Mellanox OFED for FreeBSD User Manual

Rev 2.1.6

www.mellanox.com

NOTE:
THIS HARDWARE, SOFTWARE OR TEST SUITE PRODUCT (“PRODUCT(S)”) AND ITS RELATED DOCUMENTATION ARE PROVIDED BY MELLANOX TECHNOLOGIES “AS-IS” WITH ALL FAULTS OF ANY KIND AND SOLELY FOR THE PURPOSE OF AIDING THE CUSTOMER IN TESTING APPLICATIONS THAT USE THE PRODUCTS IN DESIGNATED SOLUTIONS. THE CUSTOMER'S MANUFACTURING TEST ENVIRONMENT HAS NOT MET THE STANDARDS SET BY MELLANOX TECHNOLOGIES TO FULLY QUALIFY THE PRODUCT(S) AND/OR THE SYSTEM USING IT. THEREFORE, MELLANOX TECHNOLOGIES CANNOT AND DOES NOT GUARANTEE OR WARRANT THAT THE PRODUCTS WILL OPERATE WITH THE HIGHEST QUALITY. ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT ARE DISCLAIMED. IN NO EVENT SHALL MELLANOX BE LIABLE TO CUSTOMER OR ANY THIRD PARTIES FOR ANY DIRECT, INDIRECT, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES OF ANY KIND (INCLUDING, BUT NOT LIMITED TO, PAYMENT FOR PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY FROM THE USE OF THE PRODUCT(S) AND RELATED DOCUMENTATION EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Mellanox Technologies
350 Oakmead Parkway Suite 100
Sunnyvale, CA 94085
U.S.A.
www.mellanox.com
Tel: (408) 970-3400
Fax: (408) 970-3403

Table of Contents

List of Tables

Document Revision History

Table 1 - Document Revision History

Release	Date	Description

2.1.6	June 2015	• Added the following sections:
		  • Section 3.1.3, “EEPROM Cable Module Information Reader”, on page 21
		• Updated the following sections:
		  • 3.1.2 “Packet Pacing,” on page 18

2.1.5	January 15, 2015	• Added the following sections:
		  • Section 3, “Features Overview and Configuration”, on page 17
		• Updated the following sections:
		  • Section 1, “Overview”, on page 9
		  • Section 2, “Installation”, on page 11
		  • Section 4, “Performance Tuning”, on page 22

About this Manual

This Preface provides general information concerning the scope and organization of this User's Manual.

Intended Audience

This manual is intended for system administrators responsible for the installation, configuration, management and maintenance of the software and hardware of VPI (InfiniBand, Ethernet) adapter cards. It is also intended for application developers.

Common Abbreviations and Acronyms

Table 2 - Abbreviations and Acronyms

Abbreviation / Acronym	Whole Word / Description
B	(Capital) 'B' is used to indicate size in bytes or multiples of bytes (e.g., 1KB = 1024 bytes, and 1MB = 1048576 bytes)
b	(Small) 'b' is used to indicate size in bits or multiples of bits (e.g., 1Kb = 1024 bits)
FW	Firmware
HCA	Host Channel Adapter
HW	Hardware
IB	InfiniBand
LSB	Least significant *byte*
lsb	Least significant *bit*
MSB	Most significant *byte*
msb	Most significant *bit*
NIC	Network Interface Card
SW	Software
VPI	Virtual Protocol Interconnect
PFC	Priority Flow Control
PR	Path Record
RDS	Reliable Datagram Sockets
RoCE	RDMA over Converged Ethernet
SL	Service Level
QoS	Quality of Service
ULP	Upper Level Protocol
VL	Virtual Lane

Glossary

The following is a list of concepts and terms related to InfiniBand in general and to Subnet Managers in particular. It is included here for ease of reference, but the main reference remains the

InfiniBand Architecture Specification.

Table 3 - Glossary

Channel Adapter (CA), Host Channel Adapter (HCA)	An IB device that terminates an IB link and executes transport functions. This may be an HCA (Host CA) or a TCA (Target CA).
HCA Card	A network adapter card based on an InfiniBand channel adapter device.
IB Devices	Integrated circuit implementing InfiniBand compliant communication.
In-Band	A term assigned to administration activities traversing the IB connectivity only.
Local Port	The IB port of the HCA through which IBDIAG tools connect to the IB fabric.
Master Subnet Manager	The Subnet Manager that is authoritative, that has the reference configuration information for the subnet. See Subnet Manager.
Multicast Forwarding Tables	A table that exists in every switch providing the list of ports to forward received multicast packet. The table is organized by MLID.
Network Interface Card (NIC)	A network adapter card that plugs into the PCI Express slot and provides one or more ports to an Ethernet network.
Unicast Linear Forwarding Tables (LFT)	A table that exists in every switch providing the port through which packets should be sent to each LID.
Virtual Protocol Interconnet (VPI)	A Mellanox Technologies technology that allows Mellanox channel adapter devices (ConnectX®) to simultaneously connect to an InfiniBand subnet and a 10GigE subnet (each subnet connects to one of the adpater ports)

Related Documentation

Table 4 - Reference Documents

Document Name	Description
InfiniBand Architecture Specification, Vol. 1, Release 1.2.1	The InfiniBand Architecture Specification that is provided by IBTA

Support and Updates Webpage

Please visit http://www.mellanox.com > Products > Software > Ethernet Drivers > FreeBSD Drivers for downloads, FAQ, troubleshooting, future updates to this manual, etc.

1 Overview

This document provides information on the Mellanox driver for FreeBSD and instructions for installing the driver on Mellanox ConnectX® adapter cards supporting the following uplinks to servers:

- ConnectX®-3/ConnectX®-3 Pro:
  - InfiniBand: QDR, FDR10, FDR
  - Ethernet: 10GigE, 40GigE

The driver release introduces the following capabilities:

- Single/Dual port
- Up to 16 Rx queues per port
- Up to 32 Tx queues per port (according to number of CPUs)
- MSI-X or INTx
- Adaptive interrupt moderation
- Hardware Tx/Rx checksum calculation
- Large Send Offload (i.e., TCP Segmentation Offload)
- Large Receive Offload
- VLAN Tx/Rx acceleration (Hardware VLAN stripping/insertion)
- Net device statistics

1.1 Mellanox OFED for FreeBSD Package Contents

1.1.1 Tarball Package

Mellanox OFED for FreeBSD package includes the following directories:

- modules - contains the relevant Makefiles
- ofed - source code

1.1.2 mlx4 driver

mlx4 is the low level driver implementation for the ConnectX adapters designed by Mellanox Technologies. The ConnectX® can operate as an InfiniBand adapter and as an Ethernet NIC.

1.1.2.1 Software Components

Mellanox OFED for FreeBSD contains the following software components:

Table 5 - Mellanox OFED for FreeBSD Software Components

Components	Description
mlx4_core	Handles low-level functions like device initialization and firmware commands processing. Also controls resource allocation so that the InfiniBand, Ethernet and FC functions can share a device without interfering with each other.
mlx4_en	Handles Ethernet specific functions and plugs into the netdev mid-layer.

Table 5 - Mellanox OFED for FreeBSD Software Components

Components	Description
mlx4_ib	Handles InfiniBand specific functions supplied by ib_core in order to interact with verbs and ULPs.
Documentation	Release Notes, User Manual

2 Installation

This chapter describes how to install and test the Mellanox driver for FreeBSD package on a single host machine with Mellanox InfiniBand and/or Ethernet adapter hardware installed.

2.1 Software Dependencies

- To install the driver software, kernel sources must be installed on the machine.
- To run the Packet Pacing feature, the kernel patch and dedicated FW version included in the Packet Pacing package need to be applied.

2.2 Downloading Mellanox Driver for FreeBSD

1. Verify that the system has a Mellanox network adapter (HCA/NIC) installed.

   The following example shows a system with an installed Mellanox HCA:

   ```
   # pciconf -lv | grep Mellanox -C 3
   mlx4_core0@pci0:7:0:0:  class=0x028000 card=0x000615b3 chip=0x100315b3 rev=0x00 hdr=0x00
       vendor     = 'Mellanox Technologies'
       device     = 'MT27500 Family [ConnectX-3]'
       class      = network
   ```

2. Download the tarball image to your host.
   The image name has the format `MLNX_OFED_FreeBSD-<ver>.tgz`. You can download it from http://www.mellanox.com > Products > Software > Ethernet Drivers > FreeBSD
   or
   http://www.mellanox.com > Products > Software > InfiniBand/VPI drivers > FreeBSD
3. Use the md5sum utility to confirm the file integrity of your tarball image.

2.3 Installing Mellanox Driver for FreeBSD

Prior to installing the driver, please re-compile (and install) the kernel with NO OFED options/ devices enabled.

1. Extract the tarball.
2. Compile and load needed modules in the following order of dependencies:
   - Ethernet Driver:

     mlx4_core

     a.Go to the mlx4 directory. Run:

     ```
     # cd modules/mlx4
     ```

     b. Clean any previous dependencies. Run:

     ```
     # make -m $HEAD/share/mk SYSDIR=$HEAD/sys clean cleandepend
     ```

c. Compile the mlx4_core module. Run:

```
# make -m $HEAD/share/mk SYSDIR=$HEAD/sys
```

Note: For packet pacing support, add `CONFIG_RATELIMIT=yes`. This option has a kernel patch dependency.

d. Install the mlx4_core module. Run:

```
# make -m $HEAD/share/mk SYSDIR=$HEAD/sys install
```

e. Load the mlx4_core module. Run:

```
# kldload mlx4
```

mlxen

a.Go to the mlxen directory. Run:

```
# cd modules/mlxen
```

b. Clean any previous dependencies. Run:

```
# make -m $HEAD/share/mk SYSDIR=$HEAD/sys clean cleandepend
```

c. Compile the mlxen module. Run:

```
# make -m $HEAD/share/mk SYSDIR=$HEAD/sys
```

Note: For packet pacing support, add `CONFIG_RATELIMIT=yes`. This option has a kernel patch dependency.

d. Install the mlxen module. Run:

```
# make -m $HEAD/share/mk SYSDIR=$HEAD/sys install
```

e. Load the mlxen module. Run:

```
# kldload mlxen
```

- InfiniBand Driver:

mlx4_core

Run the same steps as specified for Ethernet driver above.

ib_core

a.Go to the ib_core directory. Run:

```
# cd modules/ibcore
```

b. Clean any previous dependencies. Run:

```
# make -m $HEAD/share/mk SYSDIR=$HEAD/sys clean cleandepend
```

c. Compile the ib_core module. Run:

```
# make -m $HEAD/share/mk SYSDIR=$HEAD/sys
```

d. Install the ib_core module. Run:

```
# make -m $HEAD/share/mk SYSDIR=$HEAD/sys install
```

e. Load the ib_core module. Run:

```
# kldload ibcore
```

mlx4_ib

a.Go to the mlx4_ib directory. Run:

```
# cd modules/mlx4ib
```

b. Clean any previous dependencies. Run:

```
# make -m $HEAD/share/mk SYSDIR=$HEAD/sys clean cleandepend
```

c. Compile the mlx4_ib module. Run:

```
# make -m $HEAD/share/mk SYSDIR=$HEAD/sys
```

d. Install the mlx4_ib module. Run:

```
# make -m $HEAD/share/mk SYSDIR=$HEAD/sys install
```

e. Load the mlx4_ib module. Run:

```
# kldload mlx4ib
```

ipoib

a.Go to the ipoib directory. Run:

```
# cd modules/ipoib
```

b. Clean any previous dependencies. Run:

```
# make -m $HEAD/share/mk SYSDIR=$HEAD/sys clean cleandepend
```

c. Compile the ipoib module. Run:

```
# make -m $HEAD/share/mk SYSDIR=$HEAD/sys
```

d. Install the ipoib module. Run:

```
# make -m $HEAD/share/mk SYSDIR=$HEAD/sys install
```

e. Load the ipoib module. Run:

```
# kldload ipoib
```

To load a module on reboot, add "mlx4_load=="YES"/mlxen_load="YES"/ibcore_load="YES"/ mlx4ib_load="YES"/ipoib="YES"" to the '/boot/ loader.conf' file (create if does not exist).

Run "kldstat" in order to verify which modules are loaded on your server.

2.4 Firmware Programming

The adapter card was shipped with the most current firmware available. This section is intended for future firmware upgrades, and provides instructions for (1) installing Mellanox firmware update tools (MFT), (2) downloading FW, and (3) updating adapter card firmware.

2.4.1 Installing Firmware Tools

Step 1. Download the current Mellanox Firmware Tools package (MFT) from www.mellanox.com > Products > Adapter IB/VPI SW > Firmware Tools.
The tools package to download is "MFT_SW for FreeBSD" (tarball name is mft-X.X.X.tgz).

Step 2. Extract the tarball and run the installation script.

2.4.2 Downloading Firmware

1. Retrieve device's PCI slot (i.e. pci0:x:0:0). Run:

```
$ mst status
```

2. Verify your card's PSID.

```
$flint -d <pci> q
```

3. Download the desired firmware from the Mellanox website.
   http://www.mellanox.com/page/firmware_download

2.4.3 Updating Firmware Using flint

1. Unzip the firmware binary file:

```
$flint -d <pci> -i <img.bin> b
```

2. Reboot the server.

2.4.4 Updating Firmware Using mlxburn (.mlx file)

1. Extract the relevant firmware package (ConnectX3/ConnectX3-Pro).
2. From the extracted directory, run:

```
$mlxburn -d <NIC's_pci_slot> -fw <.mlx file> -conf_dir_list .
```

Example

```
$mlxburn -d <NIC's_pci_slot> -fw fw-ConnectX3-rel.mlx -conf_dir_list .
```

3. Reboot the server.

2.5 Driver Usage and Configuration

- ***To assign an IP address to the interface:***

```
#> ifconfig mlxen<x> <ip>
```

Note: <x> is the OS assigned interface number

- ***To check driver and device information:***

```
#> pciconf -lv | grep mlx
#> flint -d pci<w:x:y:z> q
```

Example:

```
#> pciconf -lv | grep mlx
mlx4_core0@pci0:7:0:0:  class=0x028000 card=0x003715b3 chip=0x100315b3 rev=0x00 hdr=0x00
#> flint -d pci0:7:0:0 q
Image type:      FS2
FW Version:      2.11.1294
Device ID:       4099
Description:     Node             Port1            Port2            Sys image
GUIDs:           0002c903002ffcc0 0002c903002ffcc1 0002c903002ffcc2 0002c903002ffcc3
MACs:                                 0002c92ffcc0     0002c92ffcc1
VSD:
PSID:            MT_1020120019
```

- ***To check driver version:***

```
#> dmesg
```

Example:

```
#> dmesg
mlx4_core: Mellanox ConnectX core driver v2.1 (Aug 21 2014)
mlx4_en: Mellanox ConnectX HCA Ethernet driver v2.1 (Aug 18 2014)
```

- ***To query stateless offload status:***

```
#> ifconfig mlxen<x>
```

Note: <x> is the OS assigned interface number

- ***To set stateless offload status:***

```
#> ifconfig mlxen<x> [rxcsum|-rxcsum] [txcsum|-txcsum] [tso|-tso] [lro|-lro]
```

Note: <x> is the OS assigned interface number

- ***To query interrupt coalescing settings:***

```
#> sysctl -a | grep adaptive
```

Example:

```
#> sysctl -a | grep adaptive
hw.mlxen0.conf.adaptive_rx_coal: 1
hw.mlxen1.conf.adaptive_rx_coal: 1
```

- ***To enable/disable adaptive interrupt moderation:***

```
#>sysctl hw.mlxen<x>.conf.adaptive_rx_coal=[1/0]
```

Note: <x> is the OS assigned interface number

By default, the driver uses adaptive interrupt moderation for the receive path, which adjusts the moderation time to the traffic pattern.

- ***To query values for packet rate limits and for moderation time high and low:***

```
#> sysctl -a | grep pkt_rate
#> sysctl -a | grep rx_usecs
```

- ***To set the values for packet rate limits and for moderation time high and low:***

```
#> sysctl hw.mlxen<x>.conf.pkt_rate_low=[N]
#> sysctl hw.mlxen<x>.conf.pkt_rate_high=[N]
#> sysctl hw.mlxen<x>.conf.rx_usecs_low=[N]
#> sysctl hw.mlxen<x>.conf.rx_usecs_high=[N]
```

Note: <x> is the OS assigned interface number

Example:

```
#> sysctl -a | grep pkt_rate
hw.mlxen0.conf.pkt_rate_low: 400000
hw.mlxen0.conf.pkt_rate_high: 450000
hw.mlxen1.conf.pkt_rate_low: 400000
hw.mlxen1.conf.pkt_rate_high: 450000
sysctl -a | grep rx_usecs
hw.mlxen0.conf.rx_usecs_low: 0
hw.mlxen0.conf.rx_usecs_high: 128
hw.mlxen1.conf.rx_usecs_low: 0
hw.mlxen1.conf.rx_usecs_high: 128
```

Above an upper limit of packet rate, adaptive moderation will set the moderation time to its highest value. Below a lower limit of packet rate, the moderation time will be set to its lowest value.

- ***To query pause frame settings:***

```
#> ifconfig -m mlxen<x>
```

Note: <x> is the OS assigned interface number

- ***To set pause frame settings:***

```
#> ifconfig -m mlxen<x> [-]mediaopt [rxpause,txpause]
```

Note: <x> is the OS assigned interface number

- ***To query ring size values:***

```
#> sysctl -a | grep _size
```

Example:

```
#> sysctl -a | grep _size
hw.mlxen0.conf.rx_size: 1024
hw.mlxen0.conf.tx_size: 1024
hw.mlxen1.conf.rx_size: 1024
hw.mlxen1.conf.tx_size: 1024
```

- ***To modify rings size:***

```
#> sysctl hw.mlxen<x>.conf.rx_size=[N]
#> sysctl hw.mlxen<x>.conf.tx_size=[N]
```

Note: <x> is the OS assigned interface number

- ***To obtain additional device statistics:***

```
#> sysctl -a | grep mlx | grep stat
```

The driver defaults to the following parameters:

- Both ports are activated (i.e., a net device is created for each port)
- The number of Rx rings for each port is the nearest power of 2 of number of CPU cores, limited by 16.
- LRO is enabled with 32 concurrent sessions per Rx ring

3 Features Overview and Configuration

3.1 Ethernet Network

3.1.1 RDMA over Converged Ethernet (RoCE)

RoCE allows InfiniBand (IB) transport applications to work over an Ethernet network. RoCE encapsulates the InfiniBand transport and the GRH headers in Ethernet packets bearing a dedicated ether type (0x8915). Thus, any VERB application that works in an InfiniBand fabric can also work in an Ethernet fabric.

RoCE is enabled only for drivers that support VPI (currently, only mlx4).

➢ ***When working with RDMA applications over Ethernet link layer, the following points should be noted:***

- The presence of a Subnet Manager (SM) is not required in the fabric. Thus, operations that require communication with the SM are managed in a different way in RoCE. This does not affect the API.
- Since the SM is not present, querying a path is impossible. Therefore, the path record structure must be filled with the relevant values before establishing a connection.
  It is recommended working with RDMA-CM to establish a connection as it takes care of filling the path record structure.
- Since LID is a layer 2 attribute of the InfiniBand protocol stack, it is not set for a port and is displayed as zero when querying the port.
- With RoCE, APM is not supported.
- The GID table for each port is populated with N+1 entries where N is the number of IP addresses that are assigned to all network devices associated with the port including VLAN devices.
- The first entry in the GID table (at index 0) for each port is always present and equal to the link local IPv6 address of the net device that is associated with the port. Note that even if the link local IPv6 address is not set, index 0 is still populated.
- GID format can be of 2 types, IPv4 and IPv6. IPv4 GID is a IPv4-mapped IPv6 address[1] while IPv6 GID is the IPv6 address itself.
- Load the following modules for RoCE support: mlx4_core, ib_core, mlx4_ib, and mlx4_en.

1. For the IPv4 address A.B.C.D the corresponding IPv4-mapped IPv6 address is ::ffff.A.B.C.D

3.1.2 Packet Pacing

Packet pacing, also known as the “rate limit,” defines a maximum bandwidth allowed for a TCP connection. Limitation is done by hardware where each QP (transmit queue) has a rate limit value from which it calculates the delay between each packet sent.

3.1.2.1 Setting Properties for Packet Pacing

Before loading mlxen module, set which property (0-7) is relevant for packet pacing sockets.

Run:

```
kenv hw.mlx4_en.config_prios_for_rl_rings=<#priority>,<#priority>,<#priority>…
```

For example:

```
kenv hw.mlx4_en.config_prios_for_rl_rings=0,1,2…
```

When no priorities are being selected, the default priority supported with packet pacing sockets would be priority zero.

3.1.2.2 Setting Rates for Packet Pacing

Rates that are being used with packet pacing must be defined in advance.

New Rates Configuration

- Newly configured rates must be within a certain range, determined by the firmware, and they can be read through sysctl.
  - For a minimum value, run:

    ```
    #sysctl sys.device.<DEVICE NAME>.rate_limit_caps.min_value
    ```

  - For a maximum value, run:

    ```
    #sysctl sys.device.<DEVICE NAME>.rate_limit_caps.max_value
    ```

- The number of configured rates is also determined by the firmware. In order to check how many rates can be defined, run:

  ```
  #sysctl hw.<INTERFACE NAME>.conf.num_rates
  ```

- A new rate can be added by any the following methods:
  - Add a rate per index (from index 1 to num_rates):

    ```
    #sysctl hw.<INTERFACE NAME>.conf.rate_limit_1=400000
    #sysctl hw.<INTERFACE NAME>.conf.rate_limit_120=500000
    ```

    In order to read which rate was defined for a specific index (for example, Index 1), run:

    ```
    #sysctl hw.<INTERFACE NAME>.conf.rate_limit_1
    ```

    

    Each index can be defined with a rate only once.

  - Add a rate in an unknown index:

    ```
    #sysctl hw. <INTERFACE NAME>.conf.add_rate=600000
    ```

This will add the defined rate to the next available index. If all rates were already defined with an index, the new rate will not be added.

Rates are determined and then saved in bits per second.
Rates requested for a new socket are added in bytes per second.

Limitation: Rate values must be multiples of 1000.

- There are two burst levels that can be defined for each index:
  - Burst low: Burst capacity is limited to a lower range to allow better pacing
  - Burst high: Burst capacity is limited to a higher range to allow better bandwidth

  For changing the burst level per index, run:

```
#sysctl hw. <INTERFACE NAME>.conf.burst_size_1=burst_high/burst_low
```

  In order to read which burst level was defined for an index (for example, index 1), run:

```
#sysctl hw. <INTERFACE NAME>.conf.burst_size_1
```

- To display the packet pacing configuration, run:

```
#sysctl hw. <INTERFACE NAME>.conf.rate_limit_show
hw. <INTERFACE NAME>.conf.rate_limit_show:
INDEX   CURRENTLY USED BURST   RATE [bit/s]
       -------------------------------------------------
         1     0              HIGH    400,000
         2     0                LOW     500,000
         3     0                LOW     0
         4     0              LOW     0
```

where:

Index	Rate index
Currently used	number of rings which are currently running, configured with the index's rate
Burst	Burst level configured for the index's rate
Rate	Rate configured for the relevant index

All rates are shown in bits per second.

3.1.2.3 Using Packet Pacing Sockets

1. Create a rate-limited socket according to the desired rate using the setsockopt() interface **based on the previous section:**

```
setsockopt(s, SOL_SOCKET, SO_MAX_PACING_RATE, pacing_rate, sizeof(pacing_rate))
```

SO_MAX_PACING_RATE	Marks the socket as a rate limited socket
pacing_rate	Defined rate in bytes/sec. The type is unsigned int. **Note:** The same value entered via sysctl in bytes instead of bits.

- A rate-limited ring corresponding to the requested rate will be created and associated to the relevant socket.
- Rate-limited traffic will be transmitted when data is sent via the socket.

2. Modify the rate-limited value using the same socket.
3. Destroy the relevant ring upon TCP socket completion.

3.1.2.3.1Error Detection

Detecting failures can be done using the getsockopt() interface to query a specific socket.

```
getsockopt(s, SOL_SOCKET, SO_MAX_PACING_RATE, pacing_rate, &size_socket);
```

pacing_rate	Holds the rate limit value associated with the socket. If the value of pacing_rate=0, it means that there was an error and the rate limit could not be set on the socket, and the socket is not rate-limited. For a more detailed message, look at the output of dmesg.

To see if the socket is rate-limited, data must be sent using the socket. Only then will the getsockopt returned value be valid.

3.1.2.4 Feature Characteristics

- MLNX_OFED for FreeBSD supports up to 45,000 rate limited TCP connections.
- Each TCP connection is mapped to a specific QP
- **User interface**
  - Rate limited tx ring amount:

```
#> sysctl -a | grep rate_limit_tx_ring
hw.mlxen0.conf.rate_limit_tx_rings: 0
hw.mlxen1.conf.rate_limit_tx_rings: 0
```

  - Native tx ring amount:

```
#> sysctl -a | grep native_tx_ring
hw.mlxen0.conf.native_tx_rings: 8
hw.mlxen1.conf.native_tx_rings: 8
```

- **Debugging**
  - Setting the kernel environment variable before loading the modules:

```
#> kenv rate.limit.debug=1
```

    All rate limited rings statistics will be available via Sysctl (as debug option is enabled):
    - Rate limit value
    - Packets
    - Bytes

```
#> sysctl -a | grep tx_ring
hw.mlxen0.stat.tx_ring8.rate_limit_val: 409600
```

```
hw.mlxen0.stat.tx_ring8.packets: 2284
hw.mlxen0.stat.tx_ring8.bytes: 3177592
```

- Running sysctl in order to get more rate limit information (without enabling the debug option):
  - Show ring ID, qp number and rate, of currently running rings, in a csv format, in order to dump to excel:

    ```
    sysctl hw.mlxen0.conf.dump_rate_limit_rings=1
    ```

  - Show ring ID, qp number and rate, of currently running rings, in a table format:

    ```
    sysctl hw.mlxen0.conf.dump_rate_limit_rings=2
    ```

3.1.2.5 Limitations

- Max rate limited rings is 45,000
- Min rate: 250 Kbps
- Max rate: 50 Mbps

```
#> sysctl -a | grep rate_limit_caps
sys.device.mlx4_core0.rate_limit_caps.min_value: 250 Kbps
sys.device.mlx4_core0.rate_limit_caps.max_value: 50 Mbps
```

- Different rate limits to be configured per NIC/port: 120 divided by the number of priorities (See Section 3.1.2.1)

3.1.3 EEPROM Cable Module Information Reader

In order to read the cable EEPROM info:

- Read the cable information by enabling the following sysctl parameter:

```
sysctl hw.mlxen<X>.conf.eeprom_info=1
```

Example

```
hw.mlxen0.conf.eeprom_info: 0 -> 0
Offset          Values
------          ------
0x0000          0d 02 06 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0010          00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0020          00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0030          00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0040          00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0050          00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0060          00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0070          00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0080          0d 00 23 08 00 00 00 00 00 00 00 05 8d 00 00 00
0x0090          00 00 04 a0 4d 65 6c 6c 61 6e 6f 78 20 20 20 20
0x00a0          20 20 20 20 0f 00 02 c9 4d 43 32 32 30 37 31 32
0x00b0          36 2d 30 30 34 20 20 20 41 33 06 0a 0d 00 46 74
0x00c0          00 00 00 00 4d 54 31 31 33 39 56 53 30 30 34 38
0x00d0          32 20 20 20 31 31 30 39 32 34 20 20 00 00 00 e7
0x00e0          00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00f0          00 00 00 00 00 00 00 00 00 00 02 00 00 00 00 00
```

4 Performance Tuning

4.1 Interrupt Moderation

Interrupt moderation is used to decrease the frequency of network adapter interruptions to the CPU. Mellanox network adapters use an adaptive interrupt moderation algorithm by default. The algorithm checks the transmission (Tx), receives (Rx) packet rates and modifies the Rx interrupt moderation settings accordingly.

In order to manually set Rx interrupt moderation, use sysctl:

Step 1. Turn OFF the interrupt moderation. Run:

```
#> sysctl hw.mlxen<x>.conf.adaptive_rx_coal=0
```

Note: <x> is the OS assigned interface number

Step 2. Turn ON the interruption moderation. Run:

```
#> sysctl hw.mlxen<xa>.conf.adaptive_rx_coal=1
```

Step 3. Set the threshold values for packet rate limits and for moderation time. Run:

```
#> sysctl hw.mlxen<xa>.conf.pkt_rate_low=[N]
#> sysctl hw.mlxen<xa>.conf.rx_usecs_low=[N]
#> sysctl hw.mlxen<xa>.conf.pkt_rate_high=[N]
#> sysctl hw.mlxen<x>.conf.rx_usecs_high=[N]
```

Above an upper limit of packet rate, adaptive moderation will set the moderation time to its highest value. Below a lower limit of packet rate, the moderation time will be set to its lowest value.

4.2 Tuning for NUMA Architecture

4.2.1 Single NUMA Architecture

When using a server with single NUMA, no tuning is required. Also, make sure to avoid using core number 0 for interrupts and applications.

1. Find a CPU list:

```
sysctl -a | grep "group level=\"2\"" -A 1
<group level="2" cache-level="2">
<cpu count="12" mask="fff">0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11</cpu>
```

2. Tune Mellanox NICs to work on desirable cores

   a.Find the NIC's PCI location:

```
pciconf -lv | grep mlx
mlx4_core0@pci0:2:0:0:  class=0x028000 card=0x000315b3 chip=0x100715b3 rev=0x00 hdr=0x00
```

   b. Find the NIC's device name by its PCI location

```
sysctl -a | grep pci2
dev.mlx4_core.0.%parent: pci2
```

This means the NIC on PCI number 2 has a logic device called mlx4_core0.

a.Find the device interrupts.

```
vmstat -ia | grep mlx4_core0 | awk '{print $1}' | sed s/irq// | sed s/:://
285
286
287
…
```

b. Bind each interrupt to a desirable core.

```
cpuset -x 285 -l 1
cpuset -x 286 -l 2
cpuset -x 287 -l 3
…
```

c. Bind the application to the desirable core.

```
cpuset -l 1-11 <app name> <sever flag>
cpuset -l 1-11 <app name> <client flag> <IP>
```

Specifying a range of CPUs when using the cpuset command will allow the application to choose any of them. This is important for applications that execute on multiple threads.
The range argument is not supported for interrupt binding.

4.2.2 Dual NUMA Architecture

1. Find the CPU list closest to the NIC

a.Find the NIC's PCI location:

```
pciconf -lv | grep mlx
mlx4_core0@pci0:4:0:0:  class=0x028000 card=0x000315b3 chip=0x100715b3 rev=0x00 hdr=0x00
```

Usually, low PCI locations are closest to NUMA number 0, and high PCI locations are closest to NUMA number 1. Here is how to verify the locations:

b. Find the NIC's pcib by PCI location (in this example, try PCI 4)

```
sysctl -a | grep pci.4.%paren
dev.pci.4.%parent: pcib3
```

c. Find the NIC's pcib location:

```
sysctl -a | grep pcib.3.%location
dev.pcib.3.%location: slot=2 function=0 handle=\_SB_.PCI0.NPE3
```

In "handle", PCI0 is the value for locations near NUMA0, and PCI1 is the value for locations near NUMA1.

d. Find the cores list of the closest NUMA:

```
sysctl -a | grep "group level=\"2\"" -A 1
<group level="2" cache-level="2">
<cpu count="12" mask="fff">0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11</cpu>
--
<group level="2" cache-level="2">
<cpu count="12" mask="fff000">12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23</cpu>
```

Note: Each list of cores refers to a different NUMA.

2. Tune Mellanox NICs to work on desirable cores.

   a.Pin both interrupts and application processes to the relevant cores.

   b. Find the closest NUMA to the NIC

   c. Find the NIC's device name by its PCI location.

```
sysctl -a | grep pci4
dev.mlx4_core.0.%parent: pci4
```

   This means the NIC on PCI number 4 has a logic device called mlx4_core0.

   d. Find the device interrupts.

```
vmstat -ia | grep mlx4_core0 | awk '{print $1}' | sed s/irq// | sed s/://
338
339
340
…
```

   e. Bind each interrupt to a core from the closest NUMA cores list
   Note: It is best to avoid core number 0.

```
cpuset -x 338 -l 1
cpuset -x 339 -l 2
cpuset -x 340 -l 3
...
```

   f. Bind the application to the closest NUMA cores list.
   Note: It is best to avoid core number 0

```
cpuset -l 1-11 <app name> <sever flag>
cpuset -l 1-11 <app name> <client flag> <IP>
```

For best performance, change CPU’s BIOS configuration to performance mode.