Intelligent NIC™

Version: Readme-Linux-3.4.19v1.pdf
Date: 3/19/07
Description: Driver and Firmware Release Notes for Linux OS
Product: 10GbE Dual Port Card

Table of Contents

1.0 Introduction

This release notes describes the basic installation and configuration of the 10GbE adapter. It also includes the requirements, caveats and contents specific to this release.

2.0 Release Package Contents

NIC Linux Driver
TOE Linux Driver
Firmware
Diagnostics
Tools
Documentation

3.0 New in this Release

1. Driver Version 3.4.19
2. Firmware Version 3.4.19
3. Updated Release Notes
4. Known Issues
    a. After using the recover option from the flash tool, incorrect version number of the firmware is displayed

5. Fixed Issues & Changes

   a. The port order has been changed such that Port 0 connects to the switch in slot IO7 & Port 1 connects to the switch in slot IO9. This includes the proper order of MAC addresses assigned to the ports to reflect the order in which they are stored in the VPD.

   b. Removed the warning message printed by the driver when temperature goes to 60 degrees C, asking user to take action.

5.0 System Requirements

Table 1 Systems Requirements Summary

Hardware	**Systems**	**Validated Platforms**	**Validated OS (32bit and x64)**
PCIe: x1,x4,x8	Memory: Minimum 2GB	AMD Intel	RHEL4 U3,U4 SLES9 SP3 SLES10

Table 2 Software Requirements Summary

NIC Linux	**TOE Linux**	**Comments**
-- Linux kernel source/headers	-- Linux kernel source/headers -- TOE intercept driver	Linux NIC and TOE drivers are distributed in the source code

5.0 Caveats

1. Install the Linux kernel sources/headers to compile the source code driver. Some Linux OS distributions during installation may not install the kernel source/headers by default. Add the kernel sources/header files from the Linux Operating System Distribution disk.

2. Linux TOE Driver Ordering -- Load the Linux NIC driver first, followed by the TOE intercept driver (*nx_intercept.ko*) next. The TOE intercept driver must be running before loading and using the Linux TOE driver.

3. Updating from earlier versions of firmware

   a. If the card currently contains 3.3.9, 3.3.12, 3.4.2 or 3.4.5, 3.4.7, 3.4.15 version of the firmware, refer to the respective Release Notes released with the earlier versions.

   b. If you had been using an earlier version of the driver:

i. First uninstall the old NIC driver using *rmmod unm_nic*
ii. Next, remove the old driver from the kernel library
1. rm */lib/modules/<version>/kernel/drivers/net/unm_nic.ko*

c. Install the new driver, and **BRING UP THE NETWORK INTERFACE** (*ifconfig eth[x] up*) **BEFORE** starting the firmware update with the flash utility.
d. **REBOOT** the host after the firmware update is completed.

4. Driver loading sequence for the interface ports
   a. Each port has its own PCI funtion. Functions 2 and 3 are used. Function 0 is not used.
   By default, the two ports get the assignment of eth[x] and eth[x+y] depending on what other eth interfaces are already in the system. eth[x] connects to switch in IO7 and eth[x+y] connects to switch in IO9.

   NOTE: Linux, ifconfig has the option to let the user change the eth naming and so the default explained above can be changed to whatever order is required

5. Firmware Recovery

   During the firmware update process, do not reboot or stop the firmware process, this may corrupt the firmware.

   There are checks built into the flash utility, and during the firmware update, the MD5 recovery step can identify incorrect firmware and stop the update process. The flash tool will not start the firmware update process if the Ethernet interface is NOT UP.

   In the event of running the diagnostics, if it is found that there are identifiable errors in the log, the flash utility's "recover" option can be used to revert back to the previous version and thus restoring the firmware image.

6.0 Installation and Configuration

Updating Driver and Firmware

1. If an existing driver is already loaded, identify the current Firmware and Driver version
   a. Use the flash utility from the driver's bin subdirectory or the Linux OS ethtool
      i. ./nxflash -i eth[x] - -info
      ii. ethtool –i eth[x]
      iii. Remove the old NIC driver
         1. (rmmod *unm_nic;* rm /lib/modules/<kernel version>/kernel/drivers/net/*unm_nic.ko)*
         2. (rmmod *nx_nic;* rm /lib/modules/<kernel version>/kernel/drivers/net/*nx_nic.ko)*

2. Un-tar the driver package, build the driver source, install the driver module, load the driver

module, bring up the Ethernet interface and verify the link.

a. Un-tar the driver package
    i. tar xvzf <version>.tar.gz.nx
b. Build and install the driver module
    i. cd <version>
    ii. make
    iii. make install
c. Load the driver module
    i. insmod ./driver/nx_nic.ko
d. Bring up and verify all the Ethernet interfaces
    i. ifconfig eth[x] <ipaddress> up
    ii .lsmod | grep nx
        1. (checks status of the driver module loaded; value should be 2)
    iii. dmesg
        1. (checks for system messages related to the card)
e. Test the Ethernet interface using ping
f. Update the firmware
    i. ./bin/nxflash -i eth[x] - -info
    ii. ./bin/nxflash -i eth[x] -u
    iii. Reboot the system

Linux TOE Driver Installation

1. First, bring up the interface with the NIC driver(nx_nic.ko)
2. Verify that the NIC driver can ping
3. Next, build the Intercept and TOE driver
    a. tar xvzf <version>-tnic.tar.gz.nx
    b. cd <version>-tnic
    c. make
    d. make install
4. Load the Intercept driver
    a. insmod ./driver/nx_intercept.ko
5. Load the Linux TOE driver
    a. insmod ./driver/nx_tnic.ko
6. Use offload rules to offload the applications
7. To unload TOE driver, first remove the offload rules(see command in Fig. below), and type
    a. *rmmod nx_tnic*

NOTE: The Intercept driver must be running at all times for the TOE driver to work. Whenever the server is rebooted, check if the NIC and Intercept drivers are running before using the TOE driver.

TOE commands for offload

root@apps03 3.4.19]# nxoffload
usage: nxoffload <options>

options: -p Port number to be offloaded.
-n Application name to be offloaded.
-t TCP tuple to be offloaded.
Format: local IP, local port, remote IP, remote port.
Example:
nxoffload -a -t'0, 0, 0, 5001'.
nxoffload -a -t'0, 2000, 0, 5001'.
-a Add an offload rule.
-r Remove an offload rule.
-z Enable zero copy.
-s List all ports and applications offloaded.

Example: Offload an application
[root@apps03 3.4.19]# nxoffload -an iperf
Example: Verify list of applications that is using TOE (offloaded)
[root@apps03 3.4.19]# nxoffload -s

1 offload rule configured
==========

1. Application Name : iperf (TX zero copy off)

Linux Teaming/Bonding Mode Behavior

The actual source code of the Linux bonding driver was analyzed. Based on the source code, the behavior seen is exactly as expected for bonding driver set to Mode 1. The Mode 0 setting will give a different behavior.

Steps followed:

```
sudo insmod driver/nx_nic.ko
```

```
modprobe bonding mode=1 miimon=100 updelay=50000 primary=eth16 ifconfig bond0
25.145.1.199 netmask 255.255.255.0 broadcast 25.145.1.255 up ifenslave bond0
eth2 eth3
```

Man Page:

```
Bonding Driver : mode

MODE 1
active-backup or mode=1
Active-backup policy: Only one slave in the bond is active.
A different slave becomes active if, and only if, the active slave fails.
```

By setting mode=1, when ifenslave is executed the bonding driver calls into bond_enslave() and sets the slave link modes for eth15 and eth16 to BOND_LINK_DOWN.

Since the bond mode is 1, ie. BOND_MODE_ACTIVEBACKUP, and primary device is indicated, one slave device is configured as active slave and dmesg will show "bonding: bond0: making interface eth16 the new active one 0 ms earlier." and this will result in bond->current_slave being updated and

MII Status: up
will be reported for this link and this is what cat /proc/net/bonding/bond0 prints.

For the other port, dmesg will display,
bonding: bond0: link status up for interface eth15, enabling it in 50000 ms.
and 'cat /proc/net/bonding/bond0' will print for eth15.

MII Status: down
after 50 seconds dmesg will print,

bonding: bond0: link status definitely up for interface eth15.
bonding: bond0: making interface eth15 the new active one.
'cat /proc/net/bonding/bond0' will print MII Status: up for eth15.

MODE 0
But if the mode is set to 0 i.e. not BOND_MODE_ACTIVEBACKUP, (in this case Round Robin)

modprobe bonding mode=0 miimon=100 updelay=50000 primary=eth16

then both links delay for 50 secs before their states are set to BOND_LINK_UP state,

```
blade12$ cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v3.0.1 (January 9, 2006)

Bonding Mode: load balancing (round-robin) MII Status: up MII Polling Interval
(ms): 100 Up Delay (ms): 50000 Down Delay (ms): 0

Slave Interface: eth16
MII Status: down                                <========= down state
Link Failure Count: 0
Permanent HW addr: 00:0e:1e:00:11:d2

Slave Interface: eth15
MII Status: down                                <========= down state
Link Failure Count: 0
Permanent HW addr: 00:0e:1e:00:11:d3
```

after 50 secs:

```
blade12$ cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v3.0.1 (January 9, 2006)

Bonding Mode: load balancing (round-robin) MII Status: up MII Polling Interval
(ms): 100 Up Delay (ms): 50000 Down Delay (ms): 0
```

```
Slave Interface: eth16
MII Status: up                                  <========== up state
Link Failure Count: 0
Permanent HW addr: 00:0e:1e:00:11:d2

Slave Interface: eth15
MII Status: up                                  <========== up state
Link Failure Count: 0
Permanent HW addr: 00:0e:1e:00:11:d3
```

And dmesg will print:

```
bonding: bond0: link status definitely up for interface eth16.
bonding: bond0: link status definitely up for interface eth15.
```

7.0 Disclaimer

The information in this document is subject to change without notice.