chacune des interfaces. Les ports du switch sont groupés avec Etherchannel. Par contre la configuration est statique sans protocole PagP ou LACP (802.3ad).
Active backup	Failover seulement: Seule une des interfaces physique est active et répond au requêtes arp. Pas de propriétées particulière sur le switch.
Balance-XOR	Groupement de ports pour load balancing et Failover: Le switch est configuré en Etherchannel. Dans le sens sortant, le serveur Linux choisi l'interface physique en fonction de l'adresse mac (source ou destination, ou XOR sur les deux). Sur le switch Cisco, Etherchannel fonctionne aussi sur ce principe (le load balancing effectué sur le switch concerne le trafic entrant sur le serveur Linux).
Broadcast	Failover seulement: Le trafic est transmis sur toutes les interfaces physiques.
IEEE 802.3ad	Groupement de ports pour load balancing et Failover: Fonctionne les switchs Ethernet qui supportent cette norme. Le mécanisme de load blancing est similaire à celui du mode Balance-XOR.Il est basé sur le principe qui consiste à affecter toujours le même chemin à la même machine.

Solution	*Description*
Adaptive transmit load balancing (balance-tlb)	Load balancing dans le sens sortant uniquement: Le trafic sortant de l'interface est émis de sur l'une ou l'autre des interfaces physique en fonction de le charge. Dans le sens entrant, une seule des interfaces physique répond au requêtes ARP. Aucune configuration particulière n'est nécessaire sur le switch. Les drivers de la carte ethernet sur le serveur Linux doivent être compatible ethool.
Adaptative load balancing (balance-alb)	Load balancing dans les deux sens. Reprend le mécanisme balance-tlb dans le sens sortant. Dans le sens entrant, le bonding intercepte les requêtes ARP pour renvoyer alternativment l'adresse des interfaces phyisiques.

Tous ces modes sont listés et expliqués dans la documentation du module bonding qui est livrée avec les sources du noyau. Cette même documentation est aussi accessible dans sa toute dernière mouture sur le site http://www.sourceforge.net/projects/bonding

3 – Configuration

3.1 – Interconnexion 802.3ad

3.1.1 – Coté Cisco

Sur le switch Cisco la configuration est relativement simple. Il suffit d'ajouter le port au groupement Etherchannel en appliquant la commande **channel-group** en mode interface

```
switch18(config)# int gi 0/6
switch18(config-if)#channel-group ?
  <1-12>  Channel group number

switch18(config-if)#channel-group 2 ?
  mode  Etherchannel Mode of the interface

switch18(config-if)#channel-group 2 mo
switch18(config-if)#channel-group 2 mode ?
  active     Enable LACP unconditionally
  auto       Enable PAgP only if a PAgP device is detected
  desirable  Enable PAgP unconditionally
  on         Enable Etherchannel only
  passive    Enable LACP only if a LACP device is detected
switch18(config-if)#
switch18(config-if)#channel-group 2 mode active
switch18(config-if)#
```

Dans cet exemple nous ajoutons au groupement de ports numéro 2 le port 6 du switch18.

L'option « mode active» permet de préciser que nous utilisons le protcole LACP du standard 802.3ad.

Un interface logique nommée Port-Channel2 est associé au groupement de ports.

```
switch18#sh int po2
Port-channel2 is up, line protocol is up (connected)
  Hardware is EtherChannel, address is 0011.bb4c.1305 (bia 0011.bb4c.1305)
  MTU 1500 bytes, BW 1000000 Kbit, DLY 10 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation ARPA, loopback not set
  Full-duplex, 1000Mb/s
  input flow-control is off, output flow-control is off
  Members in this channel: Gi0/5 Gi0/6
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input never, output 00:00:01, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: fifo
  Output queue: 0/40 (size/max)
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 2000 bits/sec, 2 packets/sec
     3440516380 packets input, 2256460402 bytes, 0 no buffer
     Received 16048 broadcasts (0 multicast)
     0 runts,  ............
```

L'interface logique est visible exactement comme une interface physique. Dans cette exemple nous voyons qu'il s'agit d'une interface Etherchannel qui regroupe les ports physiques Gi0/5 et

Gi0/6.

La commande **show etherchannel** permet de voir l'état des groupement de ports

```
switch18#show etherchannel summary
Flags:  D - down        P - in port-channel
        I - stand-alone s - suspended
        H - Hot-standby (LACP only)
        R - Layer3      S - Layer2
        u - unsuitable for bundling
        U - in use      f - failed to allocate aggregator
        d - default port

Number of channel-groups in use: 2
Number of aggregators:           2

Group  Port-channel  Protocol    Ports
------+-------------+-----------+-----------------------------------------------
1      Po1(SD)         LACP      Gi0/10(D)   Gi0/11(D)   Gi0/12(D)
                                 Gi0/13(D)   Gi0/14(D)
2      Po2(SU)         LACP      Gi0/5(P)    Gi0/6(P)
```

3.1.2 – Coté Linux

Il faut charger le module bonding avec les bonnes options:

```
# bonding miimon=100 mode=4
```

mode=4 : signifie que nous utilisons le mode 802.3ad

miimon=100: une vérification de l'état des interfaces physiques est effectuée toutes les 100ms.

Cela crée une interface nommée « bond0 » à laquelle il faut joindre les interfaces physiques en utilisant la commande **ifenslave** dont le code source est intégré au noyau linux.

Nous utilisons une distribution debian sarge , voici ce que cela donne:

- Chargement du module au démarrage dans fichier de configuration « /etc/modules »:

```
node06bis:~# more /etc/modules
# /etc/modules: kernel modules to load at boot time.
#
# This file should contain the names of kernel modules that are
# to be loaded at boot time, one per line.  Comments begin with
# a "#", and everything on the line after them are ignored.

ide-cd
ide-detect
bonding miimon=100 mode=4
```

- Configuration IP et ajout des interfaces phyisiques dans le fichier de configuration du réseau « /etc/network/interfaces »:

```
node06bis:~# more /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# Fisrt bring up physical interfaces
auto eth0
auto eth1

# The primary network interface
auto bond0
iface bond0 inet static
        address 195.202.0.46
        netmask 255.255.255.128
        network 195.202.0.0
        broadcast 195.202.0.127
        gateway 195.202.0.1
        # dns-* options are implemented by the resolvconf package, if installed
        dns-nameservers 195.202.0.2
        dns-search cur-archamps.fr
        up /sbin/ifenslave bond0 eth1 eth0
```

La commande **ifenslave** est packagée sous debian.

```
node06bis:~# apt-cache search ifenslave
ifenslave - Attach and detach slave interfaces to a bonding device
ifenslave-2.4 - Attach and detach slave interfaces to a bonding device
ifenslave-2.6 - Attach and detach slave interfaces to a bonding device
```

la verison 2.x correspond à la version du noyau. Le package ifenslave permet d'avoir la commande sous son un nom indépendant de la version installé en créant un lien symbolique.

3.2 – Interconnexion en round robin

Le serveur Linux sait envoyer alternativement les paquets sur les différentes interfaces physiques d'un groupement de ports. La configuration est la suivante:

Sur le switch Cisco il faut créer un groupement Etherchannel qui n'utilise pas le protocole LACP pour négocier l'appartenance d'un port phyisque au groupement

La configuration est statique, elle est appliquée avec la commande suivante:

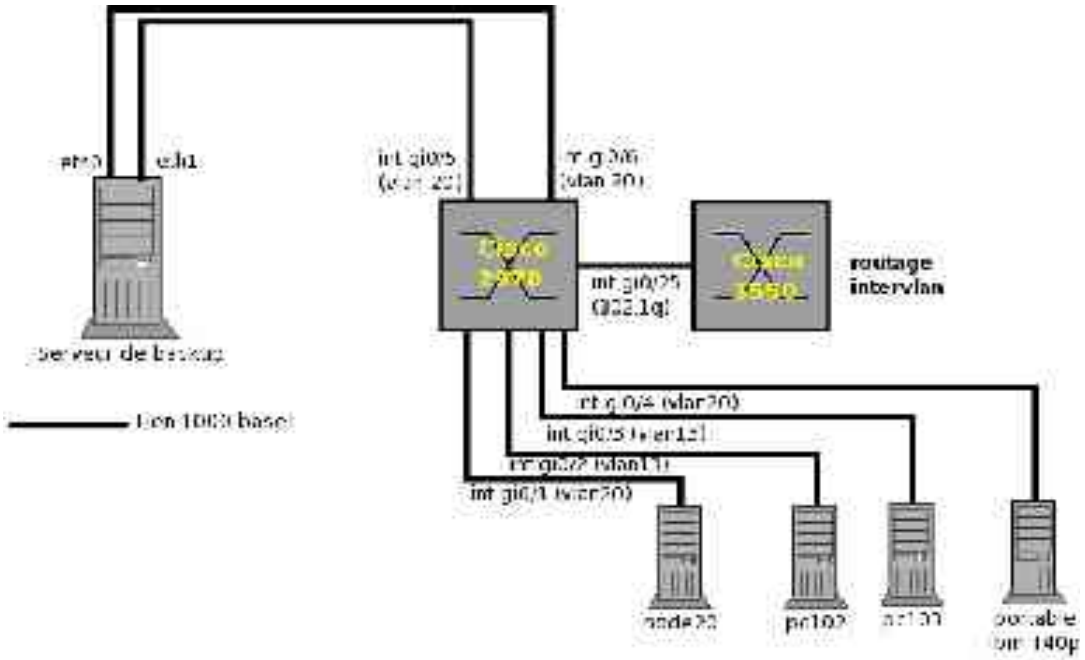
```
switch18(config)#int gigabitEthernet 0/18
switch18(config-if)#channel-group 3 mode on
switch18(config-if)#
```

Coté Linux, il suffit de changer le mode passé en paramètre du module bonding, dans ce cas le mode est 0.

4 – Les Tests

4.1 – La maquette

Les schéma ci dessous représente la maquette utilisée pour les tests:



L'aggrégat est de type 802.3ad.

4 clients gigabits dont 2 dans le VLAN du serveur de backup et 2 autres sur un VLAN différent qui devront donc être routées à travers le commutateur niveau 3 Cisco 3550. Nous pourrons ainsi constater par la même occasion l'impact du routage et filtrage IP.

L'outil utilisé pour générer du trafic est ifperf. Il s'agit d'ouvrir des sessions TCP de mémoire à mémoire pour générer un flux continue sur une période donnée. Sur les clients nous lançons la commande **iperf -s** pour recevoir ce qui est émis depuis le serveur de backup avec la commande **iperf -t 86400 -c machine** (émission sur 24h dans ce cas précis).

4.2 – Les résultats

L'analyse des courbes de trafic nous montre que pour les 3 premières machines vers lesquelles nous déclenchons l'envoie de données (node20, pc102 et pc103), le trafic passe uniquement sur l'interface eth0 du serveur de backup. Le trafic à destination de la 4ème machine (le portable ibm T40p) passe lui par l'interface eth1. Voici la répartition des débits

	Machines	*Interface utilisée*	*Débits (Mbps)*
1	node20	eth0	250
2	pctest2	eth0	375
3	pctest3	eth0	375
4	IBM T40p	eth1	800

Évidement la machine n°4 qui dispose d'une interface pour elle seule va beaucoup plus vite.

Au total le débit est 1,8 Gbps, ce qui est proche des 2 Gbps maximums théoriques.

Load balancing:

Nous constatons que la répartition des machines n'est pas fonction de la charge. C'est normal car ce sont les adresses mac qui sont utilisées pour le choix du lien physique.

Le mécanisme qui détermine le choix de l'interface physique étant basé sur des paramètres statiques (les adresses mac, voire les adresses IP) , pour une machine donnée c'est toujours la même interface physique qui est sélectionnée.

Fail over:

Nous avons appliqués la commande shutdown sur l'interface du gi0/6 du switch Cisco 2970 et immédiatement les flux ont basculé sur l'interfaces gi0/5.

4.3 – Cricket

Nous avons gardé de courbes de trafic réalisée avec Cricket qui est un outil « MRTG like ». La confguration par Defaut n'est pas prévue traiter du trafic au delà des 100Mbps, ayant rencontré le problème nous avons jugés bon d'ajouté quelques explications à ce sujet.

Cricket relève par SNMP toutes les 5mn les compteurs d'octets des interfaces de la mib2.

Ces compteurs font 32 bits et se révèlent trop juste lorsque le trafic dépasse les 100 Mbps.

La mib2 dispose maintenant maintenant de compteur d'octets 64bist (identifié ifHCInOctets et ifHCOutOctets, ils sont contenus dans la table ifXTable qui est un extension de la tables des interfaces) .

Pour relever ces compteurs 64bits nous utilisons snmp dans sa version 2 et au dela.

Voici les contenu commenté du fichier « Default » pour le switch Cisco sur lequel nous mesurons le trafic:

```
Target  default
   inst            = map(switch-interface-name)
   rrd-datafile    = %dataDir%/%auto-target-name%.rrd
   switch          = %auto-target-name%
   snmp-host       = %switch%
```

nouveau target type pour les interfaces gigabits

```
   target-type     = switch-interface-giga
```

specifier v2c pour la version du protocole

```
   snmp-version    = 2c
   snmp            =         %snmp-community%@%snmp-host%:%snmp-port%:%snmp-timeout%:%snmp-
retries%:%snmp-backoff%:%snmp-version%
```

OIDs des nouveaux compteurs d'octects HC (High Capacity):

```
OID    ifHCInOctets           1.3.6.1.2.1.31.1.1.1.6
OID    ifHCOutOctets          1.3.6.1.2.1.31.1.1.1.10
```

Collecte snmp: voici version 32 bits d'origine

```
dataSource ifInOctets
   ds-source = snmp://%snmp%/ifInOctets.%inst%
   rrd-ds-type     =   COUNTER
   rrd-heartbeat   =   1800
```

voici version 64 bits,

```
dataSource ifHCInOctets
   ds-source = snmp://%snmp%/ifHCInOctets.%inst%
   rrd-ds-type     =   COUNTER
   rrd-heartbeat   =   1800
```

Ne pas oublier de créer les graphs associés:

```
graph   ifHCOutOctets
   color       =   blue
   legend      =   "Average bits out"
   y-axis      =   "bits per second"
   units       =   "bits/sec"
   scale       =   8,*
```

et de définir la target

```
targetType switch-interface-giga
   ds = "ifHCInOctets, ifHCOutOctets, ifInErrors, ifOutErrors"
   view = "Octets: ifHCInOctets ifHCOutOctets, Errors: ifInErrors ifOutErrors"
```

5 – Conclusion

Aucun problème de stabilité et de performance avec le montage 802.3ad testé entre le serveur Linux et le switch Cisco.

Dans notre cas, s'agissant d'un serveur de backup, ce sont les flux montant qui nous intéressent. La stratégie pour la répartition de charge dépend des propriétés du switch Cisco. Nous n'avons pas le choix c'est statique car basé sur les adresses des machines. Avant de rentrer en production, il conviendra de vérifier par quelle interface passent chacunes des machines