

## [git patches] net driver fixes

**Source:** <http://linux.derkeiler.com/Mailing-Lists/Kernel/2005-08/0037.html>

---

**From:** Jeff Garzik ([jgarzik\\_at\\_pobox.com](mailto:jgarzik_at_pobox.com))

**Date:** 08/01/05

Date: Sun, 31 Jul 2005 19:30:56 -0400

To: Linus Torvalds <[torvalds@osdl.org](mailto:torvalds@osdl.org)>, Andrew Morton <[akpm@osdl.org](mailto:akpm@osdl.org)>

Please pull from the 'upstream-fixes' branch of  
<rsync://rsync.kernel.org/pub/scm/linux/kernel/git/jgarzik/netdev-2.6.git>

to obtain the fixes described in the attached diffstat/changelog/patch.

```
Documentation/networking/bonding.txt | 968 ++++++-----
drivers/net/hamradio/Kconfig | 2
drivers/net/sk98lin/skgeinit.c | 2
drivers/net/sk98lin/skxmac2.c | 8
drivers/net/skge.c | 233 +++-----
drivers/net/skge.h | 41 -
drivers/net/smc91x.h | 2
7 files changed, 806 insertions(+), 450 deletions(-)
```

commit e064cd7e3ac797df1e81b55ff4fed5fca5d106b5

Author: Ralf Baechle <[ralf@linux-mips.org](mailto:ralf@linux-mips.org)>

Date: Mon Jul 4 18:30:42 2005 +0100

[PATCH] SMP fix for 6pack driver

Drivers really only work well in SMP if they actually can be selected.  
This is a leftover from the time when the 6pack drive only used to be  
a bitrotten variant of the slip driver.

Signed-off-by: Ralf Baechle DL5RB <[ralf@linux-mips.org](mailto:ralf@linux-mips.org)>

```
Kconfig | 2 +-
1 files changed, 1 insertion(+), 1 deletion(-)
```

Signed-off-by: Jeff Garzik <[jgarzik@pobox.com](mailto:jgarzik@pobox.com)>

commit af44f5bf775e0d36aa5879c94369216ff6f717a6

Author: Tony Lindgren <[tony@atomide.com](mailto:tony@atomide.com)>

Date: Thu Jun 30 06:40:18 2005 -0700

Linux-Kernel: [git patches] net driver fixes

[PATCH] Fix OMAP specific typo in smc91x.h

--ReaqsoxgOBHFXBhH  
Content-Type: text/plain; charset=us-ascii  
Content-Disposition: inline

Hi Jeff,

Here's a little patch fixing a typo in smc91x.h.

Regards,

Tony

--ReaqsoxgOBHFXBhH  
Content-Type: text/x-chdr; charset=us-ascii  
Content-Disposition: inline; filename="patch-fix-typo-smc91x.h"  
Signed-off-by: Jeff Garzik <jgarzik@pobox.com>

commit 3f309db33e7868fe11f8fc3a0dd291703df3c662  
Author: Stephen Hemminger <shemminger@osdl.org>  
Date: Mon Jun 27 15:47:25 2005 -0700

[PATCH] sk98lin: fix workaround for yukon-lite chipset (> rev 7)

Yukon-Lite chipset needs workaround for revision 7 (or later).  
Without this patch, chip gets stuck in low power mode and never  
boots. Newer SysKonnect vendor code already had same patch.

Related bug in skge is <http://bugs.gentoo.org/87822>

Chris, please add for 2.6.12.2

Signed-off-by: Stephen Hemminger <shemminger@osdl.org>  
Signed-off-by: Jeff Garzik <jgarzik@pobox.com>

commit 00354cfb92bd819a0d09d3ef9988e509b6675fdd  
Author: Jay Vosburgh <fubar@us.ibm.com>  
Date: Thu Jul 21 12:18:02 2005 -0700

[PATCH] bonding: documentation update

Contains general updates (additional configuration info, hopefully  
better examples, updated some out of date info, and a bonus pass  
through ispell to banish the "paramters.") and info specific to  
gratuitous ARP and xmit policy functionality already in 2.6.13-rc2.

Signed-off-by: Jay Vosburgh <fubar@us.ibm.com>  
Signed-off-by: Jeff Garzik <jgarzik@pobox.com>

Linux-Kernel: [git patches] net driver fixes

commit f2e1e47d14aae1f33e98cf8d9ea93e2fe9e2f521  
Author: Stephen Hemminger <shemminger@osdl.org>  
Date: Fri Jul 22 16:26:11 2005 -0700

[PATCH] skge: version 0.8

Increase driver version to 0.8

Signed-off-by: Stephen Hemminger <shemminger@osdl.org>  
Signed-off-by: Jeff Garzik <jgarzik@pobox.com>

commit 6abebb538d317ead09cc0f3c2a0752047f9ff961  
Author: Stephen Hemminger <shemminger@osdl.org>  
Date: Fri Jul 22 16:26:10 2005 -0700

[PATCH] skge: led toggle cleanup

Cleanup code that is used to toggle LED's. Since we get called from ethtool, can use that thread rather than setting up a timer.

Signed-off-by: Stephen Hemminger <shemminger@osdl.org>  
Signed-off-by: Jeff Garzik <jgarzik@pobox.com>

commit 4cde06ed0fb58402ec1d6d117122d1058983a393  
Author: Stephen Hemminger <shemminger@osdl.org>  
Date: Fri Jul 22 16:26:09 2005 -0700

[PATCH] skge: ignore phy interrupts during negotiation

During autonegotiation set PHY interrupt mask to ignore bogus speed change interrupts.

Signed-off-by: Stephen Hemminger <shemminger@osdl.org>  
Signed-off-by: Jeff Garzik <jgarzik@pobox.com>

commit d8a09943ebbaca9befd995d8fe10dd9885256dbf  
Author: Stephen Hemminger <shemminger@osdl.org>  
Date: Fri Jul 22 16:26:08 2005 -0700

[PATCH] skge: fifo control register access fix

The code to clear fifo errors was incorrect and sending garbage to the external phy. Removed the no longer used inline's funcs.

Signed-off-by: Stephen Hemminger <shemminger@osdl.org>  
Signed-off-by: Jeff Garzik <jgarzik@pobox.com>

commit 2c66851460c9438823e39b76887376d1511fb67c  
Author: Stephen Hemminger <shemminger@osdl.org>  
Date: Fri Jul 22 16:26:07 2005 -0700

[git patches] net driver fixes

Linux-Kernel: [git patches] net driver fixes

[PATCH] skge: whitespace fixes

Minor whitespace cleanups.

Signed-off-by: Stephen Hemminger <shemminger@osdl.org>

Signed-off-by: Jeff Garzik <jgarzik@pobox.com>

commit 382317138b3ade02c9c319531ab0619e95dbc672

Author: Stephen Hemminger <shemminger@osdl.org>

Date: Fri Jul 22 16:26:06 2005 -0700

[PATCH] skge: support yukon lite rev 4

The check for Yukon lite changes was restricting itself to rev A3. It turns out that these changes are also true on A4 and later.

Signed-off-by: Stephen Hemminger <shemminger@osdl.org>

Signed-off-by: Jeff Garzik <jgarzik@pobox.com>

commit 4ff6ac052b90ee4dfce92f8e2c5cb7ef8a4d8f13

Author: Stephen Hemminger <shemminger@osdl.org>

Date: Fri Jul 22 16:26:05 2005 -0700

[PATCH] skge: phy lock deadlock

Cleanup the phy\_lock deadlock because of relocking in the nway\_reset path. Reported by Francois Romieu.

Also, don't need to do irqsave/restore for blink, just excluding bh is good enough.

Signed-off-by: Stephen Hemminger <shemminger@osdl.org>

Signed-off-by: Jeff Garzik <jgarzik@pobox.com>

commit 0eedf4ac5b536c7922263adf1b1d991d2e2397b9

Author: Stephen Hemminger <shemminger@osdl.org>

Date: Fri Jul 22 16:26:04 2005 -0700

[PATCH] skge: disable transmitter on shutdown

Here is a fix for a typo, thanks Eliot Dresselhaus. Since transmitter not active when device is down, it wasn't really noticed.

Signed-off-by: Stephen Hemminger <shemminger@osdl.org>

Signed-off-by: Jeff Garzik <jgarzik@pobox.com>

commit acdd80d514a08800380c9f92b1bf4d4c9e818125

Author: Stephen Hemminger <shemminger@osdl.org>

Date: Fri Jul 22 16:26:03 2005 -0700

Linux-Kernel: [git patches] net driver fixes

[PATCH] skge: remove SK-9EE support

The SK-9E boards use the Marvell Yukon2 chipset which is not supported by the skge driver. Thanks to Ralph Roesler for noticing.

Signed-off-by: Stephen Hemminger <shemminger@osdl.org>  
Signed-off-by: Jeff Garzik <jgarzik@pobox.com>

commit f6620cab9485d435aa93490533b8268d36dc4526  
Author: Stephen Hemminger <shemminger@osdl.org>  
Date: Fri Jul 22 16:26:02 2005 -0700

[PATCH] skge: silence mac data parity messages

Using Genesis board, I get harmless error reports. Rather than console error, turn it into a error counter.

Signed-off-by: Stephen Hemminger <shemminger@osdl.org>  
Signed-off-by: Jeff Garzik <jgarzik@pobox.com>

```
diff --git a/Documentation/networking/bonding.txt b/Documentation/networking/bonding.txt
--- a/Documentation/networking/bonding.txt
+++ b/Documentation/networking/bonding.txt
@@ -1,5 +1,7 @@
```

```
- Linux Ethernet Bonding Driver HOWTO
+ Linux Ethernet Bonding Driver HOWTO
+
+ Latest update: 21 June 2005
```

```
Initial release : Thomas Davis <tadavis at lbl.gov>
Corrections, HA extensions : 2000/10/03-15 :
@@ -11,15 +13,22 @@ Corrections, HA extensions : 2000/10/03-
```

Reorganized and updated Feb 2005 by Jay Vosburgh

-Note :

-----

+Introduction

+=====

+

+ The Linux bonding driver provides a method for aggregating  
+multiple network interfaces into a single logical "bonded" interface.  
+The behavior of the bonded interfaces depends upon the mode; generally  
+speaking, modes provide either hot standby or load balancing services.  
+Additionally, link integrity monitoring may be performed.

-The bonding driver originally came from Donald Becker's beowulf patches for  
-kernel 2.0. It has changed quite a bit since, and the original tools from  
-extreme-linux and beowulf sites will not work with this version of the driver.

[git patches] net driver fixes

- + The bonding driver originally came from Donald Becker's
- +beowulf patches for kernel 2.0. It has changed quite a bit since, and
- +the original tools from extreme–linux and beowulf sites will not work
- +with this version of the driver.

- For new versions of the driver, patches for older kernels and the updated
- userspace tools, please follow the links at the end of this file.
- + For new versions of the driver, updated userspace tools, and
- +who to ask for help, please follow the links at the end of this file.

## Table of Contents

=====

@@ –30,9 +39,13 @@ Table of Contents

### 3. Configuring Bonding Devices

#### 3.1 Configuration with sysconfig support

- +3.1.1 Using DHCP with sysconfig

- +3.1.2 Configuring Multiple Bonds with sysconfig

#### 3.2 Configuration with initscripts support

- +3.2.1 Using DHCP with initscripts

- +3.2.2 Configuring Multiple Bonds with initscripts

#### 3.3 Configuring Bonding Manually

- 3.4 Configuring Multiple Bonds

- +3.3.1 Configuring Multiple Bonds Manually

### 5. Querying Bonding Configuration

#### 5.1 Bonding Configuration

@@ –56,21 +69,30 @@ Table of Contents

### 11. Promiscuous mode

- 12. High Availability Information

- +12. Configuring Bonding for High Availability

#### 12.1 High Availability in a Single Switch Topology

- 12.1.1 Bonding Mode Selection for Single Switch Topology

- 12.1.2 Link Monitoring for Single Switch Topology

#### 12.2 High Availability in a Multiple Switch Topology

- 12.2.1 Bonding Mode Selection for Multiple Switch Topology

- 12.2.2 Link Monitoring for Multiple Switch Topology

- 12.3 Switch Behavior Issues for High Availability

- +12.2.1 HA Bonding Mode Selection for Multiple Switch Topology

- +12.2.2 HA Link Monitoring for Multiple Switch Topology

+

- +13. Configuring Bonding for Maximum Throughput

- +13.1 Maximum Throughput in a Single Switch Topology

- +13.1.1 MT Bonding Mode Selection for Single Switch Topology

- +13.1.2 MT Link Monitoring for Single Switch Topology

- +13.2 Maximum Throughput in a Multiple Switch Topology

- +13.2.1 MT Bonding Mode Selection for Multiple Switch Topology

- +13.2.2 MT Link Monitoring for Multiple Switch Topology

+

- +14. Switch Behavior Issues
  - +14.1 Link Establishment and Failover Delays
  - +14.2 Duplicated Incoming Packets
- 13. Hardware Specific Considerations
  - 13.1 IBM BladeCenter
- +15. Hardware Specific Considerations
  - +15.1 IBM BladeCenter
- 14. Frequently Asked Questions
- +16. Frequently Asked Questions
- 15. Resources and Links
- +17. Resources and Links

## 1. Bonding Driver Installation

@@ –86,16 +108,10 @@ the following steps:

### 1.1 Configure and build the kernel with bonding

---

- The latest version of the bonding driver is available in the
- + The current version of the bonding driver is available in the drivers/net/bonding subdirectory of the most recent kernel source (which is available on <http://kernel.org>).
- 
- Prior to the 2.4.11 kernel, the bonding driver was maintained largely outside the kernel tree; patches for some earlier kernels are available on the bonding sourceforge site, although those patches are still several years out of date. Most users will want to use either the most recent kernel from kernel.org or whatever kernel came with their distro.
- +(which is available on <http://kernel.org>). Most users "rolling their own" will want to use the most recent kernel from kernel.org.

Configure kernel with "make menuconfig" (or "make xconfig" or "make config"), then select "Bonding driver support" in the "Network @@ –103,8 +119,8 @@ device support" section. It is recommended driver as module since it is currently the only way to pass parameters to the driver or configure more than one bonding device.

- Build and install the new kernel and modules, then proceed to step 2.
- + Build and install the new kernel and modules, then continue below to install ifenslave.

### 1.2 Install ifenslave Control Utility

---

@@ –147,9 +163,9 @@ default kernel source include directory.

Options for the bonding driver are supplied as parameters to the bonding module at load time. They may be given as command line

arguments to the insmod or modprobe command, but are usually specified  
–in either the /etc/modprobe.conf configuration file, or in a  
–distro–specific configuration file (some of which are detailed in the  
–next section).  
+in either the /etc/modules.conf or /etc/modprobe.conf configuration  
+file, or in a distro–specific configuration file (some of which are  
+detailed in the next section).

The available bonding driver parameters are listed below. If a parameter is not specified the default value is used. When initially @@ –162,34 +178,34 @@ degradation will occur during link failure support at least miimon, so there is really no reason not to use it.

Options with textual values will accept either the text name  
– or, for backwards compatibility, the option value. E.g.,  
– "mode=802.3ad" and "mode=4" set the same mode.  
+or, for backwards compatibility, the option value. E.g.,  
+"mode=802.3ad" and "mode=4" set the same mode.

The parameters are as follows:

#### arp\_interval

– Specifies the ARP monitoring frequency in milli–seconds. If  
– ARP monitoring is used in a load–balancing mode (mode 0 or 2),  
– the switch should be configured in a mode that evenly  
– distributes packets across all links – such as round–robin. If  
– the switch is configured to distribute the packets in an XOR  
+ Specifies the ARP link monitoring frequency in milliseconds.  
+ If ARP monitoring is used in an etherchannel compatible mode  
+ (modes 0 and 2), the switch should be configured in a mode  
+ that evenly distributes packets across all links. If the  
+ switch is configured to distribute the packets in an XOR  
fashion, all replies from the ARP targets will be received on  
the same link which could cause the other team members to  
– fail. ARP monitoring should not be used in conjunction with  
– miimon. A value of 0 disables ARP monitoring. The default  
+ fail. ARP monitoring should not be used in conjunction with  
+ miimon. A value of 0 disables ARP monitoring. The default  
value is 0.

#### arp\_ip\_target

– Specifies the ip addresses to use when arp\_interval is > 0.  
– These are the targets of the ARP request sent to determine the  
– health of the link to the targets. Specify these values in  
– ddd.ddd.ddd.ddd format. Multiple ip addresses must be  
– separated by a comma. At least one IP address must be given  
– for ARP monitoring to function. The maximum number of targets  
– that can be specified is 16. The default value is no IP  
– addresses.

## Linux–Kernel: [git patches] net driver fixes

- + Specifies the IP addresses to use as ARP monitoring peers when
- + arp\_interval is > 0. These are the targets of the ARP request
- + sent to determine the health of the link to the targets.
- + Specify these values in ddd.ddd.ddd.ddd format. Multiple IP
- + addresses must be separated by a comma. At least one IP
- + address must be given for ARP monitoring to function. The
- + maximum number of targets that can be specified is 16. The
- + default value is no IP addresses.

downdelay

@@ -207,11 +223,13 @@ lacp\_rate  
are:

slow or 0

- Request partner to transmit LACPDU every 30 seconds (default)
- + Request partner to transmit LACPDU every 30 seconds

fast or 1

Request partner to transmit LACPDU every 1 second

- + The default is slow.

+

max\_bonds

Specifies the number of bonding devices to create for this

@@ -221,10 +239,11 @@ max\_bonds

miimon

- Specifies the frequency in milli–seconds that MII link
- monitoring will occur. A value of zero disables MII link
- monitoring. A value of 100 is a good starting point. The
- use\_carrier option, below, affects how the link state is
- + Specifies the MII link monitoring frequency in milliseconds.
- + This determines how often the link state of each slave is
- + inspected for link failures. A value of zero disables MII
- + link monitoring. A value of 100 is a good starting point.
- + The use\_carrier option, below, affects how the link state is
- determined. See the High Availability section for additional
- information. The default value is 0.

@@ -246,17 +265,31 @@ mode

active. A different slave becomes active if, and only if, the active slave fails. The bond's MAC address is externally visible on only one port (network adapter)

- to avoid confusing the switch. This mode provides
- fault tolerance. The primary option affects the
- behavior of this mode.
- + to avoid confusing the switch.

+

## Linux–Kernel: [git patches] net driver fixes

- + In bonding version 2.6.2 or later, when a failover
- + occurs in active–backup mode, bonding will issue one
- + or more gratuitous ARPs on the newly active slave.
- + One gratuitous ARP is issued for the bonding master
- + interface and each VLAN interfaces configured above
- + it, provided that the interface has at least one IP
- + address configured. Gratuitous ARPs issued for VLAN
- + interfaces are tagged with the appropriate VLAN id.
- +
- + This mode provides fault tolerance. The primary
- + option, documented below, affects the behavior of this
- + mode.

balance–xor or 2

- XOR policy: Transmit based on [(source MAC address
- XOR'd with destination MAC address) modulo slave
- count]. This selects the same slave for each
- destination MAC address. This mode provides load
- balancing and fault tolerance.
- + XOR policy: Transmit based on the selected transmit
- + hash policy. The default policy is a simple [(source
- + MAC address XOR'd with destination MAC address) modulo
- + slave count]. Alternate transmit policies may be
- + selected via the xmit\_hash\_policy option, described
- + below.
- +
- + This mode provides load balancing and fault tolerance.

broadcast or 3

- @@ –270,7 +303,17 @@ mode
- duplex settings. Utilizes all slaves in the active
- aggregator according to the 802.3ad specification.

- Pre–requisites:
- + Slave selection for outgoing traffic is done according
- + to the transmit hash policy, which may be changed from
- + the default simple XOR policy via the xmit\_hash\_policy
- + option, documented below. Note that not all transmit
- + policies may be 802.3ad compliant, particularly in
- + regards to the packet mis–ordering requirements of
- + section 43.2.4 of the 802.3ad standard. Differing
- + peer implementations will have varying tolerances for
- + noncompliance.
- +
- + Prerequisites:

1. Ethtool support in the base drivers for retrieving the speed and duplex of each slave.

@@ –333,7 +376,7 @@ mode



- + well as some Foundry and IBM products.
- + This algorithm is not fully 802.3ad compliant. A
- + single TCP or UDP conversation containing both
- + fragmented and unfragmented packets will see packets
- + striped across two interfaces. This may result in out
- + of order delivery. Most traffic types will not meet
- + this criteria, as TCP rarely fragments traffic, and
- + most UDP traffic is not involved in extended
- + conversations. Other implementations of 802.3ad may
- + or may not tolerate this noncompliance.
- + The default value is layer2. This option was added in bonding
- +version 2.6.3. In earlier versions of bonding, this parameter does
- +not exist, and the layer2 policy is the only policy.

### 3. Configuring Bonding Devices

@@ –448,8 +545,9 @@ Bonding devices can be managed by hand, slave devices. On SLES 9, this is most easily done by running the yast2 sysconfig configuration utility. The goal is for to create an ifcfg–id file for each slave device. The simplest way to accomplish –this is to configure the devices for DHCP. The name of the –configuration file for each device will be of the form: +this is to configure the devices for DHCP (this is only to get the +file ifcfg–id file created; see below for some issues with DHCP). The +name of the configuration file for each device will be of the form:

```
ifcfg–id–xx:xx:xx:xx:xx:xx
```

@@ –459,7 +557,7 @@ the device's permanent MAC address.

Once the set of ifcfg–id–xx:xx:xx:xx:xx:xx files has been created, it is necessary to edit the configuration files for the slave devices (the MAC addresses correspond to those of the slave devices). –Before editing, the file will contain multiple lines, and will look +Before editing, the file will contain multiple lines, and will look something like this:

```
BOOTPROTO='dhcp'  
@@ –496,16 +594,11 @@ STARTMODE="onboot"  
BONDING_MASTER="yes"  
BONDING_MODULE_OPTS="mode=active–backup miimon=100"  
BONDING_SLAVE0="eth0"  
–BONDING_SLAVE1="eth1"  
+BONDING_SLAVE1="bus–pci–0000:06:08.1"
```

Replace the sample BROADCAST, IPADDR, NETMASK and NETWORK values with the appropriate values for your network.

– Note that configuring the bonding device with BOOTPROTO='dhcp' –does not work; the scripts attempt to obtain the device address from

-DHCP prior to adding any of the slave devices. Without active slaves,  
-the DHCP requests are not sent to the network.

-

The STARTMODE specifies when the device is brought online.  
The possible values are:

@@ -531,9 +624,17 @@ for the bonding mode, link monitoring, a  
the max\_bonds bonding parameter; this will confuse the configuration  
system if you have multiple bonding devices.

- Finally, supply one BONDING\_SLAVEn="ethX" for each slave,  
-where "n" is an increasing value, one for each slave, and "ethX" is  
-the name of the slave device (eth0, eth1, etc).

+ Finally, supply one BONDING\_SLAVEn="slave device" for each  
+slave. where "n" is an increasing value, one for each slave. The  
+"slave device" is either an interface name, e.g., "eth0", or a device  
+specifier for the network device. The interface name is easier to  
+find, but the ethN names are subject to change at boot time if, e.g.,  
+a device early in the sequence has failed. The device specifiers  
+(bus-pci-0000:06:08.1 in the example above) specify the physical  
+network device, and will not change unless the device's bus location  
+changes (for example, it is moved from one PCI slot to another). The  
+example above uses one of each type for demonstration purposes; most  
+configurations will choose one or the other for all slave devices.

When all configuration files have been modified or created,  
networking must be restarted for the configuration changes to take  
@@ -544,7 +645,7 @@ effect. This can be accomplished via th

Note that the network control script (/sbin/ifdown) will  
remove the bonding module as part of the network shutdown processing,  
so it is not necessary to remove the module by hand if, e.g., the  
-module paramters have changed.  
+module parameters have changed.

Also, at this writing, YaST/YaST2 will not manage bonding  
devices (they do not show bonding interfaces on its list of network  
@@ -559,12 +660,37 @@ format can be found in an example ifcfg

Note that the template does not document the various BONDING\_  
settings described above, but does describe many of the other options.

### +3.1.1 Using DHCP with sysconfig

+-----

+

+ Under sysconfig, configuring a device with BOOTPROTO='dhcp'  
+will cause it to query DHCP for its IP address information. At this  
+writing, this does not function for bonding devices; the scripts  
+attempt to obtain the device address from DHCP prior to adding any of  
+the slave devices. Without active slaves, the DHCP requests are not  
+sent to the network.

+

### +3.1.2 Configuring Multiple Bonds with sysconfig

+-----  
+  
+ The sysconfig network initialization system is capable of  
+handling multiple bonding devices. All that is necessary is for each  
+bonding instance to have an appropriately configured ifcfg–bondX file  
+(as described above). Do not specify the "max\_bonds" parameter to any  
+instance of bonding, as this will confuse sysconfig. If you require  
+multiple bonding devices with identical parameters, create multiple  
+ifcfg–bondX files.  
+  
+ Because the sysconfig scripts supply the bonding module  
+options in the ifcfg–bondX file, it is not necessary to add them to  
+the system /etc/modules.conf or /etc/modprobe.conf configuration file.  
+  
3.2 Configuration with initscripts support  
-----

This section applies to distros using a version of initscripts with bonding support, for example, Red Hat Linux 9 or Red Hat –Enterprise Linux version 3. On these systems, the network +Enterprise Linux version 3 or 4. On these systems, the network initialization scripts have some knowledge of bonding, and can be configured to control bonding devices.

@@ –614,10 +740,11 @@ USERCTL=no

Be sure to change the networking specific lines (IPADDR, NETMASK, NETWORK and BROADCAST) to match your network configuration.

– Finally, it is necessary to edit /etc/modules.conf to load the –bonding module when the bond0 interface is brought up. The following –sample lines in /etc/modules.conf will load the bonding module, and –select its options:

+ Finally, it is necessary to edit /etc/modules.conf (or +/etc/modprobe.conf, depending upon your distro) to load the bonding +module with your desired options when the bond0 interface is brought +up. The following lines in /etc/modules.conf (or modprobe.conf) will +load the bonding module, and select its options:

```
alias bond0 bonding
options bond0 mode=balance–alb miimon=100
@@ –629,6 +756,33 @@ options for your configuration.
will restart the networking subsystem and your bond link should be now
up and running.
```

### +3.2.1 Using DHCP with initscripts

-----

+  
+ Recent versions of initscripts (the version supplied with  
+Fedora Core 3 and Red Hat Enterprise Linux 4 is reported to work) do  
+have support for assigning IP information to bonding devices via DHCP.  
+

+ To configure bonding for DHCP, configure it as described  
+above, except replace the line "BOOTPROTO=none" with "BOOTPROTO=dhcp"  
+and add a line consisting of "TYPE=Bonding". Note that the TYPE value  
+is case sensitive.

+  
+3.2.2 Configuring Multiple Bonds with initscripts

---

+ At this writing, the initscripts package does not directly  
+support loading the bonding driver multiple times, so the process for  
+doing so is the same as described in the "Configuring Multiple Bonds  
+Manually" section, below.

+  
+ NOTE: It has been observed that some Red Hat supplied kernels  
+are apparently unable to rename modules at load time (the "-obonding1"  
+part). Attempts to pass that option to modprobe will produce an  
+"Operation not permitted" error. This has been reported on some  
+Fedora Core kernels, and has been seen on RHEL 4 as well. On kernels  
+exhibiting this problem, it will be impossible to configure multiple  
+bonds with differing parameters.

### 3.3 Configuring Bonding Manually

---

@@ –638,10 +792,11 @@ scripts (the sysconfig or initscripts pa  
knowledge of bonding. One such distro is SuSE Linux Enterprise Server  
version 8.

– The general methodology for these systems is to place the  
–bonding module parameters into /etc/modprobe.conf, then add modprobe  
–and/or ifenslave commands to the system's global init script. The  
–name of the global init script differs; for sysconfig, it is  
+ The general method for these systems is to place the bonding  
+module parameters into /etc/modules.conf or /etc/modprobe.conf (as  
+appropriate for the installed distro), then add modprobe and/or  
+ifenslave commands to the system's global init script. The name of  
+the global init script differs; for sysconfig, it is  
/etc/init.d/boot.local and for initscripts it is /etc/rc.d/rc.local.

For example, if you wanted to make a simple bond of two e100  
@@ –649,7 +804,7 @@ devices (presumed to be eth0 and eth1),  
reboots, edit the appropriate file (/etc/init.d/boot.local or  
/etc/rc.d/rc.local), and add the following:

```
–modprobe bonding –obond0 mode=balance–alb miimon=100  
+modprobe bonding mode=balance–alb miimon=100  
modprobe e100  
ifconfig bond0 192.168.1.1 netmask 255.255.255.0 up  
ifenslave bond0 eth0  
@@ –657,11 +812,7 @@ ifenslave bond0 eth1
```

Replace the example bonding module parameters and bond0

network configuration (IP address, netmask, etc) with the appropriate  
–values for your configuration. The above example loads the bonding  
–module with the name "bond0," this simplifies the naming if multiple  
–bonding modules are loaded (each successive instance of the module is  
–given a different name, and the module instance names match the  
–bonding interface names).  
+values for your configuration.

Unfortunately, this method will not provide support for the  
ifup and ifdown scripts on the bond devices. To reload the bonding  
@@ –684,20 +835,23 @@ appropriate device driver modules. For  
the following:

```
# ifconfig bond0 down  
–# rmmod bond0  
+# rmmod bonding  
# rmmod e100
```

Again, for convenience, it may be desirable to create a script  
with these commands.

### –3.4 Configuring Multiple Bonds

#### –3.3.1 Configuring Multiple Bonds Manually

This section contains information on configuring multiple  
–bonding devices with differing options. If you require multiple  
–bonding devices, but all with the same options, see the "max\_bonds"  
–module parameter, documented above.  
+bonding devices with differing options for those systems whose network  
+initialization scripts lack support for configuring multiple bonds.  
+  
+ If you require multiple bonding devices, but all with the same  
+options, you may wish to use the "max\_bonds" module parameter,  
+documented above.

To create multiple bonding devices with differing options, it  
is necessary to load the bonding driver multiple times. Note that  
@@ –724,11 +878,16 @@ named "bond0" and creates the bond0 devi  
miimon of 100. The second instance is named "bond1" and creates the  
bond1 device in balance–alb mode with an miimon of 50.

+ In some circumstances (typically with older distributions),  
+the above does not work, and the second bonding instance never sees  
+its options. In that case, the second options line can be substituted  
+as follows:  
+  
+install bonding1 /sbin/modprobe bonding –obond1 mode=balance–alb miimon=50  
+

This may be repeated any number of times, specifying a new and  
–unique name in place of bond0 or bond1 for each instance.  
+unique name in place of bond1 for each subsequent instance.

– When the appropriate module paramters are in place, then  
–configure bonding according to the instructions for your distro.

## 5. Querying Bonding Configuration

=====

@@ –846,8 +1005,8 @@ tagged internally by bonding itself. As  
self generated packets.

For reasons of simplicity, and to support the use of adapters  
–that can do VLAN hardware acceleration offloading, the bonding  
–interface declares itself as fully hardware offloading capable, it gets  
+that can do VLAN hardware acceleration offloading, the bonding  
+interface declares itself as fully hardware offloading capable, it gets  
the add\_vid/kill\_vid notifications to gather the necessary  
information, and it propagates those actions to the slaves. In case  
of mixed adapter types, hardware accelerated tagged packets that  
@@ –880,7 +1039,7 @@ bond interface:  
matches the hardware address of the VLAN interfaces.

Note that changing a VLAN interface's HW address would set the  
–underlying device -- i.e. the bonding interface -- to promiscuous  
+underlying device -- i.e. the bonding interface -- to promiscuous  
mode, which might not be what you want.

@@ –923,7 +1082,7 @@ down or have a problem making it unrespo  
an additional target (or several) increases the reliability of the ARP  
monitoring.

– Multiple ARP targets must be seperated by commas as follows:  
+ Multiple ARP targets must be separated by commas as follows:

# example options for ARP monitoring with three targets

alias bond0 bonding

@@ –1045,7 +1204,7 @@ install bonding /sbin/modprobe tg3; /sbi

This will, when loading the bonding module, rather than  
performing the normal action, instead execute the provided command.  
This command loads the device drivers in the order needed, then calls  
–modprobe with --ignore–install to cause the normal action to then take  
+modprobe with --ignore–install to cause the normal action to then take  
place. Full documentation on this can be found in the modprobe.conf  
and modprobe manual pages.

@@ –1130,14 +1289,14 @@ association.

common to enable promiscuous mode on the device, so that all traffic  
is seen (instead of seeing only traffic destined for the local host).

The bonding driver handles promiscuous mode changes to the bonding

- master device (e.g., bond0), and propagates the setting to the slave
- +master device (e.g., bond0), and propagates the setting to the slave devices.

- For the balance–rr, balance–xor, broadcast, and 802.3ad modes,
- the promiscuous mode setting is propagated to all slaves.
  - +the promiscuous mode setting is propagated to all slaves.

- For the active–backup, balance–tlb and balance–alb modes, the
- promiscuous mode setting is propagated only to the active slave.
  - +promiscuous mode setting is propagated only to the active slave.

For balance–tlb mode, the active slave is the slave currently receiving inbound traffic.  
@@ –1148,46 +1307,182 @@ sending to peers that are unassigned or

- For the active–backup, balance–tlb and balance–alb modes, when the active slave changes (e.g., due to a link failure), the
- promiscuous setting will be propagated to the new active slave.
  - +promiscuous setting will be propagated to the new active slave.

## –12. High Availability Information

### +12. Configuring Bonding for High Availability

- High Availability refers to configurations that provide maximum network availability by having redundant or backup devices,
- links and switches between the host and the rest of the world.
  - 
  - There are currently two basic methods for configuring to maximize availability. They are dependent on the network topology and the primary goal of the configuration, but in general, a configuration can be optimized for maximum available bandwidth, or for maximum network availability.
  - +links or switches between the host and the rest of the world. The goal is to provide the maximum availability of network connectivity (i.e., the network always works), even though other configurations could provide higher throughput.

#### 12.1 High Availability in a Single Switch Topology

---

- If two hosts (or a host and a switch) are directly connected via multiple physical links, then there is no network availability penalty for optimizing for maximum bandwidth: there is only one switch (or peer), so if it fails, you have no alternative access to fail over to.
- 
- Example 1 : host to switch (or other host)
-

```

- +-----+ +-----+
- | eth0 eth0 | switch |
- | Host A +-----+ or |
- | +-----+ other |
- | eth1 eth1 | host |
- +-----+ +-----+

```

+ If two hosts (or a host and a single switch) are directly  
+connected via multiple physical links, then there is no availability  
+penalty to optimizing for maximum bandwidth. In this case, there is  
+only one switch (or peer), so if it fails, there is no alternative  
+access to fail over to. Additionally, the bonding load balance modes  
+support link monitoring of their members, so if individual links fail,  
+the load will be rebalanced across the remaining devices.

+  
+ See Section 13, "Configuring Bonding for Maximum Throughput"  
+for information on configuring bonding with one peer device.

+  
+12.2 High Availability in a Multiple Switch Topology

+-----

+  
+ With multiple switches, the configuration of bonding and the  
+network changes dramatically. In multiple switch topologies, there is  
+a trade off between network availability and usable bandwidth.

+  
+ Below is a sample network, configured to maximize the  
+availability of the network:

```

+ | |
+ |port3 port3|
+ +-----+ +-----+
+ | |port2 ISL port2| |
+ | switch A +-----+ switch B |
+ | | | |
+ +-----+ +-----+
+ |port1 port1|
+ | +-----+ |
+ +-----+ host1 +-----+
+ eth0 +-----+ eth1

```

+  
+ In this configuration, there is a link between the two  
+switches (ISL, or inter switch link), and multiple ports connecting to  
+the outside world ("port3" on each switch). There is no technical  
+reason that this could not be extended to a third switch.

+  
+12.2.1 HA Bonding Mode Selection for Multiple Switch Topology

+-----

+ In a topology such as the example above, the active–backup and  
+broadcast modes are the only useful bonding modes when optimizing for  
+availability; the other modes require all links to terminate on the  
+same peer for them to behave rationally.

–12.1.1 Bonding Mode Selection for single switch topology

---

+active–backup: This is generally the preferred mode, particularly if  
+ the switches have an ISL and play together well. If the  
+ network configuration is such that one switch is specifically  
+ a backup switch (e.g., has lower capacity, higher cost, etc),  
+ then the primary option can be used to insure that the  
+ preferred link is always used when it is available.  
+  
+broadcast: This mode is really a special purpose mode, and is suitable  
+ only for very specific needs. For example, if the two  
+ switches are not connected (no ISL), and the networks beyond  
+ them are totally independent. In this case, if it is  
+ necessary for some specific one–way traffic to reach both  
+ independent networks, then the broadcast mode may be suitable.

+12.2.2 HA Link Monitoring Selection for Multiple Switch Topology

---

+ The choice of link monitoring ultimately depends upon your  
+switch. If the switch can reliably fail ports in response to other  
+failures, then either the MII or ARP monitors should work. For  
+example, in the above example, if the "port3" link fails at the remote  
+end, the MII monitor has no direct means to detect this. The ARP  
+monitor could be configured with a target at the remote end of port3,  
+thus detecting that failure without switch support.

+ In general, however, in a multiple switch topology, the ARP  
+monitor can provide a higher level of reliability in detecting end to  
+end connectivity failures (which may be caused by the failure of any  
+individual component to pass traffic for any reason). Additionally,  
+the ARP monitor should be configured with multiple targets (at least  
+one for each switch in the network). This will insure that,  
+regardless of which switch is active, the ARP monitor has a suitable  
+target to query.

+13. Configuring Bonding for Maximum Throughput

---

+13.1 Maximizing Throughput in a Single Switch Topology

---

+ In a single switch configuration, the best method to maximize  
+throughput depends upon the application and network environment. The  
+various load balancing modes each have strengths and weaknesses in  
+different environments, as detailed below.

+ For this discussion, we will break down the topologies into  
+two categories. Depending upon the destination of most traffic, we

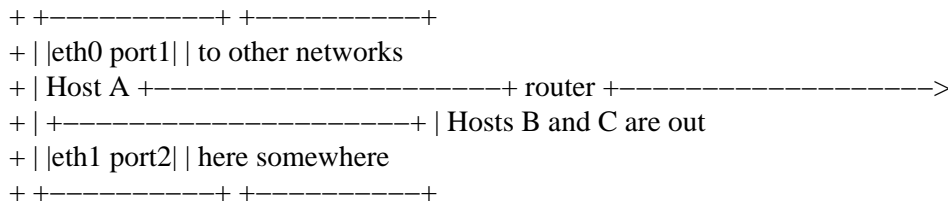
+categorize them into either "gatewayed" or "local" configurations.

+

+ In a gatewayed configuration, the "switch" is acting primarily  
 +as a router, and the majority of traffic passes through this router to  
 +other networks. An example would be the following:

+

+



+

+ The router may be a dedicated router device, or another host  
 +acting as a gateway. For our discussion, the important point is that  
 +the majority of traffic from Host A will pass through the router to  
 +some other network before reaching its final destination.

+

+ In a gatewayed network configuration, although Host A may  
 +communicate with many other systems, all of its traffic will be sent  
 +and received via one other peer on the local network, the router.

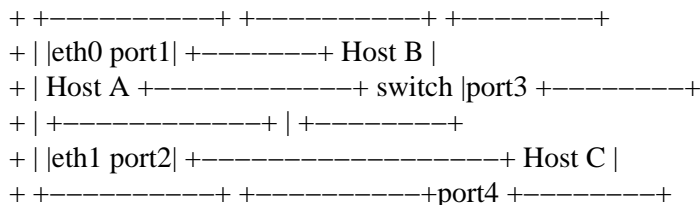
+

+ Note that the case of two systems connected directly via  
 +multiple physical links is, for purposes of configuring bonding, the  
 +same as a gatewayed configuration. In that case, it happens that all  
 +traffic is destined for the "gateway" itself, not some other network  
 +beyond the gateway.

+

+ In a local configuration, the "switch" is acting primarily as  
 +a switch, and the majority of traffic passes through this switch to  
 +reach other stations on the same network. An example would be the  
 +following:

+



+

+

+ Again, the switch may be a dedicated switch device, or another  
 +host acting as a gateway. For our discussion, the important point is  
 +that the majority of traffic from Host A is destined for other hosts  
 +on the same local network (Hosts B and C in the above example).

+

+ In summary, in a gatewayed configuration, traffic to and from  
 +the bonded device will be to the same MAC level peer on the network  
 +(the gateway itself, i.e., the router), regardless of its final  
 +destination. In a local configuration, traffic flows directly to and

+from the final destinations, thus, each destination (Host B, Host C)  
+will be addressed directly by their individual MAC addresses.

+

+ This distinction between a gatewayed and a local network  
+configuration is important because many of the load balancing modes  
+available use the MAC addresses of the local network source and  
+destination to make load balancing decisions. The behavior of each  
+mode is described below.

+

+

### +13.1.1 MT Bonding Mode Selection for Single Switch Topology

+

This configuration is the easiest to set up and to understand,  
although you will have to decide which bonding mode best suits your  
–needs. The tradeoffs for each mode are detailed below:  
+needs. The trade offs for each mode are detailed below:

balance–rr: This mode is the only mode that will permit a single  
TCP/IP connection to stripe traffic across multiple  
@@ –1206,6 +1501,23 @@ balance–rr: This mode is the only mode t  
interface's worth of throughput, even after adjusting  
tcp\_reordering.

+ Note that this out of order delivery occurs when both the  
+ sending and receiving systems are utilizing a multiple  
+ interface bond. Consider a configuration in which a  
+ balance–rr bond feeds into a single higher capacity network  
+ channel (e.g., multiple 100Mb/sec ethernet feeding a single  
+ gigabit ethernet via an etherchannel capable switch). In this  
+ configuration, traffic sent from the multiple 100Mb devices to  
+ a destination connected to the gigabit device will not see  
+ packets out of order. However, traffic sent from the gigabit  
+ device to the multiple 100Mb devices may or may not see  
+ traffic out of order, depending upon the balance policy of the  
+ switch. Many switches do not support any modes that stripe  
+ traffic (instead choosing a port based upon IP or MAC level  
+ addresses); for those devices, traffic flowing from the  
+ gigabit device to the many 100Mb devices will only utilize one  
+ interface.

+

If you are utilizing protocols other than TCP/IP, UDP for  
example, and your application can tolerate out of order  
delivery, then this mode can allow for single stream datagram  
@@ –1220,16 +1532,21 @@ active–backup: There is not much advanta  
connected to the same peer as the primary. In this case, a  
load balancing mode (with link monitoring) will provide the  
same level of network availability, but with increased  
– available bandwidth. On the plus side, it does not require  
– any configuration of the switch.  
+ available bandwidth. On the plus side, active–backup mode

## Linux–Kernel: [git patches] net driver fixes

- + does not require any configuration of the switch, so it may
- + have value if the hardware available does not support any of
- + the load balance modes.

balance–xor: This mode will limit traffic such that packets destined for specific peers will always be sent over the same interface. Since the destination is determined by the MAC addresses involved, this may be desirable if you have a large network with many hosts. It is likely to be suboptimal if all your traffic is passed through a single router, however. As with balance–rr, the switch ports need to be configured for addresses involved, this mode works best in a "local" network configuration (as described above), with destinations all on the same local network. This mode is likely to be suboptimal if all your traffic is passed through a single router (i.e., a "gatewayed" network configuration, as described above).

- + As with balance–rr, the switch ports need to be configured for "etherchannel" or "trunking."

broadcast: Like active–backup, there is not much advantage to this protocol includes automatic configuration of the aggregates, so minimal manual configuration of the switch is needed (typically only to designate that some set of devices is usable for 802.3ad). The 802.3ad standard also mandates that frames be delivered in order (within certain limits), so in general single connections will not see misordering of packets. The 802.3ad mode does have some drawbacks: the standard mandates that all devices in the aggregate operate at the same speed and duplex. Also, as with all bonding load balance modes other than balance–rr, no single connection will be able to utilize more than a single interface's worth of bandwidth. Additionally, the linux bonding 802.3ad implementation distributes traffic by peer (using an XOR of MAC addresses), so in general all traffic to a particular destination will use the same interface. Finally, the 802.3ad mode mandates the use of the MII monitor, therefore, the ARP monitor is not available in this mode.

- balance–tlb: This mode is also a good choice for this type of topology. It has no special switch configuration requirements, and balances outgoing traffic by peer, in a vaguely intelligent manner (not a simple XOR as in balance–xor or 802.3ad mode), so that unlucky MAC addresses will not all "bunch up" on a single interface. Interfaces may be of differing speeds. On the down side, in this mode all incoming traffic arrives over a single interface, this mode requires

- certain ethtool support in the network device driver of the
- slave interfaces, and the ARP monitor is not available.
- 
- balance–alb: This mode is everything that balance–tlb is, and more. It
- has all of the features (and restrictions) of balance–tlb, and
- will also balance incoming traffic from peers (as described in
- the Bonding Module Options section, above). The only extra
- down side to this mode is that the network device driver must
- support changing the hardware address while the device is
- open.
- + bandwidth.

#### –12.1.2 Link Monitoring for Single Switch Topology

---

- + Additionally, the linux bonding 802.3ad implementation
- + distributes traffic by peer (using an XOR of MAC addresses),
- + so in a "gatewayed" configuration, all outgoing traffic will
- + generally use the same device. Incoming traffic may also end
- + up on a single device, but that is dependent upon the
- + balancing policy of the peer's 802.3ad implementation. In a
- + "local" configuration, traffic will be distributed across the
- + devices in the bond.
- +
- + Finally, the 802.3ad mode mandates the use of the MII monitor,
- + therefore, the ARP monitor is not available in this mode.
- +
- +balance–tlb: The balance–tlb mode balances outgoing traffic by peer.
- + Since the balancing is done according to MAC address, in a
- + "gatewayed" configuration (as described above), this mode will
- + send all traffic across a single device. However, in a
- + "local" network configuration, this mode balances multiple
- + local network peers across devices in a vaguely intelligent
- + manner (not a simple XOR as in balance–xor or 802.3ad mode),
- + so that mathematically unlucky MAC addresses (i.e., ones that
- + XOR to the same value) will not all "bunch up" on a single
- + interface.
- +
- + Unlike 802.3ad, interfaces may be of differing speeds, and no
- + special switch configuration is required. On the down side,
- + in this mode all incoming traffic arrives over a single
- + interface, this mode requires certain ethtool support in the
- + network device driver of the slave interfaces, and the ARP
- + monitor is not available.
- +
- +balance–alb: This mode is everything that balance–tlb is, and more.
- + It has all of the features (and restrictions) of balance–tlb,
- + and will also balance incoming traffic from local network
- + peers (as described in the Bonding Module Options section,
- + above).
- +
- + The only additional down side to this mode is that the network

+ device driver must support changing the hardware address while  
+ the device is open.

+

+13.1.2 MT Link Monitoring for Single Switch Topology

+-----

The choice of link monitoring may largely depend upon which mode you choose to use. The more advanced load balancing modes do not support the use of the ARP monitor, and are thus restricted to using –the MII monitor (which does not provide as high a level of assurance –as the ARP monitor).

–

+the MII monitor (which does not provide as high a level of end to end  
+assurance as the ARP monitor).

–12.2 High Availability in a Multiple Switch Topology

-----