|  |
| --- |
| Proyecto Integrado de Administración de Sistemas Informáticos en Red |
| Configuración de NIC Bonding y Link Aggregation |
| Instalación de los nodos de almacenamiento, configuración de NIC Bonding en Debian y configuración de Link Aggregation en el switch SMC8028L2 |

|  |
| --- |
| **Aitor Lázaro Sánchez**  **IES Gonzalo Nazareno**  **25 de Junio de 2012** |

Contenido

[1. Introducción 1](#_Toc328411579)

[2. Infraestructura. 1](#_Toc328411580)

[2.1. Los nodos de computación: io, europa, ganimedes y calisto. 1](#_Toc328411581)

[2.1.1. Preparación del RAID 0 1](#_Toc328411582)

[2.1.2. Instalación de Debian Wheezy 2](#_Toc328411583)

[2.2. La red 2](#_Toc328411584)

[3. NIC Bonding 3](#_Toc328411585)

[3.1. Modos de bonding en Debian. 3](#_Toc328411586)

[4. Link Aggregation 4](#_Toc328411587)

[5. Configuración 4](#_Toc328411588)

[5.1 Bonding en Debian Wheezy 4](#_Toc328411589)

[5.2. Link Aggregation en el Switch 8](#_Toc328411590)

[6. Pruebas de rendimiento. 8](#_Toc328411591)

[7. Pruebas realizadas durante la configuración: 9](#_Toc328411592)

[7.1. Configuración de los slaves 9](#_Toc328411593)

[7.2. Intentos de configuración del Modo 802.3ad 9](#_Toc328411594)

[7.3. Configuración en el Switch 11](#_Toc328411595)

[8. Referencias 14](#_Toc328411596)

# Introducción

En esta documentación presentaré que es el NIC Bonding y el Link Agregation, así como sus modos y configuración en los cuatro nodos destinados a la ejecución de instancias. Esta es solo una de las cuatro patas del proyecto integrado realizado en el IES Gonzalo Nazareno el curso 2011/2012.

Este documento, como ya he dicho, se centra en la muestra de como hemos configurado la red para esas cuatro máquinas y todo lo que eso conlleva: Resumen de los nodos, explicación de NIC Bonding, sus modos, sus parametros, explicación de Link Aggregation (que al final no se pudo llevar a cabo), explicación de la configuración finalmente escogida y por último la explicación del proceso llevado a cabo, con los errores que hemos tenido y como buenamente hemos podido subsanarlos o evitarlos.

# Infraestructura.

## 2.1. Los nodos de computación: io, europa, ganimedes y calisto.

Estos cuatro nodos serán las que se encargarán de ejecutar las instancias y estarán a las órdenes de jupiter en todo momento.

Especificaciones de las máquinas:

Los nodos no son exactamente iguales, mientras que io y europa tienen 16 cores y 32 GB de memoria RAM, ganimedes y calisto tienen ni más ni menos que 24 cores y 64 GB de RAM cada una.

Los cuatro equipos disponen de dos discos duros de 300 GB a 15000 rpm y una controladora LSI MegaRAID SAS.

### Preparación del RAID 0

En los cuatro nodos configuramos un RAID0 para aumentar la eficiencia, ya que, tal y como se indica en la documentación[[1]](#footnote-1), en estos equipos no va a haber datos persistentes, sino discos volátiles creados para las instancias. Así pues, con dos discos de 15000rpm y un RAID0 montado se logra una velocidad realmente importante. Tras configurarlo tendremos un dispositivo virtual de 556'929GB.

### Instalación de Debian Wheezy

La instalación también es idéntica en todos. Se ha instalado Debian Wheezy de 64 bits con las siguientes particiones:

/dev/sda1 1GB /boot  
/dev/sda5 2GB swap  
/dev/sda6 545GB /

En los cuatro equipos no se ha instalado entorno gráfico, simplemente un servidor ssh y los utilidades estándar del sistema.

## 2.2. La red

Tal como viene indicado en el esquema de red[[2]](#footnote-2), el objetivo es conectar los cuatro nodos a la red privada, entendiendo la red privada como la red que usa OpenStack para comunicarse con todos sus componentes, y, logrando mediante bonding y link aggregation, hacer que estas cuatro maquinas se comuniquen entre si a más de 2Gbps. En un principio la idea era lograrlo mediante el modo 4 de bonding, el 802.3ad, pero que, como ya mostraré más adelante, no se ha podido lograr y se ha realizado al final con el modo 0, balance-rr, que funciona normalmente a 2'2 - 2'3 Gbps.

Las ip's[[3]](#footnote-3) privadas del bonding de cada máquina son las siguientes:

Io 192.168.222.11  
Europa 192.168.222.12  
Ganimedes 192.168.222.13  
Calisto 192.168.222.14

# NIC Bonding

El NIC Bonding es la técnica que consiste en la creación de una interfaz de red virtual que tiene subyugadas varias interfaces físicas, que pasan a funcionar como slave (esclavo) para lograr desde un aumento en el ancho de banda hasta tener redundancia de la conexión, por lo que si una interfaz falla el resto siguen cumpliendo su cometido.

## Modos de bonding en Debian.

Para el proyecto, al usar Debian Wheezy, hemos usado el paquete ifenslave-2.6, que permite los siguientes modos de bonding:

**Modo 0 o balance-rr:** Este método se basa en el algoritmo Round Robin, es decir, transmite los paquetes de forma secuencial desde la primera hasta la última tarjeta de red esclava. Esta opción ofrece balanceo de carga y tolerancia a fallos. Modo por defecto.

**Modo 1 o active-backup:** Solo una interfaz está activa, el resto están a la espera de que se produzca algún fallo en la interfaz activa para activarse y suplir a la que falla. Esta opción tiene tolerancia a fallos pero no ofrece balanceo de carga.

**Modo 2 o balance-xor:** Se le asigna la cada MAC a una u otra interfaz esclava, siendo esa tarjeta de red la que se encargará de llevar las comunicaciones con esa MAC. Ofrece balanceo de carga y tolerancia a fallos.

**Modo 3 o broadcast:** Simplemente envía todo por todas las interfaces esclavas. Simplemente ofrece tolerancia a fallos.

**Modo 4 o 802.3ad:** Es el standard IEEE 802.3ad (Dynamic link aggregation, también conocido como Port Trunking y LACP). Permite agregar varios enlaces para aumentar el ancho de banda. Es necesario que todos los enlaces sean de la misma velocidad y ancho de banda, además el switch debe ser compatible con este estándar. Ofrece tolerancia a fallos y balanceo de carga.

**Modo 5 o balance-tlb:** Transmite la información balanceando según la carga de las interfaces, es necesario disponer de ethtool. Ofrece tolerancia a fallos y balanceo de carga.

**Modo 6 o balance-alb:** Balancea tanto en el envío como en la recepción, para lograr esto último manipula los replies ARP donde indica la MAC de una interfaz de uno de los slaves. Puede provocar que algún sistema de seguridad basado en las tablas ARP de problemas. También ofrece tolerancia a fallos y balanceo de carga.

# Link Aggregation

En el caso del switch la idea es casi identica: configurar varios puertos para que actúen como uno solo y así lograr mejorar el rendimiento. En el caso del switch que usamos, el SMC8028L2, nos permite dos opciones, la estática, que vale para todos los modos excepto el 802.3ad y el modo LACP que funciona con el IEEE 802.3ad.

# Configuración

## Bonding en Debian Wheezy

El objetivo inicial era lograr configurar tres interfaces gigabit ethernet en cada nodo para lograr una transferencia cercana a los 3Gbps. Para ello se iba a usar el modo 802.3ad (o modo 4) y se iba a configurar el Link Aggregation en el switch con la opción LACP.

Al final, por falta de tiempo, documentación y por, seguramente, error a la hora de configurar el switch, se dejó la idea inicial y se optó por el modo balance-rr (o modo 0) con el que se ha logrado una tasa de transferencia que rondan los 2'3 Gbps habiendo llegado, en muy contadas ocasiones, casi a los 2'5 Gbps.

Para configurar el bonding en Debian Wheezy es necesaria la instalación del paquete ifenslave-2.6.

aptitude install ifenslave-2.6

Para no tener que explicar por separado los parámetros que se le pueden configurar del bonding voy a poner la lista con el breve resumen de los parámetros:

**arp\_interval:** Indica cada cuántos milisegundos se envía un ARP reply. Su valor por defecto es 0.

**arp\_ip\_target:** Indica cual será la IP destino, se pueden especificar hasta 16 valores separados por comas.

**downdelay:** Especifica los milisegundos que tardará en bajar la interfaz cuando se detecte un error. El valor por defecto es 0.

**max\_bonds:** Indica cuantas interfaces virtuales se crearán al iniciar el móduclo. Por defecto es 1.

**miimon:** Indica cada cuantos milisegundos se va a comprobar el estado de los enlaces, por defecto es 0, con lo que está desactivado y por ello se aconseja en casi toods los sitios activarlo, un buen valor es 100.

**mode:** Indica el modo de bonding.

**primary:** Solo se utiliza en el modo 1 y sirve para indicar que interfaz va a estar como activa y cual o cuales como esclavas, además también indica el orden de ejecución en caso de error.

**updelay:** Indica cuantos milisegundos se tardará en activar una tarjeta de red esclava cuando se detecte un error en una interfaz. El valor por defecto es 0 (inmediatamente).

**use\_carrier:** Su valor puede ser 0 o 1, si es 1 indica que usará una llamada netif\_carrier\_ok() del módulo de la tarjeta de red para la detección por MII. Si vale 0 se usarán llamadas ETHTOOL ioctl's, que está obsoleto (deprecated).

**xmit\_hash\_policy:** Especifica la politica de transmisión para los modos 2 y 4, pueden ser:

**Layer 2:** Actúa como en el modo 2, es decir, todo lo que vaya para una IP lo mandará por la misma interfaz.

**Layer 3+4:** Usa niveles superiores al de enlace y actúa según el tipo de tráfico. No es completamente compatible con el 802.3ad

**slaves:** Indica que interfaces actúan como esclavas en el bonding.

Una vez instalado debemos bajar todas las interfaces que queremos usar como slaves si es que estaban activas:

ifdown eth1  
 ifdown eth2  
 ifdown eth3

Ahora toca modificar el fichero /etc/network/interfaces y escribir las siguientes lineas:

iface bond1 inet static   
 address 192.168.222.13  
 netmask 255.255.255.0  
 bond-slaves eth1 eth2 eth3  
 bond-mode 0  
 mtu 9000  
 bond-miimon 100

La configuración es muy sencilla, el único parámetro que no se ha visto ya es el de la línea mtu 9000, que sirve para indicar los bytes de los jumbo frames.

Una vez realizado esto reiniciamos networking y ya deberemos tener el bonding preparado.

/etc/init.d/networking restart

Otra forma de subir el bond sería mediante comandos de la siguiente manera:

1º Activar el modulo bonding indicándole el modo y el valor de miimon

modprobe bonding mode=0 (o el nombre) miimon=100

2º Configuramos bond1 indicando ip, mascara de red y mtu

ifconfig bond1 192.168.222.13 netmask 255.255.255.0 mtu 9000 up

3º Y por último se añaden las interfaces esclavas al bonding:

ifenslave bond1 eth1 eth2 eth3

Con ifconfig podemos comprobar el estado del bonding, que debe ser algo así:

bond1 Link encap:Ethernet HWaddr 00:25:90:69:f8:a9   
 inet addr:192.168.222.13 Bcast:192.168.222.255 Mask:255.255.255.0  
 inet6 addr: fe80::225:90ff:fe69:f8a9/64 Scope:Link  
 UP BROADCAST RUNNING MASTER MULTICAST MTU:9000 Metric:1  
 RX packets:1847300 errors:0 dropped:0 overruns:0 frame:0  
 TX packets:1428607 errors:0 dropped:0 overruns:0 carrier:0  
 collisions:0 txqueuelen:0   
 RX bytes:1732704907 (1.6 GiB) TX bytes:6066067471 (5.6 GiB)

Las interfaces que están como esclavo deben tener un aspecto como este:

eth1 Link encap:Ethernet HWaddr 00:25:90:69:f8:a9   
 UP BROADCAST RUNNING SLAVE MULTICAST MTU:9000 Metric:1  
 RX packets:796589 errors:0 dropped:0 overruns:0 frame:0  
 TX packets:476431 errors:0 dropped:0 overruns:0 carrier:0  
 collisions:0 txqueuelen:1000   
 RX bytes:1412769379 (1.3 GiB) TX bytes:2023694953 (1.8 GiB)  
 Memory:fe7e0000-fe800000

Para comprobar la configuración correcta del bonding hay que comprobar los siguientes ficheros.

Configuración en general:

cat /proc/net/bonding/bond1

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: eth1  
 MII Status: up  
 Speed: 1000 Mbps  
 Duplex: full  
 Link Failure Count: 0  
 Permanent HW addr: 00:25:90:69:f8:a9  
 Slave queue ID: 0

Slave Interface: eth2  
 MII Status: up  
 Speed: 1000 Mbps  
 Duplex: full  
 Link Failure Count: 1  
 Permanent HW addr: 00:25:90:74:e0:fe  
 Slave queue ID: 0  
  
 Slave Interface: eth3  
 MII Status: up  
 Speed: 1000 Mbps  
 Duplex: full  
 Link Failure Count: 0  
 Permanent HW addr: 00:25:90:74:e0:ff  
 Slave queue ID: 0

Para el modo:

cat /sys/class/net/bond1/bonding/mode  
 balance-rr 0

Hay otros muchos ficheros para ver los parametros y la configuración del bonding, la mayoría de ficheros con los parametos se encuentran en el directorio /sys/class/net/bond1 y /sys/class/net/bond1/bonding/

En el caso de hayamos configurado algo mal, para poder realizar los cambios hay que bajar primero las interfaces esclavas, después del bonding y quitar el módulo para que se borre la configuración.

ifdown eth1  
 ifdown eth2  
 ifdown eth3  
 ifconfig bond1 down  
 modprobe -r bonding

## 5.2. Link Aggregation en el Switch

En lo relativo al switch, por la falta del tiempo y por algún fallo de configuración no logramos configurarlo en modo LACP ni en estático de forma correcta, ya que no solo no mejoraba el rendimiento respecto al switch sin configuración, sino que además actuaba como un lastre y no pasaba del Gbps.

# Pruebas de rendimiento.

Para comprobar el ancho de banda usamos iperf con 50 clientes simultaneos para exigir a la red. Estas son algunas pruebas de rendimiento. El comando para las pruebas son los siguientes.

En el servidor:

iperf -s

En el cliente:

iperf -c 192.168.222.11 -P 50

Io como servidor, Ganimedes de cliente.

[SUM] 0.0-10.0 sec 2.51 GBytes 2.16 Gbits/sec

Ganimedes como servidor y Calisto como cliente

[SUM] 0.0-10.6 sec 2.66 GBytes 2.27 Gbits/sec

Calisto como servidor e Io como cliente

[SUM] 0.0-10.4 sec 2.64 GBytes 2.18 Gbits/sec

# Pruebas realizadas durante la configuración:

Aquí recopilaré una serie de errores y configuraciones probadas mientras configurábamos bonding en Debian Wheezy y el trunking en el Switch SMC8028L2

## 7.1. Configuración de los slaves

Para empezar, a la hora de configurar las interfaces como esclavas, nos encontramos en internet con bastante documentación, enfocada principalmente a Ubuntu, en la que se indicaba que para poder poner la interfaz como slave hacía falta poner esta configuración en cada interfaz:

auto eth1  
 iface eth1 inet manual  
 bond-master bond1

Siendo la línea importante la última, donde se indicaba con el parámetro bond-master a que interfaz lógica estaba sometida. Además, en la configuración del bond1 había que indicar al parámetro bond-slaves que no había ninguno:

bond-slaves none

En debian, o al menos en nuestro caso, no nos ha hecho falta. No tenemos en el fichero /etc/network/interfaces configurado nada de las interfaces esclavas y en la línea bond-slaves de bond1 hemos indicado los esclavos. Está así:

auto bond1  
 iface bond1 inet static  
 address 192.168.222.13  
 netmask 255.255.255.0  
 bond-slaves eth1 eth2 eth3  
 bond-mode 0  
 mtu 9000  
 bond-miimon 100

## 7.2. Intentos de configuración del Modo 802.3ad

Cómo he comentado antes el objetivo era configurar el bonding en modo 802.3ad, cosa que al final no pude lograr, seguramente por no configurar correctamente el switch. Aquí voy a recopilar una serie de configuraciones y errores que tuvimos que afrontar durante la configuración del bonding. La configuración básica probada era esta:

auto bond1  
 iface bond0 inet static  
 address 192.168.222.13  
 netmask 255.255.255.0  
 bond-mode 4  
 bond-miimon 100  
 bond-slaves none

auto eth1  
 iface eth1 inet manual  
 bond-master bond0

Estos parámetros eran prácticamente invariables en casi todos los intentos que hicimos, si bien bond-slaves fue sufrió la modificación comentada en el principio de este apartado. De esas lineas no hay mucho que explicar, si un caso repetir que el bond-mode 4 se refiere al modo 802.3ad.

A partir de ahí se hicieron varias pruebas con distintos parámetros:

Desde el principio añadí las siguientes dos líneas, que no están mal pero tampoco son necesarias, y como en nuestro caso ese apartado no era muy importante al final se quitaron. Seguramente en según que escenario podrían ser importantes:

bond downdelay 200  
 bond-updelay 200

El primer parámetro indica cuantos milisegundos tardará en bajarse una interfaz que falle a partir de que se detecte el error, por defecto es 0, inmediatamente. El segundo indica cuanto tiempo tardará en levantarse una interfaz cuando se detecte el fallo en una interfaz que falla, también se calcula en milisegundos y su valor por defecto es 0.

El siguiente parámetro que probamos fue bond-lacp-rate que indica cada cuanto va a enviar los mensajes LACDPU, que se encargan de controlar el protocolo de agregación de enlaces. Hay dos valores: 0 (lento) y 1 (rápido).

bond-lacp-rate 1

También añadimos esta línea, que no pintaba nada ahí, ya que bond-primary solo sirve para el modo 1 (active-backup) e indica quien es la interfaz activa y en que orden van saltando. En este caso sería eth1 la activa y eth2 se activaría en caso de que esta fallase, y si falla eth2 la que funcionaría como backup sería eth3. Esta línea también la añadimos para hacer pruebas en las interfaces, se eliminó de ambos sitios ya que era inútil.

bond-primary eth1 eth2 eth3

Por último también añadimos la configuración de mtu a 9000

mtu 9000

Al final la configuración que dejamos para hacer las pruebas era algo así:

auto bond1  
 iface bond0 inet static  
 address 192.168.222.13  
 netmask 255.255.255.0  
 bond-mode 4  
 bond-miimon 100  
 bond-slaves eth1 eth2 eth3  
 bond-lacp-rate 1  
 mtu 9000

## 7.3. Configuración en el Switch

Importante nada más empezar: Cuando se configura el switch deben estar los puertos desconectados, al igual que lo deben estar antes de desconfigurar el switch. Si no se hace eso es posible que el switch de fallos, como nos ocurrió a nosotros donde había problemas con MAC asignadas de forma extraña (varias MAC para un bonding que debería tener solo una) y fallos que se solucionaron al reiniciar sin ningún tipo de configuración.

Al tener el modo 4 activado es necesario configurar el switch en modo LACP. Así se hizo asignando a cada grupo de tres puertos una ID distinta:

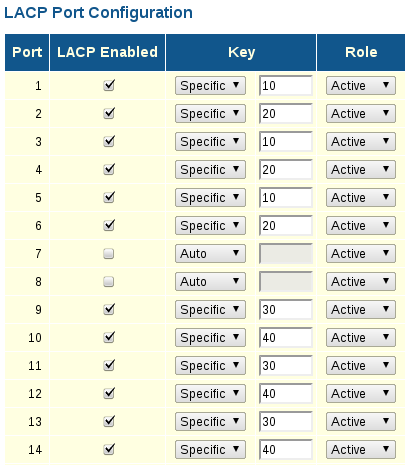
Máquina ID Puertos

- Io 10 1,3,5

- Europa 20 2,4,6

- Ganimedes 30 9,11,13

- Calisto 40 10,12,14



Una vez configurado se conectan los puertos y se levanta el bonding. Para ver si ha cogido bien la configuración del switch debemos mirar los siguientes ficheros:

cat /sys/class/net/bond1/bonding/ad\_partner\_key  
 cat /sys/class/net/bond1/bonding/ad\_actor\_key

En el primero se almacena el valor del ID asignado en el Switch, en el caso de Ganimedes, por ejemplo, el 30. En el segundo fichero nos mostrará el identificador que usa Debian, en todos los equipos nos salía el 17.

Al configurarlo así nos encontramos con que el ancho de banda del bonding no superaba los 980Mbps. Más adelante creímos dar con la solución al encontrar un parámetro para el modo 802.3ad en el que se indicaba que ese bonding iba destinado al ancho de banda, ejecutamos el cambio, pero el rendimiento fue el mismo.

Una vez hecho esto y sin lograr resolverlo probamos con distintos modos de bonding, exactamente el modo 0 (balance-rr), el 2 (broadcast), 5 (balance-tlb) y el 6 (balance-alb). El cambio es sencillo, bajar bond0, hacer modprobe -r bonding para eliminar la configuración anterior y en /etc/network/interfaces cambiar el número del bond-mode y eliminar la linea "bond-lacp-rate 1"

Además, también quitamos la configuración LACP del switch y lo configuramos como estático de la siguiente manera:

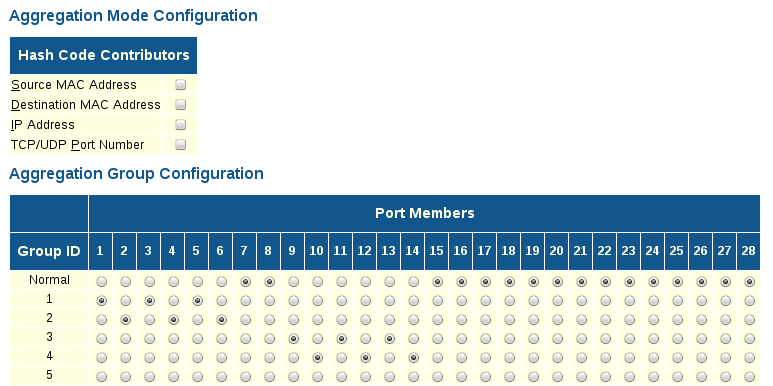
Máquina ID Puertos

- Io 1 1,3,5

- Europa 2 2,4,6

- Ganimedes 3 9,11,13

- Calisto 4 10,12,14



Cómo se puede ver no se ha asignado ningún hash, con lo que los paquetes, teóricamente, los mandará por la primera tarjeta de red que esté disponible, sin tener en cuenta la IP, el puerto, a que MAC va dirigido o cual es la MAC de origen. Aun así tampoco logramos el objetivo de aumentar el ancho de banda, incluso con el modo 2, broadcast, disminuyó notablemente.

Al final, ya con el tiempo justo**, desconfiguramos el switch** y poco después encontramos en internet la posible solución. Realizamos lo siguiente y funcionó:

Modificar el valor por defecto de /proc/sys/net/core/netdev\_max\_backlog a 3000

echo "3000" > /proc/sys/net/core/netdev\_max\_backlog

Cambiar el valor de txqueulen de 0 a 10000 en bond1

ifconfig bond1 txqueuelen 10000

Y mediante ethtool cambiar el valor de rx y tx de las interfaces del bonding a 3096:

aptitude install ethtool  
  
 ethtool -G eth1 rx 3072 tx 3072  
 ethtool -G eth2 rx 3072 tx 3072  
 ethtool -G eth3 rx 3072 tx 3072

Una vez configurado así el bonding llegó a alcanzar más de 2Gbps.

Para volver a documentar exactamente los cambios necesarios volvimos a desconfigurar todo, dejando los cambios como estaban por defecto, reiniciando routers y máquinas. Cual fue nuestra sorpresa que alcanzó, aun no teniendo esos cambios, los 2'3 Gbps. Seguramente debido algún fallo de configuración del switch que al reiniciarse desapareció.

Tras lograr que el bonding funcionase a gran velocidad volvimos a configurar el link agreggation en el switch en modo LACP y volvimos a configurar en Debian el modo 4. El intento no fructiferó y se decidió dejarlo en modo 0.

# Referencias

http://crysol.org/es/node/811  
http://wiki.debian.org/Bonding  
http://www.howtoforge.com/nic-bonding-on-debian-lenny   
http://phenobarbital.wordpress.com/2011/06/22/linux-guia-rapida-de-network-bonding-en-debian/  
http://geekomatic.ch/2011/09/23/1316778300000.html  
http://www.kernel.org/doc/Documentation/networking/bonding.txt  
http://serverfault.com/questions/287475/why-does-my-gigabit-bond-not-deliver-at-least-150-mb-s-throughput  
http://smc-australia.com.au/smc-australia/\_download/SMC8028L2/SMC8028L2\_mng.pdf

1. http://informatica.gonzalonazareno.org/openstack/wheezy/ch01s05.html [↑](#footnote-ref-1)
2. http://informatica.gonzalonazareno.org/openstack/wheezy/ch01s02.html [↑](#footnote-ref-2)
3. http://informatica.gonzalonazareno.org/openstack/wheezy/ch01s03.html [↑](#footnote-ref-3)