Mellanox OFED for Linux User Manual

Rev 2.1-1.0.0

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 PRODUCTO(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

Mellanox Technologies, Ltd.
Beit Mellanox
PO Box 586 Yokneam 20692
Israel
www.mellanox.com
Tel: +972 (0)74 723 7200
Fax: +972 (0)4 959 3245

Table of Contents

List of Figures

List of Tables

Document Revision History

Table 1 - Document Revision History

Release	Date	Description
2.1-1.0.0	December 2013	• Added the following sections: • Section 2.3.6, "Installation Logging", on page 41 • Section 4.6.2, "RoCE Time Stamping", on page 77 and its subsections • Section 4.17, "PeerDirect", on page 105 • Section 4.18, "Inline-Receive", on page 106 • Section 4.19, "Ethernet Performance Counters", on page 107 • Section 4.20, "Memory Window", on page 111 • Section 4.1.2.1.1, "SRP Module Parameters", on page 42 • Section 4.1.2.1.2, "SRP Remote Ports Parameters", on page 42 • Section 4.1.2.2.1, "SRP sysfs Parameters", on page 43 • Section , "srpd", on page 46 • Section 4.6.1.3, "Querying Time Stamping Capabilities via ethtool", on page 77 • Updated the following sections: • Section 1.5, "RDMA over Converged Ethernet (RoCE)", on page 25 • Section 2.3.3, "Installation Procedure", on page 32 • Section 4.13.2, "Setting Up SR-IOV", on page 90 • Section 5.3.1, "Compiling OpenMPI with MXM", on page 118 • Section 5.3.2, "Enabling MXM in OpenMPI", on page 119 • Section 5.3.4, "Configuring Multi-Rail Support", on page 120 • Section 4.8.4, "Setting Performance Tuning", on page 84 • Section 8.4.1, "File Format", on page 149 • Appendix C.2, "mlx4_core Parameters" page 237 • Section 4.1.2.2, "Manually Establishing an SRP Connection", on page 43 • Section 4.1.2.3, "SRP Tools - ibsrpdm, srp_daemon and srpd Service Script", on page 45 • Section 4.1.2.4, "Automatic Discovery and Connection to Targets", on page 47 • Section 4.1.2.5, "Multiple Connections from Initiator InfiniBand Port to the Target", on page 48 • Section 4.1.2.6, "High Availability (HA)", on page 48 • Section 4.1.2.7, "Shutting Down SRP", on page 49 • Section 4.15, "Ethtool", on page 103

Table 1 - Document Revision History

Release	Date	Description
2.0-3.0.0	October 2013	Removed section "Command Line Interface (CLI)" Updated the following sections: • Appendix E, "Lustre Compilation over MLNX_OFED" page 240

	August 2013	• Updated the following sections:
		  • Section 1.3.4, "ULPs", on page 23
		  • Section 4.12, "Flow Steering", on page 87 and its subsections
		  • Section 1.3.3, "Mid-layer Core", on page 23
		  • Section 4.8, "Ethernet Tunneling Over IPoIB Driver (eIPoIB)", on page 80
		  • Section 8.2.1, "opensm Syntax", on page 137
		  • Appendix C, "mlx4 Module Parameters" page 237
		• Added the following sections:
		  • Section 1.5, "RDMA over Converged Ethernet (RoCE)", on page 25
		  • Section 4.5, "Quality of Service Ethernet", on page 66 and its subsections
		  • Section 4.11, "XRC - eXtended Reliable Connected Transport Service for InfiniBand", on page 86
		  • Section 4.13.7, "Configuring Pkeys and GUIDs under SR-IOV", on page 97 and its subsections
		  • Section 4.15, "Ethtool", on page 103
		  • Appendix E, "Lustre Compilation over MLNX_OFED" page 240

2.0-2.0.5	April 2013	Initial release

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 (Sheet 1 of 2)

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
iSER	iSCSI RDMA Protocol
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
IPoIB	IP over InfiniBand
PFC	Priority Flow Control
PR	Path Record
RDS	Reliable Datagram Sockets
RoCE	RDMA over Converged Ethernet

Table 2 - Abbreviations and Acronyms (Sheet 2 of 2)

Abbreviation / Acronym	Whole Word / Description
SDP	Sockets Direct Protocol
SL	Service Level
SRP	SCSI RDMA Protocol
MPI	Message Passing Interface
EoIB	Ethernet over Infiniband
QoS	Quality of Service
ULP	Upper Level Protocol
VL	Virtual Lane
vHBA	Virtual SCSI Host Bus adapter
uDAPL	User Direct Access Programming Library

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 (Sheet 1 of 2)

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.
IB Cluster/Fabric/ Subnet	A set of IB devices connected by IB cables.
In-Band	A term assigned to administration activities traversing the IB connectivity only.
Local Identifier (ID)	An address assigned to a port (data sink or source point) by the Subnet Manager, unique within the subnet, used for directing packets within the subnet.
Local Device/Node/ System	The IB Host Channel Adapter (HCA) Card installed on the machine running IBDIAG tools.

Table 3 - Glossary (Sheet 2 of 2)

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.
Standby Subnet Manager	A Subnet Manager that is currently quiescent, and not in the role of a Master Subnet Manager, by agency of the master SM. See Subnet Manager.
Subnet Administrator (SA)	An application (normally part of the Subnet Manager) that implements the interface for querying and manipulating subnet management data.
Subnet Manager (SM)	One of several entities involved in the configuration and control of the an IB fabric.
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
IEEE Std 802.3ae™-2002 (Amendment to IEEE Std 802.3-2002) Document # PDF: SS94996	Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications Amendment: Media Access Control (MAC) Parameters, Physical Layers, and Management Parameters for 10 Gb/s Operation

Table 4 - Reference Documents

Document Name	Description
Firmware Release Notes for Mellanox adapter devices	See the Release Notes PDF file relevant to your adapter device under `docs/` folder of installed package.
MFT User's Manual	Mellanox Firmware Tools User's Manual. See under `docs/` folder of installed package.
MFT Release Notes	Release Notes for the Mellanox Firmware Tools. See under `docs/` folder of installed package.

Support and Updates Webpage

Please visit http://www.mellanox.com > Products > InfiniBand/VPI Drivers > Linux SW/Drivers for downloads, FAQ, troubleshooting, future updates to this manual, etc.

1 Mellanox OFED Overview

1.1 Introduction to Mellanox OFED

Mellanox OFED is a single Virtual Protocol Internconnect (VPI) software stack which operates across all Mellanox network adapter solutions supporting 10, 20, 40 and 56 Gb/s InfiniBand (IB); 10, 40 and 56 Gb/s Ethernet; and 2.5 or 5.0 GT/s PCI Express 2.0 and 8 GT/s PCI Express 3.0 uplinks to servers.

All Mellanox network adapter cards are compatible with OpenFabrics-based RDMA protocols and software, and are supported with major operating system distributions.

Mellanox OFED is certified with the following products:

- Mellanox Messaging Accelerator (VMA™) software: Socket acceleration library that performs OS bypass for standard socket based applications.
- Mellanox Unified Fabric Manager (UFM®) software: Powerful platform for managing demanding scale-out computing fabric environments, built on top of the OpenSM industry standard routing engine.
- Fabric Collective Accelerator (FCA) - FCA is a Mellanox MPI-integrated software package that utilizes CORE-Direct technology for implementing the MPI collectives communications.

1.2 Mellanox OFED Package

1.2.1 ISO Image

Mellanox OFED for Linux (MLNX_OFED_LINUX) is provided as ISO images or as a tarball, one per supported Linux distribution and CPU architecture, that includes *source code* and *binary* RPMs, firmware, utilities, and documentation. The ISO image contains an installation script (called `mlnxofedinstall`) that performs the necessary steps to accomplish the following:

- Discover the currently installed kernel
- Uninstall any InfiniBand stacks that are part of the standard operating system distribution or another vendor's commercial stack
- Install the MLNX_OFED_LINUX binary RPMs (if they are available for the current kernel)
- Identify the currently installed InfiniBand HCAs and perform the required firmware updates

1.2.2 Software Components

MLNX_OFED_LINUX contains the following software components:

- Mellanox Host Channel Adapter Drivers
  - mlx5, mlx4 (VPI), which is split into multiple modules:
    - mlx4_core (low-level helper)
    - mlx4_ib (IB)
    - mlx5_ib
    - mlx5_core

    - mlx4_en (Ethernet)
- Mid-layer core
    - Verbs, MADs, SA, CM, CMA, uVerbs, uMADs
- Upper Layer Protocols (ULPs)
    - IPoIB, RDS*, SRP Initiator and SRP

      * **NOTE:** RDS was not tested by Mellanox Technologies.
- MPI
    - Open MPI stack supporting the InfiniBand, RoCE and Ethernet interfaces
    - OSU MVAPICH stack supporting the InfiniBand and RoCE interfaces
    - MPI benchmark tests (OSU BW/LAT, Intel MPI Benchmark, Presta)
- OpenSM: InfiniBand Subnet Manager
- Utilities
    - Diagnostic tools
    - Performance tests
- Firmware tools (MFT)
- Source code for all the OFED software modules (for use under the conditions mentioned in the modules' LICENSE files)
- Documentation

1.2.3 Firmware

The ISO image includes the following firmware items:

- Firmware images (.mlx format) for ConnectX®-3/ConnectX®-3 Pro/Connect-IB™ network adapters
- Firmware configuration (.INI) files for Mellanox standard network adapter cards and custom cards
- FlexBoot for ConnectX®-3 HCA devices

1.2.4 Directory Structure

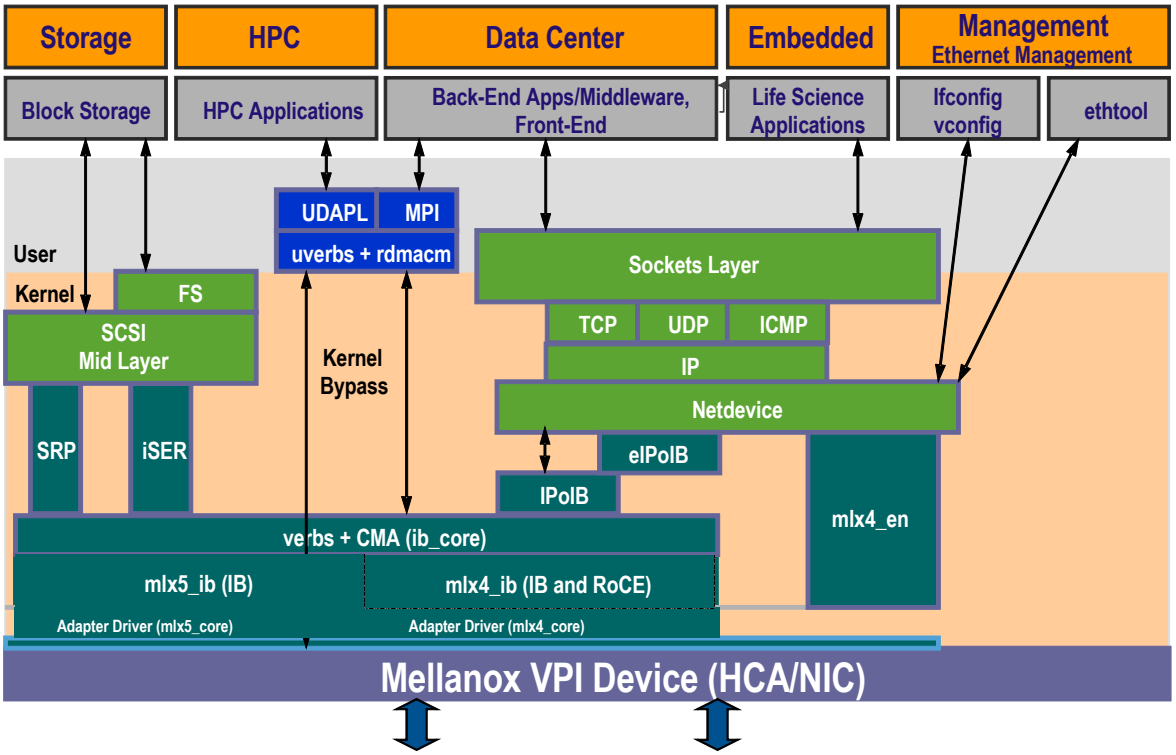
The ISO image of MLNX_OFED_LINUX contains the following files and directories:

- mlnxofedinstall - This is the MLNX_OFED_LINUX installation script.
- ofed_uninstall.sh - This is the MLNX_OFED_LINUX un-installation script.
- <RPMS folders> - Directory of binary RPMs for a specific CPU architecture.
- firmware/ - Directory of the Mellanox IB HCA firmware images (including Boot-over-IB)
- src/ - Directory of the OFED source tarball
- mlnx_add_kernel_support.sh - Script required to rebuild MLNX_OFED_LINUX for customized kernel version on supported Linux Distribution
- docs/ - Directory of Mellanox OFED related documentation

1.3 Architecture

Figure 1 shows a diagram of the Mellanox OFED stack, and how upper layer protocols (ULPs) interface with the hardware and with the kernel and user space. The application level also shows the versatility of markets that Mellanox OFED applies to.

Figure 1: Mellanox OFED Stack for ConnectX® Family Adapter Cards



The following sub-sections briefly describe the various components of the Mellanox OFED stack.

1.3.1 mlx4 VPI Driver

`mlx4` is the low level driver implementation for the ConnectX family adapters designed by Mellanox Technologies. ConnectX® family adapters can operate as an InfiniBand adapter, or as an Ethernet NIC. The OFED driver supports InfiniBand and Ethernet NIC configurations. To accommodate the supported configurations, the driver is split into the following modules:

mlx4_core

Handles low-level functions like device initialization and firmware commands processing. Also controls resource allocation so that the InfiniBand and Ethernet functions can share the device without interfering with each other.

mlx4_ib

Handles InfiniBand-specific functions and plugs into the InfiniBand midlayer

mlx4_en

A 10/40GigE driver under drivers/net/ethernet/mellanox/mlx4 that handles Ethernet specific functions and plugs into the netdev mid-layer

1.3.2 mlx5 Driver

`mlx5` is the low level driver implementation for the Connect-IB™ adapters designed by Mellanox Technologies. Connect-IB™ operates as an InfiniBand adapter. The mlx5 driver is comprised of the following kernel modules:

mlx5_core

Acts as a library of common functions (e.g. initializing the device after reset) required by the Connect-IB™ adapter card.

mlx5_ib

Handles InfiniBand-specific functions and plugs into the InfiniBand midlayer.

libmlx5

libmlx5 is the provider library that implements hardware specific user-space functionality. If there is no compatibility between the firmware and the driver, the driver will not load and a message will be printed in the dmesg.

The following are the Libmlx5 environment variables:

- MLX5_FREEZE_ON_ERROR_CQE
  - Causes the process to hang in a loop when completion with error which is not flushed with error or retry exceeded occurs/
  - Otherwise disabled
- MLX5_POST_SEND_PREFER_BF
  - Configures every work request that can use blue flame will use blue flame
  - Otherwise - blue flame depends on the size of the message and inline indication in the packet
- MLX5_SHUT_UP_BF
  - Disables blue flame feature
  - Otherwise - do not disable
- MLX5_SINGLE_THREADED
  - All spinlocks are disabled
  - Otherwise - spinlocks enabled
  - Used by applications that are single threaded and would like to save the overhead of taking spinlocks.
- MLX5_CQE_SIZE
  - 64 - completion queue entry size is 64 bytes (default)
  - 128 - completion queue entry size is 128 bytes

MLX5_SCATTER_TO_CQE

- Small buffers are scattered to the completion queue entry and manipulated by the driver. Valid for RC transport.
- Default is 1, otherwise disabled.

1.3.3 Mid-layer Core

Core services include: management interface (MAD), connection manager (CM) interface, and Subnet Administrator (SA) interface. The stack includes components for both user-mode and kernel applications. The core services run in the kernel and expose an interface to user-mode for verbs, CM and management.

1.3.4 ULPs

IPoIB

The IP over IB (IPoIB) driver is a network interface implementation over InfiniBand. IPoIB encapsulates IP datagrams over an InfiniBand connected or datagram transport service. IPoIB pre-appends the IP datagrams with an encapsulation header, and sends the outcome over the InfiniBand transport service. The transport service is Unreliable Datagram (UD) by default, but it may also be configured to be Reliable Connected (RC). The interface supports unicast, multicast and broadcast. For details, see Chapter 4.3, “IP over InfiniBand”.

iSER

iSCSI Extensions for RDMA (iSER) extends the iSCSI protocol to RDMA. It permits data to be transferred directly into and out of SCSI buffers without intermediate data copies. For further information, please refer to Chapter 4.2, “iSCSI Extensions for RDMA (iSER)”.

SRP

SCSI RDMA Protocol (SRP) is designed to take full advantage of the protocol offload and RDMA features provided by the InfiniBand architecture. SRP allows a large body of SCSI software to be readily used on InfiniBand architecture. The SRP driver—known as the SRP Initiator—differs from traditional low-level SCSI drivers in Linux. The SRP Initiator does not control a local HBA; instead, it controls a connection to an I/O controller—known as the SRP Target—to provide access to remote storage devices across an InfiniBand fabric. The SRP Target resides in an I/O unit and provides storage services. See Chapter 4.1, “SCSI RDMA Protocol” and Appendix B, “SRP Target Driver”.

uDAPL

User Direct Access Programming Library (uDAPL) is a standard API that promotes data center application data messaging performance, scalability, and reliability over RDMA interconnects: InfiniBand and RoCE. The uDAPL interface is defined by the DAT collaborative.

This release of the uDAPL reference implementation package for both DAT 1.2 and 2.0 specification is timed to coincide with OFED release of the Open Fabrics (www.openfabrics.org) software stack.

For more information about the DAT collaborative, go to the following site:

http://www.datcollaborative.org

1.3.5 MPI

Message Passing Interface (MPI) is a library specification that enables the development of parallel software libraries to utilize parallel computers, clusters, and heterogeneous networks. Mellanox OFED includes the following MPI implementations over InfiniBand:

- Open MPI – an open source MPI-2 implementation by the Open MPI Project
- OSU MVAPICH – an MPI-1 implementation by Ohio State University

Mellanox OFED also includes MPI benchmark tests such as OSU BW/LAT, Intel MPI Benchmark, and Presta.

1.3.6 InfiniBand Subnet Manager

All InfiniBand-compliant ULPs require a proper operation of a Subnet Manager (SM) running on the InfiniBand fabric, at all times. An SM can run on any node or on an IB switch. OpenSM is an InfiniBand-compliant Subnet Manager, and it is installed as part of Mellanox OFED.[1] See Chapter 8, “OpenSM – Subnet Manager”.

1.3.7 Diagnostic Utilities

Mellanox OFED includes the following two diagnostic packages for use by network and datacenter managers:

- ibutils – Mellanox Technologies diagnostic utilities
- infiniband-diags – OpenFabrics Alliance InfiniBand diagnostic tools

1.3.8 Mellanox Firmware Tools

The Mellanox Firmware Tools (MFT) package is a set of firmware management tools for a single InfiniBand node. MFT can be used for:

- Generating a standard or customized Mellanox firmware image
- Querying for firmware information
- Burning a firmware image to a single InfiniBand node

MFT includes the following tools:

- mlxburn - provides the following functions:
  - Generation of a standard or customized Mellanox firmware image for burning—in .bin (binary) or .img format
  - Burning an image to the Flash/EEPROM attached to a Mellanox HCA or switch device
  - Querying the firmware version loaded on an HCA board
  - Displaying the VPD (Vital Product Data) of an HCA board
- flint

  This tool burns a firmware binary image or an expansion ROM image to the Flash device of a Mellanox network adapter/bridge/switch device. It includes query functions to the burnt firmware image and to the binary image file.
- spark

1. OpenSM is disabled by default. See Chapter 8, “OpenSM – Subnet Manager” for details on enabling it.

This tool burns a firmware binary image to the EEPROM(s) attached to an InfiniScaleIII® switch device. It includes query functions to the burnt firmware image and to the binary image file. The tool accesses the EEPROM and/or switch device via an I2C-compatible interface or via vendor-specific MADs over the InfiniBand fabric (In-Band tool).

- Debug utilities

  A set of debug utilities (e.g., itrace, mstdump, isw, and i2c)

For additional details, please refer to the MFT User's Manual `docs/`.

1.4 Quality of Service

Quality of Service (QoS) requirements stem from the realization of I/O consolidation over an IB and Eth network. As multiple applications and ULPs share the same fabric, a means is needed to control their use of network resources.

QoS over Mellanox OFED for Linux is discussed in Chapter 8, "OpenSM – Subnet Manager".

1.5 RDMA over Converged Ethernet (RoCE)

RoCE allows InfiniBand (IB) transport applications to work over Ethernet network. RoCE encapsulates the InfiniBand transport and the GRH headers in Ethernet packets bearing a dedicated ether type (0x8195). Thus, any VERB application that works in an InfiniBand fabric can work in an Ethernet fabric as well.

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 but only the actions such as joining multicast group, that need to be taken when using the API
- 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, the alternate path is not set for RC QP and therefore APM is not supported
- 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. Hence, it is recommended working with RDMA-CM to establish a connection as it takes care of filling the path record structure
- 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, alias devices and bonding masters. The only exception to this rule is a bonding master of a slave in a DOWN state. In that case, a matching GID to the IP address of the master will not be present in the GID table of the slave's port
- 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

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

2 Installation

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

2.1 Hardware and Software Requirements

Table 1 - Software and Hardware Requirements

Requirements	Description
Platforms	A server platform with an adapter card based on one of the following Mellanox Technologies' InfiniBand HCA devices: • MT27508 ConnectX®-3 (VPI, IB, EN) (firmware: fw-ConnectX3) • MT4113 Connect-IB™ (IB) (firmware: fw-Connect-IB) For the list of supported architecture platforms, please refer to the Mellanox OFED Release Notes file.
Required Disk Space for Installation	1GB
Device ID	For the latest list of device IDs, please visit Mellanox website.
Operating System	Linux operating system. For the list of supported operating system distributions and kernels, please refer to the Mellanox OFED Release Notes file.
Installer Privileges	The installation requires administrator privileges on the target machine.

2.2 Downloading Mellanox OFED

Step 1. Verify that the system has a Mellanox network adapter (HCA/NIC) installed by ensuring that you can see ConnectX or InfiniHost entries in the display.

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

```
# lspci -v | grep Mellanox
06:00.0 Network controller: Mellanox Technologies MT27500 Family [ConnectX-3]
Subsystem: Mellanox Technologies Device 0024
```

Step 2. Download the ISO image to your host.

The image's name has the format `MLNX_OFED_LINUX-<ver>-<OS label><CPU arch>.iso`. You can download it from http://www.mellanox.com > Products > Software> InfiniBand Drivers.

Step 3. Use the md5sum utility to confirm the file integrity of your ISO image. Run the following command and compare the result to the value provided on the download page.

```
host1$ md5sum MLNX_OFED_LINUX-<ver>-<OS label>.iso
```

2.3 Installing Mellanox OFED

The installation script, `mlnxofedinstall`, performs the following:

- Discovers the currently installed kernel
- Uninstalls any software stacks that are part of the standard operating system distribution or another vendor's commercial stack
- Installs the MLNX_OFED_LINUX binary RPMs (if they are available for the current kernel)
- Identifies the currently installed InfiniBand and Ethernet network adapters and automatically[1] upgrades the firmware

2.3.1 Pre-installation Notes

- The installation script removes all previously installed Mellanox OFED packages and re-installs from scratch. You will be prompted to acknowledge the deletion of the old packages.

Pre-existing configuration files will be saved with the extension ".conf.rpmsave".

- If you need to install Mellanox OFED on an entire (homogeneous) cluster, a common strategy is to mount the ISO image on one of the cluster nodes and then copy it to a shared file system such as NFS. To install on all the cluster nodes, use cluster-aware tools (such as pdsh).
- If your kernel version does not match with any of the offered pre-built RPMs, you can add your kernel version by using the "`mlnx_add_kernel_support.sh`" script located under the docs/ directory.

On Redhat and SLES distributions with errata kernel installed there is no need to use the mlnx_add_kernel_support.sh script. The regular installation can be performed and weak-updates mechanism will create symbolic links to the MLNX_OFED kernel modules.

Usage:

```
mlnx_add_kernel_support.sh -m|--mlnx_ofed <path to MLNX_OFED directory> [--make-iso|--
make-tgz]

  [--make-iso]                                  Create MLNX_OFED ISO image.
  [--make-tgz]                                  Create MLNX_OFED tarball. (Default)
  [-t|--tmpdir <local work dir>]
  [--kmp]                                       Enable KMP format if supported.
  [-k | --kernel] <kernel version>              Kernel version to use.
  [-s | --kernel-sources] <path to the kernel sources>    Path to kernel headers.
  [-v|--verbose]
  [-n|--name]                                    Name of the package to be created.
  [-y|--yes]                                     Answer "yes" to all questions
```

1. The firmware will not be updated if you run the install script with the '--without-fw-update' option.

Example

The following command will create a MLNX_OFED_LINUX ISO image for RedHat 6.3 under the /tmp directory.

```
# ./MLNX_OFED_LINUX-2.1-1.0.0-rhel6.3-x86_64/mlnx_add_kernel_support.sh -m /tmp/
MLNX_OFED_LINUX-2.1-1.0.0-rhel6.3-x86_64/ --make-tgz
Note: This program will create MLNX_OFED_LINUX TGZ for rhel6.2 under /tmp directory.
All Mellanox, OEM, OFED, or Distribution IB packages will be removed.
Do you want to continue?[y/N]:y
See log file /tmp/mlnx_ofed_iso.21642.log

Building OFED RPMs. Please wait...
Removing OFED RPMs...
Created /tmp/MLNX_OFED_LINUX-2.1-1.0.0-rhel6.2-x86_64.tgz
```

The "`mlnx_add_kernel_support.sh`" script can be executed directly from the mlnxofedinstall script. For further information, please see '--add-kernel-support' option below.

2.3.2 Installation Script

Mellanox OFED includes an installation script called `mlnxofedinstall`. Its usage is described below. You will use it during the installation procedure described in Section 2.3.3, "Installation Procedure", on page 32.

Usage

```
./mnt/mlnxofedinstall [OPTIONS]
```

Options

-c|--config <packages config_file>	Example of the configuration file can be found under docs
-n|--net <network config_file>	Example of the network configuration file can be found under docs
-k|--kernel-version <kernel version>	Use provided kernel version instead of 'uname -r'
-p|--print-available	Print available packages for current platform and create corresponding ofed.conf file
--without-32bit	Skip 32-bit libraries installation
--without-depcheck	Skip Distro's libraries check
--without-fw-update	Skip firmware update
--fw-update-only	Update firmware. Skip driver installation
--force-fw-update	Force firmware update
--force	Force installation
--all|--hpc|--basic|--msm	Install all, hpc, basic or Mellanox Subnet manager packages correspondingly
--vma|--vma-vpi	Install packages required by VMA to support VPI
--vma-eth	Install packages required by VMA to work over Ethernet
--with-vma	Set configuration for VMA use (to be used with any installation parameter).
--guest	Install packages required by guest os
--hypervisor	Install packages required by hypervisor os
-v|-vv|-vvv	Set verbosity level
--umad-dev-rw	Grant non root users read/write permission for umad devices instead of default
--enable-affinity	Run mlnx_affinity script upon boot
--disable-affinity	Disable mlnx_affinity script (Default)
--enable-sriov	Burn SR-IOV enabled firmware
--add-kernel-support	Add kernel support (Run mlnx_add_kernel_support.sh)
--skip-distro-check	Do not check MLNX_OFED vs Distro matching

```
--hugepages-overcommit              Setting 80% of MAX_MEMORY as overcommit for huge
                                    page allocation

-q                                  Set quiet - no messages will be printed

--with-fabric-collector             Install fabric-collector package.
```

2.3.2.1 mlnxofedinstall Return Codes

Table 2 lists the `mlnxofedinstall` script return codes and their meanings.

Table 2 - mlnxofedinstall Return Codes

Return Code	Meaning
0	The Installation ended successfully
1	The installation failed
2	No firmware was found for the adapter device
22	Invalid parameter
28	Not enough free space
171	Not applicable to this system configuration. This can occur when the required hardware is not present on the system.
172	Prerequisites are not met. For example, missing the required software installed or the hardware is not configured correctly.
173	Failed to start the `mst` driver

2.3.3 Installation Procedure

Step 1. Login to the installation machine as root.

Step 2. Mount the ISO image on your machine

```
host1# mount -o ro,loop MLNX_OFED_LINUX-<ver>-<OS label>-<CPU arch>.iso /mnt
```

Step 3. Run the installation script.

```
./mlnxofedinstall
Logs dir: /tmp/MLNX_OFED_LINUX-2.1-0.0.9.10740.logs
This program will install the MLNX_OFED_LINUX package on your machine.
Note that all other Mellanox, OEM, OFED, or Distribution IB packages will be removed.
Do you want to continue?[y/N]:y

Uninstalling the previous version of MLNX_OFED_LINUX

/bin/rpm --nosignature -e --allmatches --nodeps libmverbs libmverbs.i686 libmverbs-
devel libmverbs-devel.i686 libmqe libmqe.i686 libmqe-devel libmqe-devel.i686

Starting MLNX_OFED_LINUX-2.1-0.0.9 installation ...

Installing mlnx-ofa_kernel RPM
Preparing...                ##################################################
mlnx-ofa_kernel             ##################################################
Installing kmod-mlnx-ofa_kernel 2.1 RPM
Preparing...                ##################################################
kmod-mlnx-ofa_kernel        ##################################################
Installing mlnx-ofa_kernel-devel RPM
Preparing...                ##################################################
mlnx-ofa_kernel-devel       ##################################################
Installing kmod-kernel-mft-mlnx any RPM
Preparing...                ##################################################
kmod-kernel-mft-mlnx        ##################################################
Installing knem-mlnx RPM
Preparing...                ##################################################
knem-mlnx                   ##################################################
Installing kmod-knem-mlnx 1.1.90mlnx2 RPM
Preparing...                ##################################################
kmod-knem-mlnx              ##################################################
Installing ummunotify-mlnx RPM
Preparing...                ##################################################
ummunotify-mlnx             ##################################################
Installing kmod-ummunotify-mlnx 1.0 RPM
Preparing...                ##################################################
kmod-ummunotify-mlnx        ##################################################
Installing mpi-selector RPM
Preparing...                ##################################################
mpi-selector                ##################################################
```

```
Installing user level RPMs:
Preparing...                ##################################################
ofed-scripts                ##################################################
Preparing...                ##################################################
libibverbs                  ##################################################
Preparing...                ##################################################
libibverbs                  ##################################################
Preparing...                ##################################################
libibverbs-devel            ##################################################
Preparing...                ##################################################
libibverbs-devel            ##################################################
Preparing...                ##################################################
libibverbs-devel-static     ##################################################
Preparing...                ##################################################
libibverbs-devel-static     ##################################################
Preparing...                ##################################################
libibverbs-utils            ##################################################
Preparing...                ##################################################
libmlx4                     ##################################################
Preparing...                ##################################################
libmlx4                     ##################################################
Preparing...                ##################################################
libmlx4-devel               ##################################################
Preparing...                ##################################################
libmlx4-devel               ##################################################
Preparing...                ##################################################
libmlx5                     ##################################################
Preparing...                ##################################################
libmlx5                     ##################################################
Preparing...                ##################################################
libmlx5-devel               ##################################################
Preparing...                ##################################################
libmlx5-devel               ##################################################
Preparing...                ##################################################
libcxgb3                    ##################################################
Preparing...                ##################################################
libcxgb3                    ##################################################
Preparing...                ##################################################
libcxgb3-devel              ##################################################
Preparing...                ##################################################
libcxgb3-devel              ##################################################
Preparing...                ##################################################
libcxgb4                    ##################################################
```

```
Preparing...                ##################################################
libcxgb4-devel              ##################################################
Preparing...                ##################################################
libcxgb4-devel              ##################################################
Preparing...                ##################################################
libnes                      ##################################################
Preparing...                ##################################################
libnes                      ##################################################
Preparing...                ##################################################
libnes-devel-static         ##################################################
Preparing...                ##################################################
libnes-devel-static         ##################################################
Preparing...                ##################################################
libipathverbs               ##################################################
Preparing...                ##################################################
libipathverbs               ##################################################
Preparing...                ##################################################
libipathverbs-devel         ##################################################
Preparing...                ##################################################
libipathverbs-devel         ##################################################
Preparing...                ##################################################
libibcm                     ##################################################
Preparing...                ##################################################
libibcm                     ##################################################
Preparing...                ##################################################
libibcm-devel               ##################################################
Preparing...                ##################################################
libibcm-devel               ##################################################
Preparing...                ##################################################
libibumad                   ##################################################
Preparing...                ##################################################
libibumad                   ##################################################
Preparing...                ##################################################
libibumad-devel             ##################################################
Preparing...                ##################################################
libibumad-devel             ##################################################
Preparing...                ##################################################
libibumad-static            ##################################################
Preparing...                ##################################################
libibumad-static            ##################################################
Preparing...                ##################################################
libibmad                    ##################################################
Preparing...                ##################################################
libibmad                    ##################################################
```

```
Preparing...                ##################################################
libibmad-devel              ##################################################
Preparing...                ##################################################
libibmad-devel              ##################################################
Preparing...                ##################################################
libibmad-static             ##################################################
Preparing...                ##################################################
libibmad-static             ##################################################
Preparing...                ##################################################
ibsim                       ##################################################
Preparing...                ##################################################
ibacm                       ##################################################
Preparing...                ##################################################
librdmacm                   ##################################################
Preparing...                ##################################################
librdmacm                   ##################################################
Preparing...                ##################################################
librdmacm-utils             ##################################################
Preparing...                ##################################################
librdmacm-devel             ##################################################
Preparing...                ##################################################
librdmacm-devel             ##################################################
Preparing...                ##################################################
opensm-libs                 ##################################################
Preparing...                ##################################################
opensm-libs                 ##################################################
Preparing...                ##################################################
opensm                      ##################################################
Preparing...                ##################################################
opensm-devel                ##################################################
Preparing...                ##################################################
opensm-devel                ##################################################
Preparing...                ##################################################
opensm-static               ##################################################
Preparing...                ##################################################
opensm-static               ##################################################
Preparing...                ##################################################
infiniband-diags            ##################################################
Preparing...                ##################################################
fca                         ##################################################
```

```
IMPORTANT NOTE:
===============

- The FCA Manager and FCA MPI Runtime library are installed in /opt/mellanox/fca
directory.
- The FCA Manager will not be started automatically.
- To start FCA Manager now, type:
    /etc/init.d/fca_managerd start

- There should be single process of FCA Manager running per fabric.

- To start FCA Manager automatically after boot, type:
    /etc/init.d/fca_managerd install_service

- Check /opt/mellanox/fca/share/doc/fca/README.txt for quick start instructions.

Preparing...                ##################################################
dapl                        ##################################################
Preparing...                ##################################################
dapl                        ##################################################
Preparing...                ##################################################
dapl-devel                  ##################################################
Preparing...                ##################################################
dapl-devel                  ##################################################
Preparing...                ##################################################
dapl-devel-static           ##################################################
Preparing...                ##################################################
dapl-devel-static           ##################################################
Preparing...                ##################################################
dapl-utils                  ##################################################
Preparing...                ##################################################
perftest                    ##################################################
Preparing...                ##################################################
mstflint                    ##################################################
Preparing...                ##################################################
mft                         ##################################################
Preparing...                ##################################################
srptools                    ##################################################
Preparing...                ##################################################
rds-tools                   ##################################################
Preparing...                ##################################################
rds-devel                   ##################################################
Preparing...                ##################################################
ibutils2                    ##################################################
Preparing...                ##################################################
ibutils                     ##################################################
Preparing...                ##################################################
cc_mgr                      ##################################################
Preparing...                ##################################################
dump_pr                     ##################################################
```

```
Preparing...                ##################################################
ar_mgr                      ##################################################
Preparing...                ##################################################
ibdump                      ##################################################
Preparing...                ##################################################
infiniband-diags-compat     ##################################################
Preparing...                ##################################################
qperf                       ##################################################
Preparing...                ##################################################
mxm                         ##################################################
Preparing...                ##################################################
openmpi                     ##################################################
Preparing...                ##################################################
openmpi                     ##################################################
Preparing...                ##################################################
bupc                        ##################################################
Preparing...                ##################################################
infinipath-psm              ##################################################
Preparing...                ##################################################
infinipath-psm-devel        ##################################################
Preparing...                ##################################################
mvapich2                    ##################################################
Preparing...                ##################################################
openshmem                   ##################################################
Preparing...                ##################################################
hcoll                       ##################################################
Preparing...                ##################################################
libibprof                   ##################################################
Preparing...                ##################################################
libvma                      ##################################################
- Changing max locked memory to unlimited (in /etc/security/limits.conf)
  Please log out from the shell and login again in order to update this change
  Read more about this topic in the VMA's User Manual

- VMA README.txt is installed at: /usr/share/doc/libvma-6.5.8-0/README.txt
- Please refer to VMA journal for the latest changes: /usr/share/doc/libvma-6.5.8-0/
journal.txt
Preparing...                ##################################################
mlnxofed-docs               ##################################################
Preparing...                ##################################################
mpitests_mvapich2__1_9      ##################################################
Preparing...                ##################################################
mpitests_openmpi__1_6_5     ##################################################
Preparing...                ##################################################
mpitests_openmpi__1_7_4     ##################################################
```

```
Device (06:00.0):
       06:00.0 Network controller: Mellanox Technologies MT27500 Family [ConnectX-3]
        Link Width: 8x
        PCI Link Speed: 5Gb/s
Installation finished successfully.

Attempting to perform Firmware update...
Querying Mellanox devices firmware ...

Device #1:
----------
  Device:        0000:06:00.0
  Part Number:   MCX354A-FCB_A1
 Description:   ConnectX-3 VPI adapter card; dual-port QSFP; FDR IB (56Gb/s) and
40GigE; PCIe3.0 x8 8GT/s; RoHS R6
  PSID:          MT_1090110019
  Versions:      Current        Available
     FW          2.30.7384      2.30.8000
     PXE         3.4.0146       3.4.0146

  Status:        Update required

---------
Found 1 device(s) requiring firmware update...

Device #1: Updating FW ... Done

A restart is needed for updates to take effect.
Log File: /tmp/MLNX_OFED_LINUX-2.1-0.0.9.10740.logs/fw_update.log
```

In case your machine has the latest firmware, no firmware update will occur and the installation script will print at the end of installation a message similar to the following:

```
Device #1:
----------
  Device:        0000:06:00.0
  Part Number:   MCX354A-FCB_A1
  Description:   ConnectX-3 VPI adapter card; dual-port QSFP; FDR
IB (56Gb/s) and 40GigE; PCIe3.0 x8 8GT/s; RoHS R6
  PSID:          MT_1090110019
  Versions:      Current        Available
     FW          2.30.7706      2.30.8000
     PXE         3.4.0146       3.4.0146
  Status:        Up to date
```

In case your machine has an unsupported network adapter device, no firmware update will occur and the error message below will be printed. Please contact your hardware vendor for help on firmware updates.

Error message:

```
Device #1:
----------
  Device:         0000:05:00.0
  Part Number:
  Description:
  PSID:           MT_0DB0110010
  Versions:       Current         Available
     FW           2.9.1000        N/A
  Status:         No matching image found
```

Step 4. In case the installation script performed firmware updates to your network adapter hardware, it will ask you to reboot your machine.

Step 5. The script adds the following lines to `/etc/security/limits.conf` for the userspace components such as MPI:

```
* soft memlock unlimited
* hard memlock unlimited
```

These settings unlimit the amount of memory that can be pinned by a user space application. If desired, tune the value unlimited to a specific amount of RAM.

Step 6. For your machine to be part of the InfiniBand/VPI fabric, a Subnet Manager must be running on one of the fabric nodes. At this point, Mellanox OFED for Linux has already installed the OpenSM Subnet Manager on your machine. For details on starting OpenSM, see Chapter 8, “OpenSM – Subnet Manager”.

Step 7. (InfiniBand only) Run the `hca_self_test.ofed` utility to verify whether or not the InfiniBand link is up. The utility also checks for and displays additional information such as

- HCA firmware version
- Kernel architecture
- Driver version
- Number of active HCA ports along with their states
- Node GUID

Note: For more details on `hca_self_test.ofed`, see the file `hca_self_test.readme` under `docs/`.

```
# hca_self_test.ofed

---- Performing Adapter Device Self Test ----
Number of CAs Detected ................. 1
PCI Device Check ....................... PASS
Kernel Arch ............................ x86_64
Host Driver Version .................... MLNX_OFED_LINUX-2.1-1.0.0 (OFED-2.1-1.0.0):
3.0.76-0.11-default
Host Driver RPM Check .................. PASS
Firmware on CA #0 VPI .................. v2.30.8000
Firmware Check on CA #0 (VPI) .......... PASS
Host Driver Initialization ............. PASS
Number of CA Ports Active .............. 1
Port State of Port #1 on CA #0 (VPI)..... UP 4X FDR (InfiniBand)
Port State of Port #2 on CA #0 (VPI)..... UP 4X FDR (InfiniBand)
Error Counter Check on CA #0 (VPI)...... PASS
Kernel Syslog Check .................... PASS
Node GUID on CA #0 (VPI) ............... 00:02:c9:03:00:30:0e:60
----------------- DONE --------------------
```

After the installer completes, information about the Mellanox OFED installation such as prefix, kernel version, and installation parameters can be retrieved by running the command `/etc/infiniband/info`.

2.3.4 Installation Results

Software

- Most of MLNX_OFED packages are installed under the “/usr” directory except for the following packages which are installed under the “/opt” directory:
  - openshmem, bupc, fca and ibutils
- The kernel modules are installed under
  - /lib/modules/`uname -r`/updates on SLES and Fedora Distributions
  - /lib/modules/`uname -r`/extra/mlnx-ofa_kernel on RHEL and other RedHat like Distributions
  - /lib/modules/`uname -r`/updates/dkms/ on Ubuntu

Firmware

- The firmware of existing network adapter devices will be updated if the following two conditions are fulfilled:
  a. You run the installation script in default mode; that is, *without* the option ‘--without-fw-update’.

b. The firmware version of the adapter device is older than the firmware version included with the Mellanox OFED ISO image

If an adapter's Flash was originally programmed with an Expansion ROM image, the automatic firmware update will also burn an Expansion ROM image.

- In case your machine has an unsupported network adapter device, no firmware update will occur and the error message below will be printed. Please contact your hardware vendor for help on firmware updates.

  Error message:

```
-I- Querying device ...
-E- Can't auto detect fw configuration file: ...
```

2.3.5 Post-installation Notes

- Most of the Mellanox OFED components can be configured or reconfigured after the installation by modifying the relevant configuration files. See the relevant chapters in this manual for details.
- The list of the modules that will be loaded automatically upon boot can be found in the `/etc/infiniband/openib.conf` file.

2.3.6 Installation Logging

While installing MLNX_OFED, the install log for each selected package will be saved in a separate log file.

The path to the directory containing the log files will be displayed after running the installation script in the following format: `"Logs dir: /tmp/MLNX_OFED_LINUX-<version>.<PID>.logs"`.

Example:

```
Logs dir: /tmp/MLNX_OFED_LINUX-2.1-0.0.2.31701.logs
```

2.4 Updating Firmware After Installation

In case you ran the `mlnxofedinstall` script with the '`--without-fw-update`' option and now you wish to (manually) update firmware on your adapter card(s), you need to perform the following steps:

If you need to burn an Expansion ROM image, please refer to "Burning the Expansion ROM Image" on page 221

The following steps are also appropriate in case you wish to burn newer firmware that you have downloaded from Mellanox Technologies' Web site (http://www.mellanox.com > Downloads > Firmware).

Step 1. Start `mst`.

```
host1# mst start
```

Step 2. Identify your target InfiniBand device for firmware update.

1. Get the list of InfiniBand device names on your machine.

```
host1# mst status
MST modules:
------------
    MST PCI module loaded
    MST PCI configuration module loaded
    MST Calibre (I2C) module is not loaded

MST devices:
------------
/dev/mst/mt25418_pciconf0        - PCI configuration cycles access.
                                   bus:dev.fn=02:00.0 addr.reg=88
                                   data.reg=92
                                   Chip revision is: A0
/dev/mst/mt25418_pci_cr0         - PCI direct access.
                                   bus:dev.fn=02:00.0 bar=0xdef00000
                                   size=0x100000
                                   Chip revision is: A0
/dev/mst/mt25418_pci_msix0       - PCI direct access.
                                   bus:dev.fn=02:00.0 bar=0xdeefe000
                                   size=0x2000
/dev/mst/mt25418_pci_uar0        - PCI direct access.
                                   bus:dev.fn=02:00.0 bar=0xdc800000
                                   size=0x800000
```

2. Your InfiniBand device is the one with the postfix "_pci_cr0". In the example listed above, this will be `/dev/mst/mt25418_pci_cr0`.

Step 3. Burn firmware.

- Burn a firmware image from a .mlx file using the `mlxburn` utility (that is already installed on your machine).

The following command burns firmware onto the ConnectX device with the device name obtained in the example of Step 2.

```
> flint -d /dev/mst/mt25418_pci_cr0 -i fw-25408-2_1_8000-MCX353A-FCA_A1.bin burn
```

Step 4. Reboot your machine after the firmware burning is completed.

2.5 Installing MLNX_OFED using YUM

2.5.1 Setting up MLNX_OFED YUM Repository

Step 1. Download the tarball to your host.

The image's name has the format `MLNX_OFED_LINUX-<ver>-<OS label><CPU arch>.tgz`. You can download it from http://www.mellanox.com > Products > Software> InfiniBand Drivers.

Step 2. Extract the MLNX_OFED tarball package to a shared location in your network.

```
# tar xzf MLNX_OFED_LINUX-<MLNX_OFED_version>-rhel6.4-x86_64.tgz
```

Step 3. Download and install Mellanox Technologies GPG-KEY:

The key can be downloaded via the following link:

http://www.mellanox.com/downloads/ofed/RPM-GPG-KEY-Mellanox

```
# wget http://www.mellanox.com/downloads/ofed/RPM-GPG-KEY-Mellanox
--2013-08-20 13:52:30--  http://www.mellanox.com/downloads/ofed/RPM-GPG-KEY-Mellanox
Resolving www.mellanox.com... 72.3.194.0
Connecting to www.mellanox.com|72.3.194.0|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1354 (1.3K) [text/plain]
Saving to: ?RPM-GPG-KEY-Mellanox?

100%[==============================================>] 1,354       --.-K/s   in 0s

2013-08-20 13:52:30 (247 MB/s) - ?RPM-GPG-KEY-Mellanox? saved [1354/1354]
```

Step 4. Install the key.

```
# sudo rpm --import RPM-GPG-KEY-Mellanox
```

Step 5. Check that the key was successfully imported.

```
# rpm -q gpg-pubkey --qf '%{NAME}-%{VERSION}-%{RELEASE}\t%{SUMMARY}\n' | grep Mellanox
gpg-pubkey-a9e4b643-520791ba    gpg(Mellanox Technologies <support@mellanox.com>)
```

Step 6. Create a YUM repository configuration file called `"/etc/yum.repos.d/mlnx_ofed.repo"` with the following content.

```
[mlnx_ofed]
name=MLNX_OFED Repository
baseurl=file:///<path to extracted MLNX_OFED package>
enabled=1
gpgkey=file:///<path to the downloaded key (RPM-GPG-KEY-Mellanox)>
gpgcheck=1
```

Step 7. Check that the repository was successfully added.

```
# yum repolist
Loaded plugins: product-id, security, subscription-manager
This system is not registered to Red Hat Subscription Management. You can use subscrip-
tion-manager to register.
repo id           repo name                                       status
mlnx_ofed         MLNX_OFED Repository                             108
rpmforge          RHEL 6Server - RPMforge.net - dag               4,597

repolist: 8,351
```

2.5.2 Installing MLNX_OFED using the YUM Tool

After setting up the YUM repository for MLNX_OFED package, perform the following:

Step 1. View the available package groups by invoking:

```
# yum grouplist | grep MLNX_OFED
   MLNX_OFED ALL
   MLNX_OFED BASIC
   MLNX_OFED GUEST
   MLNX_OFED HPC
   MLNX_OFED HYPERVISOR
   MLNX_OFED VMA
   MLNX_OFED VMA-ETH
   MLNX_OFED VMA-VPI
```

Step 2. Install the desired group.

```
# yum groupinstall 'MLNX_OFED ALL'
Loaded plugins: product-id, security, subscription-manager
This system is not registered to Red Hat Subscription Management. You can use subscrip-
tion-manager to register.
Setting up Group Process
Resolving Dependencies
--> Running transaction check
---> Package ar_mgr.x86_64 0:1.0-0.11.g22fff4a will be installed
..............
..............
  rds-devel.x86_64 0:2.0.6mlnx-1
  rds-tools.x86_64 0:2.0.6mlnx-1
  srptools.x86_64 0:0.0.4mlnx3-OFED.2.0.2.6.7.11.ge863cb7
Complete!
```

2.5.3 Updating Firmware After Installation

Installing MLNX_OFED using the YUM tool does not automatically update the firmware.

To update the firmware to the version included in MLNX_OFED package, you can either:

- Run the `mlnxofedinstall` script with the `"--fw-update-only"` flag

  or

- Update the firmware to the latest version available on Mellanox Technologies' Web site as described in section Section 2.4, "Updating Firmware After Installation", on page 41.

2.6 Uninstalling Mellanox OFED

Use the script `/usr/sbin/ofed_uninstall.sh` to uninstall the Mellanox OFED package. The script is part of the `ofed-scripts` RPM.

2.7 Uninstalling Mellanox OFED using the YUM Tool

If MLNX_OFED was installed using the yum tool, then it can be uninstalled as follow:

```
yum groupremove '<group name>'[1]
```

1. The "`<group name>`" must be the same group name that was previously used to install MLNX_OFED.

3 Configuration Files

For the complete list of configuration files, please refer to MLNX_OFED_configuration_files.txt at the following location:
docs/readme_and_user_manual/MLNX_OFED_configuration_files.txt

3.1 Persistent Naming for Network Interfaces

To avoid network interface renaming after boot or driver restart use the `"/etc/udev/rules.d/70-persistent-net.rules"` file.

- Example for Ethernet interfaces:

```
 # PCI device 0x15b3:0x1003 (mlx4_core)
 SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{address}=="00:02:c9:fa:c3:50",
ATTR{dev_id}=="0x0", ATTR{type}=="1", KERNEL=="eth*", NAME="eth1"
 SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{address}=="00:02:c9:fa:c3:51",
ATTR{dev_id}=="0x0", ATTR{type}=="1", KERNEL=="eth*", NAME="eth2"
 SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{address}=="00:02:c9:e9:56:a1",
ATTR{dev_id}=="0x0", ATTR{type}=="1", KERNEL=="eth*", NAME="eth3"
 SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{address}=="00:02:c9:e9:56:a2",
ATTR{dev_id}=="0x0", ATTR{type}=="1", KERNEL=="eth*", NAME="eth4"
```

- Example for IPoIB interfaces:

```
 SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{dev_id}=="0x0", ATTR{type}=="32",
NAME="ib0"
 SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{dev_id}=="0x1", ATTR{type}=="32",
NAME="ib1"
```

4 Driver Features

4.1 SCSI RDMA Protocol

4.1.1 Overview

As described in Section 1.3.4, the SCSI RDMA Protocol (SRP) is designed to take full advantage of the protocol off-load and RDMA features provided by the InfiniBand architecture. SRP allows a large body of SCSI software to be readily used on InfiniBand architecture. The SRP Initiator controls the connection to an SRP Target in order to provide access to remote storage devices across an InfiniBand fabric. The kSRP Target resides in an IO unit and provides storage services.

Section 4.1.2 describes the SRP Initiator included in Mellanox OFED for Linux. This package, however, does *not* include an SRP Target.

4.1.2 SRP Initiator

This SRP Initiator is based on open source from OpenFabrics (www.openfabrics.org) that implements the SCSI RDMA Protocol-2 (SRP-2). SRP-2 is described in Document # T10/1524-D available from http://www.t10.org.

The SRP Initiator supports

- Basic SCSI Primary Commands -3 (SPC-3)
  (www.t10.org/ftp/t10/drafts/spc3/spc3r21b.pdf)
- Basic SCSI Block Commands -2 (SBC-2)
  (www.t10.org/ftp/t10/drafts/sbc2/sbc2r16.pdf)
- Basic functionality, task management and limited error handling

4.1.2.1 Loading SRP Initiator

To load the SRP module, either execute the "`modprobe ib_srp`" command after the OFED driver is up, or change the value of SRP_LOAD in `/etc/infiniband/openib.conf` to "yes".

For the changes to take effect, run: /etc/init.d/openibd restart

When loading the ib_srp module, it is possible to set the module parameter srp_sg_tablesize. This is the maximum number of gather/scatter entries per I/O (default: 12).

4.1.2.1.1 SRP Module Parameters

When loading the SRP module, the following parameters can be set (viewable by the `"modinfo ib_srp"` command):

cmd_sg_entries	Default number of gather/scatter entries in the SRP command (default is 12, max 255)
allow_ext_sg	Default behavior when there are more than cmd_sg_entries S/G entries after mapping; fails the request when false (default false)
topspin_workarounds	Enable workarounds for Topspin/Cisco SRP target bugs
reconnect_delay	Time between successive reconnect attempts. Time between successive reconnect attempts of SRP initiator to a disconnected target until dev_loss_tmo timer expires (if enabled), after that the SCSI target will be removed.
fast_io_fail_tmo	Number of seconds between the observation of a transport layer error and failing all I/O. Increasing this timeout allows more tolerance to transport errors, however, doing so increases the total failover time in case of serious transport failure. Note: fast_io_fail_tmo value must be smaller than the value of reconnect_delay.
dev_loss_tmo	Maximum number of seconds that the SRP transport should insulate transport layer errors. After this time has been exceeded the SCSI target is removed. Normally it is advised to set this to -1 (disabled) which will never remove the scsi_host. In deployments where different SRP targets are connected and disconnected frequently, it may be required to enable this timeout in order to clean old scsi_hosts representing targets that no longer exists.

Constraints between parameters:

- `dev_loss_tmo, fast_io_fail_tmo, reconnect_delay` cannot be all disabled or negative values.
- `reconnect_delay` must be positive number.
- `fast_io_fail_tmo` must be smaller than SCSI block device timeout.
- `fast_io_fail_tmo` must be smaller than `dev_loss_tmo`.

4.1.2.1.2 SRP Remote Ports Parameters

Several SRP remote ports parameters are modifiable online on existing connection.

- ***To modify dev_loss_tmo to 600 seconds:***

```
echo 600 > /sys/class/srp_remote_ports/port-xxx/dev_loss_tmo
```

- ***To modify fast_io_fail_tmo to 15 seconds:***

```
echo 15 > /sys/class/srp_remote_ports/port-xxx/fast_io_fail_tmo
```

- ***To modify reconnect_delay to 10 seconds:***

```
echo 20 > /sys/class/srp_remote_ports/port-xxx/reconnect_delay
```

4.1.2.2 Manually Establishing an SRP Connection

The following steps describe how to manually load an SRP connection between the Initiator and an SRP Target. Section 4.1.2.4 explains how to do this automatically.

- Make sure that the `ib_srp` module is loaded, the SRP Initiator is reachable by the SRP Target, and that an SM is running.
- To establish a connection with an SRP Target and create an SRP (SCSI) device for that target under `/dev`, use the following command:

```
echo -n id_ext=[GUID value],ioc_guid=[GUID value],dgid=[port GID value],\
pkey=ffff,service_id=[service[0] value] > \
/sys/class/infiniband_srp/srp-mlx[hca number]-[port number]/add_target
```

  See Section 4.1.2.3 for instructions on how the parameters in this `echo` command may be obtained.

  Notes:

  - Execution of the above “echo” command may take some time
  - The SM must be running while the command executes
  - It is possible to include additional parameters in the echo command:
    - max_cmd_per_lun - Default: 62
    - max_sect (short for max_sectors) - sets the request size of a command
    - io_class - Default: 0x100 as in rev 16A of the specification (In rev 10 the default was 0xff00)
    - tl_retry_count - a number in the range 2..7 specifying the IB RC retry count. Default: 2
    - comp_vector, a number in the range 0..n-1 specifying the MSI-X completion vector. Some HCA's allocate multiple (n) MSI-X vectors per HCA port. If the IRQ affinity masks of these interrupts have been configured such that each MSI-X interrupt is handled by a different CPU then the comp_vector parameter can be used to spread the SRP completion workload over multiple CPU's.
    - cmd_sg_entries, a number in the range 1..255 that specifies the maximum number of data buffer descriptors stored in the SRP_CMD information unit itself. With allow_ext_sg=0 the parameter cmd_sg_entries defines the maximum S/G list length for a single SRP_CMD, and commands whose S/G list length exceeds this limit after S/G list collapsing will fail.
    - initiator_ext - Please refer to Section 9 (Multiple Connections...)
- To list the new SCSI devices that have been added by the echo command, you may use either of the following two methods:
  - Execute “fdisk -l”. This command lists all devices; the new devices are included in this listing.
  - Execute “dmesg” or look at `/var/log/messages` to find messages with the names of the new devices.

4.1.2.2.1 SRP sysfs Parameters

Interface for making ib_srp connect to a new target. One can request ib_srp to connect to a new target by writing a comma-separated list of login parameters to this sysfs attribute. The supported parameters are:

id_ext	A 16-digit hexadecimal number specifying the eight byte identifier extension in the 16-byte SRP target port identifier. The target port identifier is sent by ib_srp to the target in the SRP_LOGIN_REQ request.

ioc_guid	A 16-digit hexadecimal number specifying the eight byte I/O controller GUID portion of the 16-byte target port identifier.
dgid	A 32-digit hexadecimal number specifying the destination GID.
pkey	A four-digit hexadecimal number specifying the InfiniBand partition key.
service_id	A 16-digit hexadecimal number specifying the InfiniBand service ID used to establish communication with the SRP target. How to find out the value of the service ID is specified in the documentation of the SRP target.
max_sect	A decimal number specifying the maximum number of 512-byte sectors to be transferred via a single SCSI command.
max_cmd_per_lun	A decimal number specifying the maximum number of outstanding commands for a single LUN.
io_class	A hexadecimal number specifying the SRP I/O class. Must be either 0xff00 (rev 10) or 0x0100 (rev 16a). The I/O class defines the format of the SRP initiator and target port identifiers.
initiator_ext	A 16-digit hexadecimal number specifying the identifier extension portion of the SRP initiator port identifier. This data is sent by the initiator to the target in the SRP_LOGIN_REQ request.
cmd_sg_entries	A number in the range 1..255 that specifies the maximum number of data buffer descriptors stored in the SRP_CMD information unit itself. With allow_ext_sg=0 the parameter cmd_sg_entries defines the maximum S/G list length for a single SRP_CMD, and commands whose S/G list length exceeds this limit after S/G list collapsing will fail.
allow_ext_sg	Whether ib_srp is allowed to include a partial memory descriptor list in an SRP_CMD instead of the entire list. If a partial memory descriptor list has been included in an SRP_CMD the remaining memory descriptors are communicated from initiator to target via an additional RDMA transfer. Setting allow_ext_sg to 1 increases the maximum amount of data that can be transferred between initiator and target via a single SCSI command. Since not all SRP target implementations support partial memory descriptor lists the default value for this option is 0.
sg_tablesize	A number in the range 1..2048 specifying the maximum S/G list length the SCSI layer is allowed to pass to ib_srp. Specifying a value that exceeds cmd_sg_entries is only safe with partial memory descriptor list support enabled (allow_ext_sg=1).
comp_vector	A number in the range 0..n-1 specifying the MSI-X completion vector. Some HCA's allocate multiple (n) MSI-X vectors per HCA port. If the IRQ affinity masks of these interrupts have been configured such that each MSI-X interrupt is handled by a different CPU then the comp_vector parameter can be used to spread the SRP completion workload over multiple CPU's.

tl_retry_count	A number in the range 2..7 specifying the IB RC retry count.

4.1.2.3 SRP Tools - ibsrpdm, srp_daemon and srpd Service Script

To assist in performing the steps in Section 6, the OFED distribution provides two utilities, ibsrpdm and srp_daemon, which

- Detect targets on the fabric reachable by the Initiator (for Step 1)
- Output target attributes in a format suitable for use in the above “echo” command (Step 2)
- A service script srpd which may be started at stack startup

The utilities can be found under `/usr/sbin/`, and are part of the srptools RPM that may be installed using the Mellanox OFED installation. Detailed information regarding the various options for these utilities are provided by their man pages.

Below, several usage scenarios for these utilities are presented.

ibsrpdm

ibsrpdm is using for the following tasks:

1. Detecting reachable targets
   a. To detect all targets reachable by the SRP initiator via the default umad device (/sys/class/infiniband_mad/umad0), execute the following command:

      ```
      ibsrpdm
      ```

      This command will output information on each SRP Target detected, in human-readable form.

      Sample output:

      ```
      IO Unit Info:
           port LID:        0103
           port GID:        fe80000000000000002c90200402bd5
           change ID:       0002
           max controllers: 0x10
        controller[  1]
            GUID:      0002c90200402bd4
            vendor ID: 0002c9
            device ID: 005a44
            IO class : 0100
            ID:        LSI Storage Systems SRP Driver 200400a0b81146a1
            service entries: 1
            service[  0]: 200400a0b81146a1 / SRP.T10:200400A0B81146A1
      ```

   b. To detect all the SRP Targets reachable by the SRP Initiator via another umad device, use the following command:

      ```
      ibsrpdm -d <umad device>
      ```

2. Assistance in creating an SRP connection
   a. To generate output suitable for utilization in the “echo” command of Section 4.1.2.2, add the ‘-c’ option to ibsrpdm:

      ```
      ibsrpdm -c
      ```

Sample output:

```
id_ext=200400A0B81146A1,ioc_guid=0002c90200402bd4,
dgid=fe8000000000000002c90200402bd5,pkey=ffff,service_id=200400a0b81146a1
```

b. To establish a connection with an SRP Target using the output from the 'ibsrpdm -c' example above, execute the following command:

```
echo -n id_ext=200400A0B81146A1,ioc_guid=0002c90200402bd4,
dgid=fe8000000000000002c90200402bd5,pkey=ffff,service_id=200400a0b81146a1 > /sys/
class/infiniband_srp/srp-mlx4_0-1/add_target
```

The SRP connection should now be up; the newly created SCSI devices should appear in the listing obtained from the 'fdisk -l' command.

3. Discover reachable SRP Targets given an InfiniBand HCA name and port, rather than by just runing /sys/class/infiniband_mad/umad<N> where <N> is a digit

srpd

The srpd service script allows automatic activation and termination of the srp_daemon utility on all system live InfiniBand ports.

srp_daemon

The srp_daemon utility is based on ibsrpdm and extends its functionality. In addition to the ibsrpdm functionality described above, srp_daemon can also

- Establish an SRP connection by itself (without the need to issue the "echo" command described in Section 4.1.2.2)
- Continue running in background, detecting new targets and establishing SRP connections with them (daemon mode)
- Discover reachable SRP Targets given an infiniband HCA name and port, rather than just by
  /dev/umad<N> where <N> is a digit
- Enable High Availability operation (together with Device-Mapper Multipath)
- Have a configuration file that determines the targets to connect to

1. srp_daemon commands equivalent to ibsrpdm:

```
"srp_daemon -a -o"    is equivalent to "ibsrpdm"
"srp_daemon -c -a -o" is equivalent to "ibsrpdm -c"
```

These srp_daemon commands can behave differently than the equivalent ibsrpdm command when /etc/srp_daemon.conf is not empty.

2. srp_daemon extensions to ibsrpdm
   - To discover SRP Targets reachable from the HCA device <InfiniBand HCA name> and the port <port num>, (and to generate output suitable for 'echo',) you may execute:

```
host1# srp_daemon -c -a -o -i <InfiniBand HCA name> -p <port number>
```

To obtain the list of InfiniBand HCA device names, you can either use the ibstat tool or run ‘ls /sys/class/infiniband’.

- To both discover the SRP Targets and establish connections with them, just add the -e option to the above command.
- Executing srp_daemon over a port without the -a option will only display the reachable targets via the port and to which the initiator is not connected. If executing with the -e option it is better to omit -a.
- It is recommended to use the -n option. This option adds the initiator_ext to the connecting string. (See Section 4.1.2.5 for more details).
- srp_daemon has a configuration file that can be set, where the default is /etc/srp_daemon.conf. Use the -f to supply a different configuration file that configures the targets srp_daemon is allowed to connect to. The configuration file can also be used to set values for additional parameters (e.g., max_cmd_per_lun, max_sect).
- A continuous background (daemon) operation, providing an automatic ongoing detection and connection capability. See Section 4.1.2.4.

4.1.2.4 Automatic Discovery and Connection to Targets

- Make sure that the ib_srp module is loaded, the SRP Initiator can reach an SRP Target, and that an SM is running.
- To connect to all the existing Targets in the fabric, run “`srp_daemon -e -o`”. This utility will scan the fabric once, connect to every Target it detects, and then exit.

srp_daemon will follow the configuration it finds in /etc/srp_daemon.conf. Thus, it will ignore a target that is disallowed in the configuration file.

- To connect to all the existing Targets in the fabric and to connect to new targets that will join the fabric, execute srp_daemon -e. This utility continues to execute until it is either killed by the user or encounters connection errors (such as no SM in the fabric).
- To execute SRP daemon as a daemon on all the ports:
  - `srp_daemon.sh` (found under /usr/sbin/). `srp_daemon.sh` sends its log to `/var/log/srp_daemon.log`.
  - Start the srpd service script, run `service srpd start`
- It is possible to configure this script to execute automatically when the InfiniBand driver starts by changing the value of SRP_DAEMON_ENABLE in /etc/infiniband/openib.conf to “yes”. However, this option also enables SRP High Availability that has some more features – see Section 4.1.2.6).

  For the changes in `openib.conf` to take effect, run:
  `/etc/init.d/openibd restart`

4.1.2.5 Multiple Connections from Initiator InfiniBand Port to the Target

Some system configurations may need multiple SRP connections from the SRP Initiator to the same SRP Target: to the same Target IB port, or to different IB ports on the same Target HCA.

In case of a single Target IB port, i.e., SRP connections use the same path, the configuration is enabled using a different initiator_ext value for each SRP connection. The initiator_ext value is a 16-hexadecimal-digit value specified in the connection command.

Also in case of two physical connections (i.e., network paths) from a single initiator IB port to two different IB ports on the same Target HCA, there is need for a different initiator_ext value on each path. The conventions is to use the Target port GUID as the initiator_ext value for the relevant path.

If you use srp_daemon with -n flag, it automatically assigns initiator_ext values according to this convention. For example:

```
id_ext=200500A0B81146A1,ioc_guid=0002c90200402bec,\
dgid=fe8000000000000002c90200402bed,pkey=ffff,\
service_id=200500a0b81146a1,initiator_ext=ed2b400002c90200
```

Notes:

1. It is recommended to use the -n flag for all srp_daemon invocations.
2. ibsrpdm does not have a corresponding option.
3. srp_daemon.sh always uses the -n option (whether invoked manually by the user, or automatically at startup by setting SRP_DAEMON_ENABLE to yes).

4.1.2.6 High Availability (HA)

Overview

High Availability works using the Device-Mapper (DM) multipath and the SRP daemon. Each initiator is connected to the same target from several ports/HCAs. The DM multipath is responsible for joining together different paths to the same target and for fail-over between paths when one of them goes offline. Multipath will be executed on newly joined SCSI devices.

Each initiator should execute several instances of the SRP daemon, one for each port. At startup, each SRP daemon detects the SRP Targets in the fabric and sends requests to the ib_srp module to connect to each of them. These SRP daemons also detect targets that subsequently join the fabric, and send the ib_srp module requests to connect to them as well.

Operation

When a path (from port1) to a target fails, the ib_srp module starts an error recovery process. If this process gets to the reset_host stage and there is no path to the target from this port, ib_srp will remove this scsi_host. After the scsi_host is removed, multipath switches to another path to this target (from another port/HCA).

When the failed path recovers, it will be detected by the SRP daemon. The SRP daemon will then request ib_srp to connect to this target. Once the connection is up, there will be a new scsi_host for this target. Multipath will be executed on the devices of this host, returning to the original state (prior to the failed path).

Manual Activation of High Availability

Initialization: (Execute after each boot of the driver)

1. Execute modprobe dm-multipath
2. Execute modprobe ib-srp
3. Make sure you have created file /etc/udev/rules.d/91-srp.rules as described above.
4. Execute for each port and each HCA:

   ```
   srp_daemon -c -e -R 300 -i <InfiniBand HCA name> -p <port number>
   ```

   This step can be performed by executing srp_daemon.sh, which sends its log to /var/log/srp_daemon.log.

Now it is possible to access the SRP LUNs on `/dev/mapper/`.

It is possible for regular (non-SRP) LUNs to also be present; the SRP LUNs may be identified by their names. You can configure the /etc/multipath.conf file to change multipath behavior.

It is also possible that the SRP LUNs will not appear under /dev/mapper/. This can occur if the SRP LUNs are in the black-list of multipath. Edit the 'blacklist' section in /etc/multipath.conf and make sure the SRP LUNs are not black-listed.

Automatic Activation of High Availability

- Set the value of SRP_DAEMON_ENABLE in `/etc/infiniband/openib.conf` to "yes".

  For the changes in `openib.conf` to take effect, run:
  `/etc/init.d/openibd restart`
- Start srpd service, run: `service srpd start`
- From the next loading of the driver it will be possible to access the SRP LUNs on /dev/mapper/

It is possible that regular (not SRP) LUNs may also be present; the SRP LUNs may be identified by their name.

- It is possible to see the output of the SRP daemon in /var/log/srp_daemon.log

4.1.2.7 Shutting Down SRP

SRP can be shutdown by using "rmmod ib_srp", or by stopping the OFED driver ("`/etc/init.d/openibd stop`"), or as a by-product of a complete system shutdown.

Prior to shutting down SRP, remove all references to it. The actions you need to take depend on the way SRP was loaded. There are three cases:

1. Without High Availability

When working without High Availability, you should unmount the SRP partitions that were mounted prior to shutting down SRP.

2. After Manual Activation of High Availability

If you manually activated SRP High Availability, perform the following steps:

a. Unmount all SRP partitions that were mounted.

b. Stop service srpd (Kill the SRP daemon instances).

c. Make sure there are no multipath instances running. If there are multiple instances, wait for them to end or kill them.

d. Run: `multipath -F`

3. After Automatic Activation of High Availability

If SRP High Availability was automatically activated, SRP shutdown must be part of the driver shutdown ("/etc/init.d/openibd stop") which performs Steps 2-4 of case b above. However, you still have to unmount all SRP partitions that were mounted before driver shutdown.

4.2 iSCSI Extensions for RDMA (iSER)

iSCSI Extensions for RDMA (iSER) is currently at beta level.
Please be aware that the content below is subject to change.

4.2.1 Overview

iSCSI Extensions for RDMA (iSER) extends the iSCSI protocol to RDMA. It permits data to be transferred directly into and out of SCSI buffers without intermediate data copies.

4.2.2 iSER Initiator

The iSER initiator is controlled through the iSCSI interface available from the iscsi-initiator-utils package.

Make sure iSCSI is enabled and properly configured on your system before proceeding with iSER.

Targets settings such as `timeouts` and `retries` are set the same as any other iSCSI targets.

If targets are set to auto connect on boot, and targets are unreachable, it may take a long time to continue the boot process if `timeouts` and `max retries` are set too high.

Example for discovering and connecting targets over iSER:

```
iscsiadm -m discovery -o new -o old -t st -I iser -p <ip:port> -l
```

iSER also supports RoCE without any additional configuration required. To bond the RoCE interfaces, set the `fail_over_mac` option in the bonding driver.

4.3 IP over InfiniBand

4.3.1 Introduction

The IP over IB (IPoIB) driver is a network interface implementation over InfiniBand. IPoIB encapsulates IP datagrams over an InfiniBand Connected or Datagram transport service. The IPoIB driver, ib_ipoib, exploits the following capabilities:

- VLAN simulation over an InfiniBand network via child interfaces
- High Availability via Bonding
- Varies MTU values:
  - up to 4k in Datagram mode
  - up to 64k in Connected mode
- Uses any ConnectX® IB ports (one or two)
- Inserts IP/UDP/TCP checksum on outgoing packets
- Calculates checksum on received packets
- Support net device TSO through ConnectX® LSO capability to defragment large datagrams to MTU quantas.
- Dual operation mode - datagram and connected
- Large MTU support through connected mode

IPoIB also supports the following software based enhancements:

- Giant Receive Offload
- NAPI
- Ethtool support

4.3.2 IPoIB Mode Setting

IPoIB can run in two modes of operation: Connected mode and Datagram mode. By default, IPoIB is set to work in Datagram, except for Connect-IB™ adapter card which uses IPoIB with Connected mode as default.

For better scalability and performance we recommend using the Datagram mode. However, the mode can be changed to Connected mode by editing the file `/etc/infiniband/openib.conf` and setting `'SET_IPOIB_CM=yes'`.

The `SET_IPOIB_CM` parameter is set to `"auto"` by default to enable the Connected mode for Connect-IB™ card and Datagram for all other ConnectX® cards.

After changing the mode, you need to restart the driver by running:

```
/etc/init.d/openibd restart
```

To check the current mode used for out-going connections, enter:

```
cat /sys/class/net/ib<n>/mode
```

4.3.3 IPoIB Configuration

Unless you have run the installation script `mlnxofedinstall` with the flag '-n', then IPoIB has not been configured by the installation. The configuration of IPoIB requires assigning an IP address and a subnet mask to each HCA port, like any other network adapter card (i.e., you need to prepare a file called ifcfg-ib<n> for each port). The first port on the first HCA in the host is called interface ib0, the second port is called ib1, and so on.

An IPoIB configuration can be based on DHCP (Section 4.3.3.1) or on a static configuration (Section 4.3.3.2) that you need to supply. You can also apply a manual configuration that persists only until the next reboot or driver restart (Section 4.3.3.3).

4.3.3.1 IPoIB Configuration Based on DHCP

Setting an IPoIB interface configuration based on DHCP is performed similarly to the configuration of Ethernet interfaces. In other words, you need to make sure that IPoIB configuration files include the following line:

For RedHat:

```
BOOTPROTO=dhcp
```

For SLES:

```
BOOTPROTO='dchp'
```

If IPoIB configuration files are included, ifcfg-ib<n> files will be installed under:
/etc/sysconfig/network-scripts/ on a RedHat machine
/etc/sysconfig/network/ on a SuSE machine.

A patch for DHCP is required for supporting IPoIB. For further information, please see the REAME which is available under the docs/dhcp/ directory.

Standard DHCP fields holding MAC addresses are not large enough to contain an IPoIB hardware address. To overcome this problem, DHCP over InfiniBand messages convey a client identifier field used to identify the DHCP session. This client identifier field can be used to associate an IP address with a client identifier value, such that the DHCP server will grant the same IP address to any client that conveys this client identifier.

The length of the client identifier field is not fixed in the specification. For the *Mellanox OFED for Linux* package, it is recommended to have IPoIB use the same format that FlexBoot uses for this client identifier – see Section A.4.2, "Configuring the DHCP Server", on page 222.

4.3.3.1.1 DHCP Server

In order for the DHCP server to provide configuration records for clients, an appropriate configuration file needs to be created. By default, the DHCP server looks for a configuration file called `dhcpd.conf` under `/etc`. You can either edit this file or create a new one and provide its full path to the DHCP server using the -cf flag. See a file example at `docs/dhcpd.conf` of the *Mellanox OFED for Linux* installation.

The DHCP server must run on a machine which has loaded the IPoIB module.

To run the DHCP server from the command line, enter:

```
dhcpd <IB network interface name> -d
```

Example:

```
host1# dhcpd ib0 -d
```

4.3.3.1.2 DHCP Client (Optional)

A DHCP client can be used if you need to prepare a diskless machine with an IB driver. See Step 8 under "Example: Adding an IB Driver to initrd (Linux)".

In order to use a DHCP client identifier, you need to first create a configuration file that defines the DHCP client identifier.
Then run the DHCP client with this file using the following command:

```
dhclient -cf <client conf file> <IB network interface name>
```

Example of a configuration file for the ConnectX (PCI Device ID 26428), called `dhclient.conf`:

```
# The value indicates a hexadecimal number
interface "ib1" {
send dhcp-client-identifier
ff:00:00:00:00:00:02:00:00:02:c9:00:00:02:c9:03:00:00:10:39;
}
```

Example of a configuration file for InfiniHost III Ex (PCI Device ID 25218), called `dhclient.conf`:

```
# The value indicates a hexadecimal number
interface "ib1" {
send dhcp-client-identifier
20:00:55:04:01:fe:80:00:00:00:00:00:00:00:02:c9:02:00:23:13:92;
}
```

In order to use the configuration file, run:

```
host1# dhclient -cf dhclient.conf ib1
```

4.3.3.2 Static IPoIB Configuration

If you wish to use an IPoIB configuration that is not based on DHCP, you need to supply the installation script with a configuration file (using the '-n' option) containing the full IP configuration. The IPoIB configuration file can specify either or both of the following data for an IPoIB interface:

- A static IPoIB configuration
- An IPoIB configuration based on an Ethernet configuration

See your Linux distribution documentation for additional information about configuring IP addresses.

The following code lines are an excerpt from a sample IPoIB configuration file:

```
# Static settings; all values provided by this file
IPADDR_ib0=11.4.3.175
NETMASK_ib0=255.255.0.0
NETWORK_ib0=11.4.0.0
BROADCAST_ib0=11.4.255.255
ONBOOT_ib0=1
# Based on eth0; each '*' will be replaced with a corresponding octet
# from eth0.
LAN_INTERFACE_ib0=eth0
IPADDR_ib0=11.4.'*'.'*'
NETMASK_ib0=255.255.0.0
NETWORK_ib0=11.4.0.0
BROADCAST_ib0=11.4.255.255
ONBOOT_ib0=1
# Based on the first eth<n> interface that is found (for n=0,1,...);
# each '*' will be replaced with a corresponding octet from eth<n>.
LAN_INTERFACE_ib0=
IPADDR_ib0=11.4.'*'.'*'
NETMASK_ib0=255.255.0.0
NETWORK_ib0=11.4.0.0
BROADCAST_ib0=11.4.255.255
ONBOOT_ib0=1
```

4.3.3.3 Manually Configuring IPoIB

This manual configuration persists only until the next reboot or driver restart.

To manually configure IPoIB for the default IB partition (VLAN), perform the following steps:

Step 1. To configure the interface, enter the `ifconfig` command with the following items:

- The appropriate IB interface (ib0, ib1, etc.)
- The IP address that you want to assign to the interface
- The netmask keyword

- The subnet mask that you want to assign to the interface

The following example shows how to configure an IB interface:

```
host1$ ifconfig ib0 11.4.3.175 netmask 255.255.0.0
```

Step 2. (Optional) Verify the configuration by entering the `ifconfig` command with the appropriate interface identifier *ib#* argument.

The following example shows how to verify the configuration:

```
host1$ ifconfig ib0
b0  Link encap:UNSPEC  HWaddr 80-00-04-04-FE-80-00-00-00-00-00-00-00-00-00-00
inet addr:11.4.3.175  Bcast:11.4.255.255  Mask:255.255.0.0
UP BROADCAST MULTICAST  MTU:65520  Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:128
RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
```

Step 3. Repeat Step 1 and Step 2 on the remaining interface(s).

4.3.4 Subinterfaces

You can create subinterfaces for a primary IPoIB interface to provide traffic isolation. Each such subinterface (also called a child interface) has a different IP and network addresses from the primary (parent) interface. The default Partition Key (PKey), ff:ff, applies to the primary (parent) interface.

This section describes how to

- Create a subinterface (Section 4.3.4.1)
- Remove a subinterface (Section 4.3.4.2)

4.3.4.1 Creating a Subinterface

In the following procedure, ib0 is used as an example of an IB subinterface.

To create a child interface (subinterface), follow this procedure:

Step 1. Decide on the PKey to be used in the subnet (valid values can be 0 or any 16-bit unsigned value). The actual PKey used is a 16-bit number with the most significant bit set. For example, a value of 1 will give a PKey with the value 0x8001.

Step 2. Create a child interface by running:

```
host1$ echo <PKey> > /sys/class/net/<IB subinterface>/create_child
```

Example:

```
host1$ echo 1 > /sys/class/net/ib0/create_child
```

This will create the interface ib0.8001.

Step 3. Verify the configuration of this interface by running:

```
host1$ ifconfig <subinterface>.<subinterface PKey>
```

Using the example of Step 2:

```
host1$ ifconfig ib0.8001
ib0.8001  Link encap:UNSPEC  HWaddr 80-00-00-4A-FE-80-00-00-00-00-00-00-00-00-00-00
BROADCAST MULTICAST  MTU:2044  Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:128
RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
```

Step 4. As can be seen, the interface does not have IP or network addresses. To configure those, you should follow the manual configuration procedure described in Section 4.3.3.3.

Step 5. To be able to use this interface, a configuration of the Subnet Manager is needed so that the PKey chosen, which defines a broadcast address, be recognized (see Chapter 8, "OpenSM – Subnet Manager").

4.3.4.2 Removing a Subinterface

To remove a child interface (subinterface), run:

```
echo <subinterface PKey> /sys/class/net/<ib_interface>/delete_child
```

Using the example of Step 2:

```
echo 0x8001 > /sys/class/net/ib0/delete_child
```

Note that when deleting the interface you must use the PKey value with the most significant bit set (e.g., 0x8000 in the example above).

4.3.5 Verifying IPoIB Functionality

To verify your configuration and your IPoIB functionality, perform the following steps:

Step 1. Verify the IPoIB functionality by using the `ifconfig` command.

The following example shows how two IB nodes are used to verify IPoIB functionality. In the following example, IB node 1 is at 11.4.3.175, and IB node 2 is at 11.4.3.176:

```
host1# ifconfig ib0 11.4.3.175 netmask 255.255.0.0
host2# ifconfig ib0 11.4.3.176 netmask 255.255.0.0
```

Step 2. Enter the ping command from 11.4.3.175 to 11.4.3.176.

The following example shows how to enter the ping command:

```
host1# ping -c 5 11.4.3.176
PING 11.4.3.176 (11.4.3.176) 56(84) bytes of data.
64 bytes from 11.4.3.176: icmp_seq=0 ttl=64 time=0.079 ms
64 bytes from 11.4.3.176: icmp_seq=1 ttl=64 time=0.044 ms
64 bytes from 11.4.3.176: icmp_seq=2 ttl=64 time=0.055 ms
64 bytes from 11.4.3.176: icmp_seq=3 ttl=64 time=0.049 ms
64 bytes from 11.4.3.176: icmp_seq=4 ttl=64 time=0.065 ms
--- 11.4.3.176 ping statistics ---
```

```
5 packets transmitted, 5 received, 0% packet loss, time 3999ms rtt min/avg/max/mdev =
0.044/0.058/0.079/0.014 ms, pipe 2
```

4.3.6 Bonding IPoIB

To create an interface configuration script for the ibX and bondX interfaces, you should use the standard syntax (depending on your OS).

Bonding of IPoIB interfaces is accomplished in the same manner as would bonding of Ethernet interfaces: via the Linux Bonding Driver.

- Network Script files for IPoIB slaves are named after the IPoIB interfaces (e.g: ifcfg-ib0)
- The only meaningful bonding policy in IPoIB is High-Availability (bonding mode number 1, or active-backup)
- Bonding parameter "fail_over_mac" is meaningless in IPoIB interfaces, hence, the only supported value is the default: 0 (or "none" in SLES11)

For a persistent bonding IPoIB Network configuration, use the same Linux Network Scripts semantics, with the following exceptions/ additions:

- In the bonding master configuration file (e.g: ifcfg-bond0), in addition to Linux bonding semantics, use the following parameter: MTU=65520

65520 is a valid MTU value only if all IPoIB slaves operate in Connected mode (See Section 4.3.2, “IPoIB Mode Setting”, on page 56) and are configured with the same value. For IPoIB slaves that work in datagram mode, use MTU=2044. If you do not set the correct MTU or do not set MTU at all, performance of the interface might decrease.

- In the bonding slave configuration file (e.g: ifcfg-ib0), use the same Linux Network Scripts semantics. In particular: DEVICE=ib0
- In the bonding slave configuration file (e.g: ifcfg-ib0.8003), the line TYPE=InfiniBand is necessary when using bonding over devices configured with partitions (p_key)
- For RHEL users:

  In /etc/modprobe.b/bond.conf add the following lines:

  ```
  alias bond0 bonding
  ```

- For SLES users:

  It is necessary to update the MANDATORY_DEVICES environment variable in /etc/sysconfig/network/config with the names of the IPoIB slave devices (e.g. ib0, ib1, etc.). Otherwise, bonding master may be created before IPoIB slave interfaces at boot time.

  It is possible to have multiple IPoIB bonding masters and a mix of IPoIB bonding master and Ethernet bonding master. However, It is NOT possible to mix Ethernet and IPoIB slaves under the same bonding master

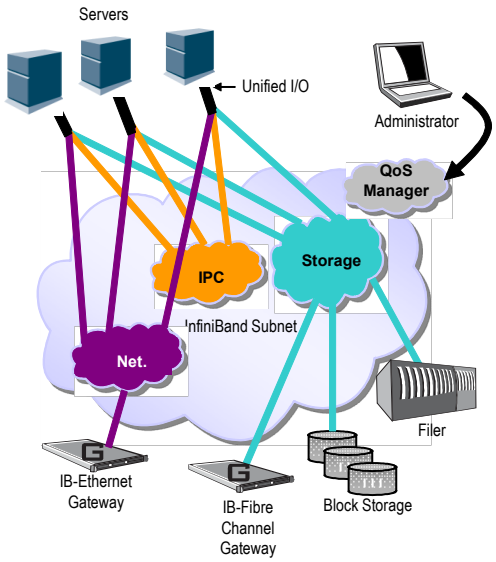
Restarting openibd does no keep the bonding configuration via Network Scripts. You have to restart the network service in order to bring up the bonding master. After the configuration is saved, restart the network service by running: /etc/init.d/network restart.

4.4 Quality of Service InfiniBand

4.4.1 Quality of Service Overview

Quality of Service (QoS) requirements stem from the realization of I/O consolidation over an IB network. As multiple applications and ULPs share the same fabric, a means is needed to control their use of network resources.

Figure 2: I/O Consolidation Over InfiniBand



QoS over Mellanox OFED for Linux is discussed in Chapter 8, “OpenSM – Subnet Manager”.

The basic need is to differentiate the service levels provided to different traffic flows, such that a policy can be enforced and can control each flow utilization of fabric resources.

The InfiniBand Architecture Specification defines several hardware features and management interfaces for supporting QoS:

- Up to 15 Virtual Lanes (VL) carry traffic in a non-blocking manner
- Arbitration between traffic of different VLs is performed by a two-priority-level weighted round robin arbiter. The arbiter is programmable with a sequence of (VL, weight) pairs and a maximal number of high priority credits to be processed before low priority is served
- Packets carry class of service marking in the range 0 to 15 in their header SL field
- Each switch can map the incoming packet by its SL to a particular output VL, based on a programmable table VL=SL-to-VL-MAP(in-port, out-port, SL)
- The Subnet Administrator controls the parameters of each communication flow by providing them as a response to Path Record (PR) or MultiPathRecord (MPR) queries

DiffServ architecture (IETF RFC 2474 & 2475) is widely used in highly dynamic fabrics. The following subsections provide the functional definition of the various software elements that enable a DiffServ-like architecture over the Mellanox OFED software stack.

4.4.2 QoS Architecture

QoS functionality is split between the SM/SA, CMA and the various ULPs. We take the "chronology approach" to describe how the overall system works.

1. The network manager (human) provides a set of rules (policy) that define how the network is being configured and how its resources are split to different QoS-Levels. The policy also define how to decide which QoS-Level each application or ULP or service use.
2. The SM analyzes the provided policy to see if it is realizable and performs the necessary fabric setup. Part of this policy defines the default QoS-Level of each partition. The SA is enhanced to match the requested Source, Destination, QoS-Class, Service-ID, PKey against the policy, so clients (ULPs, programs) can obtain a policy enforced QoS. The SM may also set up partitions with appropriate IPoIB broadcast group. This broadcast group carries its QoS attributes: SL, MTU, RATE, and Packet Lifetime.
3. IPoIB is being setup. IPoIB uses the SL, MTU, RATE and Packet Lifetime available on the multicast group which forms the broadcast group of this partition.
4. MPI which provides non IB based connection management should be configured to run using hard coded SLs. It uses these SLs for every QP being opened.
5. ULPs that use CM interface (like SRP) have their own pre-assigned Service-ID and use it while obtaining PathRecord/MultiPathRecord (PR/MPR) for establishing connections. The SA receiving the PR/MPR matches it against the policy and returns the appropriate PR/MPR including SL, MTU, RATE and Lifetime.
6. ULPs and programs (e.g. SDP) use CMA to establish RC connection provide the CMA the target IP and port number. ULPs might also provide QoS-Class. The CMA then creates Service-ID for the ULP and passes this ID and optional QoS-Class in the PR/MPR request. The resulting PR/MPR is used for configuring the connection QP.

PathRecord and MultiPathRecord Enhancement for QoS:

As mentioned above, the PathRecord and MultiPathRecord attributes are enhanced to carry the Service-ID which is a 64bit value. A new field QoS-Class is also provided.

A new capability bit describes the SM QoS support in the SA class port info. This approach provides an easy migration path for existing access layer and ULPs by not introducing new set of PR/MPR attributes.

4.4.3 Supported Policy

The QoS policy, which is specified in a stand-alone file, is divided into the following four subsections:

I. Port Group

A set of CAs, Routers or Switches that share the same settings. A port group might be a partition defined by the partition manager policy, list of GUIDs, or list of port names based on NodeDescription.

II. Fabric Setup

Defines how the SL2VL and VLArb tables should be setup.

In OFED this part of the policy is ignored. SL2VL and VLArb tables should be configured in the OpenSM options file (opensm.opts).

III. QoS-Levels Definition

This section defines the possible sets of parameters for QoS that a client might be mapped to. Each set holds SL and optionally: Max MTU, Max Rate, Packet Lifetime and Path Bits.

Path Bits are not implemented in OFED.

IV. Matching Rules

A list of rules that match an incoming PR/MPR request to a QoS-Level. The rules are processed in order such as the first match is applied. Each rule is built out of a set of match expressions which should all match for the rule to apply. The matching expressions are defined for the following fields:

- SRC and DST to lists of port groups
- Service-ID to a list of Service-ID values or ranges
- QoS-Class to a list of QoS-Class values or ranges

4.4.4 CMA Features

The CMA interface supports Service-ID through the notion of port space as a prefix to the port number, which is part of the sockaddr provided to rdma_resolve_add(). The CMA also allows the ULP (like SDP) to propagate a request for a specific QoS-Class. The CMA uses the provided QoS-Class and Service-ID in the sent PR/MPR.

4.4.4.1 IPoIB

IPoIB queries the SA for its broadcast group information and uses the SL, MTU, RATE and Packet Lifetime available on the multicast group which forms this broadcast group.

4.4.4.2 SRP

The current SRP implementation uses its own CM callbacks (not CMA). So SRP fills in the Service-ID in the PR/MPR by itself and use that information in setting up the QP.

SRP Service-ID is defined by the SRP target I/O Controller (it also complies with IBTA Service-ID rules). The Service-ID is reported by the I/O Controller in the ServiceEntries DMA attribute and should be used in the PR/MPR if the SA reports its ability to handle QoS PR/MPRs.

4.4.5 OpenSM Features

The QoS related functionality that is provided by OpenSM—the Subnet Manager described in Chapter 8 can be split into two main parts:

I. Fabric Setup

During fabric initialization, the Subnet Manager parses the policy and apply its settings to the discovered fabric elements.

II. PR/MPR Query Handling

OpenSM enforces the provided policy on client request. The overall flow for such requests is: first the request is matched against the defined match rules such that the target QoS-Level definition is found. Given the QoS-Level a path(s) search is performed with the given restrictions imposed by that level.

4.5 Quality of Service Ethernet

4.5.1 Quality of Service Overview

Quality of Service (QoS) is a mechanism of assigning a priority to a network flow (socket, rdma_cm connection) and manage its guarantees, limitations and its priority over other flows. This is accomplished by mapping the user's priority to a hardware TC (traffic class) through a 2/3 stages process. The TC is assigned with the QoS attributes and the different flows behave accordingly

4.5.2 Mapping Traffic to Traffic Classes

Mapping traffic to TCs consists of several actions which are user controllable, some controlled by the application itself and others by the system/network administrators.

The following is the general mapping traffic to Traffic Classes flow:

1. The application sets the required Type of Service (ToS).
2. The ToS is translated into a Socket Priority (`sk_prio`).
3. The `sk_prio` is mapped to a User Priority (UP) by the system administrator (some applications set `sk_prio` directly).
4. The UP is mapped to TC by the network/system administrator.
5. TCs hold the actual QoS parameters

QoS can be applied on the following types of traffic. However, the general QoS flow may vary among them:

- **Plain Ethernet** - Applications use regular inet sockets and the traffic passes via the kernel Ethernet driver
- **RoCE** - Applications use the RDMA API to transmit using QPs
- **Raw Ethernet QP** - Application use VERBs API to transmit using a Raw Ethernet QP

4.5.3 Plain Ethernet Quality of Service Mapping

Applications use regular inet sockets and the traffic passes via the kernel Ethernet driver.

The following is the Plain Ethernet QoS mapping flow:

1. The application sets the ToS of the socket using `setsockopt` (`IP_TOS`, value).
2. ToS is translated into the `sk_prio` using a fixed translation:

```
TOS 0 <=> sk_prio 0
TOS 8 <=> sk_prio 2
TOS 24 <=> sk_prio 4
TOS 16 <=> sk_prio 6
```

3. The Socket Priority is mapped to the UP:
   - If the underlying device is a VLAN device, `egress_map` is used controlled by the `vconfig` command. This is per VLAN mapping.
   - If the underlying device is not a VLAN device, the `tc` command is used. In this case, even though `tc` manual states that the mapping is from the `sk_prio` to the TC number, the `mlx4_en` driver interprets this as a `sk_prio` to UP mapping.
     Mapping the sk_prio to the UP is done by using `tc_wrap.py -i <dev name> -u 0,1,2,3,4,5,6,7`
4. The the UP is mapped to the TC as configured by the `mlnx_qos` tool or by the `lldpad` daemon if DCBX is used.

Socket applications can use `setsockopt` (`SK_PRIO`, value) to directly set the `sk_prio` of the socket. In this case the ToS to `sk_prio` fixed mapping is not needed. This allows the application and the administrator to utilize more than the 4 values possible via ToS.

In case of VLAN interface, the UP obtained according to the above mapping is also used in the VLAN tag of the traffic

4.5.4 RoCE Quality of Service Mapping

Applications use RDMA-CM API to create and use QPs.

The following is the RoCE QoS mapping flow:

1. The application sets the ToS of the QP using the `rdma_set_option` option (`RDMA_OPTION_ID_TOS`, value).
2. ToS is translated into the Socket Priority (`sk_prio`) using a fixed translation:

```
TOS 0 <=> sk_prio 0
TOS 8 <=> sk_prio 2
TOS 24 <=> sk_prio 4
TOS 16 <=> sk_prio 6
```

3. The Socket Priority is mapped to the User Priority (UP) using the `tc` command.

   In case of a VLAN device, the parent real device is used for the purpose of this mapping.

4. The the UP is mapped to the TC as configured by the `mlnx_qos` tool or by the `lldpad` daemon if DCBX is used.

With RoCE, there can only be 4 predefined ToS values for the purpose of QoS mapping.

4.5.5 Raw Ethernet QP Quality of Service Mapping

Applications open a Raw Ethernet QP using VERBs directly.

The following is the RoCE QoS mapping flow:

1. The application sets the UP of the Raw Ethernet QP during the INIT to RTR state transition of the QP:
   - Sets `qp_attrs.ah_attrs.sl = up`
   - Calls `modify_qp` with `IB_QP_AV` set in the mask
2. The UP is mapped to the TC as configured by the `mlnx_qos` tool or by the `lldpad` daemon if DCBX is used

When using Raw Ethernet QP mapping, the TOS/sk_prio to UP mapping is lost.

Performing the Raw Ethernet QP mapping forces the QP to transmit using the given UP. If packets with VLAN tag are transmitted, UP in the VLAN tag will be overwritten with the given UP.

4.5.6 Map Priorities with tc_wrap.py/mlnx_qos

Network flow that can be managed by QoS attributes is described by a User Priority (UP). A user's `sk_prio` is mapped to UP which in turn is mapped into TC.

- Indicating the UP
  - When the user uses `sk_prio`, it is mapped into a UP by the `'tc'` tool. This is done by the `tc_wrap.py` tool which gets a list of <= 16 comma separated UP and maps the `sk_prio` to the specified UP.
    For example, `tc_wrap.py -ieth0 -u 1,5` maps `sk_prio 0` of `eth0` device to UP 1 and `sk_prio 1` to UP 5.
  - Setting `set_egress_map` in VLAN, maps the `skb_priority` of the VLAN to a `vlan_qos`. The `vlan_qos` is represents a UP for the VLAN device.
  - In RoCE, `rdma_set_option` with `RDMA_OPTION_ID_TOS` could be used to set the UP
  - When creating QPs, the `sl` field in `ibv_modify_qp` command represents the UP
- Indicating the TC

- After mapping the `skb_priority` to UP, one should map the UP into a TC. This assigns the user priority to a specific hardware traffic class. In order to do that, `mlnx_qos` should be used. `mlnx_qos` gets a list of a mapping between UPs to TCs. For example, `mlnx_qos -ieth0 -p 0,0,0,0,1,1,1,1` maps UPs 0-3 to `TC0`, and Ups 4-7 to `TC1`.

4.5.7 Quality of Service Properties

The different QoS properties that can be assigned to a TC are:

- Strict Priority (see "Strict Priority")
- Minimal Bandwidth Guarantee (ETS) (see "Minimal Bandwidth Guarantee (ETS)")
- Rate Limit (see "Rate Limit")

4.5.7.1 Strict Priority

When setting a TC's transmission algorithm to be 'strict', then this TC has absolute (strict) priority over other TC strict priorities coming before it (as determined by the TC number: TC 7 is highest priority, TC 0 is lowest). It also has an absolute priority over non strict TCs (ETS).

This property needs to be used with care, as it may easily cause starvation of other TCs.

A higher strict priority TC is always given the first chance to transmit. Only if the highest strict priority TC has nothing more to transmit, will the next highest TC be considered.

Non strict priority TCs will be considered last to transmit.

This property is extremely useful for low latency low bandwidth traffic. Traffic that needs to get immediate service when it exists, but is not of high volume to starve other transmitters in the system.

4.5.7.2 Minimal Bandwidth Guarantee (ETS)

After servicing the strict priority TCs, the amount of bandwidth (BW) left on the wire may be split among other TCs according to a minimal guarantee policy.

If, for instance, TC0 is set to 80% guarantee and TC1 to 20% (the TCs sum must be 100), then the BW left after servicing all strict priority TCs will be split according to this ratio.

Since this is a minimal guarantee, there is no maximum enforcement. This means, in the same example, that if TC1 did not use its share of 20%, the reminder will be used by TC0.

4.5.7.3 Rate Limit

Rate limit defines a maximum bandwidth allowed for a TC. Please note that 10% deviation from the requested values is considered acceptable.

4.5.8 Quality of Service Tools

4.5.8.1 mlnx_qos

`mlnx_qos` is a centralized tool used to configure QoS features of the local host. It communicates directly with the driver thus does not require setting up a DCBX daemon on the system.

The `mlnx_qos` tool enables the administrator of the system to:

- Inspect the current QoS mappings and configuration

The tool will also display maps configured by TC and `vconfig set_egress_map` tools, in order to give a centralized view of all QoS mappings.

- Set UP to TC mapping
- Assign a transmission algorithm to each TC (strict or ETS)
- Set minimal BW guarantee to ETS TCs
- Set rate limit to TCs

For unlimited ratelimit set the ratelimit to 0.

Usage:

```
mlnx_qos -i <interface> [options]
```

Options:

```
  --version             show program's version number and exit
  -h, --help            show this help message and exit
  -p LIST, --prio_tc=LIST
                        maps UPs to TCs. LIST is 8 comma seperated TC numbers.
                        Example: 0,0,0,0,1,1,1,1 maps UPs 0-3 to TC0, and UPs
                        4-7 to TC1
  -s LIST, --tsa=LIST   Transmission algorithm for each TC. LIST is comma
                        seperated algorithm names for each TC. Possible
                        algorithms: strict, etc. Example: ets,strict,ets sets
                        TC0,TC2 to ETS and TC1 to strict. The rest are
                        unchanged.
  -t LIST, --tcbw=LIST  Set minimal guaranteed %BW for ETS TCs. LIST is comma
                        seperated percents for each TC. Values set to TCs that
                        are not configured to ETS algorithm are ignored, but
                        must be present. Example: if TC0,TC2 are set to ETS,
                        then 10,0,90 will set TC0 to 10% and TC2 to 90%.
                        Percents must sum to 100.
  -r LIST, --ratelimit=LIST
                        Rate limit for TCs (in Gbps). LIST is a comma
                        seperated Gbps limit for each TC. Example: 1,8,8 will
                        limit TC0 to 1Gbps, and TC1,TC2 to 8 Gbps each.
  -i INTF, --interface=INTF
                        Interface name
  -a                    Show all interface's TCs
```

Get Current Configuration:

```
tc: 0 ratelimit: unlimited, tsa: strict
         up:  0
                 skprio: 0
                 skprio: 1
                 skprio: 2 (tos: 8)
                 skprio: 3
                 skprio: 4 (tos: 24)
                 skprio: 5
                 skprio: 6 (tos: 16)
                 skprio: 7
                 skprio: 8
                 skprio: 9
                 skprio: 10
                 skprio: 11
                 skprio: 12
                 skprio: 13
                 skprio: 14
                 skprio: 15
         up:  1
         up:  2
         up:  3
         up:  4
         up:  5
         up:  6
         up:  7
```

Set ratelimit. 3Gbps for tc0 4Gbps for tc1 and 2Gbps for tc2:

```
tc: 0 ratelimit: 3 Gbps, tsa: strict
         up:  0
                 skprio: 0
                 skprio: 1
                 skprio: 2 (tos: 8)
                 skprio: 3
                 skprio: 4 (tos: 24)
                 skprio: 5
                 skprio: 6 (tos: 16)
                 skprio: 7
                 skprio: 8
                 skprio: 9
                 skprio: 10
                 skprio: 11
                 skprio: 12
                 skprio: 13
                 skprio: 14
                 skprio: 15
         up:  1
         up:  2
         up:  3
         up:  4
         up:  5
         up:  6
         up:  7
```

Configure QoS. map UP 0,7 to tc0, 1,2,3 to tc1 and 4,5,6 to tc 2. set tc0,tc1 as ets and tc2 as strict. divide ets 30% for tc0 and 70% for tc1:

```
mlnx_qos -i eth3 -s ets,ets,strict -p 0,1,1,1,2,2,2 -t 30,70
tc: 0 ratelimit: 3 Gbps, tsa: ets, bw: 30%
         up:  0
                 skprio: 0
                 skprio: 1
                 skprio: 2 (tos: 8)
                 skprio: 3
                 skprio: 4 (tos: 24)
                 skprio: 5
                 skprio: 6 (tos: 16)
                 skprio: 7
                 skprio: 8
                 skprio: 9
                 skprio: 10
                 skprio: 11
                 skprio: 12
                 skprio: 13
                 skprio: 14
                 skprio: 15
  up:  7
tc: 1 ratelimit: 4 Gbps, tsa: ets, bw: 70%
```

```
         up:  1
         up:  2
         up:  3
tc: 2 ratelimit: 2 Gbps, tsa: strict
         up:  4
         up:  5
         up:  6
```

4.5.8.2 tc and tc_wrap.py

The 'tc' tool is used to setup `sk_prio` to UP mapping, using the `mqprio` queue discipline.

In kernels that do not support `mqprio` (such as 2.6.34), an alternate mapping is created in `sysfs`. The 'tc_wrap.py' tool will use either the `sysfs` or the 'tc' tool to configure the `sk_prio` to UP mapping.

Usage:

```
tc_wrap.py -i <interface> [options]
```

Options:

```
--version                           show program's version number and exit
-h, --help                          show this help message and exit
-u SKPRIO_UP, --skprio_up=SKPRIO_UP maps sk_prio to UP. LIST is <=16 comma separated
                                    UP. index of element is sk_prio.
-i INTF, --interface=INTF           Interface name
```

Example: set skprio 0-2 to UP0, and skprio 3-7 to UP1 on eth4

```
UP  0
        skprio: 0
        skprio: 1
        skprio: 2 (tos: 8)
        skprio: 7
        skprio: 8
        skprio: 9
        skprio: 10
        skprio: 11
        skprio: 12
        skprio: 13
        skprio: 14
        skprio: 15
UP  1
        skprio: 3
        skprio: 4 (tos: 24)
        skprio: 5
        skprio: 6 (tos: 16)
```

```
UP  2
UP  3
UP  4
UP  5
UP  6
UP  7
```

4.5.8.3 Additional Tools

tc tool compiled with the `sch_mqprio` module is required to support kernel v2.6.32 or higher. This is a part of `iproute2` package v2.6.32-19 or higher. Otherwise, an alternative custom sysfs interface is available.

- `mlnx_qos tool` (package: ofed-scripts) requires python >= 2.5
- `tc_wrap.py` (package: ofed-scripts) requires python >= 2.5

4.6 Ethernet Time-Stamping

4.6.1 Ethernet Time-Stamping Service

Time Stamping is currently supported in ConnectX®-3/ConnectX®-3 Pro adapter cards only.

Time stamping is the process of keeping track of the creation of a packet. A time-stamping service supports assertions of proof that a datum existed before a particular time. Incoming packets are time-stamped before they are distributed on the PCI depending on the congestion in the PCI buffers. Outgoing packets are time-stamped very close to placing them on the wire.

4.6.1.1 Enabling Time Stamping

Time-stamping is off by default and should be enabled before use.

➢ ***To enable time stamping for a socket:***

- Call setsockopt() with SO_TIMESTAMPING and with the following flags:

```
SOF_TIMESTAMPING_TX_HARDWARE:  try to obtain send time stamp in hardware
SOF_TIMESTAMPING_TX_SOFTWARE:  if SOF_TIMESTAMPING_TX_HARDWARE is off or
                               fails, then do it in software
SOF_TIMESTAMPING_RX_HARDWARE:  return the original, unmodified time stamp
                               as generated by the hardware
SOF_TIMESTAMPING_RX_SOFTWARE:  if SOF_TIMESTAMPING_RX_HARDWARE is off or
                               fails, then do it in software
SOF_TIMESTAMPING_RAW_HARDWARE: return original raw hardware time stamp
SOF_TIMESTAMPING_SYS_HARDWARE: return hardware time stamp transformed to
                               the system time base
SOF_TIMESTAMPING_SOFTWARE:     return system time stamp generated in
                               software
SOF_TIMESTAMPING_TX/RX determine how time stamps are generated.
SOF_TIMESTAMPING_RAW/SYS determine how they are reported
```

- ***To enable time stamping for a net device:***

  Admin privileged user can enable/disable time stamping through calling ioctl(sock, SIOCSHWTSTAMP, &ifreq) with following values:

  Send side time sampling:

  - Enabled by ifreq.hwtstamp_config.tx_type when

```
/* possible values for hwtstamp_config->tx_type */
enum hwtstamp_tx_types {
        /*
         * No outgoing packet will need hardware time stamping;
         * should a packet arrive which asks for it, no hardware
         * time stamping will be done.
        */
        HWTSTAMP_TX_OFF,

        /*
         * Enables hardware time stamping for outgoing packets;
         * the sender of the packet decides which are to be
         * time stamped by setting %SOF_TIMESTAMPING_TX_SOFTWARE
         * before sending the packet.
         */
        HWTSTAMP_TX_ON,
  /*
         * Enables time stamping for outgoing packets just as
         * HWTSTAMP_TX_ON does, but also enables time stamp insertion
         * directly into Sync packets. In this case, transmitted Sync
         * packets will not received a time stamp via the socket error
         * queue.
         */
        HWTSTAMP_TX_ONESTEP_SYNC,
};
Note: for send side time stamping currently only HWTSTAMP_TX_OFF and
HWTSTAMP_TX_ON are supported.
```

Receive side time sampling:

- Enabled by ifreq.hwtstamp_config.rx_filter when

```
/* possible values for hwtstamp_config->rx_filter */
enum hwtstamp_rx_filters {
        /* time stamp no incoming packet at all */
        HWTSTAMP_FILTER_NONE,

        /* time stamp any incoming packet */
        HWTSTAMP_FILTER_ALL,
      /* return value: time stamp all packets requested plus some others */
        HWTSTAMP_FILTER_SOME,

        /* PTP v1, UDP, any kind of event packet */
        HWTSTAMP_FILTER_PTP_V1_L4_EVENT,
        /* PTP v1, UDP, Sync packet */
        HWTSTAMP_FILTER_PTP_V1_L4_SYNC,
        /* PTP v1, UDP, Delay_req packet */
        HWTSTAMP_FILTER_PTP_V1_L4_DELAY_REQ,
        /* PTP v2, UDP, any kind of event packet */
        HWTSTAMP_FILTER_PTP_V2_L4_EVENT,
        /* PTP v2, UDP, Sync packet */
        HWTSTAMP_FILTER_PTP_V2_L4_SYNC,
        /* PTP v2, UDP, Delay_req packet */
        HWTSTAMP_FILTER_PTP_V2_L4_DELAY_REQ,

        /* 802.AS1, Ethernet, any kind of event packet */
        HWTSTAMP_FILTER_PTP_V2_L2_EVENT,
        /* 802.AS1, Ethernet, Sync packet */
        HWTSTAMP_FILTER_PTP_V2_L2_SYNC,
        /* 802.AS1, Ethernet, Delay_req packet */
        HWTSTAMP_FILTER_PTP_V2_L2_DELAY_REQ,

        /* PTP v2/802.AS1, any layer, any kind of event packet */
        HWTSTAMP_FILTER_PTP_V2_EVENT,
        /* PTP v2/802.AS1, any layer, Sync packet */
        HWTSTAMP_FILTER_PTP_V2_SYNC,
        /* PTP v2/802.AS1, any layer, Delay_req packet */
        HWTSTAMP_FILTER_PTP_V2_DELAY_REQ,
};
Note: for receive side time stamping currently only HWTSTAMP_FILTER_NONE and
HWTSTAMP_FILTER_ALL are supported.
```

4.6.1.2 Getting Time Stamping

Once time stamping is enabled time stamp is placed in the socket Ancillary data. recvmsg() can be used to get this control message for regular incoming packets. For send time stamps the outgoing packet is looped back to the socket's error queue with the send time stamp(s) attached. It can be received with recvmsg(flags=MSG_ERRQUEUE). The call returns the original outgoing packet data including all headers preprended down to and including the link layer, the scm_timestamping control message and a sock_extended_err control message with ee_errno==ENOMSG and ee_origin==SO_EE_ORIGIN_TIMESTAMPING. A socket with such

a pending bounced packet is ready for reading as far as select() is concerned. If the outgoing packet has to be fragmented, then only the first fragment is time stamped and returned to the sending socket.

When time-stamping is enabled, VLAN stripping is disabled. For more info please refer to Documentation/networking/timestamping.txt in kernel.org

4.6.1.3 Querying Time Stamping Capabilities via ethtool

➢ ***To display Time Stamping capabilities via ethtool:***

- Show Time Stamping capabilities

```
ethtool -T eth<x>
```

Example:

```
ethtool -T eth0
Time stamping parameters for p2p1:
Capabilities:
        hardware-transmit     (SOF_TIMESTAMPING_TX_HARDWARE)
        software-transmit     (SOF_TIMESTAMPING_TX_SOFTWARE)
        hardware-receive      (SOF_TIMESTAMPING_RX_HARDWARE)
        software-receive      (SOF_TIMESTAMPING_RX_SOFTWARE)
        software-system-clock (SOF_TIMESTAMPING_SOFTWARE)
        hardware-raw-clock    (SOF_TIMESTAMPING_RAW_HARDWARE)
PTP Hardware Clock: none
Hardware Transmit Timestamp Modes:
        off                   (HWTSTAMP_TX_OFF)
        on                    (HWTSTAMP_TX_ON)
Hardware Receive Filter Modes:
        none                  (HWTSTAMP_FILTER_NONE)
        all                   (HWTSTAMP_FILTER_ALL)
```

4.6.2 RoCE Time Stamping

RoCE Time Stamping is currently at beta level.
Please be aware that everything listed here is subject to change.

RoCE Time Stamping allows you to stamp packets when they are sent to the wire / received from the wire. The time stamp is given in a raw hardware cycles, but could be easily converted into hardware referenced nanoseconds based time. Additionally, it enables you to query the hardware for the hardware time, thus stamp other application's event and compare time.

4.6.2.1 Query Capabilities

Time stamping is available if and only the hardware reports it is capable of reporting it. To verify whether RoCE Time Stamping is available, run i`bv_ex_query_device`.

For example:

```
struct ibv_exp_device_attr attr;
ibv_exp_query_device(context, &attr);
if (attr.comp_mask & IBV_EXP_DEVICE_ATTR_WITH_TIMESTAMP_MASK) {
if (attr.timestamp_mask) {
                /* Time stamping is supported with mask attr.timestamp_mask */
        }
}
if (attr.comp_mask & IBV_EXP_DEVICE_ATTR_WITH_HCA_CORE_CLOCK) {
        if (attr.hca_core_clock) {
                /* reporting the device's clock is supported. */
                /* attr.hca_core_clock is the frequency in MHZ */
        }
}
```

4.6.2.2 Creating Time Stamping Completion Queue

To get time stamps, a suitable extended Completion Queue (CQ) must be created via a special call to `ibv_create_cq_ex` verb.

```
cq_init_attr.flags = IBV_CQ_TIMESTAMP;
cq_init_attr.comp_mask = IBV_CQ_INIT_ATTR_FLAGS;
cq = ibv_create_cq_ex(context, cqe, node, NULL, 0, &cq_init_attr);
```

This CQ cannot report SL or SLID information. The value of `sl` and `sl_id` fields in `struct ibv_wc_ex` are invalid. Only the fields indicated by the `wc_flags` field in `struct ibv_wc_ex` contains a valid and usable value.

When using Time Stamping, several fields of `struct ibv_wc_ex` are not available resulting in RoCE UD / RoCE traffic with VLANs failure.

4.6.2.3 Polling a Completion Queue

Polling a CQ for time stamp is done via the `ibv_poll_cq_ex` verb.

```
ret = ibv_poll_cq_ex(cq, 1, &wc_ex, sizeof(wc_ex));
if (ret > 0) {
                /* CQ returned a wc */
                if (wc_ex.wc_flags & IBV_WC_WITH_TIMESTAMP) {
                                /* This wc contains a timestamp */
                                timestamp = wc_ex.timestamp;
                                /* Timestamp is given in raw hardware time */
                }
}
```

CQs that are opened with the `ibv_create_cq_ex` versb should be always be polled with the `ibv_poll_cq_ex` verb.

4.6.2.4 Querying the Hardware Time

Querying the hardware for time is done via the `ibv_query_values_ex` verb.

For example:

```
ret = ibv_query_values_ex(context, IBV_VALUES_HW_CLOCK, &queried_values);
if (!ret && queried_values.comp_mask & IBV_VALUES_HW_CLOCK)
queried_time = queried_values.hwclock;
```

To change the queried time in nanoseconds resolution, use the `IBV_VALUES_HW_CLOCK_NS` flag along with the `hwclock_ns` field.

```
ret = ibv_query_values_ex(context, IBV_VALUES_HW_CLOCK_NS, &queried_values);
if (!ret && queried_values.comp_mask & IBV_VALUES_HW_CLOCK_NS)
queried_time_ns = queried_values.hwclock_ns;
```

Querying the Hardware Time is available only on physical functions / native machines.

4.7 Atomic Operations

Atomic Operations are applicable to the mlx4 driver only.

4.7.1 Enhanced Atomic Operations

ConnectX® implements a set of Extended Atomic Operations beyond those defined by the IB spec. Atomicity guarantees, Atomic Ack generation, ordering rules and error behavior for this set of extended Atomic operations is the same as that for IB standard Atomic operations (as defined in section 9.4.5 of the IB spec).

4.7.1.1 Masked Compare and Swap (MskCmpSwap)

The MskCmpSwap atomic operation is an extension to the CmpSwap operation defined in the IB spec. MskCmpSwap allows the user to select a portion of the 64 bit target data for the "compare" check as well as to restrict the swap to a (possibly different) portion. The pseudocode below describes the operation:

```
| atomic_response = *va
| if (!((compare_add ^ *va) & compare_add_mask)) then
```

```
    *va = (*va & ~(swap_mask)) | (swap & swap_mask)

return atomic_response
```

The additional operands are carried in the Extended Transport Header. Atomic response generation and packet format for MskCmpSwap is as for standard IB Atomic operations.

4.7.1.2 Masked Fetch and Add (MFetchAdd)

The MFetchAdd Atomic operation extends the functionality of the standard IB FetchAdd by allowing the user to split the target into multiple fields of selectable length. The atomic add is done independently on each one of this fields. A bit set in the field_boundary parameter specifies the field boundaries. The pseudocode below describes the operation:

```
bit_adder(ci, b1, b2, *co)
{
  value = ci + b1 + b2
  *co = !!(value & 2)

  return value & 1
}

#define MASK_IS_SET(mask, attr)      (!!((mask)&(attr)))
bit_position = 1
carry = 0
atomic_response = 0

for i = 0 to 63
{
        if ( i != 0 )
                bit_position =  bit_position << 1

        bit_add_res = bit_adder(carry, MASK_IS_SET(*va, bit_position),
                                MASK_IS_SET(compare_add, bit_position), &new_carry)
        if (bit_add_res)
                atomic_response |= bit_position

        carry = ((new_carry) && (!MASK_IS_SET(compare_add_mask, bit_position)))
}

return atomic_response
```

4.8 Ethernet Tunneling Over IPoIB Driver (eIPoIB)

The eth_ipoib driver provides a standard Ethernet interface to be used as a Physical Interface (PIF) into the Hypervisor virtual network, and serves one or more Virtual Interfaces (VIF). This driver supports L2 Switching (Direct Bridging) as well as other L3 Switching modes (e.g. NAT). This document explains the configuration and driver behavior when configured in Bridging mode.

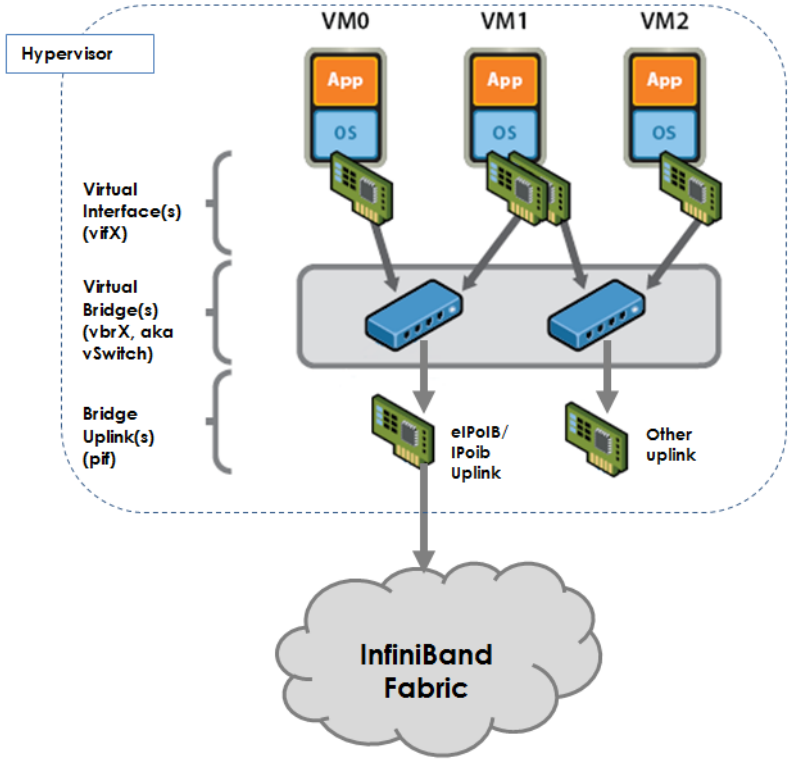
In virtualization environment, a virtual machine can be expose to the physical network by performing the next setting:

Step 1. Create a virtual bridge

Step 2. Attach the para-virtualized interface created by the eth_ipoib driver to the bridge

Step 3. Attach the Ethernet interface in the Virtual Machine to that bridge

The diagram below describes the topology that was created after these steps:



The diagram shows how the traffic from the Virtual Machine goes to the virtual-bridge in the Hypervisor and from the bridge to the eIPoIB interface. eIPoIB interface is the Ethernet interface that enslaves the IPoIB interfaces in order to send/receive packets from the Ethernet interface in the Virtual Machine to the IB fabric beneath.

4.8.1 Enabling the eIPoIB Driver

Once the mlnx_ofed driver installation is completed, perform the following:

Step 1. Open the `/etc/infiniband/openib.conf` file and include:

```
E_IPOIB_LOAD=yes
```

Step 2. Restart the InfiniBand drivers.

```
/etc/init.d/openibd restart
```

4.8.2 Configuring the Ethernet Tunneling Over IPoIB Driver

When eth_ipoib is loaded, number of eIPoIB interfaces are created, with the following default naming scheme: `ethX`, where X represents the ETH port available on the system.

To check which eIPoIB interfaces were created:

```
cat /sys/class/net/eth_ipoib_interfaces
```

For example, on a system with dual port HCA, the following two interfaces might be created; `eth4` and `eth5`.

```
cat /sys/class/net/eth_ipoib_interfaces
eth4 over IB port: ib0
eth5 over IB port: ib1
```

These interfaces can be used to configure the network for the guest. For example, if the guest has a VIF that is connected to the Virtual Bridge `br0`, then enslave the eIPoIB interface to `br0` by running:

```
# brctl addif br0 ethX
```

In RHEL KVM environment, there are other methods to create/configure your virtual network (e.g. macvtap). For additional information, please refer to the Red Hat User Manual.

The IPoIB daemon (ipoibd) detects the new virtual interface that is attached to the same bridge as the eIPoIB interface and creates a new IPoIB instances for it in order to send/receive data. As a result, number of IPoIB interfaces (ibX.Y) are shown as being created/destroyed, and are being enslaved to the corresponding `ethX` interface to serve any active VIF in the system according to the set configuration, This process is done automatically by the `ipoibd` service.

- ***To see the list of IPoIB interfaces enslaved under eth_ipoib interface.***

```
# cat /sys/class/net/ethX/eth/vifs
```

For example:

```
# cat /sys/class/net/eth5/eth/vifs
  SLAVE=ib0.1      MAC=9a:c2:1f:d7:3b:63 VLAN=N/A
  SLAVE=ib0.2      MAC=52:54:00:60:55:88 VLAN=N/A
  SLAVE=ib0.3      MAC=52:54:00:60:55:89 VLAN=N/A
```

Each ethX interface has at lease one ibX.Y slave to serve the PIF itself. In the VIFs list of ethX you will notice that ibX.1 is always created to serve applications running from the Hypervisor on top of the ethX interface directly.

For InfiniBand applications that require native IPoIB interfaces (e.g. CMA), the original IPoIB interfaces ibX can still be used. For example, CMA and ethX drivers can co-exist and make use of IPoIB ports; CMA can use ib0, while eth0.ipoib interface will use ibX.Y interfaces.

- ***To see the list of eIPoIB interfaces.***

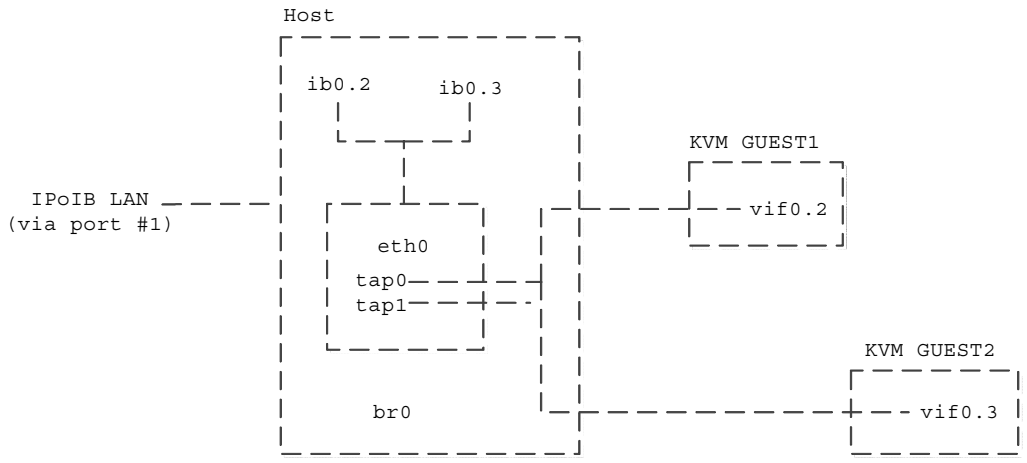
```
# cat /sys/class/net/eth_ipoib_interfaces
```

For example:

```
# cat /sys/class/net/eth_ipoib_interfaces
  eth4 over IB port: ib0
  eth5 over IB port: ib1
```

The example above shows, two eIPoIB interfaces, where eth4 runs traffic over ib0, and eth5 runs traffic over ib1.

Figure 3: An Example of a Virtual Network



The example above shows a few IPoIB instances that server the virtual interfaces at the Virtual Machines.

To display the services provided to the Virtual Machine interfaces:

```
# cat /sys/class/net/eth0/eth/vifs
```

Example:

```
# cat /sys/class/net/eth0/eth/vifs
  SLAVE=ib0.2 MAC=52:54:00:60:55:88 VLAN=N/A
```

In the example above the ib0.2 IPoIB interface serves the MAC 52:54:00:60:55:88 with no VLAN tag for that interface.

4.8.3 VLAN Configuration Over an eIPoIB Interface

eIPoIB driver supports VLAN Switch Tagging (VST) mode, which enables the virtual machine interface to have no VLAN tag over it, thus allowing VLAN tagging to be handled by the Hypervisor.

➢ ***To attach a Virtual Machine interface to a specific isolated tag:***

Step 1. Verify the VLAN tag to be used has the same pkey value that is already configured on that ib port.

```
# cat /sys/class/infiniband/mlx4_0/ports/<ib port>/pkeys/*
```

Step 2. Create a VLAN interface in the Hypervisor, over the eIPoIB interface.

```
# vconfig add <eIPoIB interface> <vlan tag>
```

Step 3. Attach the new VLAN interface to the same bridge that the virtual machine interface is already attached to.

```
# brctl addif <br-name> <interface-name>
```

For example, to create the VLAN tag 3 with pkey 0x8003 over that port in the eIPoIB interface eth4, run:

```
#vconfig add eth4 3
#brctl addif br2 eth4.3
```

4.8.4 Setting Performance Tuning

- Use 4K MTU over OpenSM. For further information, please refer to Section 8.4.1, "File Format", on page 149

  ```
  Default=0xffff, ipoib, mtu=5 : ALL=full;
  ```

- Use MTU for 4K (4092 Bytes):
  - In UD mode, the maximum MTU value is 4092 Bytes

  Make sure that all interfaces (including the guest interface and its virtual bridge) have the same MTU value (MTU 4092 Bytes). For further information of MTU settings, please refer to the Hypervisor User Manual.

- Tune the TCP/IP stack using sysctl (dom0/domu)

  ```
  # /sbin/sysctl_perf_tuning
  ```

- Other performance tuning for KVM environment such as vCPU pinning and NUMA tuning may apply. For further information, please refer to the Hypervisor User Manual.

4.9 Contiguous Pages

Contiguous Pages improves performance by allocating user memory regions over physical contiguous pages. It enables a user application to ask low level drivers to allocate contiguous memory for it as part of `ibv_reg_mr`.

Additional performance improvements can be reached by allocating Queue Pair (QP) and Completion Queue (CQ|) buffers to the Contiguous Pages.

To activate set the below environment variables with values of PREFER_CONTIG or CONTIG.

- For QP: MLX_QP_ALLOC_TYPE
- For CQ: MLX_CQ_ALLOC_TYPE

The following are all the possible values that can be allocated to the buffer:

Table 3 - Buffer Values

Possible Value[1]	Description
ANON	Use current pages ANON small ones. Default value.
HUGE	Force huge pages.
CONTIG	Force contiguous pages.
PREFER_CONTIG	Try contiguous fallback to ANON small pages.
PREFER_HUGE	Try huge fallback to ANON small pages.

Table 3 - Buffer Values

Possible Value[1]	Description
ALL	Try huge fallback to contiguous if failed fallback to ANON small pages.

1. Values are NOT case sensitive.

Usage:

The application calls the `ibv_reg_mr` API which turns on the `IBV_ACCESS_ALLOCATE_MR` bit and sets the input address to NULL. Upon success, the address field of the struct `ibv_mr` will hold the address to the allocated memory block. This block will be freed implicitly when the `ibv_dereg_mr()` is called.

The following are environment variables that can be used to control error cases / contiguity:

Table 4 - Parameters Used to Control Error Cases / Contiguity

Parameters	Description
MLX_MR_ALLOC_TYPE	Configures the allocator type. • ALL (Default) - Uses all possible allocator and selects most efficient allocator. • ANON - Enables the usage of anonymous pages and disables the allocator • CONTIG - Forces the usage of the contiguous pages allocator. If contiguous pages are not available the allocation fails
MLX_MR_MAX_LOG2_CONTIG_BSIZE	Sets the maximum contiguous block size order. • Values: 12-23 • Default: 23
MLX_MR_MIN_LOG2_CONTIG_BSIZE	Sets the minimum contiguous block size order. • Values: 12-23 • Default: 12

4.10 Shared Memory Region

Shared Memory Region is only applicable to the mlx4 driver.

Shared Memory Region (MR) enables sharing MR among applications by implementing the "Register Shared MR" verb which is part of the IB spec.

Sharing MR involves the following steps:

Step 1. Request to create a shared MR

The application sends a request via the `ibv_reg_mr` API to create a shared MR. The application supplies the allowed sharing access to that MR. If the MR was created successfully, a unique MR ID is returned as part of the struct `ibv_mr` which can be used by other applications to register with that MR.

The underlying physical pages must not be Least Recently Used (LRU) or Anonymous. To disable that, you need to turn on the `IBV_ACCESS_ALLOCATE_MR` bit as part of the sharing bits.

Usage:

- Turns on via the `ibv_reg_mr` one or more of the sharing access bits. The sharing bits are part of the `ibv_reg_mr` man page.
- Turns on the `IBV_ACCESS_ALLOCATE_MR` bit

Step 2. Request to register to a shared MR

A new verb called `ibv_reg_shared_mr` is added to enable sharing an MR. To use this verb, the application supplies the MR ID that it wants to register for and the desired access mode to that MR. The desired access is validated against its given permissions and upon successful creation, the physical pages of the original MR are shared by the new MR. Once the MR is shared, it can be used even if the original MR was destroyed.

The request to share the MR can be repeated multiple times and an arbitrary number of Memory Regions can potentially share the same physical memory locations.

Usage:

- Uses the “handle” field that was returned from the `ibv_reg_mr` as the `mr_handle`
- Supplies the desired “access mode” for that MR
- Supplies the address field which can be either NULL or any hint as the required output. The address and its length are returned as part of the `ibv_mr struct`.

  To achieve high performance it is highly recommended to supply an address that is aligned as the original memory region address. Generally, it may be an alignment to 4M address.

For further information on how to use the `ibv_reg_shared_mr` verb, please refer to the `ibv_reg_shared_mr` man page and/or to the `ibv_shared_mr` sample program which demonstrates a basic usage of this verb.
Further information on the `ibv_shared_mr` sample program can be found in the `ibv_shared_mr` man page.

4.11 XRC - eXtended Reliable Connected Transport Service for InfiniBand

XRC allows significant savings in the number of QPs and the associated memory resources required to establish all to all process connectivity in large clusters.

It significantly improves the scalability of the solution for large clusters of multicore end-nodes by reducing the required resources.

For further details, please refer to the "Annex A14 Supplement to InfiniBand Architecture Specification Volume 1.2.1"

A new API can be used by user space applications to work with the XRC transport. The legacy API is currently supported in both binary and source modes, however it is deprecated. Thus we recommend using the new API.

The new verbs to be used are:

- `ibv_open_xrcd/ibv_close_xrcd`
- `ibv_create_srq_ex`
- `ibv_get_srq_num`
- `ibv_create_qp_ex`

- `ibv_open_qp`

Please use ibv_xsrq_pingpong for basic tests and code reference. For detailed information regarding the various options for these verbs, please refer to their appropriate man pages.

4.12 Flow Steering

Flow Steering is applicable to the mlx4 driver only.

Flow steering is a new model which steers network flows based on flow specifications to specific QPs. Those flows can be either unicast or multicast network flows. In order to maintain flexibility, domains and priorities are used. Flow steering uses a methodology of flow attribute, which is a combination of L2-L4 flow specifications, a destination QP and a priority. Flow steering rules may be inserted either by using ethtool or by using InfiniBand verbs. The verbs abstraction uses a different terminology from the flow attribute (ibv_flow_attr), defined by a combination of specifications (struct ibv_flow_spec_*).

4.12.1 Enable/Disable Flow Steering

Flow Steering is disabled by default and regular L2 steering is performed instead (B0 Steering). When using SR-IOV, flow steering is enabled if there is an adequate amount of space to store the flow steering table for the guest/master.

➢ ***To enable Flow Steering:***

Step 1. Open the `/etc/modprobe.d/mlnx.conf` file.

Step 2. Set the parameter `log_num_mgm_entry_size` to `-1` by writing the option `mlx4_core log_num_mgm_entry_size=-1`.

Step 3. Restart the driver

➢ ***To disable Flow Steering:***

Step 1. Open the `/etc/modprobe.d/mlnx.conf` file.

Step 2. Remove the `options mlx4_core log_num_mgm_entry_size= -1`.

Step 3. Restart the driver

4.12.2 Flow Domains and Priorities

Flow steering defines the concept of domain and priority. Each domain represents a user agent that can attach a flow. The domains are prioritized. A higher priority domain will always supersede a lower priority domain when their flow specifications overlap. Setting a lower priority value will result in higher priority.

In addition to the domain, there is priority within each of the domains. Each domain can have at most 2^12 priorities in accordance with its needs.

The following are the domains at a descending order of priority:

- **User Verbs** allows a user application QP to be attached into a specified flow when using `ibv_create_flow` and `ibv_destroy_flow` verbs

- `ibv_create_flow`

```
struct ibv_flow *ibv_create_flow(struct ibv_qp *qp, struct ibv_flow_attr *flow)
```

Input parameters:

- `struct ibv_qp` - the attached QP.
- `struct ibv_flow_attr` - attaches the QP to the flow specified. The flow contains mandatory control parameters and optional L2, L3 and L4 headers. The optional headers are detected by setting the size and `num_of_specs` fields:

  `struct ibv_flow_attr` can be followed by the optional flow headers structs:

```
struct ibv_flow_spec_ib
struct ibv_flow_spec_eth
struct ibv_flow_spec_ipv4
struct ibv_flow_spec_tcp_udp
```

  For further information, please refer to the `ibv_create_flow` man page.

Be advised that from MLNX_OFED v2.0-3.0.0 and higher, the parameters (both the value and the mask) should be set in big-endian format.

  Each header struct holds the relevant network layer parameters for matching. To enforce the match, the user sets a mask for each parameter. The supported masks are:

  - All one mask - include the parameter value in the attached rule
    Note: Since the VLAN ID in the Ethernet header is 12bit long, the following parameter should be used: `flow_spec_eth.mask.vlan_tag = htons(0x0fff).`
  - All zero mask - ignore the parameter value in the attached rule

  When setting the flow type to NORMAL, the incoming traffic will be steered according to the rule specifications. ALL_DEFAULT and MC_DEFAULT rules options are valid only for Ethernet link type since InfiniBand link type packets always include QP number.

  For further information, please refer to the relevant man pages.

- `ibv_destroy_flow`

```
int ibv_destroy_flow(struct ibv_flow *flow_id)
```

Input parameters:

`ibv_destroy_flow` requires `struct ibv_flow` which is the return value of `ibv_create_flow` in case of success.

Output parameters:

Returns 0 on success, or the value of errno on failure.

For further information, please refer to the `ibv_destroy_flow` man page.

- **Ethtool**

Ethtool domain is used to attach an RX ring, specifically its QP to a specified flow.

Please refer to the most recent ethtool manpage for all the ways to specify a flow.

Examples:

- ethtool –U eth5 flow-type ether dst 00:11:22:33:44:55 loc 5 action 2

All packets that contain the above destination MAC address are to be steered into rx-ring 2 (its underlying QP), with priority 5 (within the ethtool domain)

- ethtool –U eth5 flow-type tcp4 src-ip 1.2.3.4 dst-port 8888 loc 5 action 2

  All packets that contain the above destination IP address and source port are to be steered into rx-ring 2. When destination MAC is not given, the user's destination MAC is filled automatically.

- ethtool –u eth5

  Shows all of ethtool's steering rule

When configuring two rules with the same priority, the second rule will overwrite the first one, so this ethtool interface is effectively a table. Inserting Flow Steering rules in the kernel requires support from both the ethtool in the user space and in kernel (v2.6.28).

MLX4 Driver Support

The mlx4 driver supports only a subset of the flow specification the ethtool API defines. Asking for an unsupported flow specification will result with an "invalid value" failure.

The following are the flow specific parameters:

Table 5 - Flow Specific Parameters

	ether	tcp4/udp4	ip4
Mandatory	dst		src-ip/dst-ip
Optional	vlan	src-ip, dst-ip, src-port, dst-port, vlan	src-ip, dst-ip, vlan

- **RFS**

  RFS is an in-kernel-logic responsible for load balancing between CPUs by attaching flows to CPUs that are used by flow's owner applications. This domain allows the RFS mechanism to use the flow steering infrastructure to support the RFS logic by implementing the `ndo_rx_flow_steer`, which, in turn, calls the underlying flow steering mechanism with the RFS domain.

  Enabling the RFS requires enabling the `'ntuple'` flag via the ethtool,

  For example, to enable ntuple for eth0, run:

  ```
  ethtool -K eth0 ntuple on
  ```

  RFS requires the kernel to be compiled with the `CONFIG_RFS_ACCEL` option. This options is available in kernels 2.6.39 and above. Furthermore, RFS requires Device Managed Flow Steering support.

RFS cannot function if LRO is enabled. LRO can be disabled via ethtool.

- **All of the rest**

  The lowest priority domain serves the following users:

  - **The mlx4 Ethernet driver** attaches its unicast and multicast MACs addresses to its QP using L2 flow specifications

- **The mlx4 ipoib driver** when it attaches its QP to his configured GIDS

Fragmented UDP traffic cannot be steered. It is treated as 'other' protocol by hardware (from the first packet) and not considered as UDP traffic.

We recommend using `libibverbs` v2.0-3.0.0 and `libmlx4` v2.0-3.0.0 and higher as of MLNX_OFED v2.0-3.0.0 due to API changes.

4.13 Single Root IO Virtualization (SR-IOV)

Single Root IO Virtualization (SR-IOV) is a technology that allows a physical PCIe device to present itself multiple times through the PCIe bus. This technology enables multiple virtual instances of the device with separate resources. Mellanox adapters are capable of exposing in ConnectX®-3 adapter cards 63 virtual instances called Virtual Functions (VFs). These virtual functions can then be provisioned separately. Each VF can be seen as an addition device connected to the Physical Function. It shares the same resources with the Physical Function, and its number of ports equals those of the Physical Function.

SR-IOV is commonly used in conjunction with an SR-IOV enabled hypervisor to provide virtual machines direct hardware access to network resources hence increasing its performance.

In this chapter we will demonstrate setup and configuration of SR-IOV in a Red Hat Linux environment using Mellanox ConnectX® VPI adapter cards family.

4.13.1 System Requirements

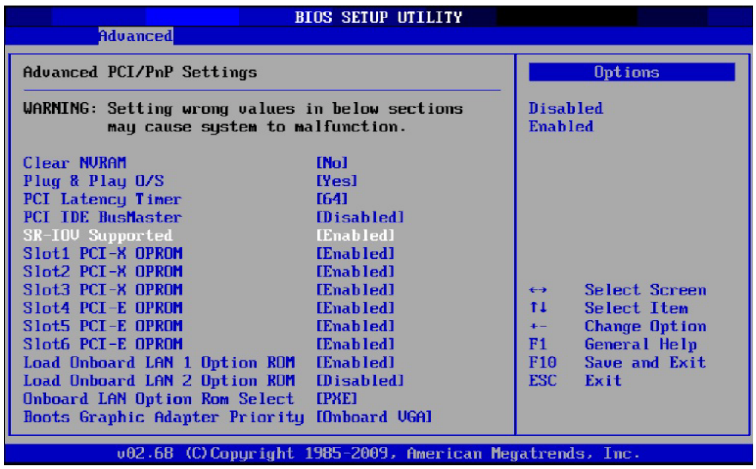
To set up an SR-IOV environment, the following is required:

- MLNX_OFED Driver
- A server/blade with an SR-IOV-capable motherboard BIOS
- Hypervisor that supports SR-IOV such as: Red Hat Enterprise Linux Server Version 6.*
- Mellanox ConnectX® VPI Adapter Card family with SR-IOV capability

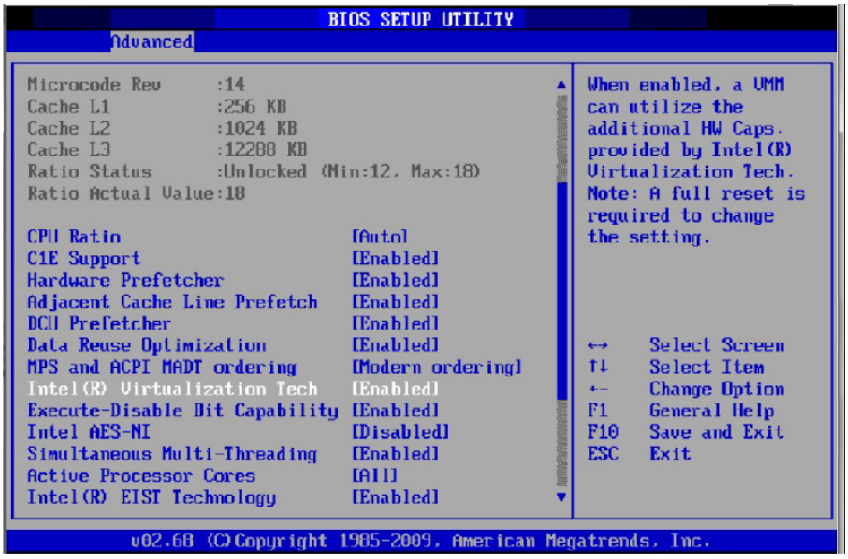
4.13.2 Setting Up SR-IOV

Depending on your system, perform the steps below to set up your BIOS. The figures used in this section are for illustration purposes only. For further information, please refer to the appropriate BIOS User Manual:

Step 1. Enable "SR-IOV" in the system BIOS.



Step 2. Enable "Intel Virtualization Technology".



Step 3. Install a hypervisor that supports SR-IOV.

Step 4. Depending on your system, update the /boot/grub/grub.conf file to include a similar command line load parameter for the Linux kernel.

For example, to Intel systems, add:

```
default=0
timeout=5
splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu
title Red Hat Enterprise Linux Server (2.6.32-36.x86-645)
        root (hd0,0)
        kernel /vmlinuz-2.6.32-36.x86-64 ro root=/dev/VolGroup00/LogVol00 rhgb quiet
        intel_iommu=on[1]
        initrd /initrd-2.6.32-36.x86-64.img
```

1. Please make sure the parameter "intel_iommu=on" exists when updating the /boot/grub/grub.conf file, otherwise SR-IOV cannot be loaded.

Step 5. Install the MLNX_OFED driver for Linux that supports SR-IOV.

Step 6. Verify the HCA is configured to support SR-IOV.

```
[root@selene ~]# mstflint -dev <PCI Device> dc
```

- Verify in the [HCA] section the following fields appear[1],[2]:

```
[HCA]
num_pfs = 1
total_vfs = <0-63>
sriov_en = true
```

HCA parameters can be configured during firmware update using the `mlnxofedinstall` script and running the `'--enable-sriov'` installation parameter.

If the current firmware version is the same as one provided with MLNX_OFED, run it in combination with the `'--force-fw-update'` parameter.

This configuration option is supported only in HCAs whose configuration file (INI) is included in MLNX_OFED.

Parameter	Recommended Value
num_pfs	1 **Note:** This field is optional and might not always appear.
total_vfs	63
sriov_en	true

- If the HCA does not support SR-IOV, please contact Mellanox Support: support@mellanox.com

Step 7. Create the text file /etc/modprobe.d/mlx4_core.conf if it does not exist, otherwise delete its contents.

Step 8. Insert an "option" line in the /etc/modprobe.d/mlx4_core.conf file to set the number of VFs. the protocol type per port, and the allowed number of virtual functions to be used by the physical function driver (probe_vf).

For example:

```
options mlx4_core num_vfs=5 port_type_array=1,2 probe_vf=1
```

1. If the fields in the example above do not appear in the [HCA] section, meaning SR-IOV is not supported in the used INI.
2. If SR-IOV is supported, to enable SR-IOV (if it is not enabled), it is sufficient to set “sriov_en = true” in the INI.

Parameter	Recommended Value

num_vfs	• If absent, or zero: no VFs will be available
	• If its value is a single number in the range of 0-63: The driver will enable the `num_vfs` VFs on the HCA and this will be applied to all ConnectX® HCAs on the host.
	• If its format is a string: The string specifies the `num_vfs` parameter separately per installed HCA.
	• The string format is: "bb:dd.f-v,bb:dd.f-v,…"
	  • bb:dd.f = bus:device.function of the PF of the HCA
	  • v = number of VFs to enable for that HCA
	For example:
	• `num_vfs=5` - The driver will enable 5 VFs on the HCA and this will be applied to all ConnectX® HCAs on the host
	• `num_vfs=00:04.0-5,00:07.0-8` - The driver will enable 5 VFs on the HCA positioned in BDF 00:04.0 and 8 on the one in 00:07.0)
	Note: PFs not included in the above list will not have SR-IOV enabled.

port_type_array	Specifies the protocol type of the ports. It is either one array of 2 port types `'t1,t2'` for all devices or list of BDF to `port_type_array 'bb:dd.f-t1;t2,...'`. (string) Valid port types: 1-ib, 2-eth, 3-auto, 4-N/A If only a single port is available, use the N/A port type for port2 (e.g '1,4').

probe_vf	• If absent or zero: no VFs will be used by the PF driver
	• If its value is a single number in the range of 0-63: Physical Function driver will use `probe_vf` VFs and this will be applied to all ConnectX® HCAs on the host.
	• If its format is a string: the string specifies the `probe_vf` parameter separately per installed HCA.
	• The string format is: "bb:dd.f-v,bb:dd.f-v,…
	  • bb:dd.f = bus:device.function of the PF of the HCA
	  • v = number of VFs to use in the PF driver for that HCA
	For example:
	• `probe_vfs=5` - The PF driver will probe 5 VFs on the HCA and this will be applied to all ConnectX® HCAs on the host
	• `probe_vfs=00:04.0-5,00:07.0-8` - The PF driver will probe 5 VFs on the HCA positioned in BDF 00:04.0 and 8 for the one in 00:07.0)
	Note: PFs not included in the above list will not use any of their VFs in the PF driver.

The example above loads the driver with 5 VFs (num_vfs). The standard use of a VF is a single VF per a single VM. However, the number of VFs varies upon the working mode requirements.

The protocol types are:

- Port 1 = IB
- Port 2 = Ethernet
  - port_type_array=2,2 (Ethernet, Ethernet)

- port_type_array=1,1 (IB, IB)
- port_type_array=1,2 (VPI: IB, Ethernet)
- NO port_type_array module parameter: ports are IB

Step 9. Reboot the server.

If the SR-IOV is not supported by the server, the machine might not come out of boot/load.

Step 10. Load the driver and verify the SR-IOV is supported. Run:

```
lspci | grep Mellanox
03:00.0 InfiniBand: Mellanox Technologies MT26428 [ConnectX VPI PCIe 2.0 5GT/s - IB QDR
/ 10GigE] (rev b0)
03:00.1 InfiniBand: Mellanox Technologies MT27500 Family [ConnectX-3 Virtual Function]
(rev b0)
03:00.2 InfiniBand: Mellanox Technologies MT27500 Family [ConnectX-3 Virtual Function]
(rev b0)
03:00.3 InfiniBand: Mellanox Technologies MT27500 Family [ConnectX-3 Virtual Function]
(rev b0)
03:00.4 InfiniBand: Mellanox Technologies MT27500 Family [ConnectX-3 Virtual Function]
(rev b0)
03:00.5 InfiniBand: Mellanox Technologies MT27500 Family [ConnectX-3 Virtual Function]
(rev b0)
```

Where:

- “03:00" represents the Physical Function
- “03:00.**X**" represents the Virtual Function connected to the Physical Function

4.13.3 Enabling SR-IOV and Para Virtualization on the Same Setup

➢ ***To enable SR-IOV and Para Virtualization on the same setup:***

Step 1. Create a bridge.

```
vim /etc/sysconfig/network-scripts/ifcfg-bridge0
DEVICE=bridge0
TYPE=Bridge
IPADDR=12.195.15.1
NETMASK=255.255.0.0
BOOTPROTO=static
ONBOOT=yes
NM_CONTROLLED=no
DELAY=0
```

Step 2. Change the related interface (in the example below bridge0 is created over eth5).

```
DEVICE=eth5
BOOTPROTO=none
STARTMODE=on
HWADDR=00:02:c9:2e:66:52
TYPE=Ethernet
NM_CONTROLLED=no
ONBOOT=yes
BRIDGE=bridge0
```

Step 3. Restart the service network.

Step 4. Attach a virtual NIC to VM.

```
ifconfig -a
…
eth6      Link encap:Ethernet  HWaddr 52:54:00:E7:77:99
          inet addr:13.195.15.5  Bcast:13.195.255.255  Mask:255.255.0.0
          inet6 addr: fe80::5054:ff:fee7:7799/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
         RX packets:481 errors:0 dropped:0 overruns:0 frame:0
          TX packets:450 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:22440 (21.9 KiB)  TX bytes:19232 (18.7 KiB)
          Interrupt:10 Base address:0xa000
…
```

4.13.4 Assigning a Virtual Function to a Virtual Machine

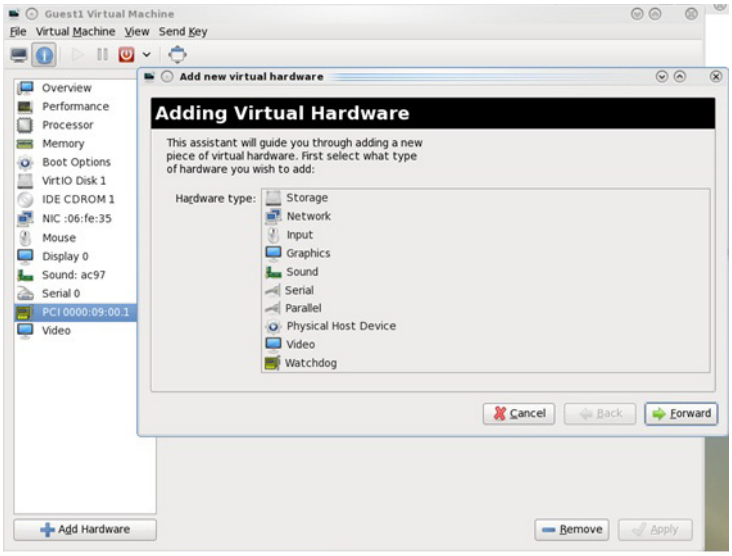
This section will describe a mechanism for adding a SR-IOV VF to a Virtual Machine.

4.13.4.1 Assigning the SR-IOV Virtual Function to the Red Hat KVM VM Server

Step 1. Run the virt-manager.

Step 2. Double click on the virtual machine and open its Properties.

Step 3. Go to Details->Add hardware ->PCI host device.



Step 4. Choose a Mellanox virtual function according to its PCI device (e.g., 00:03.1)

Step 5. If the Virtual Machine is up reboot it, otherwise start it.

Step 6. Log into the virtual machine and verify that it recognizes the Mellanox card. Run:

```
lspci | grep Mellanox

00:03.0 InfiniBand: Mellanox Technologies MT27500 Family [ConnectX-3 Virtual Function]
(rev b0)
```

Step 7. Add the device to the `/etc/sysconfig/network-scripts/ifcfg-ethX` configuration file. The MAC address for every virtual function is configured randomly, therefore it is not necessary to add it.

4.13.5 Uninstalling SR-IOV Driver

➢ ***To uninstall SR-IOV driver, perform the following:***

Step 1. For Hypervisors, detach all the Virtual Functions (VF) from all the Virtual Machines (VM) or stop the Virtual Machines that use the Virtual Functions.

Please be aware, stopping the driver when there are VMs that use the VFs, will cause machine to hang.

Step 2. Run the script below. Please be aware, uninstalling the driver deletes the entire driver's file, but does not unload the driver.

```
[root@swl022 ~]# /usr/sbin/ofed_uninstall.sh
This program will uninstall all OFED packages on your machine.
Do you want to continue?[y/N]:y
Running /usr/sbin/vendor_pre_uninstall.sh
Removing OFED Software installations
Running /bin/rpm -e --allmatches kernel-ib kernel-ib-devel libibverbs libibverbs-devel
libibverbs-devel-static libibverbs-utils libmlx4 libmlx4-devel libibcm libibcm-devel
libibumad libibumad-devel libibumad-static libibmad libibmad-devel libibmad-static
librdmacm librdmacm-utils librdmacm-devel ibacm opensm-libs opensm-devel perftest com-
pat-dapl compat-dapl-devel dapl dapl-devel dapl-devel-static dapl-utils srptools infini-
band-diags-guest ofed-scripts opensm-devel
warning: /etc/infiniband/openib.conf saved as /etc/infiniband/openib.conf.rpmsave
Running /tmp/2818-ofed_vendor_post_uninstall.sh
```

Step 3. Restart the server.

4.13.6 Burning Firmware with SR-IOV

The following procedure explains how to create a binary image with SR-IOV enabled that has 63 VFs. However, the number of VFs varies according to the working mode requirements.

➢ ***To burn the firmware:***

Step 1. Verify you have MFT installed in your machine.

Step 2. Enter the firmware directory, according to the HCA type (e.g. ConnectX®-3).

The path is: /mlnx_ofed/firmware/<device>/<FW version>

Step 3. Find the ini file that contains the HCA's PSID. Run:

```
#  ibv_devinfo | grep board_id
board_id: MT_1090120019
```

If such ini file cannot be found in the firmware directory, you may want to dump the configuration file using mstflint. Run:

```
#  mstflint -dev <PCI device> dc > <ini device file>
```

Step 4. Edit the ini file that you found in the previous step, and add the following lines to the [HCA] section in order to support 63 VFs.

```
;; SRIOV enable
total_vfs = 63[1]
num_pfs = 1
sriov_en = true
```

1. Some servers might have issues accepting 63 Virtual Functions or more. In such case, please set the number of "total_vfs" to any required value.

Step 5. Create a binary image using the modified ini file. Run:

```
#  mlxburn -fw ./<fw name>mlx -conf <modified ini file> -wrimage <file name>.bin
```

The file <file name>.bin is a firmware binary file with SR-IOV enabled that has 63 VFs. It can be spread across all machines and can be burnt using mstflint, which is part of the bundle, using the following command:

```
#  mstflint -dev <PCI device> -image <file name>.bin  b
```

After burning the firmware, the machine **must be rebooted**. If the driver is only restarted, the machine may hang and a reboot using power OFF/ON might be required.

4.13.7 Configuring Pkeys and GUIDs under SR-IOV

4.13.7.1 Port Type Management

Port Type management is static when enabling SR-IOV (the `connectx_port_config` script will not work). The port type is set on the Host via a module parameter, `port_type_array`, in `mlx4_core`. This parameter may be used to set the port type uniformly for all installed ConnectX® HCAs, or it may specify an individual configuration for each HCA.

This parameter should be specified as an options line in the file `/etc/modprobe.d/mlx4_core.conf`.

For example, to configure all HCAs to have Port1 as IB and Port2 as ETH, insert the following line:

```
options mlx4_core port_type_array=1,2
```

To set HCAs individually, you may use a string of `Domain:bus:device.function=x;y`

For example, if you have a pair of HCAs, whose PFs are 0000:04:00.0 and 0000:05:00.0, you may specify that the first will have both ports as IB, and the second will have both ports as ETH as follows:

```
options mlx4_core port_type_array='0000:04:00.0-1;1,0000:05:00.0-2;2
```

Only the PFs are set via this mechanism. The VFs inherit their port types from their associated PF.

4.13.7.2 Virtual Function InfiniBand Ports

Each VF presents itself as an independent vHCA to the host, while a single HCA is observable by the network which is unaware of the vHCAs. No changes are required by the InfiniBand subsystem, ULPs, and applications to support SR-IOV, and vHCAs are interoperable with any existing (non-virtualized) IB deployments.

Sharing the same physical port(s) among multiple vHCAs is achieved as follows:

- Each vHCA port presents its own virtual GID table

  The virtual GID table for the InfiniBand ports consists of a single entry (at index 0) that maps to a unique index in the physical GID table. The vHCA of the PF maps to physical GID index 0. To obtain GIDs for other vHCAs, alias GUIDs are requested from the SM. These GIDs are mapped to vHCAs as follows:

  - vHCA number x is assigned the GID/GUID at index x of the physical GID table.
- Each vHCA port presents its own virtual PKey table

  The virtual PKey table (presented to a VF) is a mapping of selected indexes of the physical PKey table. The host admin can control which PKey indexes are mapped to which virtual indexes using a sysfs interface (see Section , on page 98). The physical PKey table may contain both full and partial memberships of the same PKey to allow different membership types in different virtual tables.
- Each vHCA port has its own virtual port state

  A vHCA port is up if the following conditions apply:

  - The physical port is up
  - The virtual GID table contains the GIDs requested by the host admin
  - The SM has acknowledged the requested GIDs since the last time that the physical port went up
- Other port attributes are shared, such as: GID prefix, LID, SM LID, LMC mask

To allow the host admin to control the virtual GID and PKey tables of vHCAs, a new sysfs 'iov' sub-tree has been added under the PF InfiniBand device.

4.13.7.2.1SRIOV sysfs Administration Interfaces on the Hypervisor

Administration of GUIDs and PKeys is done via the sysfs interface in the Hypervisor (Dom0). This interface is under:

```
/sys/class/infiniband/<infiniband device>/iov
```

Under this directory, the following subdirectories can be found:

- `ports` - The actual (physical) port resource tables

  Port GID tables:

  - `ports/<n>/gids/<n> where 0 <= n <= 127` (the physical port gids)

- ports/<n>/admin_guids/<n> where 0 <= n <= 127 (allows examining or changing the administrative state of a given GUID>
- ports/<n>/pkeys/<n> where 0 <= n <= 126 (displays the contents of the physical pkey table)

- <pci id> directories - one for Dom0 and one per guest. Here, you may see the mapping between virtual and physical pkey indices, and the virtual to physical gid 0.

  Currently, the GID mapping cannot be modified, but the pkey virtual to physical mapping can .

  These directories have the structure:

  - <pci_id>/port/<m>/gid_idx/0 where m = 1..2 (this is read-only)

    and

  - <pci_id>/port/<m>/pkey_idx/<n>, where m = 1..2 and n = 0..126

For instructions on configuring pkey_idx, please see below.

4.13.7.2.2Configuring an Alias GUID (under ports/<n>/admin_guids)

Step 1. Determine the GUID index of the PCI Virtual Function that you want to pass through to a guest.

For example, if you want to pass through PCI function 02:00.3 to a certain guest, you initially need to see which GUID index is used for this function.

To do so:

```
cat /sys/class/infiniband/iov/0000:02:00.3/port/<port_num>/gid_idx/0
```

The value returned will present which guid index to modify on Dom0.

Step 2. Modify the physical GUID table via the admin_guids sysfs interface.

To configure the GUID at index <n> on port <port_num>:

```
cd /sys/class/infiniband/mlx4_0/iov/ports/<port_num>/admin_guids
echo <your desired guid>  > n
```

Example:

```
cd /sys/class/infiniband/mlx4_0/iov/ports/1/admin_guids
echo[1] "0x002fffff8118" > 3
```

1. echo "0x0" means let the SM assign a value to that GUID
   echo "0xffffffffffffffff" means delete that GUID
   echo <any other value> means request the SM to assign this GUID to this index

Step 3. Read the administrative status of the GUID index.

To read the administrative status of GUID index m on port n:

```
cat /sys/class/infiniband/mlx4_0/iov/ports/<n>/admin_guids/<m>
```

Step 4. Check the operational state of a GUID.

```
 /sys/class/infiniband/mlx4_0/iov/ports/<n>/gids  (where n = 1 or 2)
```

The values indicate what gids are actually configured on the firmware/hardware, and all the entries are R/O.

Step 5. Compare the value you read under the "admin_guids" directory at that index with the value under the "gids" directory, to verify the change requested in Step 3 has been accepted by the SM, and programmed into the hardware port GID table.

If the value under `admin_guids/<m>` is different that the value under `gids/<m>`, the request is still in progress.

4.13.7.2.3Partitioning IPoIB Communication using PKeys

PKeys are used to partition IPoIB communication between the Virtual Machines and the Dom0 by mapping a non-default full-membership PKey to virtual index 0, and mapping the default PKey to a virtual pkey index other than zero.

The below describes how to set up two hosts, each with 2 Virtual Machines. Host-1/vm-1 will be able to communicate via IPoIB only with Host2/vm1,and Host1/vm2 only with Host2/vm2.

In addition, Host1/Dom0 will be able to communicate only with Host2/Dom0 over ib0. vm1 and vm2 will not be able to communicate with each other, nor with Dom0.

This is done by configuring the virtual-to-physical PKey mappings for all the VMs, such that at virtual PKey index 0, both vm-1s will have the same pkey and both vm-2s will have the same PKey (different from the vm-1's), and the Dom0's will have the default pkey (different from the vm's pkeys at index 0).

OpenSM must be used to configure the physical Pkey tables on both hosts.

- The physical Pkey table on both hosts (Dom0) will be configured by OpenSM to be:

```
index 0 = 0xffff
index 1 = 0xb000
index 2 = 0xb030
```

- The vm1's virt-to-physical PKey mapping will be:

```
pkey_idx 0 = 1
pkey_idx 1 = 0
```

- The vm2's virt-to-phys pkey mapping will be:

```
pkey_idx 0 = 2
pkey_idx 1 = 0
```

  so that the default pkey will reside on the vms at index 1 instead of at index 0.

The IPoIB QPs are created to use the PKey at index 0. As a result, the Dom0, vm1 and vm2 IPoIB QPs will all use different PKeys.

➢ ***To partition IPoIB communication using PKeys:***

Step 1. Create a file `"/etc/opensm/partitions.conf"` on the host on which OpenSM runs, containing lines.

```
Default=0x7fff,ipoib : ALL=full ;
Pkey1=0x3000,ipoib : ALL=full;
Pkey3=0x3030,ipoib : ALL=full;
```

This will cause OpenSM to configure the physical Port Pkey tables on all physical ports on the network as follows:

```
pkey idx | pkey value
---------|---------
       0 | 0xFFFF
       1 | 0xB000
       2 | 0xB030
```

(the most significant bit indicates if a PKey is a full PKey).

The `",ipoib"` causes OpenSM to pre-create IPoIB the broadcast group for the indicated PKeys.

Step 2. Configure (on Dom0) the virtual-to-physical PKey mappings for the VMs.

Step a. Check the PCI ID for the Physical Function and the Virtual Functions.

```
lspci | grep Mel
```

Step b. Assuming that on Host1, the physical function displayed by lspci is "0000:02:00.0", and that on Host2 it is "0000:03:00.0"
On Host1 do the following.

```
cd /sys/class/infiniband/mlx4_0/iov
0000:02:00.0  0000:02:00.1 0000:02:00.2 ...[1]
```

1. 0000:02:00.0 contains the virtual-to-physical mapping tables for the physical function.
   0000:02:00.X contain the virt-to-phys mapping tables for the virtual functions.

Do not touch the Dom0 mapping table (under <nnnn>:<nn>:00.0). Modify only tables under 0000:02:00.1 and/or 0000:02:00.2. We assume that vm1 uses VF 0000:02:00.1 and vm2 uses VF 0000:02:00.2

Step c. Configure the virtual-to-physical PKey mapping for the VMs.

```
echo 0 > 0000:02:00.1/ports/1/pkey_idx/1
echo 1 > 0000:02:00.1/ports/1/pkey_idx/0
echo 0 > 0000:02:00.2/ports/1/pkey_idx/1
echo 2 > 0000:02:00.2/ports/1/pkey_idx/0
```

vm1 pkey index 0 will be mapped to physical pkey-index 1, and vm2 pkey index 0 will be mapped to physical pkey index 2. Both vm1 and vm2 will have their pkey index 1 mapped to the default pkey.

Step d. On Host2 do the following.

```
cd /sys/class/infiniband/mlx4_0/iov
echo 0 > 0000:03:00.1/ports/1/pkey_idx/1
echo 1 > 0000:03:00.1/ports/1/pkey_idx/0
echo 0 > 0000:03:00.2/ports/1/pkey_idx/1
echo 2 > 0000:03:00.2/ports/1/pkey_idx/0
```

Step e. Once the VMs are running, you can check the VM's virtualized PKey table by doing (on the vm).

```
cat /sys/class/infiniband/mlx4_0/ports/[1,2]/pkeys/[0,1]
```

Step 3. Start up the VMs (and bind VFs to them).

Step 4. Configure IP addresses for ib0 on the host and on the guests.

4.13.7.3 Ethernet Virtual Function Configuration when Running SR-IOV

4.13.7.3.1VLAN Guest Tagging (VGT) and VLAN Switch Tagging (VST)

When running ETH ports on VGT, the ports may be configured to simply pass through packets as is from VFs (Vlan Guest Tagging), or the administrator may configure the Hypervisor to silently force packets to be associated with a VLan/Qos (Vlan Switch Tagging).

In the latter case, untagged or priority-tagged outgoing packets from the guest will have the VLAN tag inserted, and incoming packets will have the VLAN tag removed. Any vlan-tagged packets sent by the VF are silently dropped. The default behavior is VGT.

The feature may be controlled on the Hypervisor from userspace via iprout2 / netlink:

```
ip link set { dev DEVICE | group DEVGROUP } [ { up | down } ]
   ...
   [ vf NUM [ mac LLADDR ]
      [ vlan VLANID [ qos VLAN-QOS ] ]
          ...
   [ spoofchk { on | off} ] ]
          ...
```

use:

```
ip link set dev <PF device> vf <NUM> vlan <vlan_id> [qos <qos>]
```

- where `NUM = 0..max-vf-num`
- vlan_id = `0..4095` (4095 means `"set VGT"`)
- qos = `0..7`

For example:

- `ip link set dev eth2 vf 2 qos 3` - sets VST mode for VF #2 belonging to PF eth2, with qos = 3
- `ip link set dev eth2 vf 2 4095` - sets mode for VF 2 back to VGT

4.13.7.3.2Additional Ethernet VF Configuration Options

- Guest MAC configuration

  By default, guest MAC addresses are configured to be all zeroes. In the mlnx_ofed guest driver, if a guest sees a zero MAC, it generates a random MAC address for itself. If the administrator wishes the guest to always start up with the same MAC, he/she should configure guest MACs before the guest driver comes up.

  The guest MAC may be configured by using:

  ```
  ip link set dev <PF device> vf <NUM> mac <LLADDR>
  ```

  For legacy guests, which do not generate random MACs, the adminstrator should always configure their MAC addresses via ip link, as above.

- Spoof checking

  Spoof checking is currently available only on upstream kernels newer than 3.1.

  ```
  ip link set dev <PF device> vf <NUM> spoofchk [on | off]
  ```

4.13.7.3.3RoCE Support

RoCE is supported on Virtual Functions and VLANs may be used with it. For RoCE, the hypervisor GID table size is of 16 entries while the VFs share the remaining 112 entries. When the

number of VFs is larger than 56 entries, some of them will have GID table with only a single entry which is inadequate if VF's Ethernet device is assigned with an IP address.

When setting num_vfs in mlx4_core module parameter it is important to check that the number of the assigned IP addresses per VF does not exceed the limit for GID table size.

4.14 CORE-Direct

4.14.1 CORE-Direct Overview

CORE-Direct provides a solution for off loading the MPI collectives operations from the software library to the network. CORE-Direct accelerates MPI applications and solves the scalability issues in large scale systems by eliminating the issues of operating systems noise and jitter.

It addresses the collectives communication scalability problem by off loading a sequence of data-dependent communications to the Host Channel Adapter (HCA). This solution provides the hooks needed to support computation and communication overlap. Additionally, it provides a means to reduce the effects of system noise and application skew on application scalability.

The relevant verbs to be used for CORE-Direct:

- ibv_create_qp_ex
- ibv_modify_cq
- ibv_query_device_ex
- ibv_post_task

Samples programs for reference:

- ibv_task_pingpong, ibv_cc_pingpong

4.15 Ethtool

ethtool is a standard Linux utility for controlling network drivers and hardware, particularly for wired Ethernet devices. It can be used to:

- Get identification and diagnostic information
- Get extended device statistics
- Control speed, duplex, autonegotiation and flow control for Ethernet devices
- Control checksum offload and other hardware offload features
- Control DMA ring sizes and interrupt moderation

The following are the ethtool supported options:

Table 6 - ethtool Supported Options

Options	Description

ethtool -i eth<x>	Checks driver and device information.
	For example:
	`#> ethtool -i eth2`
	`driver: mlx4_en (MT_0DD0120009_CX3)`
	`version: 2.1.6 (Aug 2013)`
	`firmware-version: 2.30.3000`
	`bus-info: 0000:1a:00.0`

ethtool -k eth<x>	Queries the stateless offload status.
ethtool -K eth<x> [rx on|off] [tx on|off] [sg on|off] [tso on|off] [lro on|off] [gro on|off] [gso on|off]	Sets the stateless offload status. TCP Segmentation Offload (TSO), Generic Segmentation Offload (GSO): increase outbound throughput by reducing CPU overhead. It works by queuing up large buffers and letting the network interface card split them into separate packets. Large Receive Offload (LRO): increases inbound throughput of high-bandwidth network connections by reducing CPU overhead. It works by aggregating multiple incoming packets from a single stream into a larger buffer before they are passed higher up the networking stack, thus reducing the number of packets that have to be processed. LRO is available in kernel versions < 3.1 for untagged traffic. **Note:** LRO will be done whenever possible. Otherwise GRO will be done. Generic Receive Offload (GRO) is available throughout all kernels.
ethtool -c eth<x>	Queries interrupt coalescing settings.
ethtool -C eth<x> adaptive-rx on|off	Enables/disables adaptive interrupt moderation. By default, the driver uses adaptive interrupt moderation for the receive path, which adjusts the moderation time to the traffic pattern.
ethtool -C eth<x> [pkt-rate-low N] [pkt-rate-high N] [rx-usecs-low N] [rx-usecs-high N]	Sets the values for packet rate limits and for moderation time high and low values. • 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.

Table 6 - ethtool Supported Options

Options	Description
ethtool -C eth<x> [rx-usecs N] [rx-frames N]	Sets the interrupt coalescing settings when the adaptive moderation is disabled. **Note:** usec settings correspond to the time to wait after the *last* packet is sent/received before triggering an interrupt.
ethtool -a eth<x>	Queries the pause frame settings.
ethtool -A eth<x> [rx on|off] [tx on|off]	Sets the pause frame settings.
ethtool -g eth<x>	Queries the ring size values.
ethtool -G eth<x> [rx <N>] [tx <N>]	Modifies the rings size.
ethtool -S eth<x>	Obtains additional device statistics.
ethtool -t eth<x>	Performs a self diagnostics test.
ethtool -s eth<x> msglvl [N]	Changes the current driver message level.
ethtool -T eth<x>	Shows time stamping capabilities
ethtool -l eth<x>	Shows the number of channels
ethtool -L eth<x> [rx <N>] [tx <N>]	Sets the number of channels

4.16 Dynamically Connected Transport Service

Dynamically Connected transport (DCT) is currently at beta level.
Please be aware that the content below is subject to change.

Dynamically Connected transport (DCT) service is an extension to transport services to enable a higher degree of scalability while maintaining high performance for sparse traffic. Utilization of DCT reduces the total number of QPs required system wide by having Reliable type QPs dynamically connect and disconnect from any remote node. DCT connections only stay connected while they are active. This results in smaller memory footprint, less overhead to set connections and higher on-chip cache utilization and hence increased performance. DCT is supported only in mlx5 and is at beta level.

4.17 PeerDirect

PeerDirect uses an API between IB CORE and peer memory clients, (e.g. GPU cards) to provide access to an HCA to read/write peer memory for data buffers. As a result, it allows RDMA-based

(over InfiniBand/RoCE) application to use peer device computing power, and RDMA interconnect at the same time without copying the data between the P2P devices.

For example, PeerDirect is being used for GPUDirect RDMA.

Detailed description for that API exists under MLNX OFED installation, please see docs/readme_and_user_manual/PEER_MEMORY_API.txt

4.18 Inline-Receive

When Inline-Receive is active, the HCA may write received data in to the receive WQE or CQE. Using Inline-Receive saves PCIe read transaction since the HCA does not need to read the scatter list, therefore it improves performance in case of short receive-messages.

On poll CQ, the driver copies the received data from WQE/CQE to the user's buffers. Therefore, apart from querying Inline-Receive capability and Inline-Receive activation the feature is transparent to user application.

When Inline-Receive is active, user application must provide a valid virtual address for the receive buffers to allow the driver moving the inline-received message to these buffers. The validity of these addresses is not checked therefore the result of providing non-valid virtual addresses is unexpected.

Connect-IB™ supports Inline-Receive on both the requestor and the responder sides. Since data is copied at the poll CQ verb, Inline-Receive on the requestor side is possible only if the user chooses IB(V)_SIGNAL_ALL_WR.

4.18.1 Querying Inline-Receive Capability

User application can use the `ibv_exp_query_device` function to get the maximum possible Inline-Receive size. To get the size, the application needs to set the `IBV_EXP_DEVICE_ATTR_INLINE_RECV_SZ` bit in the `ibv_exp_device_attr comp_mask`.

4.18.2 Activating Inline-Receive

To activate the Inline-Receive, you need to set the required message size in the `max_inl_recv` field in the `ibv_exp_qp_init_attr struct` when calling `ibv_exp_create_qp` function. The value returned by the same field is the actual Inline-Receive size applied.

Setting the message size may affect the WQE/CQE size.

4.19 Ethernet Performance Counters

Counters are used to provide information about how well an operating system, an application, a service, or a driver is performing. The counter data helps determine system bottlenecks and fine-tune the system and application performance. The operating system, network, and devices provide counter data that an application can consume to provide users with a graphical view of how well the system is performing.

The counter index is a QP attribute given in the QP context. Multiple QPs may be associated with the same counter set, If multiple QPs share the same counter its value represents the cumulative total.

- ConnectX®-3 support 127 different counters which allocated:
  - 4 counters reserved for PF - 2 counters for each port
  - 2 counters reserved for VF - 1 counter for each port
  - All other counters if exist are allocated by demand
- RoCE counters are available only through sysfs located under:
  - # /sys/class/infiniband/mlx4_*/ports/*/counters/
  - # /sys/class/infiniband/mlx4_*/ports/*/counters_ext/
- Physical Function can also read Virtual Functions' port counters through sysfs located under:
  - # /sys/class/net/eth*/vf*_statistics/

To display the network device Ethernet statistics, you can run:

```
Ethtool -S <devname>
```

Table 7 - Port IN Counters

Counter	Description
rx_packets	Total packets successfully received.
rx_bytes	Total bytes in successfully received packets.
rx_multicast_packets	Total multicast packets successfully received.
rx_broadcast_packets	Total broadcast packets successfully received.
rx_errors	Number of receive packets that contained errors preventing them from being deliverable to a higher-layer protocol.
rx_dropped	Number of receive packets which were chosen to be discarded even though no errors had been detected to prevent their being deliverable to a higher-layer protocol.
rx_length_errors	Number of received frames that were dropped due to an error in frame length
rx_over_errors	Number of received frames that were dropped due to overflow
rx_crc_errors	Number of received frames with a bad CRC that are not runts, jabbers, or alignment errors

Table 7 - Port IN Counters

Counter	Description
rx_jabbers	Number of received frames with a length greater than MTU octets and a bad CRC
rx_in_range_length_error	Number of received frames with a length/type field value in the (decimal) range [1500:46] (42 is also counted for VLANtagged frames)
rx_out_range_length_error	Number of received frames with a length/type field value in the (decimal) range [1535:1501]
rx_lt_64_bytes_packets	Number of received 64-or-less-octet frames
rx_127_bytes_packets	Number of received 65-to-127-octet frames
rx_255_bytes_packets	Number of received 128-to-255-octet frames
rx_511_bytes_packets	Number of received 256-to-511-octet frames
rx_1023_bytes_packets	Number of received 512-to-1023-octet frames
rx_1518_bytes_packets	Number of received 1024-to-1518-octet frames
rx_1522_bytes_packets	Number of received 1519-to-1522-octet frames
rx_1548_bytes_packets	Number of received 1523-to-1548-octet frames
rx_gt_1548_bytes_packets	Number of received 1549-or-greater-octet frames

Table 8 - Port OUT Counters

Counter	Description
tx_packets	Total packets successfully transmitted.
tx_bytes	Total bytes in successfully transmitted packets.
tx_multicast_packets	Total multicast packets successfully transmitted.
tx_broadcast_packets	Total broadcast packets successfully transmitted.
tx_errors	Number of frames that failed to transmit
tx_dropped	Number of transmitted frames that were dropped
tx_lt_64_bytes_packets	Number of transmitted 64-or-less-octet frames
tx_127_bytes_packets	Number of transmitted 65-to-127-octet frames
tx_255_bytes_packets	Number of transmitted 128-to-255-octet frames
tx_511_bytes_packets	Number of transmitted 256-to-511-octet frames
tx_1023_bytes_packets	Number of transmitted 512-to-1023-octet frames

Table 8 - Port OUT Counters

Counter	Description
tx_1518_bytes_packets	Number of transmitted 1024-to-1518-octet frames
tx_1522_bytes_packets	Number of transmitted 1519-to-1522-octet frames
tx_1548_bytes_packets	Number of transmitted 1523-to-1548-octet frames
tx_gt_1548_bytes_packets	Number of transmitted 1549-or-greater-octet frames

Table 9 - Port VLAN Priority Tagging (where <i> is in the range 0…7)

Counter	Description
rx_prio_<i>_packets	Total packets successfully received with priority i.
rx_prio_<i>_bytes	Total bytes in successfully received packets with priority i.
rx_novlan_packets	Total packets successfully received with no VLAN priority.
rx_novlan_bytes	Total bytes in successfully received packets with no VLAN priority.
tx_prio_<i>_packets	Total packets successfully transmitted with priority i.
tx_prio_<i>_bytes	Total bytes in successfully transmitted packets with priority i.
tx_novlan_packets	Total packets successfully transmitted with no VLAN priority.
tx_novlan_bytes	Total bytes in successfully transmitted packets with no VLAN priority.

Table 10 - Port Pause (where <i> is in the range 0…7)

Counter	Description
rx_pause_prio_<i>	The total number of PAUSE frames received from the far-end port
rx_pause_duration_prio_<i>	The total time in microseconds that far-end port was requested to pause transmission of packets.
rx_pause_transition_prio_<i>	The number of receiver transitions from XON state (paused) to XOFF state (non-paused)
tx_pause_prio_<i>	The total number of PAUSE frames sent to the far-end port
tx_pause_duration_prio_<i>	The total time in microseconds that transmission of packets has been paused

Table 10 - Port Pause (where <i> is in the range 0…7)

Counter	Description
tx_pause_transition_prio_<i>	The number of transmitter transitions from XON state (paused) to XOFF state (non-paused)

Table 11 - VPort Statistics (where <i>=<empty_string> is the PF, and ranges 1…NumOfVf per VF)

Counter	Description
vport<i>_rx_unicast_packets	Unicast packets received successfully
vport<i>_rx_unicast_bytes	Unicast packet bytes received successfully
vport<i>_rx_multicast_packets	Multicast packets received successfully
vport<i>_rx_multicast_bytes	Multicast packet bytes received successfully
vport<i>_rx_broadcast_packets	Broadcast packets received successfully
vport<i>_rx_broadcast_bytes	Broadcast packet bytes received successfully
vport<i>_rx_dropped	Received packets discarded due to out-of-buffer condition
vport<i>_rx_errors	Received packets discarded due to receive error condition
vport<i>_tx_unicast_packets	Unicast packets sent successfully
vport<i>_tx_unicast_bytes	Unicast packet bytes sent successfully
vport<i>_tx_multicast_packets	Multicast packets sent successfully
vport<i>_tx_multicast_bytes	Multicast packet bytes sent successfully
vport<i>_tx_broadcast_packets	Broadcast packets sent successfully
vport<i>_tx_broadcast_bytes	Broadcast packet bytes sent successfully
vport<i>_tx_errors	Packets dropped due to transmit errors

Table 12 - SW Statistics

Counter	Description
rx_lro_aggregated	Number of packets aggregated
rx_lro_flushed	Number of LRO flush to the stack
rx_lro_no_desc	Number of times LRO description was not found
rx_alloc_failed	Number of times failed preparing receive descriptor
rx_csum_good	Number of packets received with good checksum
rx_csum_none	Number of packets received with no checksum indication
tx_chksum_offload	Number of packets transmitted with checksum offload
tx_queue_stopped	Number of times transmit queue suspended
tx_wake_queue	Number of times transmit queue resumed
tx_timeout	Number of times transmitter timeout
tx_tso_packets	Number of packet that were aggregated

Table 13 - Per Ring (SW) Statistics (where <i> is the ring I – per configuration)

Counter	Description
rx<i>_packets	Total packets successfully received on ring i
rx<i>_bytes	Total bytes in successfully received packets on ring i.
tx<i>_packets	Total packets successfully transmitted on ring i.
tx<i>_bytes	Total bytes in successfully transmitted packets on ring i.

4.20 Memory Window

Memory Window allows the application to have a more flexible control over remote access to its memory. It is available only on physical functions / native machines The two types of Memory Windows supported are: type 1 and type 2B.

Memory Windows are intended for situations where the application wants to:

- grant and revoke remote access rights to a registered region in a dynamic fashion with less of a performance penalty
- grant different remote access rights to different remote agents and/or grant those rights over different ranges within registered region

For further information, please refer to the InfiniBand specification document.

Memory Windows API cannot co-work with peer memory clients (PeerDirect).

4.20.1 Query Capabilities

Memory Windows are available if and only the hardware supports it. To verify whether Memory Windows are available, run `ibv_query_device`.

For example:

```
truct ibv_device_attr device_attr = {};
ibv_query_device (context, & device_attr);
if (device_attr.device_cap_flags & IBV_DEVICE_MEM_WINDOW ||
                device_attr.device_cap_flags & IBV_DEVICE_MW_TYPE_2B) {
/* Memory window is supported */
```

4.20.2 Allocating Memory Window

Allocating memory window is done by calling the `ibv_alloc_mw` verb.

```
type_mw = IBV_MW_TYPE_2/ IBV_MW_TYPE_1
mw = ibv_alloc_mw(pd, type_mw);
```

4.20.3 Binding Memory Windows

After allocated, memory window should be bound to a registered memory region. Memory Region should have been registered using the `IBV_ACCESS_MW_BIND` access flag.

- Binding Memory Window **type 1** is done via the `ibv_bind_mw` verb.

```
struct ibv_mw_bind mw_bind;
ret = ibv_bind_mw(qp, mw, &mw_bind);
```

- Binding memory window **type 2B** is done via the `ibv_post_send` verb and a specific Work Request (WR) with `opcode = IBV_WR_BIND_MW`

Prior to binding, please make sure to update the existing rkey.

```
ibv_inc_rkey(mw->rkey)
```

4.20.4 Invalidating Memory Window

Before rebinding Memory Window type 2, it must be invalidated using the `ibv_post_send` verb and a specific WR with `opcode = IBV_WR_LOCAL_INV`.

4.20.5 Deallocating Memory Window

Deallocating memory window is done using the `ibv_dealloc_mw` verb.

```
ibv_dealloc_mw(mw);
```

5 HPC Features

5.1 Shared Memory Access

The Shared Memory Access (SHMEM) routines provide low-latency, high-bandwidth communication for use in highly parallel scalable programs. The routines in the SHMEM Application Programming Interface (API) provide a programming model for exchanging data between cooperating parallel processes. The SHMEM API can be used either alone or in combination with MPI routines in the same parallel program.

The SHMEM parallel programming library is an easy-to-use programming model which uses highly efficient one-sided communication APIs to provide an intuitive global-view interface to shared or distributed memory systems. SHMEM's capabilities provide an excellent low level interface for PGAS applications.

A SHMEM program is of a single program, multiple data (SPMD) style. All the SHMEM processes, referred as processing elements (PEs), start simultaneously and run the same program. Commonly, the PEs perform computation on their own sub-domains of the larger problem, and periodically communicate with other PEs to exchange information on which the next communication phase depends.

The SHMEM routines minimize the overhead associated with data transfer requests, maximize bandwidth, and minimize data latency (the period of time that starts when a PE initiates a transfer of data and ends when a PE can use the data).

SHMEM routines support remote data transfer through:

- "put" operations - data transfer to a different PE
- "get" operations - data transfer from a different PE, and remote pointers, allowing direct references to data objects owned by another PE

Additional supported operations are collective broadcast and reduction, barrier synchronization, and atomic memory operations. An atomic memory operation is an atomic read-and-update operation, such as a fetch-and-increment, on a remote or local data object.

SHMEM libraries implement active messaging. The sending of data involves only one CPU where the source processor puts the data into the memory of the destination processor. Likewise, a processor can read data from another processor's memory without interrupting the remote CPU. The remote processor is unaware that its memory has been read or written unless the programmer implements a mechanism to accomplish this.

5.1.1 Mellanox ScalableSHMEM

The ScalableSHMEM programming library is a one-side communications library that supports a unique set of parallel programming features including point-to-point and collective routines, synchronizations, atomic operations, and a shared memory paradigm used between the processes of a parallel programming application.

Mellanox ScalableSHMEM is based on the API defined by the OpenSHMEM.org consortium. The library works with the OpenFabrics RDMA for Linux stack (OFED), and also has the ability to utilize MellanoX Messaging libraries (MXM) as well as Mellanox Fabric Collective Accelerations (FCA), providing an unprecedented level of scalability for SHMEM programs running over InfiniBand.

The latest ScalableSHMEM software can be downloaded from the Mellanox website.

5.1.2 Running SHMEM with FCA

The Mellanox Fabric Collective Accelerator (FCA) is a unique solution for offloading collective operations from the Message Passing Interface (MPI) or ScalableSHMEM process onto Mellanox InfiniBand managed switch CPUs. As a system-wide solution, FCA utilizes intelligence on Mellanox InfiniBand switches, Unified Fabric Manager and MPI nodes without requiring additional hardware. The FCA manager creates a topology based collective tree, and orchestrates an efficient collective operation using the switch-based CPUs on the MPI/ScalableSHMEM nodes.

FCA accelerates MPI/ScalableSHMEM collective operation performance by up to 100 times providing a reduction in the overall job runtime. Implementation is simple and transparent during the job runtime.

FCA is disabled by default and *must* be configured prior to using it from the ScalableSHMEM.

➢ ***To enable FCA by default in the ScalableSHMEM:***

1. Edit the `/opt/mellanox/openshmem/2.2/etc/openmpi-mca-params.conf` file.
2. Set the `scoll_fca_enable` parameter to `1`.

```
scoll_fca_enable=1
```

3. Set the `scoll_fca_np` parameter to 0.

```
scoll_fca_np=0
```

➢ ***To enable FCA in the*** `shmemrun` ***command line, add the following:***

```
-mca scoll_fca_enable=1
-mca scoll_fca_enable_np 0
```

➢ ***To disable FCA:***

```
-mca scoll_fca_enable 0 -mca coll_fca_enable 0
```

For more details on FCA installation and configuration, please refer to the FCA User Manual found in the Mellanox website.

5.1.3 Running ScalableSHMEM with MXM

MellanoX Messaging (MXM) library provides enhancements to parallel communication libraries by fully utilizing the underlying networking infrastructure provided by Mellanox HCA/switch hardware. This includes a variety of enhancements that take advantage of Mellanox networking hardware including:

- Multiple transport support including RC, XRC and UD
- Proper management of HCA resources and memory structures
- Efficient memory registration
- One-sided communication semantics
- Connection management
- Receive side tag matching
- Intra-node shared memory communication

These enhancements significantly increase the scalability and performance of message communi-cations in the network, alleviating bottlenecks within the parallel communication libraries

5.1.4 Running SHMEM with Contiguous Pages

Contiguous Pages improves performance by allocating user memory regions over contiguous pages. It enables a user application to ask low level drivers to allocate contiguous memory for it as part of `ibv_reg_mr`.

- ***To activate MLNX_OFED 2.0 and the contiguous pages allocator with SHMEM:***

  Run the following argument to enable compound pages with SHMEM:

  ```
  % /opt/mellanox/openshmem/2.1/bin/shmemrun -mca shmalloc_use_hugepages 5
  ```

  If using compound pages is not possible, then the user will fall back to regular hugepages mechanism.

- ***To force use of compound pages allocator***

  Run the following command:

  ```
  % /opt/mellanox/openshmem/2.1/bin/shmemrun -mca shmalloc_use_hugepages 5 -x
  MR_FORCE_CONTIG_PAGES=1
  ```

  For further information on the Contiguous Pages, please refer to Section 4.9, “Contiguous Pages”, on page 84.

5.1.5 Running ScalableSHMEM Application

The ScalableSHMEM framework contains the `shmemrun` utility which launches the executable from a service node to compute nodes. This utility accepts the same command line parameters as `mpirun` from the OpenMPI package.

For further information, please refer to OpenMPI MCA parameters documentation at:

http://www.open-mpi.org/faq/?category=running.

Run `"shmemrun --help"` to obtain ScalableSHMEM job launcher runtime parameters.

ScalableSHMEM contains support for environment module system (http://modules.sf.net/). The modules configuration file can be found at: `/opt/mellanox/openshmem/2.2/etc/shmem_modulefile`

5.2 Message Passing Interface

5.2.1 Overview

Mellanox OFED for Linux includes the following Message Passing Interface (MPI) implementations over InfiniBand:

- Open MPI 1.4.6 & 1.6.1 – an open source MPI-2 implementation by the Open MPI Project
- OSU MVAPICH2 1.7 – an MPI-1 implementation by Ohio State University

These MPI implementations, along with MPI benchmark tests such as OSU BW/LAT, Intel MPI Benchmark, and Presta, are installed on your machine as part of the Mellanox OFED for Linux installation. Table 14 lists some useful MPI links.

Table 14 - Useful MPI Links

MPI Standard	http://www-unix.mcs.anl.gov/mpi
Open MPI	http://www.open-mpi.org
MVAPICH 2 MPI	http://mvapich.cse.ohio-state.edu/
MPI Forum	http://www.mpi-forum.org

This chapter includes the following sections:

5.2.2 Prerequisites for Running MPI

For launching multiple MPI processes on multiple remote machines, the MPI standard provides a launcher program that requires automatic login (i.e., password-less) onto the remote machines. SSH (Secure Shell) is both a computer program and a network protocol that can be used for logging and running commands on remote computers and/or servers.

5.2.2.1 SSH Configuration

The following steps describe how to configure password-less access over SSH:

Step 1. Generate an ssh key on the initiator machine (host1).

```
host1$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/<username>/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/<username>/.ssh/id_rsa.
Your public key has been saved in /home/<username>/.ssh/id_rsa.pub.
The key fingerprint is:
38:1b:29:df:4f:08:00:4a:0e:50:0f:05:44:e7:9f:05 <username>@host1
```

Step 2. Check that the public and private keys have been generated.

```
host1$ cd /home/<username>/.ssh/
host1$ ls
host1$ ls -la
total 40
drwx------  2 root root 4096 Mar  5 04:57 .
drwxr-x--- 13 root root 4096 Mar  4 18:27 ..
-rw-------  1 root root 1675 Mar  5 04:57 id_rsa
```

```
-rw-r--r--  1 root root  404 Mar  5 04:57 id_rsa.pub
```

Step 3. Check the public key.

```
host1$ cat id_rsa.pub
ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEA1zVY8VBHQh9okZN7OA1ibUQ74RXm4zHeczyVxpYHaDPyDmqezbYMKrCIVz
d10bH+ZkC0rpLYviU0oUHd3fvNTfMs0gcGg08PysUf+12FyYjira2P1xyg6mkHLGGqVutfEMmABZ3wNCUg6J2X
3G/uiuSWXeubZmbXcMrP/
w4IWByfH8ajwo6A5WioNbFZElbYeeNfPZf4UNcgMOAMWp64sL58tkt32F+RGmyLXQWZL27Synsn6dHpxMqBorX
NC0ZBe4kTnUqm63nQ2z1qVMdL9FrCma1xIOu9+SQJAjwONevaMzFKEHe7YHg6YrNfXunfdbEurzB524TpPcrod
ZlfCQ== <username>@host1
```

Step 4. Now you need to add the public key to the `authorized_keys2` file on the *target* machine.

```
host1$ cat id_rsa.pub | xargs ssh host2 \"echo  >>/home/<username>/.ssh/
authorized_keys2"
                                                  <username>@host2's pass-
word:                // Enter password
host1$
```

For a local machine, simply add the key to `authorized_keys2.`

```
host1$ cat id_rsa.pub >> authorized_keys2
```

Step 5. Test.

```
host1$ ssh  host2  uname
Linux
```

5.2.3 MPI Selector - Which MPI Runs

Mellanox OFED contains a simple mechanism for system administrators and end-users to select which MPI implementation they want to use. The MPI selector functionality is not specific to any MPI implementation; it can be used with any implementation that provides shell startup files that correctly set the environment for that MPI. The Mellanox OFED installer will automatically add MPI selector support for each MPI that it installs. Additional MPI's not known by the Mellanox OFED installer can be listed in the MPI selector; see the mpi-selector(1) man page for details.

Note that MPI selector only affects the default MPI environment for *future* shells. Specifically, if you use MPI selector to select MPI implementation ABC, this default selection will not take effect until you start a new shell (e.g., logout and login again). Other packages (such as environment modules) provide functionality that allows changing your environment to point to a new MPI implementation in the current shell. The MPI selector was not meant to duplicate or replace that functionality.

The MPI selector functionality can be invoked in one of two ways:

1. The mpi-selector-menu command.

   This command is a simple, menu-based program that allows the selection of the system-wide MPI (usually only settable by root) and a per-user MPI selection. It also shows what the current selections are. This command is recommended for all users.

2. The mpi-selector command.

   This command is a CLI-equivalent of the mpi-selector-menu, allowing for the same functionality as mpi-selector-menu but without the interactive menus and prompts. It is suitable for scripting.

5.2.4 Compiling MPI Applications

Compiling MVAPICH Applications

Please refer to http://mvapich.cse.ohio-state.edu/support/mvapich_user_guide.html.

To review the default configuration of the installation, check the default configuration file:

```
/usr/mpi/<compiler>/mvapich-<mvapich-ver>/etc/mvapich.conf
```

Compiling Open MPI Applications

Please refer to http://www.open-mpi.org/faq/?category=mpi-apps.

5.3 MellanoX Messaging

MellanoX Messaging (MXM) provides enhancements to parallel communication libraries by fully utilizing the underlying networking infrastructure provided by Mellanox HCA/switch hardware. This includes a variety of enhancements that take advantage of Mellanox networking hardware including:

- Multiple transport support including RC and UD
- Proper management of HCA resources and memory structures
- Efficient memory registration
- One-sided communication semantics
- Connection management
- Receive side tag matching
- Intra-node shared memory communication

These enhancements significantly increase the scalability and performance of message communications in the network, alleviating bottlenecks within the parallel communication libraries.

The latest MXM software can be downloaded from the Mellanox website.

MLNX_OFED v2.0 or later comes with a pre-installed version of MXM v2.x and OpenMPI compiled with MXM v2.x.

5.3.1 Compiling OpenMPI with MXM

Step 1. Install MXM from RPM.

```
% rpm -ihv mxm-x.y.z-1.x86_64.rpm
```

MXM will be installed automatically in the `/opt/mellanox/mxm` folder.

Step 2. Enter OpenMPI source directory and run:

```
% cd $OMPI_HOME
% ./configure --with-mxm=/opt/mellanox/mxm <... other configure parameters...>
% make all && make install
```

MLNX_OFED v2.0 or later comes with a pre-installed version of MXM v2.x and OpenMPI compiled with MXM v2.x.

To check the version of MXM installed on your host, run:

```
rpm -qi mxm
```

➢ ***To upgrade MLNX_OFED v2.0 or later with a newer MXM:***

Step 1. Remove MXM v1.1.

```
rpm -e mxm
```

Step 2. Remove the pre-compiled OpenMPI.

```
rpm -e mlnx-openmpi_gcc
```

Step 3. Install the new MXM and compile the OpenMPI with it.

To run OpenMPI without MXM, run:

```
% mpirun -mca mtl ^mxm <...>
```

When upgrading to MXM v2.1, OpenMPI compiled with the previous versions of the MXM should be recompiled with MXM v2.1.

5.3.2 Enabling MXM in OpenMPI

MXM v2.1 is automatically selected by OpenMPI (up to v1.6) when the Number of Processes (NP) is higher or equal to 128. To enable MXM for any NP use the following OpenMPI parameter `"-mca mtl_mxm_np <number>"`.

From OpenMPI v1.7, MXM is selected when the number of processes is higher or equal to 0. i.e. by default.

➢ ***To activate MXM for any NP, run:***

```
% mpirun -mca mtl_mxm_np 0 <...other mpirun parameters ...>
```

5.3.3 Tuning MXM Settings

The default MXM settings are already optimized. To check the available MXM parameters and their default values, run the `/opt/mellanox/mxm/bin/mxm_dump_config` utility which is part of the MXM RPM.

MXM parameters can be modified in one of the following methods:

- Modifying the default MXM parameters value as part of the mpirun:

```
% mpirun -x MXM_UD_RX_MAX_BUFFERS=128000 <...>
```

- Modifying the default MXM parameters value from SHELL:

```
% export MXM_UD_RX_MAX_BUFFERS=128000
% mpirun <...>
```

5.3.4 Configuring Multi-Rail Support

Multi-Rail support enables the user to use more than one of the active ports on the card, by making a better use of the resources. It provides a combined throughput among the used ports.

➢ ***To configure dual rail support:***

- Specify the list of ports you would like to use to enable multi rail support.

```
-x MXM_RDMA_PORTS=cardName:portNum
```

or

```
-x MXM_IB_PORTS=cardName:portNum
```

5.3.5 Configuring MXM over the Ethernet Fabric

➢ ***To configure MXM over the Ethernet fabric:***

Step 1. Make sure the Ethernet port is active.

```
ibv_devinfo
```

ibv_devinfo displays the list of cards and ports in the system. Please make sure (in the ibv_devinfo output) that the desired port has Ethernet at the `link_layer` field and that its state is `PORT_ACTIVE`.

Step 2. Specify the ports you would like to use, if there is a non Ethernet active port in the card.

```
-x MXM_RDMA_PORTS=mlx4_0:1
```

or

```
-x MXM_IB_PORTS=mlx4_0:1
```

5.4 Fabric Collective Accelerator

The Mellanox Fabric Collective Accelerator (FCA) is a unique solution for offloading collective operations from the Message Passing Interface (MPI) process to the server CPUs. As a system-wide solution, FCA does not require any additional hardware. The FCA manager creates a topology based collective tree, and orchestrates an efficient collective operation using the CPUs in the servers that are part of the collective operation. FCA accelerates MPI collective operation performance by up to 100 times providing a reduction in the overall job runtime. Implementation is simple and transparent during the job runtime.

MLNX_OFED v2.0 or later comes with a pre-installed version of FCA v2.x

FCA is built on the following main principles:

- Topology-aware Orchestration

  The MPI collective logical tree is matched to the physical topology. The collective logical tree is constructed to assure:

  - Maximum utilization of fast inter-core communication
  - Distribution of the results.
- Communication Isolation

Collective communications are isolated from the rest of the traffic in the fabric using a private virtual network (VLane) eliminating contention with other types of traffic.

After MLNX_OFED installation, FCA can be found at `/opt/mellanox/fca` folder.

For further information on configuration instructions, please refer to the FCA User Manual.

5.5 ScalableUPC

Unified Parallel C (UPC) is an extension of the C programming language designed for high performance computing on large-scale parallel machines.The language provides a uniform programming model for both shared and distributed memory hardware. The programmer is presented with a single shared, partitioned address space, where variables may be directly read and written by any processor, but each variable is physically associated with a single processor. UPC uses a Single Program Multiple Data (SPMD) model of computation in which the amount of parallelism is fixed at program startup time, typically with a single thread of execution per processor.

In order to express parallelism, UPC extends ISO C 99 with the following constructs:

- An explicitly parallel execution model
- A shared address space
- Synchronization primitives and a memory consistency model
- Memory management primitives

The UPC language evolved from experiences with three other earlier languages that proposed parallel extensions to ISO C 99: AC, Split-C, and Parallel C Preprocessor (PCP). UPC is not a superset of these three languages, but rather an attempt to distill the best characteristics of each. UPC combines the programmability advantages of the shared memory programming paradigm and the control over data layout and performance of the message passing programming paradigm.

Mellanox ScalableUPC is based on Berkely UPC package (see http://upc.lbl.gov/) and contains the following enhancements:

- GasNet library used within UPC integrated with Mellanox FCA which off-loads from UPC collective operations.
  For further information on FCA, please refer to the Mellanox website.
- GasNet library contains MXM conduit which offloads from UPC all P2P operations as well as some synchronization routines. For further information on MXM, please refer to the Mellanox website.

Mellanox OFED 1.8 includes ScalableUPC 2.1, which is installed under: `/opt/mellanox/bupc`.
If you have installed OFED 1.8, you do *not* need to download and install ScalableUPC.

Mellanox ScalableUPC is distributed as source RPM as well and can be downloaded from the Mellanox website.

5.5.1 Installing ScalableUPC

Mellanox ScalableUPC is installed as part of MLNX_OFED package.

Mellanox OFED 1.8.5 includes ScalableUPC Rev 2.2, which is installed under: `/opt/mellanox/bupc`.
If you have installed OFED 1.8.5, you do not need to download and install ScalableUPC.

Mellanox ScalableUPC is distributed as source RPM as well and can be downloaded from the Mellanox website.

Please note, the binary distribution of ScalableUPC is compiled with the following defaults:

- FCA support. FCA is disabled at runtime by default and must be configured prior to using it from the ScalableUPC. For further information, please refer to FCA User Manual.
- MXM support enabled by default

5.5.2 FCA Runtime Parameters

The following parameters can be passed to `"upcrun"` in order to change FCA support behavior:

Table 15 - Runtime Parameters

Parameter	Description
`-fca_enable <0|1>`	Disables/Enables FCA support at runtime (default: disable).
`-fca_np <value>`	Enables FCA support for collective operations if the number of processes in the job is greater than the `fca_np` value (default: 64).
`-fca_verbose <level>`	Sets verbosity level for the FCA modules
`-fca_ops <+/->[op_list]`	`op_list` - comma separated list of collective operations. • `-fca_ops <+/->[op_list]` - Enables/disables only the specified operations • `-fca_ops <+/->` - Enables/disables all operations By default all operations are enabled. Allowed operation names are: barrier (br), bcast (bt), reduce (rc), allgather (ag). Each operation can be also enabled/disabled via environment variable: • GASNET_FCA_ENABLE_BARRIER • GASNET_FCA_ENABLE_BCAST, • GASNET_FCA_ENABLE_REDUCE, **Note:** All the operations are enabled by default.

5.5.2.1 Enabling FCA Operations through Environment Variables in ScalableUPC

This method can be used to control UPC FCA offload from environment using job scheduler srun utility. The valid values are: 1 - enable, 0 - disable.

➢ ***To enable a specific operation with shell environment variables in ScalableUPC:***

```
% export GASNET_FCA_ENABLE_BARRIER=1
% export GASNET_FCA_ENABLE_BCAST=1
% export GASNET_FCA_ENABLE_REDUCE=1
```

5.5.2.2 Controlling FCA Offload in ScalableUPC using Environment Variables

- ***To enable FCA module under ScalableUPC:***

```
% export GASNET_FCA_ENABLE_CMD_LINE=1
```

- ***To set FCA verbose level:***

```
% export GASNET_FCA_VERBOSE_CMD_LINE=10
```

- ***To set the minimal number of processes threshold to activate FCA:***

```
% export GASNET_FCA_NP_CMD_LINE=1
```

ScalableUPC contains modules configuration file (http://modules.sf.net) which can be found at `/opt/mellanox/bupc/2.2/etc/bupc_modulefile`.

5.5.3 Various Executable Examples

The following are various executable examples.

- ***To run a ScalableUPC application without FCA support:***

```
% upcrun -np 128 -fca_enable 0 <executable filename>
```

- ***To run UPC applications with FCA enabled for any number of processes:***

```
% export GASNET_FCA_ENABLE_CMD_LINE=1 GASNET_FCA_NP_CMD_LINE=0
% upcrun -np 64 <executable filename>
```

- ***To run UPC application on 128 processes, verbose mode:***

```
% upcrun -np 128 -fca_enable 1 -fca_np 10 -fca_verbose 5 <executable filename>
```

- ***To run UPC application, offload to FCA Barrier and Broadcast only:***

```
% upcrun -np 128 -fca_ops +barrier,bt <executable filename>
```

6 Working With VPI

VPI allows ConnectX ports to be independently configured as either IB or Eth.

6.1 Port Type Management

ConnectX ports can be individually configured to work as InfiniBand or Ethernet ports. By default both ConnectX ports are initialized as InfiniBand ports. If you wish to change the port type use the `connectx_port_config` script after the driver is loaded.

Running "`/sbin/connectx_port_config -s`" will show current port configuration for all ConnectX devices.

Port configuration is saved in the file: `/etc/infiniband/connectx.conf`. This saved configuration is restored at driver restart only if restarting via "`/etc/init.d/openibd restart`".

Possible port types are:

- eth – Ethernet
- ib – Infiniband
- auto – Link sensing mode - Detect port type based on the attached network type. If no link is detected, the driver retries link sensing every few seconds.

The port link type can be configured for each device in the system at run time using the "`/sbin/connectx_port_config`" script. This utility will prompt for the PCI device to be modified (if there is only one it will be selected automatically).

In the next stage the user will be prompted for the desired mode for each port. The desired port configuration will then be set for the selected device.

This utility also has a non-interactive mode:

```
/sbin/connectx_port_config [[-d|--device <PCI device ID>] -c|--conf <port1,port2>]"
```

6.2 Auto Sensing

Auto Sensing enables the NIC to automatically sense the link type (InfiniBand or Ethernet) based on the link partner and load the appropriate driver stack (InfiniBand or Ethernet).

For example, if the first port is connected to an InfiniBand switch and the second to Ethernet switch, the NIC will automatically load the first switch as InfiniBand and the second as Ethernet.

6.2.1 Enabling Auto Sensing

Upon driver start up:

1. Sense the adapter card's port type:

   If a valid cable or module is connected (QSFP, SFP+, or SFP with EEPROM in the cable/module):

   - Set the port type to the sensed link type (IB/Ethernet)

   Otherwise:

   - Set the port type as default (Ethernet)

During driver run time:

- Sense a link every 3 seconds if no link is sensed/detected
- If sensed, set the port type as sensed

7 Performance

7.1 General System Configurations

The following sections describe recommended configurations for system components and/or interfaces. Different systems may have different features, thus some recommendations below may not be applicable.

7.1.1 PCI Express (PCIe) Capabilities

Table 16 - Recommended PCIe Configuration

PCIe Generation	3.0
Speed	8GT/s
Width	x8 or x16
Max Payload size	256
Max Read Request	4096

For ConnectX3® based network adapters, 40GbE Ethernet adapters it is recommended to use an x16 PCIe slot to benefit from the additional buffers allocated by the CPU.

7.1.2 Memory Configuration

For high performance it is recommended to use the highest memory speed with fewest DIMMs and populate all memory channels for every CPU installed.

For further information, please refer to your vendor's memory configuration instructions or memory configuration tool available Online.

7.1.3 Recommended BIOS Settings

These performance optimizations may result in higher power consumption.

7.1.3.1 General

Set BIOS power management to Maximum Performance.

7.1.3.2 Intel® Sandy Bridge Processors

The following table displays the recommended BIOS settings in machines with Intel code name Sandy Bridge based processors.

Table 17 - Recommended BIOS Settings for Intel Sandy Bridge Processors

	BIOS Option	Values
General	Operating Mode /Power profile	Maximum Performance
Processor	C-States	Disabled
	Turbo mode	Enabled
	Hyper-Threading[1]	HPC: disabled Data Centers: enabled
	CPU frequency select	Max performance
Memory	Memory speed	Max performance
	Memory channel mode	Independent
	Node Interleaving	Disabled / NUMA
	Channel Interleaving	Enabled
	Thermal Mode	Performance

1. Hyper-Threading can increase message rate for multi process applications by having more logical cores. It might increase the latency of a single process, due to lower frequency of a single logical core when hyper-threading is enabled.

7.1.3.3 Intel® Nehalem/Westmere Processors

The following table displays the recommended BIOS settings in machines with Intel Nehalem-based processors.Configuring the Completion Queue Stall Delay.

Table 18 - Recommended BIOS Settings for Intel® Nehalem/Westmere Processors

	BIOS Option	Values
General	Operating Mode /Power profile	Maximum Performance
Processor	C-States	Disabled
	Turbo mode	Disabled
	Hyper-Threading[1]	Disabled Recommended for latency and message rate sensitive applications.
	CPU frequency select	Max performance
Memory	Memory speed	Max performance
	Memory channel mode	Independent
	Node Interleaving	Disabled / NUMA
	Channel Interleaving	Enabled
	Thermal Mode	Performance

1. Hyper-Threading can increase message rate for multi process applications by having more logical cores. It might increase the latency of a single process, due to lower frequency of a single logical core when hyper-threading is enabled.

7.1.3.4 AMD Processors

The following table displays the recommended BIOS settings in machines with AMD based processors.

Table 19 - Recommended BIOS Settings for AMD Processors

	BIOS Option	Values
General	Operating Mode /Power profile	Maximum Performance
Processor	C-States	Disabled
	Turbo mode	Disabled
	HPC Optimizations	Enabled
	CPU frequency select	Max performance

Table 19 - Recommended BIOS Settings for AMD Processors

	BIOS Option	Values
Memory	Memory speed	Max performance
	Memory channel mode	Independent
	Node Interleaving	Disabled / NUMA
	Channel Interleaving	Enabled
	Thermal Mode	Performance

7.2 Performance Tuning for Linux

You can use the Linux *sysctl* command to modify default system network parameters that are set by the operating system in order to improve IPv4 and IPv6 traffic performance. Note, however, that changing the network parameters may yield different results on different systems. The results are significantly dependent on the CPU and chipset efficiency.

7.2.1 Tuning the Network Adapter for Improved IPv4 Traffic Performance

The following changes are recommended for improving IPv4 traffic performance:

- Disable the TCP timestamps option for better CPU utilization:

```
sysctl -w net.ipv4.tcp_timestamps=0
```

- Enable the TCP selective acks option for better throughput:

```
sysctl -w net.ipv4.tcp_sack=1
```

- Increase the maximum length of processor input queues:

```
sysctl -w net.core.netdev_max_backlog=250000
```

- Increase the TCP maximum and default buffer sizes using setsockopt():

```
sysctl  -w net.core.rmem_max=4194304
sysctl  -w net.core.wmem_max=4194304
sysctl  -w net.core.rmem_default=4194304
sysctl  -w net.core.wmem_default=4194304
sysctl  -w net.core.optmem_max=4194304
```

- Increase memory thresholds to prevent packet dropping:

```
sysctl -w net.ipv4.tcp_rmem="4096 87380 4194304"
sysctl -w net.ipv4.tcp_wmem="4096 65536 4194304"
```

- "Enable low latency mode for TCP:

```
sysctl -w net.ipv4.tcp_low_latency=1
```

7.2.2 Tuning the Network Adapter for Improved IPv6 Traffic Performance

The following changes are recommended for improving IPv6 traffic performance:

- Disable the TCP timestamps option for better CPU utilization:

```
sysctl -w net.ipv4.tcp_timestamps=0
```

- Enable the TCP selective acks option for better CPU utilization:

```
sysctl -w net.ipv4.tcp_sack=1
```

7.2.3 Preserving Your Performance Settings after a Reboot

To preserve your performance settings after a reboot, you need to add them to the file `/etc/sysctl.conf` as follows:

```
<sysctl name1>=<value1>
<sysctl name2>=<value2>
<sysctl name3>=<value3>
<sysctl name4>=<value4>
```

For example, "Tuning the Network Adapter for Improved IPv4 Traffic Performance" on page 129 lists the following setting to disable the TCP timestamps option:

```
sysctl -w net.ipv4.tcp_timestamps=0
```

In order to keep the TCP timestamps option disabled after a reboot, add the following line to `/etc/sysctl.conf`:

```
net.ipv4.tcp_timestamps=0
```

7.2.4 Tuning Power Management

Check that the output CPU frequency for each core is equal to the maximum supported and that all core frequencies are consistent.

- Check the maximum supported CPU frequency:

```
#cat /sys/devices/system/cpu/cpu*/cpufreq/cpuinfo_max_freq
```

- Check that core frequencies are consistent:

```
#cat /proc/cpuinfo | grep "cpu MHz"
```

- Check that the output frequencies are the same as the maximum supported.

  If the CPU frequency is not at the maximum, check the BIOS settings according to tables in is section "Recommended BIOS Settings" on page 126 to verify that power state is disabled.

- Check the current CPU frequency to check whether it is configured to max available frequency:

```
#cat /sys/devices/system/cpu/cpu*/cpufreq/cpuinfo_cur_freq
```

7.2.4.1 Setting the Scaling Governor

If the following modules are loaded, CPU scaling is supported, and you can improve performance by setting the scaling mode to performance:

- freq_table
- acpi_cpufreq: this module is architecture dependent.

It is also recommended to disable the module cpuspeed; this module is also architecture dependent.

➢ ***To set the scaling mode to performance, use:***

```
# echo performance > /sys/devices/system/cpu/cpu7/cpufreq/scaling_governor
```

➢ ***To disable cpuspeed, use:***

```
# service cpuspeed stop
```

7.2.4.2 Kernel Idle Loop Tuning

The mlx4_en kernel module has an optional parameter that can tune the kernel idle loop for better latency. This will improve the CPU wake up time but may result in higher power consumption.

To tune the kernel idle loop, set the following options in the `/etc/modprobe.d/mlx4.conf` file:

- For MLNX_OFED 2.0.x

```
options mlx4_core enable_sys_tune=1
```

- For MLNX_EN 1.5.10

```
options mlx4_en enable_sys_tune=1
```

7.2.4.3 OS Controlled Power Management

Some operating systems can override BIOS power management configuration and enable c-states by default, which results in a higher latency.

➢ ***To resolve the high latency issue, please follow the instructions below:***

1. Edit the /boot/grub/grub.conf file or any other bootloader configuration file.
2. Add the following kernel parameters to the bootloader command.

```
intel_idle.max_cstate=0 processor.max_cstate=1
```

3. Reboot the system.

   Example:

```
title RH6.2x64
root (hd0,0)
kernel /vmlinuz-RH6.2x64-2.6.32-220.el6.x86_64
root=UUID=817c207b-c0e8-4ed9-9c33-c589c0bb566f console=tty0
console=ttyS0,115200n8 rhgb intel_idle.max_cstate=0 processor.max_cstate=1
```

7.2.5 Interrupt Moderation

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

To manually set Tx and/or Rx interrupt moderation, use the ethtool utility. For example, the following commands first show the current (default) setting of interrupt moderation on the interface eth1, then turns off Rx interrupt moderation, and last shows the new setting.

```
> ethtool -c eth1
Coalesce parameters for eth1:
Adaptive RX: on  TX: off
...
pkt-rate-low: 400000
pkt-rate-high: 450000

rx-usecs: 16
rx-frames: 88
rx-usecs-irq: 0
rx-frames-irq: 0
...

> ethtool -C eth1 adaptive-rx off rx-usecs 0 rx-frames 0

> ethtool -c eth1
Coalesce parameters for eth1:
Adaptive RX: off TX: off
...
pkt-rate-low: 400000
pkt-rate-high: 450000

rx-usecs: 0
rx-frames: 0
rx-usecs-irq: 0
rx-frames-irq: 0
...
```

7.2.6 Tuning for NUMA Architecture

7.2.6.1 Tuning for Intel® Sandy Bridge Platform

The Intel Sandy Bridge processor has an integrated PCI express controller. Thus every PCIe adapter OS is connected directly to a NUMA node.

On a system with more than one NUMA node, performance will be better when using the local NUMA node to which the PCIe adapter is connected.

In order to identify which NUMA node is the adapter's node the system BIOS should support ACPI SLIT.

- ***To see if your system supports PCIe adapter's NUMA node detection:***

```
# cat /sys/class/net/[interface]/device/numa_node
# cat /sys/devices/[PCI root]/[PCIe function]/numa_node
```

Example for supported system:

```
# cat /sys/class/net/eth3/device//numa_node
0
```

Example for unsupported system:

```
# cat /sys/class/net/ib0/device/numa_node
-1
```

7.2.6.1.1 Improving Application Performance on Remote NUMA Node

Verbs API applications that mostly use polling, will have an impact when using the remote NUMA node.

libmlx4 has a build-in enhancement that recognizes an application that is pinned to a remote NUMA node and activates a flow that improves the out-of-the-box latency and throughput.

However, the NUMA node recognition must be enabled as described in section "Tuning for Intel® Sandy Bridge Platform" on page 132.

In systems which do not support SLIT, the following environment variable should be applied:

```
MLX4_LOCAL_CPUS=0x[bit mask of local NUMA node]
```

Example for local NUMA node which its cores are 0-7:

```
MLX4_LOCAL_CPUS=0xff
```

Additional modification can apply to impact this feature by changing the following environment variable:

```
MLX4_STALL_NUM_LOOP=[integer] (default: 400)
```

The default value is optimized for most applications. However, several applications might benefit from increasing/decreasing this value.

7.2.6.2 Tuning for AMD® Architecture

On AMD architecture there is a difference between a 2 socket system and a 4 socket system.

- With a 2 socket system the PCIe adapter will be connected to socket 0 (nodes 0,1).
- With a 4 socket system the PCIe adapter will be connected either to socket 0 (nodes 0,1) or to socket 3 (nodes 6,7).

7.2.6.3 Recognizing NUMA Node Cores

➢ ***To recognize NUMA node cores, run the following command:***

```
# cat /sys/devices/system/node/node[X]/cpulist | cpumap
```

Example:

```
# cat /sys/devices/system/node/node1/cpulist
1,3,5,7,9,11,13,15
# cat /sys/devices/system/node/node1/cpumap
0000aaaa
```

7.2.6.3.1 Running an Application on a Certain NUMA Node

In order to run an application on a certain NUMA node, the process affinity should be set in either in the command line or an external tool.

For example, if the adapter's NUMA node is 1 and NUMA 1 cores are 8-15 then an application should run with process affinity that uses 8-15 cores only.

➢ ***To run an application, run the following commands:***

```
taskset -c 8-15 ib_write_bw -a
```

or:

```
taskset 0xff00 ib_write_bw -a
```

7.2.7 IRQ Affinity

The affinity of an interrupt is defined as the set of processor cores that service that interrupt. To improve application scalability and latency, it is recommended to distribute interrupt requests (IRQs) between the available processor cores. To prevent the Linux IRQ balancer application from interfering with the interrupt affinity scheme, the IRQ balancer must be turned off.

The following command turns off the IRQ balancer:

```
> /etc/init.d/irqbalance stop
```

The following command assigns the affinity of a single interrupt vector:

```
> echo <hexadecimal bit mask> > /proc/irq/<irq vector>/smp_affinity
```

Bit i in <hexadecimal bit mask> indicates whether processor core i is in <irq vector>'s affinity or not.

7.2.7.1 IRQ Affinity Configuration

It is recommended to set each IRQ to a different core.

For Sandy Bridge or AMD systems set the irq affinity to the adapter's NUMA node:

- For optimizing single-port traffic, run:

```
set_irq_affinity_bynode.sh <numa node> <interface>
```

- For optimizing dual-port traffic, run:

```
set_irq_affinity_bynode.sh <numa node> <interface1> <interface2>
```

- To show the current irq affinity settings, run:

```
show_irq_affinity.sh <interface>
```

7.2.7.2 Auto Tuning Utility

MLNX_OFED 2.0.x introduces a new affinity tool called mlnx_affinity. This tool can automatically adjust your affinity settings for each network interface according to the system architecture.

Usage:

- Start

```
# mlnx_affinity start
```

- Stop

```
# mlnx_affinity stop
```

- Restart

```
# mlnx_affinity restart
```

mlnx_affinity can also be started by driver load/unload

➢ ***To enable mlnx_affinity by default:***

- Add the line below to the /etc/infiniband/openib.conf file.

```
RUN_AFFINITY_TUNER=yes
```

7.2.7.3 Tuning for Multiple Adapters

When optimizing the system performance for using more than one adapter. It is recommended to separate the adapter's core utilization so there will be no interleaving between interfaces.

The following script can be used to separate each adapter's IRQs to different set of cores.

```
# set_irq_affinity_cpulist.sh <cpu list>
<interface>
<cpu list> can be either a comma separated list of single core numbers (0,1,2,3)
or core groups (0-3)
```

Example:

If the system has 2 adapters on the same NUMA node (0-7) each with 2 interfaces run the following:

```
# /etc/init.d/irqbalancer stop
# set_irq_affinity_cpulist.sh 0-1 eth2
# set_irq_affinity_cpulist.sh 2-3 eth3
# set_irq_affinity_cpulist.sh 4-5 eth4
# set_irq_affinity_cpulist.sh 6-7 eth5
```

7.2.8 Tuning Multi-Threaded IP Forwarding

➢ ***To optimize NIC usage as IP forwarding:***

1. Set the following options in /etc/modprobe.d/mlx4.conf:
   - For MLNX_OFED-2.0.x:

```
options mlx4_en  inline_thold=0
options mlx4_core high_rate_steer=1
```

   - For MLNX_EN-1.5.10:

```
options mlx4_en num_lro=0 inline_thold=0
options mlx4_core high_rate_steer=1
```

2. Apply interrupt affinity tuning.
3. Forwarding on the same interface:

```
# set_irq_affinity_bynode.sh <numa node> <interface>
```

4. Forwarding from one interface to another:

```
# set_irq_affinity_bynode.sh <numa node> <interface1> <interface2>
```

5. Disable adaptive interrupt moderation and set status values, using:

```
# ethtool -C adaptive-rx off
```

8 OpenSM – Subnet Manager

8.1 Overview

OpenSM is an InfiniBand compliant Subnet Manager (SM). It is provided as a fixed flow executable called *opensm*, accompanied by a testing application called *osmtest*. OpenSM implements an InfiniBand compliant SM according to the InfiniBand Architecture Specification chapters: Management Model (13), Subnet Management (14), and Subnet Administration (15).

8.2 opensm Description

opensm is an InfiniBand compliant Subnet Manager and Subnet Administrator that runs on top of the Mellanox OFED stack. **opensm** performs the InfiniBand specification's required tasks for initializing InfiniBand hardware. One SM must be running for each InfiniBand subnet.

opensm also provides an experimental version of a performance manager.

opensm defaults were designed to meet the common case usage on clusters with up to a few hundred nodes. Thus, in this default mode, **opensm** will scan the IB fabric, initialize it, and sweep occasionally for changes.

opensm attaches to a specific IB port on the local machine and configures only the fabric connected to it. (If the local machine has other IB ports, **opensm** will ignore the fabrics connected to those other ports). If no port is specified, **opensm** will select the first "best" available port. **opensm** can also present the available ports and prompt for a port number to attach to.

By default, the **opensm** run is logged to two files: `/var/log/messages` and `/var/log/opensm.log`. The first file will register only general major events, whereas the second file will include details of reported errors. All errors reported in this second file should be treated as indicators of IB fabric health issues. (Note that when a fatal and non-recoverable error occurs, **opensm** will exit). Both log files should include the message "SUBNET UP" if **opensm** was able to setup the subnet correctly.

8.2.1 opensm Syntax

```
opensm [OPTIONS]
```

where OPTIONS are:

```
--version
          Prints OpenSM version and exits.

--config, -F <file-name>
          The name of the OpenSM config file. When not specified
          /etc/opensm/opensm.conf will be used (if exists).

--create-config, -c <file-name>
          OpenSM will dump its configuration to the specified file and exit.
          This is a way to generate OpenSM configuration file template.

--guid, -g <GUID in hex>
          This option specifies the local port GUID value
          with which OpenSM should bind.  OpenSM may be
```

```
          bound to 1 port at a time.
          If GUID given is 0, OpenSM displays a list
          of possible port GUIDs and waits for user input.
          Without -g, OpenSM tries to use the default port.

--lmc, -l <LMC>
          This option specifies the subnet's LMC value.
          The number of LIDs assigned to each port is 2^LMC.
          The LMC value must be in the range 0-7.
          LMC values > 0 allow multiple paths between ports.
          LMC values > 0 should only be used if the subnet
          topology actually provides multiple paths between
          ports, i.e. multiple interconnects between switches.
          Without -l, OpenSM defaults to LMC = 0, which allows
          one path between any two ports.

--priority, -p <PRIORITY>
          This option specifies the SM's PRIORITY.
          This will effect the handover cases, where master
          is chosen by priority and GUID.  Range goes
          from 0 (lowest priority) to 15 (highest).

--smkey, -k <SM_Key>
          This option specifies the SM's SM_Key (64 bits).
          This will effect SM authentication.
          Note that OpenSM version 3.2.1 and below used the
          default value '1' in a host byte order, it is fixed
          now but you may need this option to interoperate
          with old OpenSM running on a little endian machine.

--reassign_lids, -r
          This option causes OpenSM to reassign LIDs to all
          end nodes. Specifying -r on a running subnet
          may disrupt subnet traffic.
          Without -r, OpenSM attempts to preserve existing
          LID assignments resolving multiple use of same LID.

--routing_engine, -R <engine name>
          This option chooses routing engine(s) to use instead of default
          Min Hop algorithm.  Multiple routing engines can be specified
          separated by commas so that specific ordering of routing
          algorithms will be tried if earlier routing engines fail.
          If all configured routing engines fail, OpenSM will always
          attempt to route with Min Hop unless 'no_fallback' is
          included in the list of routing engines.
          Supported engines: updn, file, ftree, lash, dor, torus-2QoS
```

```
--do_mesh_analysis
          This option enables additional analysis for the lash
          routing engine to precondition switch port assignments
          in regular cartesian meshes which may reduce the number
          of SLs required to give a deadlock free routing

--lash_start_vl <vl number>
          Sets the starting VL to use for the lash routing algorithm.
          Defaults to 0.
--sm_sl <sl number>
          Sets the SL to use to communicate with the SM/SA. Defaults to 0.

--connect_roots, -z
          This option enforces routing engines (up/down and
          fat-tree) to make connectivity between root switches
          and in this way be IBA compliant. In many cases,
          this can violate "pure" deadlock free algorithm, so
          use it carefully.

--ucast_cache, -A
          This option enables unicast routing cache to prevent
          routing recalculation (which is a heavy task in a
          large cluster) when there was no topology change
          detected during the heavy sweep, or when the topology
          change does not require new routing calculation,
          e.g. in case of host reboot.
          This option becomes very handy when the cluster size
          is thousands of nodes.

--lid_matrix_file, -M <file name>
          This option specifies the name of the lid matrix dump file
          from where switch lid matrices (min hops tables will be
          loaded.

--lfts_file, -U <file name>
          This option specifies the name of the LFTs file
          from where switch forwarding tables will be loaded.

--sadb_file, -S <file name>
          This option specifies the name of the SA DB dump file
          from where SA database will be loaded.

--root_guid_file, -a <path to file>
          Set the root nodes for the Up/Down or Fat-Tree routing
          algorithm to the guids provided in the given file (one
```

```
          to a line)

--cn_guid_file, -u <path to file>
          Set the compute nodes for the Fat-Tree routing algorithm
          to the guids provided in the given file (one to a line)

--io_guid_file, -G <path to file>
          Set the I/O nodes for the Fat-Tree routing algorithm
          to the guids provided in the given file (one to a line)

--port-shifting
          Attempt to shift port routes around to remove alignment problems
          in routing tables

--scatter-ports <random seed>
          Randomize best port chosen for a route

--max_reverse_hops, -H <hop_count>
          Set the max number of hops the wrong way around
          an I/O node is allowed to do (connectivity for I/O nodes on top swithces)

--ids_guid_file, -m <path to file>
          Name of the map file with set of the IDs which will be used
          by Up/Down routing algorithm instead of node GUIDs
          (format: <guid> <id> per line)

--guid_routing_order_file, -X <path to file>
          Set the order port guids will be routed for the MinHop
          and Up/Down routing algorithms to the guids provided in the
          given file (one to a line)

--torus_config <path to file>
          This option defines the file name for the extra configuration
          info needed for the torus-2QoS routing engine.   The default
          name is '/etc/opensm/torus-2QoS.conf'

--once, -o
          This option causes OpenSM to configure the subnet
          once, then exit.  Ports remain in the ACTIVE state.

--sweep, -s <interval>
          This option specifies the number of seconds between
          subnet sweeps.  Specifying -s 0 disables sweeping.
          Without -s, OpenSM defaults to a sweep interval of
          10 seconds.
```

```
--timeout, -t <milliseconds>
          This option specifies the time in milliseconds
          used for transaction timeouts.
          Specifying -t 0 disables timeouts.
          Without -t, OpenSM defaults to a timeout value of
          200 milliseconds.

--retries <number>
          This option specifies the number of retries used
          for transactions.
          Without --retries, OpenSM defaults to 3 retries
          for transactions.

--maxsmps, -n <number>
          This option specifies the number of VL15 SMP MADs
          allowed on the wire at any one time.
          Specifying --maxsmps 0 allows unlimited outstanding
          SMPs.
          Without --maxsmps, OpenSM defaults to a maximum of
          4 outstanding SMPs.

--rereg_on_guid_migr
          This option if enabled, forces OpenSM to send port info
          with client reregister bit set to all nodes in the
          fabric when alias Guid migrates from one physical port
          to another.

--aguid_inout_notice
          This option enables sending GID IN/OUT notices on Alias GUIDs
          register/delete request to registered clients.

--sm_assign_guid_func [uniq_count | base_port]
          Specifies the algorithm that SM will use when it comes to choose
          SM assigned alias GUIDs. The default is uniq_count.

--console, -q [off|local]
          This option activates the OpenSM console (default off).

--ignore-guids, -i <equalize-ignore-guids-file>
          This option provides the means to define a set of ports
          (by guid) that will be ignored by the link load
          equalization algorithm.

--hop_weights_file, -w <path to file>
          This option provides the means to define a weighting
          factor per port for customizing the least weight
          hops for the routing.
```

```
--port_search_ordering_file, -O <path to file>
          This option provides the means to define a mapping
          between ports and dimension (Order) for controlling
          Dimension Order Routing (DOR).
          Moreover this option provides the means to define non
          default routing port order.

--dimn_ports_file, -O <path to file> (DEPRECATED)
          This option provides the means to define a mapping
          between ports and dimension (Order) for controlling
          Dimension Order Routing (DOR).

--honor_guid2lid, -x
          This option forces OpenSM to honor the guid2lid file,
          when it comes out of Standby state, if such file exists
          under OSM_CACHE_DIR, and is valid. By default, this is FALSE.

--const_multicast
          This option forces OpenSM to conserver previously built
          multicast trees

--log_file, -f <log-file-name>
          This option defines the log to be the given file.
          By default, the log goes to /var/log/opensm.log.
          For the log to go to standard output use -f stdout.

--log_limit, -L <size in MB>
          This option defines maximal log file size in MB. When
          specified the log file will be truncated upon reaching
          this limit.

--erase_log_file, -e
          This option will cause deletion of the log file
          (if it previously exists). By default, the log file
          is accumulative.

--Pconfig, -P <partition-config-file>
          This option defines the optional partition configuration file.
          The default name is '/etc/opensm/partitions.conf'.

--no_part_enforce, -N (DEPRECATED)
          This option disables partition enforcement on switch external ports.
```

```
--part_enforce, -Z [both, in, out, off]
          This option indicates the partition enforcement type (for switches)
          Enforcement type can be outbound only (out), inbound only (in), both or
          disabled (off). Default is both.

--allow_both_pkeys, -W
          This option indicates whether both full and limited membership
          on the same partition can be configured in the PKeyTable.
          Default is not to allow both pkeys.

--qos, -Q
          This option enables QoS setup.

--qos_policy_file, -Y <QoS-policy-file>
          This option defines the optional QoS policy file.
          The default name is '/etc/opensm/qos-policy.conf'.

--congestion_control
          (EXPERIMENTAL) This option enables congestion control configuration.

--cc_key <key>
          (EXPERIMENTAL) This option configures the CCkey to use when configuring
          congestion control.

--stay_on_fatal, -y
          This option will cause SM not to exit on fatal initialization
          issues: if SM discovers duplicated guids or 12x link with
          lane reversal badly configured.
          By default, the SM will exit on these errors.

--daemon, -B
          Run in daemon mode - OpenSM will run in the background.

--inactive, -I
           Start SM in inactive rather than normal init SM state.

--perfmgr
           Start with PerfMgr enabled.

--perfmgr_sweep_time_s <sec.>
           PerfMgr sweep interval in seconds.

--prefix_routes_file <path to file>
          This option specifies the prefix routes file.
          Prefix routes control how the SA responds to path record
          queries for off-subnet DGIDs.  Default file is:
              /etc/opensm/prefix-routes.conf
```

```
--consolidate_ipv6_snm_req
          Use shared MLID for IPv6 Solicited Node Multicast groups
          per MGID scope and P_Key.

--consolidate_ipv4_mask
          Use mask for IPv4 multicast groups multiplexing
          per MGID scope and P_Key.

--pid_file <path to file>
          Specifies the file that contains the process ID of the
          opensm daemon.The default is /var/run/opensm.pid
--max_seq_redisc
          Specifies the maximum number of failed discovery loops
          done by the SM, before completing the whole heavy sweep cycle

--mc_secondary_root_guid <GUID in hex>,
          This option defines the guid of the multicast secondary root switch

--mc_primary_root_guid <GUID in hex>,
          This option defines the guid of the multicast primary root switch

--guid_routing_order_no_scatter
           Don't use scatter for ports defined in guid_routing_order file

--sa_pr_full_world_queries_allowed
          This option allows OpenSM to respond full World Path Record queries
          (path record for each pair of ports in a fabric).

--enable_crashd
          This option causes OpenSM to run Crash Daemon child process that allows
          backtrace dump in case of fatal terminating signals.

--log_prefix <prefix text>
          Prefix to syslog messages from OpenSM.

--verbose, -v
          This option increases the log verbosity level.
          The -v option may be specified multiple times
          to further increase the verbosity level.
          See the -D option for more information about
          log verbosity.

--V, -V
          This option sets the maximum verbosity level and
          forces log flushing.
          The -V is equivalent to '-D 0xFF -d 2'.
          See the -D option for more information about
          log verbosity.

--D, -D <flags>
```

```
          This option sets the log verbosity level.
          A flags field must follow the -D option.
          A bit set/clear in the flags enables/disables a
          specific log level as follows:
          BIT    LOG LEVEL ENABLED
          ----   ----------------
          0x01 - ERROR (error messages)
          0x02 - INFO (basic messages, low volume)
          0x04 - VERBOSE (interesting stuff, moderate volume)
          0x08 - DEBUG (diagnostic, high volume)
          0x10 - FUNCS (function entry/exit, very high volume)
          0x20 - FRAMES (dumps all SMP and GMP frames)
          0x40 - ROUTING (dump FDB routing information)
          0x80 - currently unused.
          Without -D, OpenSM defaults to ERROR + INFO (0x3).
          Specifying -D 0 disables all messages.
          Specifying -D 0xFF enables all messages (see -V).
          High verbosity levels may require increasing
          the transaction timeout with the -t option.

--debug, -d <number>
          This option specifies a debug option.
          These options are not normally needed.
          The number following -d selects the debug
          option to enable as follows:
          OPT   Description
          ---    ----------------
          -d0  - Ignore other SM nodes
          -d1  - Force single threaded dispatching
          -d2  - Force log flushing after each log message
          -d3  - Disable multicast support
          -d10 - Put OpenSM in testability mode
          Without -d, no debug options are enabled

--help, -h, -?
          Display this usage info then exit.
```

8.2.2 Environment Variables

The following environment variables control `opensm` behavior:

- OSM_TMP_DIR

  Controls the directory in which the temporary files generated by opensm are created. These files are: `opensm-subnet.lst`, `opensm.fdbs`, and `opensm.mcfdbs`. By default, this directory is `/var/log`.

- OSM_CACHE_DIR

`opensm` stores certain data to the disk such that subsequent runs are consistent. The default directory used is `/var/cache/opensm`. The following file is included in it:

- `guid2lid` – stores the LID range assigned to each GUID

8.2.3 Signaling

When OpenSM receives a HUP signal, it starts a new heavy sweep as if a trap has been received or a topology change has been found.

Also, SIGUSR1 can be used to trigger a reopen of `/var/log/opensm.log` for logrotate purposes.

8.2.4 Running opensm

The defaults of **opensm** were designed to meet the common case usage on clusters with up to a few hundred nodes. Thus, in this default mode, **opensm** will scan the IB fabric, initialize it, and sweep occasionally for changes.

To run **opensm** in the default mode, simply enter:

```
host1# opensm
```

Note that **opensm** needs to be run on at least one machine in an IB subnet.

By default, an **opensm** run is logged to two files: `/var/log/messages` and `/var/log/opensm.log`. The first file, `message`, registers only general major events; the second file, `opensm.log`, includes details of reported errors. All errors reported in `opensm.log` should be treated as indicators of IB fabric health. Both log files should include the message “SUBNET UP” if opensm was able to setup the subnet correctly.

If a fatal, non-recoverable error occurs, opensm exits.

8.2.4.1 Running OpenSM As Daemon

OpenSM can also run as daemon. To run OpenSM in this mode, enter:

```
host1# /etc/init.d/opensmd start
```

8.3 osmtest Description

osmtest is a test program for validating the InfiniBand Subnet Manager and Subnet Administrator. **osmtest** provides a test suite for **opensm**. It can create an inventory file of all available nodes, ports, and PathRecords, including all their fields. It can also verify the existing inventory with all the object fields, and matches it to a pre-saved one. See Section 8.3.2.

osmtest has the following test flows:

- Multicast Compliancy test
- Event Forwarding test
- Service Record registration test
- RMPP stress test

- Small SA Queries stress test

8.3.1 Syntax

```
osmtest [OPTIONS]
```

where OPTIONS are:

```
-f, --flow       This option directs osmtest to run a specific flow:
                 Flow  Description:
                 c = create an inventory file with all nodes, ports and
                  paths
                 a = run all validation tests (expecting an input
                  inventory)
                 v = only validate the given inventory file
                 s = run service registration, deregistration, and lease
                  test
                 e = run event forwarding test
                 f = flood the SA with queries according to the stress mode
                 m = multicast flow
                 q = QoS info: dump VLArb and SLtoVL tables
                 t = run trap 64/65 flow (this flow requires running of
                  external tool)
                    Default = all flows except QoS
-w, --wait       This option specifies the wait time for trap 64/65 in
                seconds. It is used only when running -f t - the trap 64/
                65 flow Default = 10 sec
-d, --debug      This  option  specifies  a  debug option.  These options
               are not normally needed.  The number  following  -d
               selects  the  debug option to enable as follows:
                 OPT   Description
                 ---   -----------------
                 -d0   Ignore other SM nodes
                 -d1   Force single threaded dispatching
                 -d2   Force log flushing after each log message
                 -d3   Disable multicast support
-m, --max_lid    This  option specifies the maximal LID number to be
               searched for during inventory file build (Default = 100)
-g, --guid       This option specifies the  local  port  GUID  value  with
               which OpenSM  should  bind.   OpenSM may be bound to
                1 port at a time. If GUID given is 0, OpenSM displays  a
                  list  of  possible  port GUIDs  and waits for user input.
               Without -g, OpenSM tries to use the default port.
-p, --port       This option displays a menu of possible local port  GUID
               values with which osmtest could bind
-i, --inventory   This  option  specifies the name of the inventory file
                Normally, osmtest expects to find an inventory file,
                which osmtest uses to validate real-time information
```

```
                received from the SA during testing. If -i  is  not
                specified,  osmtest  defaults  to  the  file
                osmtest.dat.See -c option for related information
-s, --stress      This option runs the specified stress test instead of the
               normal test suite Stress test options are as follows:
                  OPT    Description
                  ---    -----------------
                  -s1    Single-MAD response SA queries
                  -s2    Multi-MAD (RMPP) response SA queries
                  -s3    Multi-MAD (RMPP) Path Record SA queries
                  Without -s, stress testing is not performed
-M, --Multicast_ModeThis option specify length of Multicast test:
                  OPT    Description
                  ---    -----------------
                  -M1    Short Multicast Flow (default) - single mode
                  -M2    Short Multicast Flow - multiple mode
                  -M3    Long Multicast Flow - single mode
                  -M4    Long Multicast Flow - multiple mode
                Single mode - Osmtest is tested alone, with no other  apps
                  that interact with OpenSM MC
                  Multiple  mode  -  Could  be  run  with other apps using
               MC with OpenSM. Without -M, default flow testing is per
               formed
-t, --timeout     This option specifies the time in milliseconds used for
               transaction  timeouts.  Specifying -t 0 disables
               timeouts. Without -t, OpenSM defaults to a timeout value
               of 200 milliseconds.
-l, --log_file    This option defines the log to be the given  file.   By
                default the log goes to /var/log/osm.log.  For the log to
               go to standard output use -f stdout.
-v, --verbose     This option increases the log verbosity level.   The  -v
               option may  be  specified  multiple  times to further
               increase the verbosity level.  See the -vf option for
               more  information  about log verbosity.
-V                This  option  sets  the  maximum  verbosity level and
               forces log flushing.  The -V is equivalent to '-vf 0xFF -
               d 2'.  See the -vf option for more information about log
               verbosity.
-vf               This  option  sets  the log verbosity level.  A flags
               field must follow the -D option.  A bit set/clear in the
               flags enables/disables a specific log level as follows:
                  BIT    LOG LEVEL ENABLED
                  ----   -----------------
                  0x01 - ERROR (error messages)
                  0x02 - INFO (basic messages, low volume)
                  0x04 - VERBOSE (interesting stuff, moderate volume)
```

```
                    0x08 - DEBUG (diagnostic, high volume)
                    0x10 - FUNCS (function entry/exit, very high volume)
                    0x20 - FRAMES (dumps all SMP and GMP frames)
                    0x40 - ROUTING (dump FDB routing information)
                    0x80 - currently unused.
                  Without  -vf,  osmtest defaults to ERROR + INFO (0x3)
                  Specifying -vf 0 disables all messages Specifying
                  -vf 0xFF enables all messages  (see -V) High verbosity
                  levels may require increasing the transaction timeout
                  with the -t option
-h, --help        Display this usage info then exit.
```

8.3.2 Running osmtest

To run **osmtest** in the default mode, simply enter:

```
host1# osmtest
```

The default mode runs all the flows except for the Quality of Service flow (see Section 8.6).

After installing **opensm** (and if the InfiniBand fabric is stable), it is recommended to run the following command in order to generate the inventory file:

```
host1# osmtest -f c
```

Immediately afterwards, run the following command to test **opensm**:

```
host1# osmtest -f a
```

Finally, it is recommended to occasionally run "**`osmtest -v`**" (with verbosity) to verify that nothing in the fabric has changed.

8.4 Partitions

OpenSM enables the configuration of partitions (PKeys) in an InfiniBand fabric. By default, OpenSM searches for the partitions configuration file under the name `/usr/etc/opensm/partitions.conf`. To change this filename, you can use **opensm** with the '--Pconfig' or '-P' flags.

The default partition is created by OpenSM unconditionally, even when a partition configuration file does not exist or cannot be accessed.

The default partition has a P_Key value of 0x7fff. The port out of which runs OpenSM is assigned full membership in the default partition. All other end-ports are assigned partial membership.

8.4.1 File Format

Notes:

- Line content followed after '#' character is comment and ignored by parser.

General File Format

```
<Partition Definition>:<PortGUIDs list> ;
```

Partition Definition:

```
[PartitionName][=PKey][,flag[=value]][,defmember=full|limited]
```

where

```
PartitionName     string, will be used with logging. When omitted, an           empty
string will be used.
PKey              P_Key value for this partition. Only low 15 bits will
be used. When omitted, P_Key will be autogenerated.
flag              used to indicate IPoIB capability of this partition.
defmember=full|limited
                  specifies default membership for port guid list. Default is limited.
```

Currently recognized flags are:

```
ipoib             indicates that this partition may be used for IPoIB, as a
result IPoIB capable MC group will be created.
rate=<val>        specifies rate for this IPoIB MC group (default is 3 (10GBps))
mtu=<val>         specifies MTU for this IPoIB MC group (default is 4 (2048))
sl=<val>          specifies SL for this IPoIB MC group (default is 0)
scope=<val>       specifies scope for this IPoIB MC group (default is 2 (link local))
```

Note that values for rate, MTU, and scope should be specified as defined in the IBTA specification (for example, mtu=4 for 2048). To use 4K MTU, edit that entry to `"mtu=5"` (5 indicates 4K MTU to that specific partition).

PortGUIDs list:

```
PortGUID          GUID of partition member EndPort. Hexadecimal numbers should start
from 0x, decimal numbers are accepted too.
full or limited   indicates full or limited membership for this port. When omitted (or
unrecognized) limited membership is assumed.
```

There are two useful keywords for PortGUID definition:

- 'ALL' means all end-ports in this subnet
- 'SELF' means subnet manager's port

An empty list means that there are no ports in this partition.

Notes:

- White space is permitted between delimiters ('=', ',',':',';').
- The line can be wrapped after ':' after a Partition Definition and between.
- A PartitionName *does not* need to be unique, but PKey *does* need to be unique.
- If a PKey is repeated then the associated partition configurations will be merged and the first PartitionName will be used (see also next note).
- It is possible to split a partition configuration in more than one definition, but then they PKey should be explicitly specified (otherwise different PKey values will be generated for those definitions).

Examples:

```
Default=0x7fff : ALL, SELF=full ;
NewPartition , ipoib : 0x123456=full, 0x3456789034=limi,  0x2134af2306;

YetAnotherOne = 0x300 : SELF=full ;
YetAnotherOne = 0x300 : ALL=limited ;

ShareIO = 0x80 , defmember=full : 0x123451, 0x123452;
# 0x123453, 0x123454 will be limited
ShareIO = 0x80 : 0x123453, 0x123454, 0x123455=full;
# 0x123456, 0x123457 will be limited
ShareIO   =   0x80   :   defmember=limited   :   0x123456,   0x123457,
0x123458=full;
ShareIO = 0x80 , defmember=full : 0x123459, 0x12345a;
ShareIO  =  0x80  ,  defmember=full  :   0x12345b,   0x12345c=limited,
0x12345d;
```

The following rule is equivalent to how OpenSM used to run prior to the partition manager:

```
Default=0x7fff,ipoib:ALL=full;
```

8.5 Routing Algorithms

OpenSM offers six routing engines:

1. "Min Hop Algorithm"

   Based on the minimum hops to each node where the path length is optimized.

2. "UPDN Algorithm"

   Based on the minimum hops to each node, but it is constrained to ranking rules. This algorithm should be chosen if the subnet is not a pure Fat Tree, and a deadlock may occur due to a loop in the subnet.

3. "Fat-tree Routing Algorithm"

   This algorithm optimizes routing for a congestion-free "shift" communication pattern. It should be chosen if a subnet is a symmetrical Fat Tree of various types, not just a K-ary-N-Tree: non-constant K, not fully staffed, and for any CBB ratio. Similar to UPDN, Fat Tree routing is constrained to ranking rules.

4. "LASH Routing Algorithm"

   Uses InfiniBand virtual layers (SL) to provide deadlock-free shortest-path routing while also distributing the paths between layers. LASH is an alternative deadlock-free, topology-agnostic routing algorithm to the non-minimal UPDN algorithm. It avoids the use of a potentially congested root node.

5. "DOR Routing Algorithm"

   Based on the Min Hop algorithm, but avoids port equalization except for redundant links between the same two switches. This provides deadlock free routes for hypercubes when the fabric is cabled as a hypercube and for meshes when cabled as a mesh.

6. "Torus-2QoS Routing Algorithm"

   Based on the DOR Unicast routing algorithm specialized for 2D/3D torus topologies. Torus-2QoS provides deadlock-free routing while supporting two quality of service (QoS) levels. Additionally,

   it can route around multiple failed fabric links or a single failed fabric switch without introducing deadlocks, and without changing path SLvalues granted before the failure.

   OpenSM provides an optional unicast routing cache (enabled by -A or --ucast_cache options). When enabled, unicast routing cache prevents routing recalculation (which is a heavy task in a large cluster) when there was no topology change detected during the heavy sweep, or when the topology change does not require new routing calculation, e.g. when one or more CAs/RTRs/leaf switches going down, or one or more of these nodes coming back after being down. A very common case that is handled by the unicast routing cache is host reboot, which otherwise would cause two full routing recalculations: one when the host goes down, and the other when the host comes back online.

OpenSM also supports a file method which can load routes from a table – see Modular Routing Engine below.

The basic routing algorithm is comprised of two stages:

1. MinHop matrix calculation. How many hops are required to get from each port to each LID? The algorithm to fill these tables is different if you run standard (min hop) or Up/Down. For standard routing, a "relaxation" algorithm is used to propagate min hop from every destination LID through neighbor switches. For Up/Down routing, a BFS from every target is used. The BFS tracks link direction (up or down) and avoid steps that will perform up after a down step was used.
2. Once MinHop matrices exist, each switch is visited and for each target LID a decision is made as to what port should be used to get to that LID. This step is common to standard and

Up/Down routing. Each port has a counter counting the number of target LIDs going through it. When there are multiple alternative ports with same MinHop to a LID, the one with less previously assigned ports is selected.

If LMC > 0, more checks are added. Within each group of LIDs assigned to same target port:

a. Use only ports which have same MinHop
b. First prefer the ones that go to different systemImageGuid (then the previous LID of the same LMC group)
c. If none, prefer those which go through another NodeGuid
d. Fall back to the number of paths method (if all go to same node).

8.5.1 Effect of Topology Changes

OpenSM will preserve existing routing in any case where there is no change in the fabric switches unless the -r (--reassign_lids) option is specified.

```
-r, --reassign_lids
                This option causes OpenSM to reassign LIDs to all end nodes. Specify-
ing -r on a running subnet may disrupt subnet traffic. Without -r, OpenSM attempts to
preserve existing LID assignments resolving multiple use of same LID.
```

If a link is added or removed, OpenSM does not recalculate the routes that do not have to change. A route has to change if the port is no longer UP or no longer the MinHop. When routing changes are performed, the same algorithm for balancing the routes is invoked.

In the case of using the file based routing, any topology changes are currently ignored. The 'file' routing engine just loads the LFTs from the file specified, with no reaction to real topology. Obviously, this will not be able to recheck LIDs (by GUID) for disconnected nodes, and LFTs for non-existent switches will be skipped. Multicast is not affected by 'file' routing engine (this uses min hop tables).

8.5.2 Min Hop Algorithm

The Min Hop algorithm is invoked by default if no routing algorithm is specified. It can also be invoked by specifying '-R minhop'.

The Min Hop algorithm is divided into two stages: computation of min-hop tables on every switch and LFT output port assignment. Link subscription is also equalized with the ability to override based on port GUID. The latter is supplied by:

```
-i <equalize-ignore-guids-file>
-ignore-guids <equalize-ignore-guids-file>
                This option provides the means to define a set of ports (by guids)
that will be ignored by the link load equalization algorithm.
```

LMC awareness routes based on (remote) system or switch basis.

8.5.3 UPDN Algorithm

The UPDN algorithm is designed to prevent deadlocks from occurring in loops of the subnet. A loop-deadlock is a situation in which it is no longer possible to send data between any two hosts connected through the loop. As such, the UPDN routing algorithm should be used if the subnet is not a pure Fat Tree, and one of its loops may experience a deadlock (due, for example, to high pressure).

The UPDN algorithm is based on the following main stages:

1. Auto-detect root nodes - based on the CA hop length from any switch in the subnet, a statistical histogram is built for each switch (hop num vs number of occurrences). If the histogram reflects a specific column (higher than others) for a certain node, then it is marked as a root node. Since the algorithm is statistical, it may not find any root nodes. The list of the root nodes found by this auto-detect stage is used by the ranking process stage.

The user can override the node list manually

If this stage cannot find any root nodes, and the user did not specify a guid list file, OpenSM defaults back to the Min Hop routing algorithm.

2. Ranking process - All root switch nodes (found in stage 1) are assigned a rank of 0. Using the BFS algorithm, the rest of the switch nodes in the subnet are ranked incrementally. This ranking aids in the process of enforcing rules that ensure loop-free paths.
3. Min Hop Table setting - after ranking is done, a BFS algorithm is run from each (CA or switch) node in the subnet. During the BFS process, the FDB table of each switch node traversed by BFS is updated, in reference to the starting node, based on the ranking rules and guid values.

At the end of the process, the updated FDB tables ensure loop-free paths through the subnet.

Up/Down routing does not allow LID routing communication between switches that are located inside spine “switch systems”. The reason is that there is no way to allow a LID route between them that does not break the Up/Down rule. One ramification of this is that you cannot run SM on switches other than the leaf switches of the fabric.

8.5.3.1 UPDN Algorithm Usage

Activation through OpenSM

- Use '-R updn' option (instead of old '-u') to activate the UPDN algorithm.
- Use '-a <root_guid_file>' for adding an UPDN guid file that contains the root nodes for ranking. If the `-a' option is not used, OpenSM uses its auto-detect root nodes algorithm.

Notes on the guid list file:

1. A valid guid file specifies one guid in each line. Lines with an invalid format will be discarded.
2. The user should specify the root switch guids. However, it is also possible to specify CA guids; OpenSM will use the guid of the switch (if it exists) that connects the CA to the subnet as a root node.

8.5.4 Fat-tree Routing Algorithm

The fat-tree algorithm optimizes routing for "shift" communication pattern. It should be chosen if a subnet is a symmetrical or almost symmetrical fat-tree of various types. It supports not just K-ary-N-Trees, by handling for non-constant K, cases where not all leafs (CAs) are present, any Constant Bisectional Ratio (CBB)ratio. As in UPDN, fat-tree also prevents credit-loop-deadlocks.

If the root guid file is not provided ('`-a`' or '`--root_guid_file`' options), the topology has to be pure fat-tree that complies with the following rules:

- Tree rank should be between two and eight (inclusively)
- Switches of the same rank should have the same number of UP-going port groups[1], unless they are root switches, in which case the shouldn't have UP-going ports at all.
- Switches of the same rank should have the same number of DOWN-going port groups, unless they are leaf switches.
- Switches of the same rank should have the same number of ports in each UP-going port group.
- Switches of the same rank should have the same number of ports in each DOWN-going port group.
- All the CAs have to be at the same tree level (rank).

If the root guid file is provided, the topology does not have to be pure fat-tree, and it should only comply with the following rules:

- Tree rank should be between two and eight (inclusively)
- All the Compute Nodes[2] have to be at the same tree level (rank). Note that non-compute node CAs are allowed here to be at different tree ranks.

Topologies that do not comply cause a fallback to min hop routing. Note that this can also occur on link failures which cause the topology to no longer be a “pure” fat-tree.

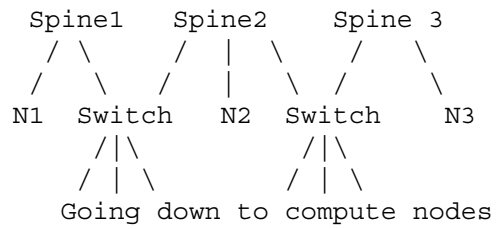
Note that although fat-tree algorithm supports trees with non-integer CBB ratio, the routing will not be as balanced as in case of integer CBB ratio. In addition to this, although the algorithm allows leaf switches to have any number of CAs, the closer the tree is to be fully populated, the more effective the "shift" communication pattern will be. In general, even if the root list is provided, the closer the topology to a pure and symmetrical fat-tree, the more optimal the routing will be.

The algorithm also dumps compute node ordering file (`opensm-ftree-ca-order.dump`) in the same directory where the OpenSM log resides. This ordering file provides the CN order that may be used to create efficient communication pattern, that will match the routing tables.

1. Ports that are connected to the same remote switch are referenced as ‘port group’
2. List of compute nodes (CNs) can be specified by ‘-u’ or ‘--cn_guid_file’ OpenSM options.

8.5.4.1 Routing between non-CN Nodes

The use of the cn_guid_file option allows non-CN nodes to be located on different levels in the fat tree. In such case, it is not guaranteed that the Fat Tree algorithm will route between two non-CN nodes. In the scheme below, N1, N2 and N3 are non-CN nodes. Although all the CN have routes to and from them, there will not necessarily be a route between N1,N2 and N3. Such routes would require to use at least one of the switches the wrong way around.



To solve this problem, a list of non-CN nodes can be specified by \'-G\' or \'--io_guid_file\' option. These nodes will be allowed to use switches the wrong way around a specific number of times (specified by \'-H\' or \'--max_reverse_hops\'. With the proper max_reverse_hops and io_guid_file values, you can ensure full connectivity in the Fat Tree. In the scheme above, with a max_reverse_hop of 1, routes will be instanciated between N1<->N2 and N2<->N3. With a max_reverse_hops value of 2, N1,N2 and N3 will all have routes between them.

Using max_reverse_hops creates routes that use the switch in a counter-stream way. This option should never be used to connect nodes with high bandwidth traffic between them! It should only be used to allow connectivity for HA purposes or similar. Also having routes the other way around can cause credit loops.

8.5.4.2 Activation through OpenSM

- Use '-R ftree' option to activate the fat-tree algorithm

LMC > 0 is not supported by fat-tree routing. If this is specified, the default routing algorithm is invoked instead.

8.5.5 LASH Routing Algorithm

LASH is an acronym for LAyered SHortest Path Routing. It is a deterministic shortest path routing algorithm that enables topology agnostic deadlock-free routing within communication networks.

When computing the routing function, LASH analyzes the network topology for the shortest-path routes between all pairs of sources / destinations and groups these paths into virtual layers in such a way as to avoid deadlock.

LASH analyzes routes and ensures deadlock freedom between switch pairs. The link from HCA between and switch does not need virtual layers as deadlock will not arise between switch and HCA.

In more detail, the algorithm works as follows:

1. LASH determines the shortest-path between all pairs of source / destination switches. Note, LASH ensures the same SL is used for all SRC/DST - DST/SRC pairs and there is no guarantee that the return path for a given DST/SRC will be the reverse of the route SRC/DST.
2. LASH then begins an SL assignment process where a route is assigned to a layer (SL) if the addition of that route does not cause deadlock within that layer. This is achieved by maintaining and analysing a channel dependency graph for each layer. Once the potential addition of a path could lead to deadlock, LASH opens a new layer and continues the process.
3. Once this stage has been completed, it is highly likely that the first layers processed will contain more paths than the latter ones. To better balance the use of layers, LASH moves paths from one layer to another so that the number of paths in each layer averages out.

Note that the implementation of LASH in opensm attempts to use as few layers as possible. This number can be less than the number of actual layers available.

In general LASH is a very flexible algorithm. It can, for example, reduce to Dimension Order Routing in certain topologies, it is topology agnostic and fares well in the face of faults.

It has been shown that for both regular and irregular topologies, LASH outperforms Up/Down. The reason for this is that LASH distributes the traffic more evenly through a network, avoiding the bottleneck issues related to a root node and always routes shortest-path.

The algorithm was developed by Simula Research Laboratory.

Use ‘-R lash -Q’ option to activate the LASH algorithm

QoS support has to be turned on in order that SL/VL mappings are used.

LMC > 0 is not supported by the LASH routing. If this is specified, the default routing algorithm is invoked instead.

For open regular cartesian meshes the DOR algorithm is the ideal routing algorithm. For toroidal meshes on the other hand there are routing loops that can cause deadlocks. LASH can be used to route these cases. The performance of LASH can be improved by preconditioning the mesh in cases where there are multiple links connecting switches and also in cases where the switches are not cabled consistently. To invoke this, use '-R lash -Q --do_mesh_analysis'. This will add an additional phase that analyses the mesh to try to determine the dimension and size of a mesh. If it determines that the mesh looks like an open or closed cartesian mesh it reorders the ports in dimension order before the rest of the LASH algorithm runs.

8.5.6 DOR Routing Algorithm

The Dimension Order Routing algorithm is based on the Min Hop algorithm and so uses shortest paths. Instead of spreading traffic out across different paths with the same shortest distance, it chooses among the available shortest paths based on an ordering of dimensions. Each port must be consistently cabled to represent a hypercube dimension or a mesh dimension. Paths are grown from a destination back to a source using the lowest dimension (port) of available paths at each step. This provides the ordering necessary to avoid deadlock. When there are multiple links between any two switches, they still represent only one dimension and traffic is balanced across them unless port equalization is turned off. In the case of hypercubes, the same port must be used throughout the fabric to represent the hypercube dimension and match on both ends of the cable. In the case of meshes, the dimension should consistently use the same pair of ports, one port on one end of the cable, and the other port on the other end, continuing along the mesh dimension.

Use '-R dor' option to activate the DOR algorithm.

8.5.7 Torus-2QoS Routing Algorithm

Torus-2QoS is a routing algorithm designed for large-scale 2D/3D torus fabrics. The torus-2QoS routing engine can provide the following functionality on a 2D/3D torus:

- Free of credit loops routing
- Two levels of QoS, assuming switches support 8 data VLs
- Ability to route around a single failed switch, and/or multiple failed links, without:
  - introducing credit loops
  - changing path SL values
- Very short run times, with good scaling properties as fabric size increases

8.5.7.1 Unicast Routing

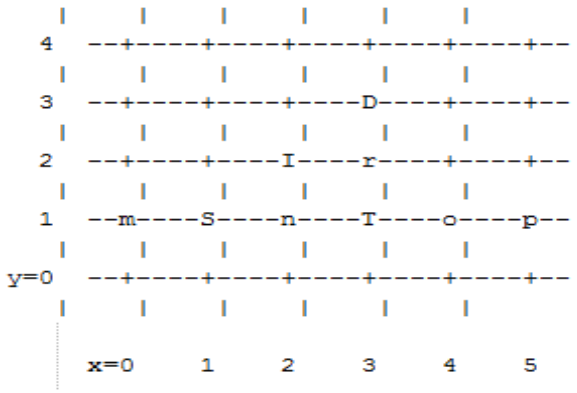
Torus-2QoS is a DOR-based algorithm that avoids deadlocks that would otherwise occur in a torus using the concept of a dateline for each torus dimension. It encodes into a path SL which datelines the path crosses as follows:

```
sl = 0;
for (d = 0; d < torus_dimensions; d++)
/* path_crosses_dateline(d) returns 0 or 1 */
sl |= path_crosses_dateline(d) << d;
```

For a 3D torus, that leaves one SL bit free, which torus-2QoS uses to implement two QoS levels. Torus-2QoS also makes use of the output port dependence of switch SL2VL maps to encode into one VL bit the information encoded in three SL bits. It computes in which torus coordinate direction each inter-switch link "points", and writes SL2VL maps for such ports as follows:

```
for (sl = 0; sl < 16; sl ++)
/* cdir(port) reports which torus coordinate direction a switch port
* "points" in, and returns 0, 1, or 2 */
sl2vl(iport,oport,sl) = 0x1 & (sl >> cdir(oport));
```

Thus, on a pristine 3D torus, i.e., in the absence of failed fabric switches, torus-2QoS consumes 8 SL values (SL bits 0-2) and 2 VL values (VL bit 0) per QoS level to provide deadlock-free routing on a 3D torus. Torus-2QoS routes around link failure by "taking the long way around" any 1D ring interrupted by a link failure. For example, consider the 2D 6x5 torus below, where switches are denoted by [+a-zA-Z]:



For a pristine fabric the path from S to D would be S-n-T-r-D. In the event that either link S-n or n-T has failed, torus-2QoS would use the path S-m-p-o-T-r-D.

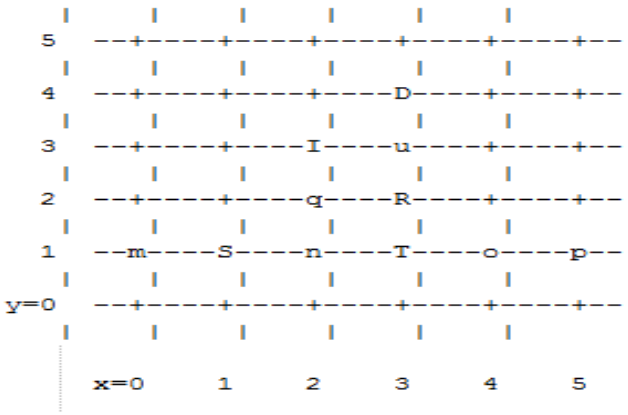
Note that it can do this without changing the path SL value; once the 1D ring m-S-n-T-o-p-m has been broken by failure, path segments using it cannot contribute to deadlock, and the x-direction dateline (between, say, x=5 and x=0) can be ignored for path segments on that ring. One result of this is that torus-2QoS can route around many simultaneous link failures, as long as no 1D ring is broken into disjoint segments. For example, if links n-T and T-o have both failed, that ring has been broken into two disjoint segments, T and o-p-m-S-n. Torus-2QoS checks for such issues, reports if they are found, and refuses to route such fabrics.

Note that in the case where there are multiple parallel links between a pair of switches, torus-2QoS will allocate routes across such links in a round-robin fashion, based on ports at the path destination switch that are active and not used for inter-switch links. Should a link that is one of severalsuch parallel links fail, routes are redistributed across the remaining links. When the last of such a set of parallel links fails, traffic is rerouted as described above.

Handling a failed switch under DOR requires introducing into a path at least one turn that would be otherwise "illegal", i.e. not allowed by DOR rules. Torus-2QoS will introduce such a turn as close as possible to the failed switch in order to route around it. n the above example, suppose switch T has failed, and consider the path from S to D. Torus-2QoS will produce the path S-n-I-r-D, rather than the S-n-T-r-D path for a pristine torus, by introducing an early turn at n. Normal DOR rules will cause traffic arriving at switch I to be forwarded to switch r; for traffic arriving from I due to the "early" turn at n, this will generate an "illegal" turn at I.

Torus-2QoS will also use the input port dependence of SL2VL maps to set VL bit 1 (which would be otherwise unused) for y-x, z-x, and z-y turns, i.e., those turns that are illegal under DOR. This causes the first hop after any such turn to use a separate set of VL values, and prevents deadlock in the presence of a single failed switch. For any given path, only the hops after a turn that is illegal under DOR can contribute to a credit loop that leads to deadlock. So in the example above with failed switch T, the location of the illegal turn at I in the path from S to D requires that any credit loop caused by that turn must encircle the failed switch at T. Thus the second and later hops after the illegal turn at I (i.e., hop r-D) cannot contribute to a credit loop

because they cannot be used to construct a loop encircling T. The hop I-r uses a separate VL, so it cannot contribute to a credit loop encircling T. Extending this argument shows that in addition to being capable of routing around a single switch failure without introducing deadlock, torus-2QoS can also route around multiple failed switches on the condition they are adjacent in the last dimension routed by DOR. For example, consider the following case on a 6x6 2D torus:



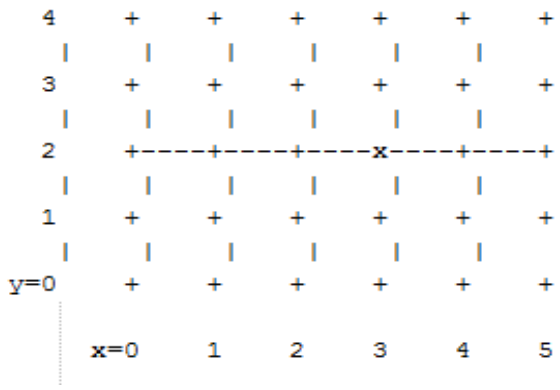
Suppose switches T and R have failed, and consider the path from S to D. Torus-2QoS will generate the path S-n-q-I-u-D, with an illegal turn at switch I, and with hop I-u using a VL with bit 1 set. As a further example, consider a case that torus-2QoS cannot route without deadlock: two failed switches adjacent in a dimension that is not the last dimension routed by DOR; here the failed switches are O and T:

In a pristine fabric, torus-2QoS would generate the path from S to D as S-n-O-T-r-D. With failed switches O and T, torus-2QoS will generate the path S-n-I-q-r-D, with illegal turn at switch I, and with hop I-q using a VL with bit 1 set. In contrast to the earlier examples, the second hop after the illegal turn, q-r, can be used to construct a credit loop encircling the failed switches.

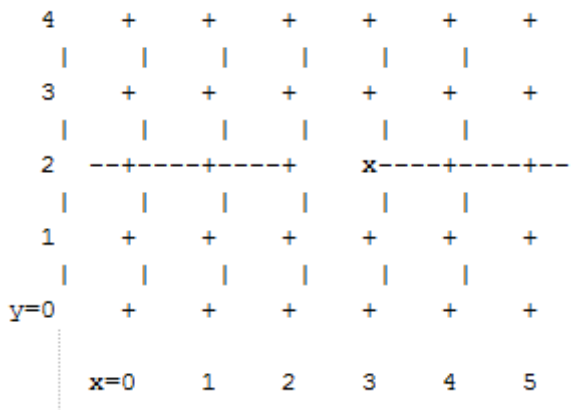
8.5.7.2 Multicast Routing

Since torus-2QoS uses all four available SL bits, and the three data VL bits that are typically available in current switches, there is no way to use SL/VL values to separate multicast traffic from unicast traffic. Thus, torus-2QoS must generate multicast routing such that credit loops can-

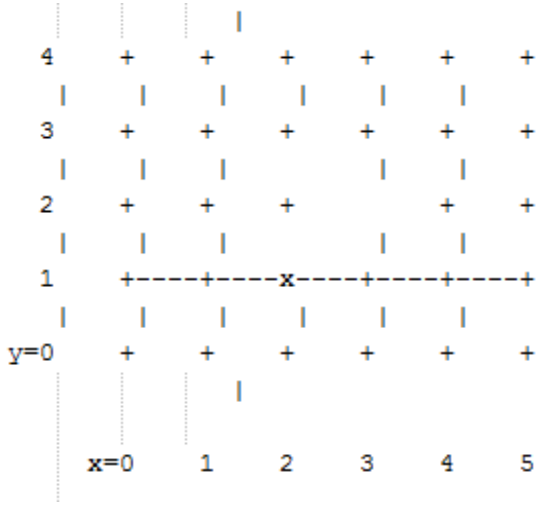
not arise from a combination of multicast and unicast path segments. It turns out that it is possible to construct spanning trees for multicast routing that have that property. For the 2D 6x5 torus example above, here is the full-fabric spanning tree that torus-2QoS will construct, where "x" is the root switch and each "+" is a non-root switch:



For multicast traffic routed from root to tip, every turn in the above spanning tree is a legal DOR turn. For traffic routed from tip to root, and some traffic routed through the root, turns are not legal DOR turns. However, to construct a credit loop, the union of multicast routing on this spanning tree with DOR unicast routing can only provide 3 of the 4 turns needed for the loop. In addition, if none of the above spanning tree branches crosses a dateline used for unicast credit loop avoidance on a torus, and if multicast traffic is confined to SL 0 or SL 8 (recall that torus-2QoS uses SL bit 3 to differentiate QoS level), then multicast traffic also cannot contribute to the "ring" credit loops that are otherwise possible in a torus. Torus-2QoS uses these ideas to create a master spanning tree. Every multicast group spanning tree will be constructed as a subset of the master tree, with the same root as the master tree. Such multicast group spanning trees will in general not be optimal for groups which are a subset of the full fabric. However, this compromise must be made to enable support for two QoS levels on a torus while preventing credit loops. In the presence of link or switch failures that result in a fabric for which torus-2QoS can generate credit-loop-free unicast routes, it is also possible to generate a master spanning tree for multicast that retains the required properties. For example, consider that same 2D 6x5 torus, with the link from (2,2) to (3,2) failed. Torus-2QoS will generate the following master spanning tree:



Two things are notable about this master spanning tree. First, assuming the x dateline was between x=5 and x=0, this spanning tree has a branch that crosses the dateline. However, just as for unicast, crossing a dateline on a 1D ring (here, the ring for y=2) that is broken by a failure cannot contribute to a torus credit loop. Second, this spanning tree is no longer optimal even for multicast groups that encompass the entire fabric. That, unfortunately, is a compromise that must be made to retain the other desirable properties of torus-2QoS routing. In the event that a single switch fails, torus-2QoS will generate a master spanning tree that has no "extra" turns by appropriately selecting a root switch. In the 2D 6x5 torus example, assume now that the switch at (3,2), i.e. the root for a pristine fabric, fails. Torus-2QoS will generate the following master spanning tree for that case:



Assuming the y dateline was between y=4 and y=0, this spanning tree has a branch that crosses a dateline. However, again this cannot contribute to credit loops as it occurs on a 1D ring (the ring for x=3) that is broken by a failure, as in the above example.

8.5.7.3 Torus Topology Discovery

The algorithm used by torus-2QoS to construct the torus topology from the undirected graph representing the fabric requires that the radix of each dimension be configured via torus-2QoS.conf. It also requires that the torus topology be "seeded"; for a 3D torus this requires configuring four switches that define the three coordinate directions of the torus. Given this starting information, the algorithm is to examine the cube formed by the eight switch locations bounded by the corners (x,y,z) and (x+1,y+1,z+1). Based on switches already placed into the torus topology at some of these locations, the algorithm examines 4-loops of interswitch links to find the one that is consistent with a face of the cube of switch locations, and adds its swiches to the discovered topology in the correct locations.

Because the algorithm is based on examining the topology of 4-loops of links, a torus with one or more radix-4 dimensions requires extra initial seed configuration. See torus-2QoS.conf(5) for details. Torus-2QoS will detect and report when it has insufficient configuration for a torus with radix-4 dimensions.

In the event the torus is significantly degraded, i.e., there are many missing switches or links, it may happen that torus-2QoS is unable to place into the torus some switches and/or links that were discovered in the fabric, and will generate a warning in that case. A similar condition

occurs if torus-2QoS is misconfigured, i.e., the radix of a torus dimension as configured does not match the radix of that torus dimension as wired, and many switches/links in the fabric will not be placed into the torus.

8.5.7.4 Quality Of Service Configuration

OpenSM will not program switchs and channel adapters with SL2VL maps or VL arbitration configuration unless it is invoked with -Q. Since torus-2QoS depends on such functionality for correct operation, always invoke OpenSM with -Q when torus-2QoS is in the list of routing engines. Any quality of service configuration method supported by OpenSM will work with torus-2QoS, subject to the following limitations and considerations. For all routing engines supported by OpenSM except torus-2QoS, there is a one-to-one correspondence between QoS level and SL. Torus-2QoS can only support two quality of service levels, so only the high-order bit of any SL value used for unicast QoS configuration will be honored by torus-2QoS. For multicast QoS configuration, only SL values 0 and 8 should be used with torus-2QoS.

Since SL to VL map configuration must be under the complete control of torus-2QoS, any configuration via qos_sl2vl, qos_swe_sl2vl, etc., must and will be ignored, and a warning will be generated. Torus-2QoS uses VL values 0-3 to implement one of its supported QoS levels, and VL values 4-7 to implement the other. Hard-to-diagnose application issues may arise if traffic is not delivered fairly across each of these two VL ranges. Torus-2QoS will detect and warn if VL arbitration is configured unfairly across VLs in the range 0-3, and also in the range 4-7. Note that the default OpenSM VL arbitration configuration does not meet this constraint, so all torus-2QoS users should configure VL arbitration via qos_vlarb_high, qos_vlarb_low, etc.

8.5.7.5 Operational Considerations

Any routing algorithm for a torus IB fabric must employ path SL values to avoid credit loops. As a result, all applications run over such fabrics must perform a path record query to obtain the correct path SL for connection setup. Applications that use rdma_cm for connection setup will automatically meet this requirement.

If a change in fabric topology causes changes in path SL values required to route without credit loops, in general all applications would need to repath to avoid message deadlock. Since torus-2QoS has the ability to reroute after a single switch failure without changing path SL values, repathing by running applications is not required when the fabric is routed with torus-2QoS.

Torus-2QoS can provide unchanging path SL values in the presence of subnet manager failover provided that all OpenSM instances have the same idea of dateline location. See torus-2QoS.conf(5) for details. Torus-2QoS will detect configurations of failed switches and links that prevent routing that is free of credit loops, and will log warnings and refuse to route. If "no_fallback" was configured in the list of OpenSM routing engines, then no other routing engine will attempt to route the fabric. In that case all paths that do not transit the failed components will continue to work, and the subset of paths that are still operational will continue to remain free of credit loops. OpenSM will continue to attempt to route the fabric after every sweep interval, and after any change (such as a link up) in the fabric topology. When the fabric components are repaired, full functionality will be restored. In the event OpenSM was configured to allow some other engine to route the fabric if torus-2QoS fails, then credit loops and message deadlock are likely if torus-2QoS had previously routed the fabric successfully. Even if the other engine is capable of routing a torus without credit loops, applications that built connections with path SL values granted under torus-2QoS will likely experience message deadlock under routing generated by a different engine, unless they repath. To verify that a torus fabric is routed free of credit loops, use ibdmchk to analyze data collected via ibdiagnet -vlr.

8.5.7.6 Torus-2QoS Configuration File Syntax

The file torus-2QoS.conf contains configuration information that is specific to the OpenSM routing engine torus-2QoS. Blank lines and lines where the first non-whitespace character is "#" are ignored. A token is any contiguous group of non-whitespace characters. Any tokens on a line following the recognized configuration tokens described below are ignored.

```
[torus|mesh] x_radix[m|M|t|T] y_radix[m|M|t|T] z_radix[m|M|t|T]
```

Either torus or mesh must be the first keyword in the configuration, and sets the topology that torus-2QoS will try to construct. A 2D topology can be configured by specifying one of x_radix, y_radix, or z_radix as 1. An individual dimension can be configured as mesh (open) or torus (looped) by suffixing its radix specification with one of m, M, t, or T. Thus, "mesh 3T 4 5" and "torus 3 4M 5M" both specify the same topology.

Note that although torus-2QoS can route mesh fabrics, its ability to route around failed components is severely compromised on such fabrics. A failed fabric componentis very likely to cause a disjoint ring; see UNICAST ROUTING in torus-2QoS(8).

```
xp_link sw0_GUID sw1_GUID
yp_link sw0_GUID sw1_GUID
zp_link sw0_GUID sw1_GUID
xm_link sw0_GUID sw1_GUID
ym_link sw0_GUID sw1_GUID
zm_link sw0_GUID sw1_GUID
```

These keywords are used to seed the torus/mesh topology. For example, "xp_link 0x2000 0x2001" specifies that a link from the switch with node GUID 0x2000 to the switch with node GUID 0x2001 would point in the positive x direction, while "xm_link 0x2000 0x2001" specifies that a link from the switch with node GUID 0x2000 to the switch with node GUID 0x2001 would point in the negative x direction. All the link keywords for a given seed must specify the same "from" switch.

In general, it is not necessary to configure both the positive and negative directions for a given coordinate; either is sufficient. However, the algorithm used for topology discovery needs extra information for torus dimensions of radix four (see TOPOLOGY DISCOVERY in torus-2QoS(8)). For such cases both the positive and negative coordinate directions must be specified.

Based on the topology specified via the torus/mesh keyword, torus-2QoS will detect and log when it has insufficient seed configuration.

```
x_dateline position
y_dateline position
z_dateline position
```

In order for torus-2QoS to provide the guarantee that path SL values do not change under any conditions for which it can still route the fabric, its idea of dateline position must not change relative to physical switch locations. The dateline keywords provide the means to configure such behavior.

The dateline for a torus dimension is always between the switch with coordinate 0 and the switch with coordinate radix-1 for that dimension. By default, the common switch in a torus seed is taken as the origin of the coordinate system used to describe switch location. The position param-

eter for a dateline keyword moves the origin (and hence the dateline) the specified amount relative to the common switch in a torus seed.

```
next_seed
```

If any of the switches used to specify a seed were to fail torus-2QoS would be unable to complete topology discovery successfully. The next_seed keyword specifies that the following link and dateline keywords apply to a new seed specification.

For maximum resiliency, no seed specification should share a switch with any other seed specification. Multiple seed specifications should use dateline configuration to ensure that torus-2QoS can grant path SL values that are constant, regardless of which seed was used to initiate topology discovery.

portgroup_max_ports max_ports - This keyword specifies the maximum number of parallel inter-switch links, and also the maximum number of host ports per switch, that torus-2QoS can accommodate. The default value is 16. Torus-2QoS will log an error message during topology discovery if this parameter needs to be increased. If this keyword appears multiple times, the last instance prevails.

port_order p1 p2 p3 ... - This keyword specifies the order in which CA ports on a destination switch are visited when computing routes. When the fabric contains switches connected with multiple parallel links, routes are distributed in a round-robin fashion across such links, and so changing the order that CA ports are visited changes the distribution of routes across such links. This may be advantageous for some specific traffic patterns.

The default is to visit CA ports in increasing port order on destination switches. Duplicate values in the list will be ignored.

EXAMPLE

```
# Look for a 2D (since x radix is one) 4x5 torus.
torus 1 4 5
# y is radix-4 torus dimension, need both
# ym_link and yp_link configuration.
yp_link 0x200000 0x200005 # sw @ y=0,z=0 -> sw @ y=1,z=0
ym_link 0x200000 0x20000f # sw @ y=0,z=0 -> sw @ y=3,z=0
# z is not radix-4 torus dimension, only need one of
# zm_link or zp_link configuration.
zp_link 0x200000 0x200001 # sw @ y=0,z=0 -> sw @ y=0,z=1
next_seed
yp_link 0x20000b 0x200010 # sw @ y=2,z=1 -> sw @ y=3,z=1
ym_link 0x20000b 0x200006 # sw @ y=2,z=1 -> sw @ y=1,z=1
zp_link 0x20000b 0x20000c # sw @ y=2,z=1 -> sw @ y=2,z=2
y_dateline -2 # Move the dateline for this seed
z_dateline -1 # back to its original position.
# If OpenSM failover is configured, for maximum resiliency
# one instance should run on a host attached to a switch
# from the first seed, and another instance should run
# on a host attached to a switch from the second seed.
# Both instances should use this torus-2QoS.conf to ensure
# path SL values do not change in the event of SM failover.
# port_order defines the order on which the ports would be
# chosen for routing.
port_order 7 10 8 11 9 12 25 28 26 29 27 30
```

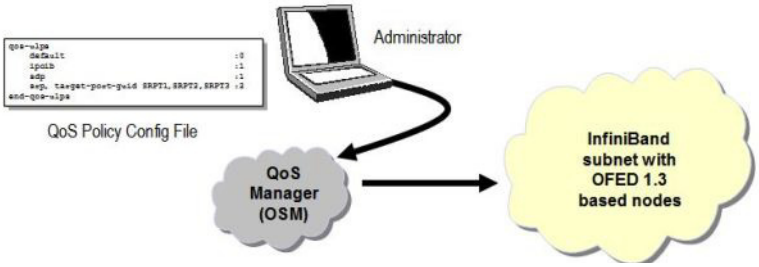
8.6 Quality of Service Management in OpenSM

8.6.1 Overview

When Quality of Service (QoS) in OpenSM is enabled (using the '-Q' or '--qos' flags), OpenSM looks for a QoS Policy file. During fabric initialization and at every heavy sweep, OpenSM parses the QoS policy file, applies its settings to the discovered fabric elements, and enforces the provided policy on client requests. The overall flow for such requests is as follows:

- The request is matched against the defined matching rules such that the QoS Level definition is found
- Given the QoS Level, a path(s) search is performed with the given restrictions imposed by that level

Figure 4: QoS Manager



There are two ways to define QoS policy:

- Advanced – the advanced policy file syntax provides the administrator various ways to match a PathRecord/MultiPathRecord (PR/MPR) request, and to enforce various QoS constraints on the requested PR/MPR
- Simple – the simple policy file syntax enables the administrator to match PR/MPR requests by various ULPs and applications running on top of these ULPs

8.6.2 Advanced QoS Policy File

The QoS policy file has the following sections:

I) Port Groups (denoted by port-groups)

This section defines zero or more port groups that can be referred later by matching rules (see below). Port group lists ports by:

- Port GUID
- Port name, which is a combination of NodeDescription and IB port number
- PKey, which means that all the ports in the subnet that belong to partition with a given PKey belong to this port group
- Partition name, which means that all the ports in the subnet that belong to partition with a given name belong to this port group
- Node type, where possible node types are: CA, SWITCH, ROUTER, ALL, and SELF (SM's port).

II) QoS Setup (denoted by qos-setup)

This section describes how to set up SL2VL and VL Arbitration tables on various nodes in the fabric. However, this is not supported in OFED. SL2VL and VLArb tables should be configured in the OpenSM options file (default location - /var/cache/opensm/opensm.opts).

III) QoS Levels (denoted by qos-levels)

Each QoS Level defines Service Level (SL) and a few optional fields:

- MTU limit
- Rate limit
- PKey
- Packet lifetime

When path(s) search is performed, it is done with regards to restriction that these QoS Level parameters impose. One QoS level that is mandatory to define is a DEFAULT QoS level. It is applied to a PR/MPR query that does not match any existing match rule. Similar to any other QoS Level, it can also be explicitly referred by any match rule.

IV) QoS Matching Rules (denoted by qos-match-rules)

Each PathRecord/MultiPathRecord query that OpenSM receives is matched against the set of matching rules. Rules are scanned in order of appearance in the QoS policy file such as the first match takes precedence.

Each rule has a name of QoS level that will be applied to the matching query. A default QoS level is applied to a query that did not match any rule.

Queries can be matched by:

- Source port group (whether a source port is a member of a specified group)
- Destination port group (same as above, only for destination port)
- PKey
- QoS class
- Service ID

To match a certain matching rule, PR/MPR query has to match ALL the rule's criteria. However, not all the fields of the PR/MPR query have to appear in the matching rule.

For instance, if the rule has a single criterion - Service ID, it will match any query that has this Service ID, disregarding rest of the query fields. However, if a certain query has only Service ID (which means that this is the only bit in the PR/MPR component mask that is on), it will not match any rule that has other matching criteria besides Service ID.

8.6.3 Simple QoS Policy Definition

Simple QoS policy definition comprises of a single section denoted by qos-ulps. Similar to the advanced QoS policy, it has a list of match rules and their QoS Level, but in this case a match rule has only one criterion - its goal is to match a certain ULP (or a certain application on top of this ULP) PR/MPR request, and QoS Level has only one constraint - Service Level (SL).

The simple policy section may appear in the policy file in combine with the advanced policy, or as a stand-alone policy definition. See more details and list of match rule criteria below.

8.6.4 Policy File Syntax Guidelines

- Leading and trailing blanks, as well as empty lines, are ignored, so the indentation in the example is just for better readability.
- Comments are started with the pound sign (#) and terminated by EOL.
- Any keyword should be the first non-blank in the line, unless it's a comment.
- Keywords that denote section/subsection start have matching closing keywords.
- Having a QoS Level named "DEFAULT" is a must - it is applied to PR/MPR requests that didn't match any of the matching rules.
- Any section/subsection of the policy file is optional.

8.6.5 Examples of Advanced Policy File

As mentioned earlier, any section of the policy file is optional, and the only mandatory part of the policy file is a default QoS Level.

Here's an example of the shortest policy file:

```
qos-levels
    qos-level
        name: DEFAULT
        sl: 0
    end-qos-level
end-qos-levels
```

Port groups section is missing because there are no match rules, which means that port groups are not referred anywhere, and there is no need defining them. And since this policy file doesn't have any matching rules, PR/MPR query will not match any rule, and OpenSM will enforce default QoS level. Essentially, the above example is equivalent to not having a QoS policy file at all.

The following example shows all the possible options and keywords in the policy file and their syntax:

```
#
# See the comments in the following example.
# They explain different keywords and their meaning.
#
port-groups

    port-group # using port GUIDs
        name: Storage
        # "use" is just a description that is used for logging
        #  Other than that, it is just a comment
        use: SRP Targets
        port-guid: 0x10000000000001, 0x10000000000005-0x1000000000FFFA
        port-guid: 0x1000000000FFFF
    end-port-group
```

```
        port-group
            name: Virtual Servers
            # The syntax of the port name is as follows:
            #   "node_description/Pnum".
            # node_description is compared to the NodeDescription of the node,
            # and "Pnum" is a port number on that node.
            port-name: vs1 HCA-1/P1, vs2 HCA-1/P1
        end-port-group

        # using partitions defined in the partition policy
        port-group
            name: Partitions
            partition: Part1
            pkey: 0x1234
        end-port-group

        # using node types: CA, ROUTER, SWITCH, SELF (for node that runs SM)
        # or ALL (for all the nodes in the subnet)
        port-group
            name: CAs and SM
            node-type: CA, SELF
        end-port-group

    end-port-groups

    qos-setup
        # This section of the policy file describes how to set up SL2VL and VL
        # Arbitration tables on various nodes in the fabric.
        # However, this is not supported in OFED - the section is parsed
        # and ignored. SL2VL and VLArb tables should be configured in the
        # OpenSM options file (by default - /var/cache/opensm/opensm.opts).
    end-qos-setup

    qos-levels

        # Having a QoS Level named "DEFAULT" is a must - it is applied to
        # PR/MPR requests that didn't match any of the matching rules.
        qos-level
            name: DEFAULT
            use: default QoS Level
            sl: 0
        end-qos-level

        # the whole set: SL, MTU-Limit, Rate-Limit, PKey, Packet Lifetime
        qos-level
            name: WholeSet
```

```
            sl: 1
            mtu-limit: 4
            rate-limit: 5
            pkey: 0x1234
            packet-life: 8
        end-qos-level

    end-qos-levels

    # Match rules are scanned in order of their apperance in the policy file.
    # First matched rule takes precedence.
    qos-match-rules

        # matching by single criteria: QoS class
        qos-match-rule
            use: by QoS class
            qos-class: 7-9,11
            # Name of qos-level to apply to the matching PR/MPR
            qos-level-name: WholeSet
        end-qos-match-rule

        # show matching by destination group and service id
        qos-match-rule
            use: Storage targets
            destination: Storage
            service-id: 0x10000000000001, 0x10000000000008-0x10000000000FFF
            qos-level-name: WholeSet
        end-qos-match-rule

        qos-match-rule
            source: Storage
            use: match by source group only
            qos-level-name: DEFAULT
        end-qos-match-rule

        qos-match-rule
            use: match by all parameters
            qos-class: 7-9,11
            source: Virtual Servers
            destination: Storage
            service-id: 0x0000000000010000-0x000000000001FFFF
            pkey: 0x0F00-0x0FFF
            qos-level-name: WholeSet
        end-qos-match-rule

    end-qos-match-rules
```

8.6.6 Simple QoS Policy - Details and Examples

Simple QoS policy match rules are tailored for matching ULPs (or some application on top of a ULP) PR/MPR requests. This section has a list of per-ULP (or per-application) match rules and the SL that should be enforced on the matched PR/MPR query.

Match rules include:

- Default match rule that is applied to PR/MPR query that didn't match any of the other match rules
- SDP
- SDP application with a specific target TCP/IP port range
- SRP with a specific target IB port GUID
- RDS
- IPoIB with a default PKey
- IPoIB with a specific PKey
- Any ULP/application with a specific Service ID in the PR/MPR query
- Any ULP/application with a specific PKey in the PR/MPR query
- Any ULP/application with a specific target IB port GUID in the PR/MPR query

Since any section of the policy file is optional, as long as basic rules of the file are kept (such as no referring to nonexisting port group, having default QoS Level, etc), the simple policy section (qos-ulps) can serve as a complete QoS policy file.

The shortest policy file in this case would be as follows:

```
qos-ulps
    default  : 0 #default SL
end-qos-ulps
```

It is equivalent to the previous example of the shortest policy file, and it is also equivalent to not having policy file at all. Below is an example of simple QoS policy with all the possible keywords:

```
qos-ulps
default                       : 0 # default SL
sdp, port-num 30000           : 0 # SL for application running on
                                  # top of SDP when a destination
                                  # TCP/IPport is 30000
sdp, port-num 10000-20000     : 0
sdp                           : 1 # default SL for any other
                                  # application running on top of SDP
rds                           : 2 # SL for RDS traffic
ipoib, pkey 0x0001            : 0 # SL for IPoIB on partition with
                                  # pkey 0x0001
ipoib                         : 4 # default IPoIB partition,
                                  # pkey=0x7FFF
any, service-id 0x6234        : 6 # match any PR/MPR query with a
                                  # specific Service ID
```

```
any, pkey 0x0ABC             : 6 # match any PR/MPR query with a
                                 # specific PKey
srp, target-port-guid 0x1234  : 5 # SRP when SRP Target is located
                                 # on a specified IB port GUID
any, target-port-guid 0x0ABC-0xFFFFF : 6 # match any PR/MPR query
                                 # with a specific target port GUID
end-qos-ulps
```

Similar to the advanced policy definition, matching of PR/MPR queries is done in order of appearance in the QoS policy file such as the first match takes precedence, except for the "default" rule, which is applied only if the query didn't match any other rule. All other sections of the QoS policy file take precedence over the qos-ulps section. That is, if a policy file has both qos-match-rules and qos-ulps sections, then any query is matched first against the rules in the qos-match-rules section, and only if there was no match, the query is matched against the rules in qos-ulps section.

Note that some of these match rules may overlap, so in order to use the simple QoS definition effectively, it is important to understand how each of the ULPs is matched.

8.6.6.1 IPoIB

IPoIB query is matched by PKey or by destination GID, in which case this is the GID of the multicast group that OpenSM creates for each IPoIB partition.

Default PKey for IPoIB partition is 0x7fff, so the following three match rules are equivalent:

```
ipoib              : <SL>
ipoib, pkey 0x7fff : <SL>
any,   pkey 0x7fff : <SL>
```

8.6.6.2 SDP

SDP PR query is matched by Service ID. The Service-ID for SDP is 0x000000000001PPPP, where PPPP are 4 hex digits holding the remote TCP/IP Port Number to connect to. The following two match rules are equivalent:

```
sdp                                                  : <SL>
any, service-id 0x0000000000010000-0x000000000001ffff : <SL>
```

8.6.6.3 RDS

Similar to SDP, RDS PR query is matched by Service ID. The Service ID for RDS is 0x000000000106PPPP, where PPPP are 4 hex digits holding the remote TCP/IP Port Number to connect to. Default port number for RDS is 0x48CA, which makes a default Service-ID 0x00000000010648CA. The following two match rules are equivalent:

```
rds                               : <SL>
any, service-id 0x00000000010648CA : <SL>
```

8.6.6.4 SRP

Service ID for SRP varies from storage vendor to vendor, thus SRP query is matched by the target IB port GUID. The following two match rules are equivalent:

```
srp, target-port-guid 0x1234  : <SL>
any, target-port-guid 0x1234  : <SL>
```

Note that any of the above ULPs might contain target port GUID in the PR query, so in order for these queries not to be recognized by the QoS manager as SRP, the SRP match rule (or any match rule that refers to the target port guid only) should be placed at the end of the qos-ulps match rules.

8.6.6.5 MPI

SL for MPI is manually configured by MPI admin. OpenSM is not forcing any SL on the MPI traffic, and that's why it is the only ULP that did not appear in the qos-ulps section.

8.6.7 SL2VL Mapping and VL Arbitration

OpenSM cached options file has a set of QoS related configuration parameters, that are used to configure SL2VL mapping and VL arbitration on IB ports. These parameters are:

- Max VLs: the maximum number of VLs that will be on the subnet
- High limit: the limit of High Priority component of VL Arbitration table (IBA 7.6.9)
- VLArb low table: Low priority VL Arbitration table (IBA 7.6.9) template
- VLArb high table: High priority VL Arbitration table (IBA 7.6.9) template
- SL2VL: SL2VL Mapping table (IBA 7.6.6) template. It is a list of VLs corresponding to SLs 0-15 (Note that VL15 used here means drop this SL).

There are separate QoS configuration parameters sets for various target types: CAs, routers, switch external ports, and switch's enhanced port 0. The names of such parameters are prefixed by "qos_<type>_" string. Here is a full list of the currently supported sets:

- qos_ca_ - QoS configuration parameters set for CAs.
- qos_rtr_ - parameters set for routers.
- qos_sw0_ - parameters set for switches' port 0.
- qos_swe_ - parameters set for switches' external ports.

Here's the example of typical default values for CAs and switches' external ports (hard-coded in OpenSM initialization):

```
qos_ca_max_vls 15
qos_ca_high_limit 0
qos_ca_vlarb_high 0:4,1:0,2:0,3:0,4:0,5:0,6:0,7:0,8:0,9:0,10:0,11:0,12:0,13:0,14:0
qos_ca_vlarb_low 0:0,1:4,2:4,3:4,4:4,5:4,6:4,7:4,8:4,9:4,10:4,11:4,12:4,13:4,14:4
qos_ca_sl2vl 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,7
qos_swe_max_vls 15
qos_swe_high_limit 0
qos_swe_vlarb_high 0:4,1:0,2:0,3:0,4:0,5:0,6:0,7:0,8:0,9:0,10:0,11:0,12:0,13:0,14:0
qos_swe_vlarb_low 0:0,1:4,2:4,3:4,4:4,5:4,6:4,7:4,8:4,9:4,10:4,11:4,12:4,13:4,14:4
```

```
qos_swe_sl2vl 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,7
```

VL arbitration tables (both high and low) are lists of VL/Weight pairs. Each list entry contains a VL number (values from 0-14), and a weighting value (values 0-255), indicating the number of 64 byte units (credits) which may be transmitted from that VL when its turn in the arbitration occurs. A weight of 0 indicates that this entry should be skipped. If a list entry is programmed for VL15 or for a VL that is not supported or is not currently configured by the port, the port may either skip that entry or send from any supported VL for that entry.

Note, that the same VLs may be listed multiple times in the High or Low priority arbitration tables, and, further, it can be listed in both tables. The limit of high-priority VLArb table (qos_<type>_high_limit) indicates the number of high-priority packets that can be transmitted without an opportunity to send a low-priority packet. Specifically, the number of bytes that can be sent is high_limit times 4K bytes.

A high_limit value of 255 indicates that the byte limit is unbounded.

If the 255 value is used, the low priority VLs may be starved.

A value of 0 indicates that only a single packet from the high-priority table may be sent before an opportunity is given to the low-priority table.

Keep in mind that ports usually transmit packets of size equal to MTU. For instance, for 4KB MTU a single packet will require 64 credits, so in order to achieve effective VL arbitration for packets of 4KB MTU, the weighting values for each VL should be multiples of 64.

Below is an example of SL2VL and VL Arbitration configuration on subnet:

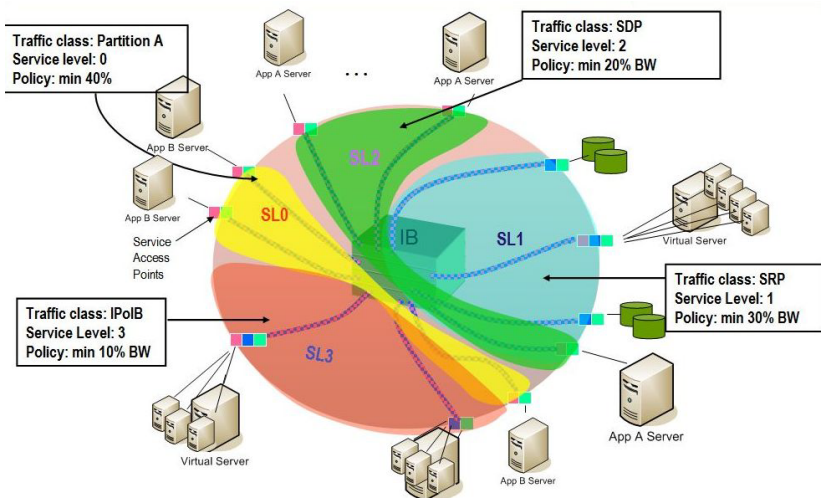
```
qos_ca_max_vls 15
qos_ca_high_limit 6
qos_ca_vlarb_high 0:4
qos_ca_vlarb_low 0:0,1:64,2:128,3:192,4:0,5:64,6:64,7:64
qos_ca_sl2vl 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,7
qos_swe_max_vls 15
qos_swe_high_limit 6
qos_swe_vlarb_high 0:4
qos_swe_vlarb_low 0:0,1:64,2:128,3:192,4:0,5:64,6:64,7:64
qos_swe_sl2vl 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,7
```

In this example, there are 8 VLs configured on subnet: VL0 to VL7. VL0 is defined as a high priority VL, and it is limited to 6 x 4KB = 24KB in a single transmission burst. Such configuration would suilt VL that needs low latency and uses small MTU when transmitting packets. Rest of VLs are defined as low priority VLs with different weights, while VL4 is effectively turned off.

8.6.8 Deployment Example

Figure 5 shows an example of an InfiniBand subnet that has been configured by a QoS manager to provide different service levels for various ULPs.

Figure 5: Example QoS Deployment on InfiniBand Subnet



8.7 QoS Configuration Examples

The following are examples of QoS configuration for different cluster deployments. Each example provides the QoS level assignment and their administration via OpenSM configuration files.

8.7.1 Typical HPC Example: MPI and Lustre

Assignment of QoS Levels

- MPI
  - Separate from I/O load
  - Min BW of 70%
- Storage Control (Lustre MDS)
  - Low latency
- Storage Data (Lustre OST)
  - Min BW 30%

Administration

- MPI is assigned an SL via the command line

```
host1# mpirun -sl 0
```

- OpenSM QoS policy file

In the following policy file example, replace OST* and MDS* with the real port GUIDs.

```
qos-ulps
    default                                                 :0 # default SL (for
MPI)
    any, target-port-guid OST1,OST2,OST3,OST4:1 # SL for Lustre OST
    any, target-port-guid MDS1,MDS2                         :2 # SL for Lustre
MDS
end-qos-ulps
```

- OpenSM options file

```
 qos_max_vls 8
 qos_high_limit 0
 qos_vlarb_high 2:1
 qos_vlarb_low 0:96,1:224
 qos_sl2vl 0,1,2,3,4,5,6,7,15,15,15,15,15,15,15,15
```

8.7.2 EDC SOA (2-tier): IPoIB and SRP

The following is an example of QoS configuration for a typical enterprise data center (EDC) with service oriented architecture (SOA), with IPoIB carrying all application traffic and SRP used for storage.

QoS Levels

- Application traffic
  - IPoIB (UD and CM) and SDP
  - Isolated from storage
  - Min BW of 50%
- SRP
  - Min BW 50%
  - Bottleneck at storage nodes

Administration

- OpenSM QoS policy file

In the following policy file example, replace SRPT* with the real SRP Target port GUIDs.

```
qos-ulps
    default                                 :0
    ipoib                                   :1
    sdp                                     :1
    srp, target-port-guid SRPT1,SRPT2,SRPT3 :2
```

```
end-qos-ulps
```

- OpenSM options file

```
qos_max_vls 8
 qos_high_limit 0
 qos_vlarb_high 1:32,2:32
 qos_vlarb_low 0:1,
 qos_sl2vl 0,1,2,3,4,5,6,7,15,15,15,15,15,15,15,15
```

8.7.3 EDC (3-tier): IPoIB, RDS, SRP

The following is an example of QoS configuration for an enterprise data center (EDC), with IPoIB carrying all application traffic, RDS for database traffic, and SRP used for storage.

QoS Levels

- Management traffic (ssh)
  - IPoIB management VLAN (partition A)
  - Min BW 10%
- Application traffic
  - IPoIB application VLAN (partition B)
  - Isolated from storage and database
  - Min BW of 30%
- Database Cluster traffic
  - RDS
  - Min BW of 30%
- SRP
  - Min BW 30%
  - Bottleneck at storage nodes

Administration

- OpenSM QoS policy file

In the following policy file example, replace SRPT* with the real SRP Initiator port GUIDs.

```
qos-ulps
    default                                 : 0
    ipoib, pkey 0x8001                      : 1
    ipoib, pkey 0x8002                      : 2
    rds                                     : 3
    srp, target-port-guid SRPT1, SRPT2, SRPT3  : 4
```

```
end-qos-ulps
```

- OpenSM options file

```
qos_max_vls 8
qos_high_limit 0
qos_vlarb_high 1:32,2:96,3:96,4:96
qos_vlarb_low 0:1
qos_sl2vl 0,1,2,3,4,5,6,7,15,15,15,15,15,15,15,15
```

- Partition configuration file

```
Default=0x7fff,     ipoib : ALL=full;
PartA=0x8001, sl=1, ipoib : ALL=full;
```

8.8 Adaptive Routing

8.8.1 Overview

Adaptive Routing is at beta stage.

Adaptive Routing (AR) enables the switch to select the output port based on the port's load. AR supports two routing modes:

- Free AR: No constraints on output port selection.
- Bounded AR: The switch does not change the output port during the same transmission burst. This mode minimizes the appearance of out-of-order packets.

Adaptive Routing Manager enables and configures Adaptive Routing mechanism on fabric switches. It scans all the fabric switches, deduces which switches support Adaptive Routing and configures the AR functionality on these switches.

Currently, Adaptive Routing Manager supports only link aggregation algorithm. Adaptive Routing Manager configures AR mechanism to allow switches to select output port out of all the ports that are linked to the same remote switch. This algorithm suits any topology with several links between switches. Especially, it suits 3D torus/mesh, where there are several link in each direction of the X/Y/Z axis.

If some switches do not support AR, they will slow down the AR Manager as it may get timeouts on the AR-related queries to these switches.

8.8.2 Installing the Adaptive Routing

Adaptive Routing Manager is a Subnet Manager plug-in, i.e. it is a shared library (libarmgr.so) that is dynamically loaded by the Subnet Manager. Adaptive Routing Manager is installed as a part of Mellanox OFED installation.

8.8.3 Running Subnet Manager with Adaptive Routing Manager

Adaptive Routing (AR) Manager can be enabled/disabled through SM options file.

8.8.3.1 Enabling Adaptive Routing

To enable Adaptive Routing, perform the following:

1. Create the Subnet Manager options file. Run:

```
opensm -c <options-file-name>
```

2. Add '**armgr**' to the '**event_plugin_name**' option in the file:

```
# Event plugin name(s)
event_plugin_name armgr
```

3. Run Subnet Manager with the new options file:

```
opensm -F <options-file-name>
```

Adaptive Routig Manager can read options file with various configuration parameters to fine-tune AR mechanism and AR Manager behavior. Default location of the AR Manager options file is **/etc/opensm/ar_mgr.conf**.

To provide an alternative location, please perform the following:

1. Add '**armgr --conf_file <ar-mgr-options-file-name>**' to the '**event_plugin_options**' option in the file **# Options** string that would be passed to the plugin(s) **event_plugin_options armgr --conf_file <ar-mgr-options-file-name>**
2. Run Subnet Manager with the new options file:

```
opensm -F <options-file-name>
```

See an example of AR Manager options file with all the default values in “Example of Adaptive Routing Manager Options File” on page 182.

8.8.3.2 Disabling Adaptive Routing

There are two ways to disable Adaptive Routing Manager:

1. By disabling it explicitly in the Adaptive Routing configuration file.
2. By removing the 'armgr' option from the Subnet Manager options file.

Adaptive Routing mechanism is automatically disabled once the switch receives setting of the usual linear routing table (LFT).

Therefore, no action is required to clear Adaptive Routing configuration on the switches if you do not wish to use Adaptive Routing.

8.8.4 Querying Adaptive Routing Tables

When Adaptive Routing is active, the content of the usual Linear Forwarding Routing Table on the switch is invalid, thus the standard tools that query LFT (e.g. smpquery, dump_lfts.sh, and others) cannot be used. To query the switch for the content of its Adaptive Routing table, use the 'smparquery' tool that is installed as a part of the Adaptive Routing Manager package. To see its usage details, run 'smparquery -h'.

8.8.5 Adaptive Routing Manager Options File

The default location of the AR Manager options file is /etc/opensm/ar_mgr.conf. To set an alternative location, please perform the following:

1. Add 'armgr --conf_file **<ar-mgr-options-file-name>** to the **event_plugin_option**' option in the file **# Options** string that would be passed to the plugin(s) **event_plugin_options armgr --conf_file <ar-mgr-options-file-name>**
2. Run Subnet Manager with the new options file:

```
opensm -F <options-file-name>'
```

AR Manager options file contains two types of parameters:

1. General options - Options which describe the AR Manager behavior and the AR parameters that will be applied to all the switches in the fabric.
2. Per-switch options - Options which describe specific switch behavior.

Note the following:

- Adaptive Routing configuration file is case sensitive.
- You can specify options for nonexisting switch GUID. These options will be ignored until a switch with a matching GUID will be added to the fabric.
- Adaptive Routing configuration file is parsed every AR Manager cycle, which in turn is executed at every heavy sweep of the Subnet Manager.
- If the AR Manager fails to parse the options file, default settings for all the options will be used.

8.8.5.1 General AR Manager Options

Table 20 - Adaptive Routing Manager Options File

Option File	Description	Values
ENABLE: <true|false>	Enable/disable Adaptive Routing on fabric switches. Note that if a switch was identified by AR Manager as device that does not support AR, AR Manager will not try to enable AR on this switch. If the firmware of this switch was updated to support the AR, the AR Manager will need to be restarted (by restarting Subnet Manager) to allow it to configure the AR on this switch. This option can be changed on-the-fly.	Default: true
AR_MODE: <bounded|free>	Adaptive Routing Mode: • free: no constraints on output port selection • bounded: the switch does not change the output port during the same transmission burst. This mode minimizes the appearance of out-of-order packets This option can be changed on-the-fly.	Default: bounded
AGEING_TIME: <usec>	Applicable to bounded AR mode only. Specifies how much time there should be no traffic in order for the switch to declare a transmission burst as finished and allow changing the output port for the next transmission burst (32-bit value). This option can be changed on-the-fly.	Default: 30
MAX_ERRORS: <N> ERROR_WINDOW: <N>	When number of errors exceeds 'MAX_ERRORS' of send/receive errors or timeouts in less than 'ERROR_WINDOW' seconds, the AR Manager will abort, returning control back to the Subnet Manager. This option can be changed on-the-fly.	Values for both options: [0-0xffff] • MAX_ERRORS = 0: zero tollerance - abort configuration on first error. Default:10 • ERROR_WINDOW = 0: mechanism disabled - no error checking. Default: 5
LOG_FILE: <full path>	AR Manager log file. This option can be changed on-the-fly.	Default: /var/log/armgr.log
LOG_SIZE: <size in MB>	This option defines maximal AR Manager log file size in MB. The logfile will be truncated and restarted upon reaching this limit. This option cannot be changed on-the-fly.	0: unlimited log file size. Default: 5

8.8.5.1.1 Per-switch AR Options

A user can provide per-switch configuration options with the following syntax:

```
SWITCH <GUID> {
                <switch option 1>;
                <switch option 2>;
                ...
                }
```

The following are the per-switch options:

Table 21 - Adaptive Routing Manager Pre-Switch Options File

Option File	Description	Values
ENABLE: <true|false>	Allows you to enable/disable the AR on this switch. If the general ENABLE option value is set to 'false', then this per-switch option is ignored. This option can be changed on the fly.	Default: true
AGEING_TIME: <usec>	Applicable to bounded AR mode only. Specifies how much time there should be no traffic in order for the switch to declare a transmission burst as finished and allow changing the output port for the next transmission burst (32-bit value). In the pre-switch options file this option refers to the particular switch only This option can be changed on-the-fly.	Default: 30

8.8.5.1.2 Example of Adaptive Routing Manager Options File

```
ENABLE: true;
 LOG_FILE: /tmp/ar_mgr.log;
 LOG_SIZE: 100;
 MAX_ERRORS: 10;
 ERROR_WINDOW: 5;

SWITCH 0x12345 {
ENABLE: true;
AGEING_TIME: 77;
 }

SWITCH 0x0002c902004050f8 {
AGEING_TIME: 44;
 }

SWITCH 0xabcde {
ENABLE: false;
 }
```

8.9 Congestion Control

8.9.1 Congestion Control Overview

Congestion Control Manager is a Subnet Manager (SM) plug-in, i.e. it is a shared library (libcc-mgr.so) that is dynamically loaded by the Subnet Manager. Congestion Control Manager is installed as part of Mellanox OFED installation.

The Congestion Control mechanism controls traffic entry into a network and attempts to avoid oversubscription of any of the processing or link capabilities of the intermediate nodes and networks. Additionally, is takes resource reducing steps by reducing the rate of sending packets. Congestion Control Manager enables and configures Congestion Control mechanism on fabric nodes (HCAs and switches).

8.9.2 Running OpenSM with Congestion Control Manager

Congestion Control (CC) Manager can be enabled/disabled through SM options file. To do so, perform the following:

1. Create the file. Run:

```
opensm -c <options-file-name>'
```

2. Find the '**event_plugin_name**' option in the file, and add '**ccmgr**' to it.

```
# Event plugin name(s)
event_plugin_name ccmgr
```

3. Run the SM with the new options file: '**opensm -F <options-file-name>**'

Once the Congestion Control is enabled on the fabric nodes, to completely disable Congestion Control, you will need to actively turn it off. Running the SM w/o the CC Manager is not sufficient, as the hardware still continues to function in accordance to the previous CC configuration.

For further information on how to turn OFF CC, please refer to Section 8.9.3, “Configuring Congestion Control Manager”, on page 183

8.9.3 Configuring Congestion Control Manager

Congestion Control (CC) Manager comes with a predefined set of setting. However, you can fine-tune the CC mechanism and CC Manager behavior by modifying some of the options. To do so, perform the following:

1. Find the '**event_plugin_options**' option in the SM options file, and add the following:

```
conf_file <cc-mgr-options-file-name>':
# Options string that would be passed to the plugin(s)
event_plugin_options ccmgr --conf_file <cc-mgr-options-file-name>
```

2. Run the SM with the new options file: '**opensm -F <options-file-name>'**

To turn CC OFF, set 'enable' to 'FALSE' in the Congestion Control Manager configuration file, and run OpenSM ones with this configuration.

For the full list of CC Manager options with all the default values, See "Configuring Congestion Control Manager" on page 183.

For further details on the list of CC Manager options, please refer to the IB spec.

8.9.4 Configuring Congestion Control Manager Main Settings

To fine-tune CC mechanism and CC Manager behavior, and set the CC manager main settings, perform the following:

- To enables/disables Congestion Control mechanism on the fabric nodes, set the following parameter:

```
enable
```

  - The values are: **<TRUE | FALSE>**.
  - The default is: **True**

- CC manager configures CC mechanism behavior based on the fabric size. The larger the fabric is, the more aggressive CC mechanism is in its response to congestion. To manually modify CC manager behavior by providing it with an arbitrary fabric size, set the following parameter:

```
num_hosts
```

  - The values are: **[0-48K]**.
  - The default is: **0** (base on the CCT calculation on the current subnet size)

- The smaller the number value of the parameter, the faster HCAs will respond to the congestion and will throttle the traffic. Note that if the number is too low, it will result in suboptimal bandwidth. To change the mean number of packets between marking eligible packets with a FECN, set the following parameter:

```
marking_rate
```

  - The values are: **[0-0xffff]**.
  - The default is: **0xa**

- You can set the minimal packet size that can be marked with FECN. Any packet less than this size [bytes] will not be marked with FECN. To do so, set the following parameter:

```
packet_size
```

  - The values are: **[0-0x3fc0]**.
  - The default is: **0x200**

- When number of errors exceeds 'max_errors' of send/receive errors or timeouts in less than 'error_window' seconds, the CC MGR will abort and will allow OpenSM to proceed. To do so, set the following parameter:

```
max_errors
error_window
```

  - The values are:

    max_errors = 0: zero tollerance - abort configuration on first error
    error_window = 0: mechanism disabled - no error checking.[0-48K]

  - The default is: `5`

8.9.4.1 Congestion Control Manager Options File

Table 22 - Congestion Control Manager General Options File

Option File	Description	Values
enable	Enables/disables Congestion Control mechanism on the fabric nodes.	Values: <TRUE | FALSE> Default: True
num_hosts	Indicates the number of nodes. The CC table values are calculated based on this number.	Values: [0-48K] Default: 0 (base on the CCT calculation on the current subnet size)

Table 23 - Congestion Control Manager Switch Options File

Option File	Description	Values
threshold	Indicates how aggressive the congestion marking should be.	[0-0xf] • 0 - no packet marking, • 0xf - very aggressive Default: 0xf
marking_rate	The mean number of packets between marking eligible packets with a FECN	Values: [0-0xffff] Default: 0xa
packet_size	Any packet less than this size [bytes] will not be marked with FECN.	Values: [0-0x3fc0] Default: 0x200

Table 24 - Congestion Control Manager CA Options File

Option File	Desctiption	Values
port_control	Specifies the Congestion Control attribute for this port	Values: • 0 - QP based congestion control, • 1 - SL/Port based congestion control Default: 0

Table 24 - Congestion Control Manager CA Options File

Option File	Desctiption	Values
ca_control_map	An array of sixteen bits, one for each SL. Each bit indicates whether or not the corresponding SL entry is to be modified.	Values: 0xffff
ccti_increase	Sets the CC Table Index (CCTI) increase.	Default: 1
trigger_threshold	Sets the trigger threshold.	Default: 2
ccti_min	Sets the CC Table Index (CCTI) minimum.	Default: 0
cct	Sets all the CC table entries to a specified value . The first entry will remain 0, whereas last value will be set to the rest of the table.	Values: <comma-separated list> Default: 0 When the value is set to 0, the CCT calculation is based on the number of nodes.
ccti_timer	Sets for all SL's the given ccti timer.	Default: 0 When the value is set to 0, the CCT calculation is based on the number of nodes.

Table 25 - Congestion Control Manager CC MGR Options File

Option File	Desctiption	Values
max_errors error_window	When number of errors exceeds 'max_errors' of send/receive errors or timeouts in less than 'error_window' seconds, the CC MGR will abort and will allow OpenSM to proceed.	Values: • max_errors = 0: zero tollerance - abort configuration on first error. • error_window = 0: mechanism disabled - no error checking. Default: 5
cc_statistics_cycle	Enables CC MGR to collect statistics from all nodes every cc_statistics_cycle [seconds]	Default: 0 When the value is set to 0, no statistics are collected.

9 InfiniBand Fabric Diagnostic Utilities

9.1 Overview

The diagnostic utilities described in this chapter provide means for debugging the connectivity and status of InfiniBand (IB) devices in a fabric.

9.2 Utilities Usage

This section first describes common configuration, interface, and addressing for all the tools in the package. Then it provides detailed descriptions of the tools themselves including: operation, synopsis and options descriptions, error codes, and examples.

9.2.1 Common Configuration, Interface and Addressing

Topology File (Optional)

An InfiniBand fabric is composed of switches and channel adapter (HCA/TCA) devices. To identify devices in a fabric (or even in one switch system), each device is given a GUID (a MAC equivalent). Since a GUID is a non-user-friendly string of characters, it is better to alias it to a meaningful, user-given name. For this objective, the IB Diagnostic Tools can be provided with a "topology file", which is an optional configuration file specifying the IB fabric topology in user-given names.

For diagnostic tools to fully support the topology file, the user may need to provide the local system name (if the local hostname is not used in the topology file).

To specify a topology file to a diagnostic tool use one of the following two options:

1. On the command line, specify the file name using the option '`-t <topology file name>`'
2. Define the environment variable IBDIAG_TOPO_FILE

To specify the local system name to an diagnostic tool use one of the following two options:

1. On the command line, specify the system name using the option '`-s <local system name>`'
2. Define the environment variable IBDIAG_SYS_NAME

9.2.2 InfiniBand Interface Definition

The diagnostic tools installed on a machine connect to the IB fabric by means of an HCA port through which they send MADs. To specify this port to an IB diagnostic tool use one of the following options:

1. On the command line, specify the port number using the option '-p <local port number>' (see below)
2. Define the environment variable IBDIAG_PORT_NUM

In case more than one HCA device is installed on the local machine, it is necessary to specify the device's index to the tool as well. For this use on of the following options:

1. On the command line, specify the index of the local device using the following option: '-i <index of local device>'
2. Define the environment variable IBDIAG_DEV_IDX

9.2.3 Addressing

This section applies to the ibdiagpath tool only. A tool command may require defining the destination device or port to which it applies.

The following addressing modes can be used to define the IB ports:

- Using a Directed Route to the destination: (Tool option '-d')

  This option defines a directed route of output port numbers from the local port to the destination.

- Using port LIDs: (Tool option '-l'):

  In this mode, the source and destination ports are defined by means of their LIDs. If the fabric is configured to allow multiple LIDs per port, then using any of them is valid for defining a port.

- Using port names defined in the topology file: (Tool option '-n')

  This option refers to the source and destination ports by the names defined in the topology file. (Therefore, this option is relevant only if a topology file is specified to the tool.) In this mode, the tool uses the names to extract the port LIDs from the matched topology, then the tool operates as in the '-l' option.

9.3 ibdiagnet (of ibutils2) - IB Net Diagnostic

This version of ibdiagnet is included in the ibutils2 package, and it is run by default after installing Mellanox OFED. To use this ibdiagnet version, run: ibdiagnet

Please see ibutils2_release_notes.txt for additional information and known issues.

ibdiagnet scans the fabric using directed route packets and extracts all the available information regarding its connectivity and devices. It then produces the following files in the output directory (which is defined by the -o option described below).

Synopsis

```
[-i|--device <dev-name>] [-p|--port <port-num>]
   [-g|--guid <GUID in hex>] [--vlr <file>]
   [-r|--routing] [-u|--fat_tree] [-o|--output_path <directory>]
    [--skip <stage>] [--skip_plugin <library name>]
    [--pc] [-P|--counter <<PM>=<value>>]
    [--pm_pause_time <seconds>] [--ber_test]
    [--ber_use_data] [--ber_thresh <value>]
    [--extended_speeds <dev-type>] [--pm_per_lane]
    [--ls <2.5|5|10|14|25|FDR10>] [--lw <1x|4x|8x|12x>]
    [-w|--write_topo_file <file name>]
    [-t|--topo_file <file>] [--out_ibnl_dir <directory>]
    [--screen_num_errs <num>] [--smp_window <num>]
    [--gmp_window <num>] [--max_hops <max-hops>]
    [-V|--version] [-h|--help] [-H|--deep_help]
```

Options

```
-i|--device <dev-name>       : Specifies the name of the device of the port
                               used to connect to the IB fabric (in case
                               of multiple devices on he local system).
-p|--port <port-num>         : Specifies the local device's port number
                               used to connect to the IB fabric.
-g|--guid <GUID in hex>      : Specifies the local port GUID value of the
                                port used to connect to the IB fabric. If
                                GUID given is 0 than ibdiagnet displays
                                a list of possible port GUIDs and waits
                                for user input.
--vlr <file>                : Specifies opensm-path-records.dump file path,
                              src-dst to SL mapping generated by SM plugin.
                              ibdiagnet will use this mapping for MADs
                              sending and credit loop check (if -r option
                               selected).
-r|--routing                : Provides a report of the fabric qualities.
-u|--fat_tree               : Indicates that UpDown credit loop checking
                              should be done against automatically determined
                                roots.
-o|--output_path <directory>  : Specifies the directory where the output
                                files will be placed.
                                (default="/var/tmp/ibdiagnet2/").
--skip <stage>              : Skip the executions of the given stage. Applicable
                              skip stages:
                              (all | dup_guids | dup_node_desc | lids
                               | links | sm | pm | nodes_info | speed_width_check
                               | pkey | aguid).
--skip_plugin <library name>  : Skip the load of the given library name.
                                Applicable skip plugins:
                                (libibdiagnet_cable_diag_plugin |
                                libibdiagnet_cable_diag_plugin-2.1.1).
--pc                           : Reset all the fabric PM counters.
-P|--counter <<PM>=<value>>    : If any of the provided PM is greater then
                               its provided value than print it.
--pm_pause_time <seconds>     : Specifies the seconds to wait between first
                               counters sample and second counters sample.
                               If seconds given is 0 than no second counters
                               sample will be done.
                               (default=1).
```

```
--ber_test                    : Provides a BER test for each port. Calculate
                                BER for each port and check no BER value
                                has exceeds the BER threshold.
                                (default threshold="10^-12").
--ber_use_data                : Indicates that BER test will use the received
                                 data for calculation.
--ber_thresh <value>          : Specifies the threshold value for the BER
                                test. The reciprocal number of the BER should
                                be provided. Example: for 10^-12 than value
                                need to be 1000000000000 or 0xe8d4a51000
                                (10^12).If threshold given is 0 than all
                                 BER values for all ports will be reported.
--extended_speeds <dev-type>  : Collect and test port extended speeds counters.
                                dev-type: (sw | all).
--pm_per_lane                 : List all counters per lane (when available).
--ls <2.5|5|10|14|25|FDR10>   : Specifies the expected link speed.
--lw <1x|4x|8x|12x>           : Specifies the expected link width.
-w|--write_topo_file <file name>: Write out a topology file for the discovered
                                  topology.
-t|--topo_file <file>         : Specifies the topology file name.
--out_ibnl_dir <directory>    : The topology file custom system definitions
                                        (ibnl) directory.
--screen_num_errs <num>       : Specifies the threshold for printing errors
                                        to screen.
                                        (default=5).
--smp_window <num>            : Max smp MADs on wire. (default=8).
--gmp_window <num>            : Max gmp MADs on wire. (default=128).
--max_hops <max-hops>         : Specifies the maximum hops for the discovery
                                        process.
                                        (default=64).
-V|--version                  : Prints the version of the tool.
-h|--help                     : Prints help information (without plugins
                                        help if exists).
-H|--deep_help                : Prints deep help information (including plugins
                                        help).
```

Output Files

Table 26 lists the ibdiagnet output files that are placed under /var/tmp/ibdiagnet2.

Table 26 - ibdiagnet (of ibutils2) Output Files

Output File	Description
ibdiagnet2.lst	Fabric links in LST format
ibdiagnet2.sm	Subnet Manager
ibdiagnet2.pm	Ports Counters
ibdiagnet2.fdbs	Unicast FDBs
ibdiagnet2.mcfdbs	Multicast FDBx
ibdiagnet2.nodes_info	Information on nodes

Table 26 - ibdiagnet (of ibutils2) Output Files

Output File	Description
ibdiagnet2.db_csv	ibdiagnet internal database

An ibdiagnet run performs the following stages:

- Fabric discovery
- Duplicated GUIDs detection
- Links in INIT state and unresponsive links detection
- Counters fetch
- Error counters check
- Routing checks
- Link width and speed checks
- Alias GUIDs check
- Subnet Manager check
- Partition keys check
- Nodes information

Return Codes

```
0 - Success
1 - Failure (with description)
```

9.4 ibdiagnet (of ibutils) - IB Net Diagnostic

This version of ibdiagnet is included in the ibutils package, and it is not run by default after installing Mellanox OFED. To use this ibdiagnet version and not that of the ibutils package, you need to specify the full path: /opt/bin/ibdiagnet

ibdiagnet scans the fabric using directed route packets and extracts all the available information regarding its connectivity and devices. It then produces the following files in the output directory (which is defined by the -o option described below).

Synopsis

```
ibdiagnet [-c <count>] [-v] [-r] [-o <out-dir>] [-t <topo-file>]
          [-s <sys-name>] [-i <dev-index>] [-p <port-num>] [-wt]
          [-pm] [-pc] [-P <<PM>=<Value>>] [-lw <1x|4x|12x>] [-ls
          <2.5|5|10>]
          [-skip <ibdiag_check/s>] [-load_db <db_file>]
```

Options

```
-c <count>        Min number of packets to be sent across each link (default
                = 10)
-v                Enable verbose mode
-r                Provides a report of the fabric qualities
-t <topo-file>    Specifies the topology file name
-s <sys-name>     Specifies the local system name. Meaningful only if a
                topology file is specified
-i <dev-index>    Specifies the index of the device of the port used to
                connect to the IB fabric (in case of multiple devices on
                the local system)
-p <port-num>     Specifies the local device's port num used to connect to
                the IB fabric
-o <out-dir>      Specifies the directory where the output files will be
                placed (default = /tmp)
-lw <1x|4x|12x>   Specifies the expected link width
-ls <2.5|5|10>    Specifies the expected link speed
-pm               Dump all the fabric links, pm Counters into ibdiagnet.pm
-pc               Reset all the fabric links pmCounters
-P <PM=<Trash>>   If any of the provided pm is greater then its provided
                value, print it to screen
-skip <skip-option(s)>Skip the executions of the selected checks. Skip
                options (one or more can be specified): dup_guids
                zero_guids pm logical_state part ipoib all
-wt <file-name>   Write out the discovered topology into the given file.
                This flag is useful if you later want to check for
                changes from the current state of the fabric. A directory
                named ibdiag_ibnl is also created by this option, and
                holds the IBNL files required to load this topology.
                To use these files you will need to set the environment
                variable named IBDM_IBNL_PATH to that directory.
                The directory is located in /tmp or in the output
                directory provided by the -o flag.
-load_db <file-name>>Load subnet data from the given .db file, and skip
                subnet discovery stage.
                  Note: Some of the checks require actual subnet discovery,
                and therefore would not run when load_db is specified.
                These checks are: Duplicated/zero guids, link state, SMs
                status.
-h|--help         Prints the help page information
-V|--version      Prints the version of the tool
--vars            Prints the tool's environment variables and their values
```

Output Files

Table 27 - ibdiagnet (of ibutils) Output Files

Output File	Description
ibdiagnet.log	A dump of all the application reports generate according to the provided flags
ibdiagnet.lst	List of all the nodes, ports and links in the fabric

Table 27 - ibdiagnet (of ibutils) Output Files

Output File	Description
ibdiagnet.fdbs	A dump of the unicast forwarding tables of the fabric switches
ibdiagnet.mcfdbs	A dump of the multicast forwarding tables of the fabric switches
ibdiagnet.masks	In case of duplicate port/node Guids, these file include the map between masked Guid and real Guids
ibdiagnet.sm	List of all the SM (state and priority) in the fabric
ibdiagnet.pm	A dump of the pm Counters values, of the fabric links
ibdiagnet.pkey	A dump of the existing partitions and their member host ports
ibdiagnet.mcg	A dump of the multicast groups, their properties and member host ports
ibdiagnet.db	A dump of the internal subnet database. This file can be loaded in later runs using the -load_db option

In addition to generating the files above, the discovery phase also checks for duplicate node/port GUIDs in the IB fabric. If such an error is detected, it is displayed on the standard output. After the discovery phase is completed, directed route packets are sent multiple times (according to the -c option) to detect possible problematic paths on which packets may be lost. Such paths are explored, and a report of the suspected bad links is displayed on the standard output.

After scanning the fabric, if the `-r` option is provided, a full report of the fabric qualities is displayed. This report includes:

- SM report
- Number of nodes and systems
- Hop-count information: maximal hop-count, an example path, and a hop-count histogram
- All CA-to-CA paths traced
- Credit loop report
- mgid-mlid-HCAs multicast group and report
- Partitions report
- IPoIB report

In case the IB fabric includes only one CA, then CA-to-CA paths are not reported. Furthermore, if a topology file is provided, ibdiagnet uses the names defined in it for the output reports.

Error Codes

```
1 - Failed to fully discover the fabric
2 - Failed to parse command line options
3 - Failed to intract with IB fabric
4 - Failed to use local device or local port
5 - Failed to use Topology File
6 - Failed to load requierd Package
```

9.5 ibdiagpath - IB diagnostic path

`ibdiagpath` traces a path between two end-points and provides information regarding the nodes and ports traversed along the path. It utilizes device specific health queries for the different devices along the path.

The way `ibdiagpath` operates depends on the addressing mode used on the command line. If directed route addressing is used (-d flag), the local node is the source node and the route to the destination port is known apriori. On the other hand, if LID-route (or by-name) addressing is employed, then the source and destination ports of a route are specified by their LIDs (or by the names defined in the topology file). In this case, the actual path from the local port to the source port, and from the source port to the destination port, is defined by means of Subnet Management Linear Forwarding Table queries of the switch nodes along that path. Therefore, the path cannot be predicted as it may change.

`ibdiagpath` should not be supplied with contradicting local ports by the -p and -d flags (see synopsis descriptions below). In other words, when ibdiagpath is provided with the options -p and -d together, the first port in the direct route must be equal to the one specified in the "-p" option. Otherwise, an error is reported.

When ibdiagpath queries for the performance counters along the path between the source and destination ports, it always traverses the LID route, even if a directed route is specified. If along the LID route one or more links are not in the ACTIVE state, ibdiagpath reports an error.

Moreover, the tool allows omitting the source node in LID-route addressing, in which case the local port on the machine running the tool is assumed to be the source.

Synopsis

```
ibdiagpath {-n <[src-name,]dst-name>|-l <[src-lid,]dst-lid>|-d
            <p1,p2,p3,...>} [-c <count>] [-v] [-t <topo-file>]
            [-s <sys-name>] [-ic<dev-index>]c[-p <port-num>]
            [-o <out-dir>][-lw <1x|4x|12x>] [-ls <2.5|5|10>][-pm] [-pc]
            [-P <<PM counter>=<Trash Limit>>]
```

Options

```
-n <[src-name,]dst-name>
                          Names of the source and destination ports (as
                        defined in the topology file; source may be omit
                        ted -> local port is assumed to be the source)
-l <[src-lid,]dst-lid>
                          Source and destination LIDs (source may be omit
                        ted --> the local port is assumed to be the
                        source)
-d <p1,p2,p3,...>        Directed route from the local node (which is the
                        source) and the destination node
-c <count>                The minimal number of packets to be sent across
                        each link (default = 100)
-v                        Enable verbose mode
-t <topo-file>            Specifies the topology file name
-s <sys-name>              Specifies the local system name. Meaningful only
                        if a topology file is specified
-i <dev-index>            Specifies the index of the device of the port
                        used to connect to the IB fabric (in case of
                        multiple devices on the local system)
-p <port-num>              Specifies the local device's port number used to
                        connect to the IB fabric
-o <out-dir>              Specifies the directory where the output files
                        will be placed (default = /tmp)
-lw <1x|4x|12x>           Specifies the expected link width
-ls <2.5|5|10>            Specifies the expected link speed
-pm                       Dump all the fabric links, pm Counters into
                        ibdiagnet.pm
-pc                       Reset all the fabric links pmCounters
-P <PM=<Trash>>           If any of the provided pm is greater then its
                        provided value, print it to screen
-h|--help                 Prints the help page information
-V|--version              Prints the version of the tool
--vars                    Prints the tool's environment variables and
                        their values
```

Output Files

Table 28 - ibdiagpath Output Files

Output File	Description
ibdiagpath.log	A dump of all the application reports generated according to the provided flags
ibdiagnet.pm	A dump of the Performance Counters values, of the fabric links

Error Codes

```
1 - The path traced is un-healthy
2 - Failed to parse command line options
3 - More then 64 hops are required for traversing the local port to the
    "Source" port and then to the "Destination" port
4 - Unable to traverse the LFT data from source to destination
5 - Failed to use Topology File
6 - Failed to load required Package
```

9.6 ibv_devices

Lists InfiniBand devices available for use from userspace, including node GUIDs.

Synopsis

```
ibv_devices
```

Examples

1. List the names of all available InfiniBand devices.

```
> ibv_devices
    device                 node GUID
    ------              ----------------
    mthca0              0002c9000101d150
    mlx4_0              0000000000073895
```

9.7 ibv_devinfo

Queries InfiniBand devices and prints about them information that is available for use from user-space.

Synopsis

```
ibv_devinfo [-d <device>] [-i <port>] [-l] [-v]
```

Output Files

Table 29 lists the various flags of the command.

Table 29 - ibv_devinfo Flags and Options

Flag	Optional / Mandatory	Default (If Not Specified)	Description
-d <device> --ib-dev=<device>	Optional	First found device	Run the command for the provided IB device ‘device’
-i <port> --ib-port=<port>	Optional	All device ports	Query the specified device port <port>

Table 29 - ibv_devinfo Flags and Options

Flag	Optional / Mandatory	Default (If Not Specified)	Description
-l --list	Optional	Inactive	Only list the names of InfiniBand devices
-v --verbose	Optional	Inactive	Print all available information about the InfiniBand device(s)

Examples

1. List the names of all available InfiniBand devices.

```
> ibv_devinfo -l
2 HCAs found:
        mthca0
        mlx4_0
```

2. Query the device mlx4_0 and print user-available information for its Port 2.

```
> ibv_devinfo -d mlx4_0 -i 2
hca_id: mlx4_0
        fw_ver:                         2.5.944
        node_guid:                      0000:0000:0007:3895
        sys_image_guid:                 0000:0000:0007:3898
        vendor_id:                      0x02c9
        vendor_part_id:                 25418
        hw_ver:                         0xA0
        board_id:                       MT_04A0140005
        phys_port_cnt:                  2
                port:   2
                        state:                  PORT_ACTIVE (4)
                        max_mtu:                2048 (4)
                        active_mtu:             2048 (4)
                        sm_lid:                 1
                        port_lid:               1
                        port_lmc:               0x00
```

9.8 ibdev2netdev

ibdev2netdev enables association between IB devices and ports and the associated net device. Additionally it reports the state of the net device link.

Synopsys

```
ibdev2netdev [-v] [-h]
```

Options

```
-v Enable verbose mode. Adds additional information such as: Device ID, Part Number,
Card Name, Firmware version, IB port state.
-h Print help messages.
```

Example:

```
sw417:~/BXOFED-1.5.2-20101128-1524 # ibdev2netdev -v
mlx4_0 (MT26428 - MT1006X00034) FALCON QDR      fw 2.7.9288 port 1 (ACTIVE) ==> eth5
(Down)
mlx4_0 (MT26428 - MT1006X00034) FALCON QDR      fw 2.7.9288 port 1 (ACTIVE) ==> ib0
(Down)
mlx4_0 (MT26428 - MT1006X00034) FALCON QDR      fw 2.7.9288 port 2 (DOWN  ) ==> ib1
(Down)
mlx4_1 (MT26448 - MT1023X00777) Hawk Dual Port  fw 2.7.9400 port 1 (DOWN  ) ==> eth2
(Down)
mlx4_1 (MT26448 - MT1023X00777) Hawk Dual Port  fw 2.7.9400 port 2 (DOWN  ) ==> eth3
(Down)
sw417:~/BXOFED-1.5.2-20101128-1524 # ibdev2netdev
mlx4_0 port 1 ==> eth5 (Down)
mlx4_0 port 1 ==> ib0 (Down)
mlx4_0 port 2 ==> ib1 (Down)
mlx4_1 port 1 ==> eth2 (Down)
mlx4_1 port 2 ==> eth3 (Down)
```

9.9 ibstatus

Displays basic information obtained from the local InfiniBand driver. Output includes LID, SMLID, port state, port physical state, port width and port rate.

Synopsis

```
ibstatus  [-h] [<device name>[:<port>]]*
```

Output Files

Table 30 lists the various flags of the command.

Table 30 - ibstatus Flags and Options

Flag	Optional / Mandatory	Default (If Not Specified)	Description
-h	Optional		Print the help menu
<device>	Optional	All devices	Print information for the specified device. May specify more than one device

Table 30 - ibstatus Flags and Options

Flag	Optional / Mandatory	Default (If Not Specified)	Description
<port>	Optional, but requires specifying a device name	All ports of the specified device	Print information for the specified port only (of the specified device)

Examples

1. List the status of all available InfiniBand devices and their ports.

```
> ibstatus
Infiniband device 'mlx4_0' port 1 status:
        default gid:     fe80:0000:0000:0000:0000:0000:0007:3896
        base lid:        0x3
        sm lid:          0x3
        state:           4: ACTIVE
        phys state:      5: LinkUp
        rate:            20 Gb/sec (4X DDR)

Infiniband device 'mlx4_0' port 2 status:
        default gid:     fe80:0000:0000:0000:0000:0000:0007:3897
        base lid:        0x1
        sm lid:          0x1
        state:           4: ACTIVE
        phys state:      5: LinkUp
        rate:            20 Gb/sec (4X DDR)

Infiniband device 'mthca0' port 1 status:
        default gid:     fe80:0000:0000:0000:0002:c900:0101:d151
        base lid:        0x0
        sm lid:          0x0
        state:           2: INIT
        phys state:      5: LinkUp
        rate:            10 Gb/sec (4X)

Infiniband device 'mthca0' port 2 status:
        default gid:     fe80:0000:0000:0000:0002:c900:0101:d152
        base lid:        0x0
        sm lid:          0x0
        state:           2: INIT
        phys state:      5: LinkUp
        rate:            10 Gb/sec (4X)
```

2. List the status of specific ports of specific devices.

```
> ibstatus mthca0:1 mlx4_0:2
Infiniband device 'mthca0' port 1 status:
        default gid:     fe80:0000:0000:0000:0002:c900:0101:d151
        base lid:        0x0
        sm lid:          0x0
        state:           2: INIT
        phys state:      5: LinkUp
        rate:            10 Gb/sec (4X)

Infiniband device 'mlx4_0' port 2 status:
        default gid:     fe80:0000:0000:0000:0000:0000:0007:3897
        base lid:        0x1
        sm lid:          0x1
        state:           4: ACTIVE
        phys state:      5: LinkUp
        rate:            20 Gb/sec (4X DDR)
```

9.10 ibportstate

Enables querying the logical (link) and physical port states of an InfiniBand port. It also allows adjusting the link speed that is enabled on any InfiniBand port.

If the queried port is a *switch* port, then ibportstate can be used to

- disable, enable or reset the port
- validate the port's link width and speed against the peer port

Synopsis

```
ibportstate [-d] [-e] [-v] [-V] [-D] [-G] [-s <smlid>] \           [-C <ca_name>] [-P
<ca_port>] [-t <timeout_ms>]      \                          [<dest dr_path|lid|guid>]
<portnum> [<op> [<value>]]
```

Output Files

Table 31 lists the various flags of the command.

Table 31 - ibportstate Flags and Options

Flag	Optional / Mandatory	Default (If Not Specified)	Description
-h(help)	Optional		Print the help menu
-d(ebug)	Optional		Raise the IB debug level. May be used several times for higher debug levels (-ddd or -d -d -d)
-e(rr_show)	Optional		Show send and receive errors (time-outs and others)

Table 31 - ibportstate Flags and Options (Continued)

Flag	Optional / Mandatory	Default (If Not Specified)	Description
-v(erbose)	Optional		Increase verbosity level. May be used several times for additional verbosity (-vvv or -v -v -v)
-V(ersion)	Optional		Show version info
-D(irect)	Optional		Use directed path address arguments. The path is a comma separated list of out ports. Examples: '0' – self port '0,1,2,1,4' – out via port 1, then 2, ...
-G(uid)	Optional		Use GUID address argument. In most cases, it is the Port GUID. Example: '0x08f1040023'
-s <smlid>	Optional		Use <smlid> as the target lid for SM/SA queries
-C <ca_name>	Optional		Use the specified channel adapter or router
-P <ca_port>	Optional		Use the specified port
-t <timeout_ms>	Optional		Override the default timeout for the solicited MADs [msec]
<dest dr_path | lid | guid>	Optional		Destination's directed path, LID, or GUID.
<portnum>	Optional		Destination's port number
<op> [<value>]	Optional	query	Define the allowed port operations: enable, disable, reset, speed, and query

In case of multiple channel adapters (CAs) or multiple ports without a CA/port being specified, a port is chosen by the utility according to the following criteria:

1. The first ACTIVE port that is found.
2. If not found, the first port that is UP (physical link state is LinkUp).

Examples

1. Query the status of Port 1 of CA mlx4_0 (using ibstatus) and use its output (the LID – 3 in this case) to obtain additional link information using ibportstate.

```
>  ibstatus mlx4_0:1
Infiniband device 'mlx4_0' port 1 status:
        default gid:     fe80:0000:0000:0000:0000:0000:9289:3895
        base lid:        0x3
        sm lid:          0x3
        state:           2: INIT
        phys state:      5: LinkUp
        rate:            20 Gb/sec (4X DDR)

> ibportstate -C mlx4_0 3 1 query
PortInfo:
# Port info: Lid 3 port 1
LinkState:.....................Initialize
PhysLinkState:.................LinkUp
LinkWidthSupported:............1X or 4X
LinkWidthEnabled:..............1X or 4X
LinkWidthActive:...............4X
LinkSpeedSupported:............2.5 Gbps or 5.0 Gbps
LinkSpeedEnabled:..............2.5 Gbps or 5.0 Gbps
LinkSpeedActive:...............5.0 Gbps
```

2. Query the status of two channel adapters using directed paths.

```
>  ibportstate -C mlx4_0 -D 0 1
PortInfo:
# Port info: DR path slid 65535; dlid 65535; 0 port 1
LinkState:.....................Initialize
PhysLinkState:.................LinkUp
LinkWidthSupported:............1X or 4X
LinkWidthEnabled:..............1X or 4X
LinkWidthActive:...............4X
LinkSpeedSupported:............2.5 Gbps or 5.0 Gbps
LinkSpeedEnabled:..............2.5 Gbps or 5.0 Gbps
LinkSpeedActive:...............5.0 Gbps

>  ibportstate -C mthca0 -D 0 1
PortInfo:
# Port info: DR path slid 65535; dlid 65535; 0 port 1
LinkState:.....................Down
PhysLinkState:.................Polling
LinkWidthSupported:............1X or 4X
LinkWidthEnabled:..............1X or 4X
LinkWidthActive:...............4X
LinkSpeedSupported:............2.5 Gbps
LinkSpeedEnabled:..............2.5 Gbps
```

```
LinkSpeedActive:.................2.5 Gbps
```

3. Change the speed of a port.

```
# First query for current configuration
>  ibportstate -C mlx4_0 -D 0 1
PortInfo:
# Port info: DR path slid 65535; dlid 65535; 0 port 1
LinkState:.......................Initialize
PhysLinkState:...................LinkUp
LinkWidthSupported:..............1X or 4X
LinkWidthEnabled:................1X or 4X
LinkWidthActive:.................4X
LinkSpeedSupported:..............2.5 Gbps or 5.0 Gbps
LinkSpeedEnabled:................2.5 Gbps or 5.0 Gbps
LinkSpeedActive:.................5.0 Gbps

# Now change the enabled link speed
>  ibportstate -C mlx4_0 -D 0 1 speed 2
ibportstate -C mlx4_0 -D 0 1 speed 2
Initial PortInfo:
# Port info: DR path slid 65535; dlid 65535; 0 port 1
LinkSpeedEnabled:................2.5 Gbps

After PortInfo set:
# Port info: DR path slid 65535; dlid 65535; 0 port 1
LinkSpeedEnabled:................5.0 Gbps (IBA extension)

# Show the new configuration
>  ibportstate -C mlx4_0 -D 0 1
PortInfo:
# Port info: DR path slid 65535; dlid 65535; 0 port 1
LinkState:.......................Initialize
PhysLinkState:...................LinkUp
LinkWidthSupported:..............1X or 4X
LinkWidthEnabled:................1X or 4X
LinkWidthActive:.................4X
LinkSpeedSupported:..............2.5 Gbps or 5.0 Gbps
LinkSpeedEnabled:................5.0 Gbps (IBA extension)
LinkSpeedActive:.................5.0 Gbps
```

9.11 ibroute

Uses SMPs to display the forwarding tables—unicast (LinearForwardingTable or LFT) or multicast (MulticastForwardingTable or MFT)—for the specified switch LID and the optional lid (mlid) range. The default range is all valid entries in the range 1 to FDBTop.

Synopsis

```
ibroute [-h] [-d] [-v] [-V] [-a] [-n] [-D] [-G] [-M] [-s <smlid>] \[-C <ca_name>] [-P
<ca_port>] [     -t <timeout_ms>] \              [<dest dr_path|lid|guid> [<star-
tlid> [<endlid>]]]
```

Output Files

Table 32 lists the various flags of the command.

Table 32 - ibportstate Flags and Options

Flag	Optional / Mandatory	Default (If Not Specified)	Description
-h(help)	Optional		Print the help menu
-d(ebug)	Optional		Raise the IB debug level. May be used several times for higher debug levels (-ddd or -d -d -d)
-a(ll)	Optional		Show all LIDs in range, including invalid entries
-v(erbose)	Optional		Increase verbosity level. May be used several times for additional verbosity (-vvv or -v -v -v)
-V(ersion)	Optional		Show version info
-a(ll)	Optional		Show all LIDs in range, including invalid entries
-n(o_dests)	Optional		Do not try to resolve destinations
-D(irect)	Optional		Use directed path address arguments. The path is a comma separated list of out ports. Examples: ‘0’ – self port ‘0,1,2,1,4’ – out via port 1, then 2, ...
-G(uid)	Optional		Use GUID address argument. In most cases, it is the Port GUID. Example: ‘0x08f1040023’
-M(ulticast)	Optional		Show multicast forwarding tables. The parameters <startlid> and <endlid> specify the MLID range.
-s <smlid>	Optional		Use <smlid> as the target LID for SM/SA queries
-C <ca_name>	Optional		Use the specified channel adapter or router
-P <ca_port>	Optional		Use the specified port

Table 32 - ibportstate Flags and Options

Flag	Optional / Mandatory	Default (If Not Specified)	Description
-t <timeout_ms>	Optional		Override the default timeout for the solicited MADs [msec]
<dest dr_path | lid | guid>	Optional		Destination's directed path, LID, or GUID
<startlid>	Optional		Starting LID in an MLID range
<endlid>	Optional		Ending LID in an MLID range

Examples

1. Dump all Lids with valid out ports of the switch with Lid 2.

```
> ibroute 2
Unicast lids [0x0-0x8] of switch Lid 2 guid 0x0002c902fffff00a (MT47396 Infiniscale-III
Mellanox Technologies):
  Lid  Out   Destination
       Port     Info
0x0002 000 : (Switch portguid 0x0002c902fffff00a: 'MT47396 Infiniscale-III Mellanox
Technologies')
0x0003 021 : (Switch portguid 0x000b8cffff004016: 'MT47396 Infiniscale-III Mellanox
Technologies')
0x0006 007 : (Channel Adapter portguid 0x0002c90300001039: 'sw137 HCA-1')
0x0007 021 : (Channel Adapter portguid 0x0002c9020025874a: 'sw157 HCA-1')
0x0008 008 : (Channel Adapter portguid 0x0002c902002582cd: 'sw136 HCA-1')
5 valid lids dumped
```

2. Dump all Lids with valid out ports of the switch with Lid 2.

```
> ibroute 2
Unicast lids [0x0-0x8] of switch Lid 2 guid 0x0002c902fffff00a (MT47396 Infiniscale-III
Mellanox Technologies):
  Lid  Out   Destination
       Port     Info
0x0002 000 : (Switch portguid 0x0002c902fffff00a: 'MT47396 Infiniscale-III Mellanox
Technologies')
0x0003 021 : (Switch portguid 0x000b8cffff004016: 'MT47396 Infiniscale-III Mellanox
Technologies')
0x0006 007 : (Channel Adapter portguid 0x0002c90300001039: 'sw137 HCA-1')
0x0007 021 : (Channel Adapter portguid 0x0002c9020025874a: 'sw157 HCA-1')
0x0008 008 : (Channel Adapter portguid 0x0002c902002582cd: 'sw136 HCA-1')
5 valid lids dumped
```

3. Dump all Lids in the range 3 to 7 with valid out ports of the switch with Lid 2.

```
> ibroute 2 3 7
```

```
Unicast lids [0x3-0x7] of switch Lid 2 guid 0x0002c902fffff00a (MT47396 Infiniscale-III
Mellanox Technologies):
  Lid  Out   Destination
       Port     Info
0x0003 021 : (Switch portguid 0x000b8cffff004016: 'MT47396 Infiniscale-III Mellanox
Technologies')
0x0006 007 : (Channel Adapter portguid 0x0002c90300001039: 'sw137 HCA-1')
0x0007 021 : (Channel Adapter portguid 0x0002c9020025874a: 'sw157 HCA-1')
3 valid lids dumped
```

4. Dump all Lids with valid out ports of the switch with portguid `0x000b8cffff004016`.

```
> ibroute -G 0x000b8cffff004016
Unicast lids [0x0-0x8] of switch Lid 3 guid 0x000b8cffff004016 (MT47396 Infiniscale-III
Mellanox Technologies):
  Lid  Out   Destination
       Port     Info
0x0002 023 : (Switch portguid 0x0002c902fffff00a: 'MT47396 Infiniscale-III Mellanox
Technologies')
0x0003 000 : (Switch portguid 0x000b8cffff004016: 'MT47396 Infiniscale-III Mellanox
Technologies')
0x0006 023 : (Channel Adapter portguid 0x0002c90300001039: 'sw137 HCA-1')
0x0007 020 : (Channel Adapter portguid 0x0002c9020025874a: 'sw157 HCA-1')
0x0008 024 : (Channel Adapter portguid 0x0002c902002582cd: 'sw136 HCA-1')
5 valid lids dumped
```

5. Dump all non-empty mlids of switch with Lid 3.

```
> ibroute -M 3
Multicast mlids [0xc000-0xc3ff] of switch Lid 3 guid 0x000b8cffff004016 (MT47396
Infiniscale-III Mellanox Technologies):
            0                   1                   2
     Ports: 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4
 MLid
0xc000                                              x
0xc001                                              x
0xc002                                              x
0xc003                                              x
0xc020                 x
0xc021                 x
0xc022                 x
0xc023                 x
0xc024                 x
0xc040                 x
0xc041                 x
0xc042                 x
12 valid mlids dumped
```

9.12 smpquery

Provides a basic subset of standard SMP queries to query Subnet management attributes such as node info, node description, switch info, and port info.

Synopsis

```
smpquery [-h] [-d] [-e] [-v] [-D] [-G] [-s <smlid>] [-V]         [-C <ca_name>] [-P
<ca_port>] [-t <timeout_ms>]         [--node-name-map <node-name-map>]         <op>
<dest dr_path|lid|guid> [op params]
```

Output Files

Table 33 lists the various flags of the command.

Table 33 - smpquery Flags and Options

Flag	Optional / Mandatory	Default (If Not Specified)	Description
-h(help)	Optional		Print the help menu
-d(ebug)	Optional		Raise the IB debug level. May be used several times for higher debug levels (-ddd or -d -d -d)
-e(rr_show)	Optional		Show send and receive errors (timeouts and others)
-v(erbose)	Optional		Increase verbosity level. May be used several times for additional verbosity (-vvv or -v -v -v)
-D(irect)	Optional		Use directed path address arguments. The path is a comma separated list of out ports. Examples: ‘0’ – self port ‘0,1,2,1,4’ – out via port 1, then 2, ...
-G(uid)	Optional		Use GUID address argument. In most cases, it is the Port GUID. Example: ‘0x08f1040023’
-s <smlid>	Optional		Use <smlid> as the target LID for SM/SA queries
-V(ersion)	Optional		Show version info
-C <ca_name>	Optional		Use the specified channel adapter or router
-P <ca_port>	Optional		Use the specified port

Table 33 - smpquery Flags and Options

Flag	Optional / Mandatory	Default (If Not Specified)	Description
-t <timeout_ms>	Optional		Override the default timeout for the solicited MADs [msec]
<op>	Mandatory		Supported operations: nodeinfo <addr> nodedesc <addr> portinfo <addr> [<portnum>] switchinfo <addr> pkeys <addr> [<portnum>] sl2vl <addr> [<portnum>] vlarb <addr> [<portnum>] guids <addr> mepi <addr> [<portnum>]
<dest dr_path | lid | guid>	Optional		Destination's directed path, LID, or GUID

Examples

1. Query PortInfo by LID, with port modifier.

```
> smpquery portinfo 1 1
# Port info: Lid 1 port 1
Mkey:.............................0x0000000000000000
GidPrefix:........................0xfe80000000000000
Lid:..............................0x0001
SMLid:............................0x0001
CapMask:..........................0x251086a
                                 IsSM
                                 IsTrapSupported
                                 IsAutomaticMigrationSupported
                                 IsSLMappingSupported
                                 IsSystemImageGUIDsupported
                                 IsCommunicatonManagementSupported
                                 IsVendorClassSupported
                                 IsCapabilityMaskNoticeSupported
                                 IsClientRegistrationSupported
DiagCode:.........................0x0000
MkeyLeasePeriod:..................0
LocalPort:........................1
LinkWidthEnabled:.................1X or 4X
LinkWidthSupported:...............1X or 4X
LinkWidthActive:..................4X
```

```
LinkSpeedSupported:.............2.5 Gbps or 5.0 Gbps
LinkState:......................Active
PhysLinkState:..................LinkUp
LinkDownDefState:...............Polling
ProtectBits:....................0
LMC:............................0
LinkSpeedActive:................5.0 Gbps
LinkSpeedEnabled:...............2.5 Gbps or 5.0 Gbps
NeighborMTU:....................2048
SMSL:...........................0
VLCap:..........................VL0-7
InitType:.......................0x00
VLHighLimit:....................4
VLArbHighCap:...................8
VLArbLowCap:....................8
InitReply:......................0x00
MtuCap:.........................2048
VLStallCount:...................0
HoqLife:........................31
OperVLs:........................VL0-3
PartEnforceInb:.................0
PartEnforceOutb:................0
FilterRawInb:...................0
FilterRawOutb:..................0
MkeyViolations:.................0
PkeyViolations:.................0
QkeyViolations:.................0
GuidCap:........................128
ClientReregister:...............0
SubnetTimeout:..................18
RespTimeVal:....................16
LocalPhysErr:...................8
OverrunErr:.....................8
MaxCreditHint:..................0
RoundTrip:......................0
```

2. Query SwitchInfo by GUID.

```
> smpquery -G switchinfo 0x000b8cffff004016
# Switch info: Lid 3
LinearFdbCap:...................49152
RandomFdbCap:...................0
McastFdbCap:....................1024
LinearFdbTop:...................8
DefPort:........................0
DefMcastPrimPort:...............0
DefMcastNotPrimPort:............0
```

```
LifeTime:.....................18
StateChange:..................0
LidsPerPort:..................0
PartEnforceCap:...............32
InboundPartEnf:...............1
OutboundPartEnf:..............1
FilterRawInbound:.............1
FilterRawOutbound:............1
EnhancedPort0:................0
```

3. Query NodeInfo by direct route.

```
> smpquery -D nodeinfo 0
# Node info: DR path slid 65535; dlid 65535; 0
BaseVers:.....................1
ClassVers:....................1
NodeType:.....................Channel Adapter
NumPorts:.....................2
SystemGuid:...................0x0002c9030000103b
Guid:.........................0x0002c90300001038
PortGuid:.....................0x0002c90300001039
PartCap:......................128
DevId:........................0x634a
Revision:.....................0x000000a0
LocalPort:....................1
VendorId:.....................0x0002c9
```

9.13 perfquery

Queries InfiniBand ports' performance and error counters. Optionally, it displays aggregated counters for all ports of a node. It can also reset counters after reading them or simply reset them.

Synopsis

```
perfquery [-h] [-d] [-G] [-a] [-l] [-r] [-C <ca_name>] [-P <ca_port>] [-R]        [-t
<timeout_ms>] [-V] [<lid|guid> [[port][reset_mask]]]
```

Output Files

Table 34 lists the various flags of the command.

Table 34 - perfquery Flags and Options

Flag	Optional / Mandatory	Default (If Not Specified)	Description
-h(help)	Optional		Print the help menu
-d(ebug)	Optional		Raise the IB debug level. May be used several times for higher debug levels (-ddd or -d -d -d)

Table 34 - perfquery Flags and Options

Flag	Optional / Mandatory	Default (If Not Specified)	Description
-G(uid)	Optional		Use GUID address argument. In most cases, it is the Port GUID. Example: '0x08f1040023'
-a	Optional		Apply query to all ports
-l	Optional		Loop ports
-r	Optional		Reset the counters after reading them
-C <ca_name>	Optional		Use the specified channel adapter or router
-P <ca_port>	Optional		Use the specified port
-R	Optional		Reset the counters
-t <timeout_ms>	Optional		Override the default timeout for the solicited MADs [msec]
-V(ersion)	Optional		Show version info
<lid | guid> [[port][reset_mask]]	Optional		LID or GUID

Examples

```
perfquery -r 32 1       # read performance counters and reset
perfquery -e -r 32 1    # read extended performance counters and reset
perfquery -R 0x20 1     # reset performance counters of port 1 only
perfquery -e -R 0x20 1  # reset extended performance counters of port 1 only
perfquery -R -a 32      # reset performance counters of all ports
perfquery -R 32 2 0x0fff        # reset only error counters of port 2
perfquery -R 32 2 0xf000        # reset only non-error counters of port 2
```

1. Read local port's performance counters.

```
> perfquery
# Port counters: Lid 6 port 1
PortSelect:......................1
CounterSelect:...................0x1000
SymbolErrors:....................0
LinkRecovers:....................0
LinkDowned:......................0
RcvErrors:.......................0
RcvRemotePhysErrors:.............0
```

```
RcvSwRelayErrors:.............0
XmtDiscards:..................0
XmtConstraintErrors:..........0
RcvConstraintErrors:..........0
LinkIntegrityErrors:..........0
ExcBufOverrunErrors:..........0
VL15Dropped:..................0
XmtData:.....................55178210
RcvData:.....................55174680
XmtPkts:.....................766366
RcvPkts:.....................766315
```

2. Read performance counters from LID 2, all ports.

```
> smpquery -a 2
# Port counters: Lid 2 port 255
PortSelect:..................255
CounterSelect:...............0x0100
SymbolErrors:................65535
LinkRecovers:................255
LinkDowned:..................16
RcvErrors:...................657
RcvRemotePhysErrors:..........0
RcvSwRelayErrors:.............70
XmtDiscards:.................488
XmtConstraintErrors:..........0
RcvConstraintErrors:..........0
LinkIntegrityErrors:..........0
ExcBufOverrunErrors:..........0
VL15Dropped:..................0
XmtData:.....................129840354
RcvData:.....................129529906
XmtPkts:.....................1803332
RcvPkts:.....................1799018
```

3. Read then reset performance counters from LID 2, port 1.

```
> perfquery  -r 2 1
# Port counters: Lid 2 port 1
PortSelect:..................1
CounterSelect:...............0x0100
SymbolErrors:................0
LinkRecovers:................0
LinkDowned:..................0
RcvErrors:...................0
RcvRemotePhysErrors:..........0
RcvSwRelayErrors:.............0
XmtDiscards:.................3
XmtConstraintErrors:..........0
```

```
RcvConstraintErrors:.............0
LinkIntegrityErrors:.............0
ExcBufOverrunErrors:.............0
VL15Dropped:.....................0
XmtData:.........................0
RcvData:.........................0
XmtPkts:.........................0
RcvPkts:.........................0
```

9.14 ibcheckerrs

Validates an IB port (or node) and reports errors in counters above threshold.

Check specified port (or node) and report errors that surpassed their predefined threshold. Port address is lid unless -G option is used to specify a GUID address. The predefined thresholds can be dumped using the -s option, and a user defined threshold_file (using the same format as the dump) can be specified using the -t <file> option.

Synopsis

```
ibcheckerrs [-h] [-b] [-v] [-G] [-T <threshold_file>] [-s] [-N |-nocolor] [-C ca_name]
[-P ca_port] [-t timeout_ms] <lid|guid>  [<port>]
```

Output Files

Table 35 lists the various flags of the command.

Table 35 - ibcheckerrs Flags and Options

Flag	Optional / Mandatory	Default (If Not Specified)	Description
-h(help)	Optional		Print the help menu
-b	Optional		Print in brief mode. Reduce the output to show only if errors are present, not what they are
-v(erbose)	Optional		Increase verbosity level. May be used several times for additional verbosity (-vvv or -v -v -v)
-G(uid)	Optional		Use GUID address argument. In most cases, it is the Port GUID. Example: ‘0x08f1040023’
-T <threshold_file>	Optional		Use specified threshold file
-s	Optional		Show the predefined thresholds
-N | -nocolor	Optional	color mode	Use mono mode rather than color mode

Table 35 - ibcheckerrs Flags and Options

Flag	Optional / Mandatory	Default (If Not Specified)	Description
-C <ca_name>	Optional		Use the specified channel adapter or router
-P <ca_port>	Optional		Use the specified port
-t <timeout_ms>	Optional		Override the default timeout for the solicited MADs [msec]
<lid | guid>	Mandatory with -G flag		Use the specified port's or node's LID/GUID (with -G option)
[<port>]	Mandatory without -G flag		Use the specified port

Examples

1. Check aggregated node counter for LID 0x2.

```
> ibcheckerrs 2
#warn: counter SymbolErrors = 65535     (threshold 10) lid 2 port 255
#warn: counter LinkRecovers = 255       (threshold 10) lid 2 port 255
#warn: counter LinkDowned = 12  (threshold 10) lid 2 port 255
#warn: counter RcvErrors = 565  (threshold 10) lid 2 port 255
#warn: counter XmtDiscards = 441        (threshold 100) lid 2 port 255
Error check on lid 2 (MT47396 Infiniscale-III Mellanox Technologies) port all:  FAILED
```

2. Check port counters for LID 2 Port 1.

```
> ibcheckerrs -v 2 1
Error check on lid 2 (MT47396 Infiniscale-III Mellanox Technologies) port 1:  OK
```

3. Check the LID2 Port 1 using the specified threshold file.

```
> cat thresh1
SymbolErrors=10
LinkRecovers=10
LinkDowned=10
RcvErrors=10
RcvRemotePhysErrors=100
RcvSwRelayErrors=100
XmtDiscards=100
XmtConstraintErrors=100
RcvConstraintErrors=100
LinkIntegrityErrors=10
ExcBufOverrunErrors=10
VL15Dropped=100
```

```
> ibcheckerrs -v -T thresh1 2 1
Error check on lid 2 (MT47396 Infiniscale-III Mellanox Technologies) port 1:  OK
```

9.15 mstflint

Queries and burns a binary firmware-image file on non-volatile (Flash) memories of Mellanox InfiniBand and Ethernet network adapters. The tool requires root privileges for Flash access.

If you purchased a standard Mellanox Technologies network adapter card, please download the firmware image from www.mellanox.com > Downloads > Firmware. If you purchased a non-standard card from a vendor other than Mellanox Technologies, please contact your vendor.

To run mstflint, you must know the device location on the PCI bus. See Example 1 for details.

Synopsis

```
mstflint [switches...] <command> [parameters...]
```

Output Files

Table 36 lists the various switches of the utility, and Table 37 lists its commands.

Table 36 - mstflint Switches (Sheet 1 of 3)

Switch	Affected/ Relevant Commands	Description
-h		Print the help menu
-hh		Print an extended help menu
-d[evice] <device>	All	Specify the device to which the Flash is connected.
-guid <GUID>	burn, sg	GUID base value. 4 GUIDs are automatically assigned to the following values: guid -> node GUID guid+1 -> port1 guid+2 -> port2 guid+3 -> system image GUID. Note: Port2 guid will be assigned even for a single port HCA; the HCA ignores this value.
-guids <GUIDs...>	burn, sg	4 GUIDs must be specified here. The specified GUIDs are assigned the following values, repectively: node, port1, port2 and system image GUID. Note: Port2 guid must be specified even for a single port HCA; the HCA ignores this value. It can be set to 0x0.

Table 36 - mstflint Switches (Sheet 2 of 3)

Switch	Affected/ Relevant Commands	Description
-mac <MAC>	burn, sg	MAC address base value. Two MACs are automatically assigned to the following values: mac -> port1 mac+1 -> port2 Note: This switch is applicable only for Mellanox Technologies Ethernet products.
-macs <MACs...>	burn, sg	Two MACs must be specified here. The specified MACs are assigned to port1 and port2, repectively. Note: This switch is applicable only for Mellanox Technologies Ethernet products.
-blank_guids	burn	Burn the image with blank GUIDs and MACs (where applicable). These values can be set later using the sg command – see Table 37 below.
-clear_semaphore	No commands allowed	Force clear the Flash semaphore on the device. No command is allowed when this switch is used. Warning: May result in system instability or Flash corruption if the device or another application is currently using the Flash.
-i[mage] <image>	burn, verify	Binary image file
-qq	burn, query	Run a quick query. When specified, mstflint will not perform full image integrity checks during the query operation. This may shorten execution time when running over slow interfaces (e.g., I2C, MTUSB-1).
-nofs	burn	Burn image in a non-failsafe manner
-skip_is	burn	Allow burning the firmware image without updating the invariant sector. This is to ensure failsafe burning even when an invariant sector difference is detected.
-byte_mode	burn, write	Shift address when accessing Flash internal registers. May be required for burn/write commands when accessing certain Flash types.
-s[ilent]	burn	Do not print burn progress messages
-y[es]	All	Non-interactive mode: Assume the answer is “yes” to all questions.
-no	All	Non-interactive mode: Assume the answer is “no” to all questions.

Table 36 - mstflint Switches (Sheet 3 of 3)

Switch	Affected/ Relevant Commands	Description
-vsd <string>	burn	Write this string of up to 208 characters to VSD upon a burn command.
-use_image_ps	burn	Burn vsd as it appears in the given image - do not keep existing VSD on Flash.
-dual_image	burn	Make the burn process burn two images on Flash. The current default failsafe burn process burns a single image (in alternating locations).
-v		Print version info

Table 37 - mstflint Commands

Command	Description
b[urn]	Burn Flash
q[uery]	Query miscellaneous Flash/firmware characteristics
v[erify]	Verify the entire Flash
bb	Burn Block: Burn the given image as is, without running any checks
sg	Set GUIDs
ri <out-file>	Read the firmware image on the Flash into the specified file
dc <out-file>	Dump Configuration: Print a firmware configuration file for the given image to the specified output file
e[rase] <addr>	Erase sector
rw <addr>	Read one DWORD from Flash
ww <addr> < data>	Write one DWORD to Flash
wwne <addr>	Write one DWORD to Flash without sector erase
wbne <addr> <size> <data...>	Write a data block to Flash without sector erase
rb <addr> <size> [out-file]	Read a data block from Flash
swreset	SW reset the target InfniScale® IV device. This command is supported only in the In-Band access method.

Possible command return values are:

```
0 - successful completion
1 - error has occurred
7 - the burn command was aborted because firmware is current
```

Examples

1. Find Mellanox Technologies's ConnectX® VPI cards with PCI Express running at 2.5GT/s and InfiniBand ports at DDR / or Ethernet ports at 10GigE.

```
> /sbin/lspci -d 15b3:634a
04:00.0 InfiniBand: Mellanox Technologies MT25418 [ConnectX IB DDR, PCIe 2.0 2.5GT/s]
(rev a0).
```

   In the example above, 15b3 is Mellanox Technologies's vendor number (in hexadecimal), and 634a is the device's PCI Device ID (in hexadecimal). The number string 04:00.0 identifies the device in the form bus:dev.fn.

The PCI Device IDs of Mellanox Technologies' devices can be obtained from the PCI ID Repository Website at http://pci-ids.ucw.cz/read/PC/15b3.

2. Verify the ConnectX firmware using its ID (using the results of the example above).

```
> mstflint -d 04:00.0 v
     ConnectX failsafe image. Start address: 80000. Chunk size 80000:

NOTE: The addresses below are contiguous logical addresses. Physical addresses on flash
may be different, based on the image start address and chunk size

     /0x00000038-0x000010db (0x0010a4)/ (BOOT2) - OK
     /0x000010dc-0x00004947 (0x00386c)/ (BOOT2) - OK
     /0x00004948-0x000052c7 (0x000980)/ (Configuration) - OK
     /0x000052c8-0x0000530b (0x000044)/ (GUID) - OK
     /0x0000530c-0x0000542f (0x000124)/ (Image Info) - OK
     /0x00005430-0x0000634f (0x000f20)/ (DDR) - OK
     /0x00006350-0x0000f29b (0x008f4c)/ (DDR) - OK
     /0x0000f29c-0x0004749b (0x038200)/ (DDR) - OK
     /0x0004749c-0x0005913f (0x011ca4)/ (DDR) - OK
     /0x00059140-0x0007a123 (0x020fe4)/ (DDR) - OK
     /0x0007a124-0x0007bdff (0x001cdc)/ (DDR) - OK
     /0x0007be00-0x0007eb97 (0x002d98)/ (DDR) - OK
     /0x0007eb98-0x0007f0af (0x000518)/ (Configuration) - OK
     /0x0007f0b0-0x0007f0fb (0x00004c)/ (Jump addresses) - OK
     /0x0007f0fc-0x0007f2a7 (0x0001ac)/ (FW Configuration) - OK

FW image verification succeeded. Image is bootable.
```

9.16 ibv_asyncwatch

Display asynchronous events forwarded to userspace for an InfiniBand device.

Synopsis

```
ibv_asyncwatch
```

Examples

1. Display asynchronous events.

```
> ibv_asyncwatch
mlx4_0: async event FD 4
```

9.17 ibdump

Dump InfiniBand traffic that flows to and from Mellanox Technologies ConnectX® family adapters InfiniBand ports. The dump file can be loaded by the Wireshark tool for graphical traffic analysis.

The following describes a work flow for local HCA (adapter) sniffing:

- Run ibdump with the desired options
- Run the application that you wish its traffic to be analyzed
- Stop ibdump (CTRL-C) or wait for the data buffer to fill (in --mem-mode)
- Open Wireshark and load the generated file

How to Get Wireshark:

Download the current release from www.wireshark.org for a Linux or Windows environment. See the ibdump_release_notes.txt file for more details.

Although ibdump is a Linux application, the generated .pcap file may be analyzed on either operating system.

Synopsis

```
ibdump [options]
```

Output Files

```
-d, --ib-dev=<dev>          use RDMA device <dev> (default first device found)
                            The relevant devices can be listed by running the
                            'ibv_devinfo' command.
-i, --ib-port=<port>        use port <port> of IB device (default 1)
-w, --write=<file>          dump file name (default "sniffer.pcap")
                            '-' stands for stdout - enables piping to tcpdump or
                            tshark.
-o, --output=<file>         alias for the '-w' option. Do not use - for backward
                            compatibility
```

```
-b, --max-burst=<log2 burst>    og2 of the maximal burst size that can be captured with
l                               no packets loss.
                                Each entry takes ~ MTU bytes of memory (default 12 -
                                4096 entries)
-s, --silent                    do not print progress indication.
--mem-mode <size>               when specified, packets are written to file only after
                                the capture is stopped. It is faster than default mode
                                (less chance for packet loss), but takes more memory. In
                                this mode, ibdump stops after <size> bytes are captured
--decap                         Decapsulate port mirroring headers. Should be used when
                                capturing RSPAN traffic.
-h, --help                      Display this help screen.
-v, --version                   Print version information.
```

Examples

1. Run ibdump.

```
> ibdump
 ------------------------------------------------
 IB device                      : "mlx4_0"
 IB port                        : 1
 Dump file                      : sniffer.pcap
 Sniffer WQEs (max burst size)  : 4096
 ------------------------------------------------

Initiating resources ...
searching for IB devices in host
Port active_mtu=2048
MR was registered with addr=0x60d850, lkey=0x28042601, rkey=0x28042601, flags=0x1
QP was created, QP number=0x4004a

Ready to capture (Press ^c to stop):
```

Appendix A: Mellanox FlexBoot

A.1 Overview

Mellanox FlexBoot is a multiprotocol remote boot technology. FlexBoot supports remote Boot over InfiniBand (BoIB) and over Ethernet.

Using Mellanox Virtual Protocol Interconnect (VPI) technologies available in ConnectX® adapters, FlexBoot gives IT Managers' the choice to boot from a remote storage target (iSCSI target) or a LAN target (Ethernet Remote Boot Server) using a single ROM image on Mellanox ConnectX products.

FlexBoot is based on the open source project iPXE available at http://www.ipxe.org.

FlexBoot first initializes the adapter device, senses the port protocol – Ethernet or InfiniBand, and brings up the port. Then it connects to a DHCP server to obtain its assigned IP address and network parameters, and also to obtain the source location of the kernel/OS to boot from. The DHCP server instructs FlexBoot to access the kernel/OS through a TFTP server, an iSCSI target, or some other service.

For an InfiniBand port, Mellanox FlexBoot implements a network driver with IP over IB acting as the transport layer. IP over IB is part of the *Mellanox OFED for Linux* software package (see www.mellanox.com > Products > InfiniBand/VPI SW/Drivers).

The binary code is exported by the device as an expansion ROM image.

A.2 FlexBoot Package

The FlexBoot package is provided as a tarball (.tgz extension). Uncompress it using the command "tar zxf <package file name>". The tarball contains PXE binary files (with the *.mrom extension) for the supported adapter devices. See the release notes file FlexBoot-<flexboot_version>_release_notes.txt for details. The package includes the following files:

- `dhcpd.conf` – sample DHCP configuration file
- `dhcp.patch` – patch file for DHCP v3.1.3

A.3 Burning the Expansion ROM Image

A.3.1 Burning the Image on ConnectX®-2 / ConnectX®-3

This section is valid for ConnectX®-2 devices with firmware versions 2.9.1000 or later and ConnectX®-3 firmware versions 2.30.3000 or later.

Prerequisites

1. Expansion ROM Image

   The expansion ROM images are provided as part of the Mellanox FlexBoot package and are listed in the release notes file `FlexBoot-<flexboot_version>_release_notes.txt`.

2. Firmware Burning Tools

You need to install the Mellanox Firmware Tools (MFT) package (version 3.0.0 or later) in order to burn the PXE ROM image. To download MFT, see Firmware Tools under www.mellanox.com > Products > InfiniBand/VPI Drivers > Firmware Tools.

Image Burning Procedure

To burn the composite image, perform the following steps:

1. Obtain the MST device name. Run:

```
# mst start
# mst status
```

   The device name will be of the form: `mt<dev_id>_pci{_cr0|conf0}`.[1]

2. Create and burn the composite image. Run:

```
flint -dev <mst device name> brom <expansion ROM image>
```

   Example on Linux:

```
flint -dev /dev/mst/mt26428_pci_cr0 brom ConnectX_26428_ROM-X_X_XXX.mrom
```

   Example on Windows:

```
flint -dev mt26428_pci_cr0 brom ConnectX_26428_ROM-X_X_XXX.mrom
```

Removing the Expansion ROM Image

Remove the expansion ROM image. Run:

```
flint -dev <mst device name> drom
```

When removing the expansion ROM image, you also remove Flexboot from the boot device list.

A.4 Preparing the DHCP Server in Linux Environment

The DHCP server plays a major role in the boot process by assigning IP addresses for FlexBoot clients and instructing the clients where to boot from. FlexBoot requires that the DHCP server run on a machine which supports IP over IB.

A.4.1 Installing the DHCP Server

Install DHCP client/server in embedded within the Linux Distribution.

A.4.2 Configuring the DHCP Server

A.4.2.1 For ConnectX Family Devices

When a FlexBoot client boots, it sends the DHCP server various information, including its DHCP client identifier. This identifier is used to distinguish between the various DHCP sessions.

1. Depending on the OS, the device name may be superceded with a prefix.

The value of the client identifier is composed of a prefix — ff:00:00:00:00:00:02:00:00:02:c9:00 — and an 8-byte port GUID (all separated by colons and represented in hexadecimal digits).

Extracting the Port GUID – Method I

➢ ***To obtain the port GUID:***

Step 1. Start mst.

```
host1# mst start
host1# mst status
```

The following MFT commands assume that the Mellanox Firmware Tools (MFT) package has been installed on the client machine.

Step 2. Obtain the Port GUID using the device name. The device name will be of the form: /dev/mst/mt<dev_id>_pci{_cr0|conf0}.

```
flint -d <MST_DEVICE_NAME> q
```

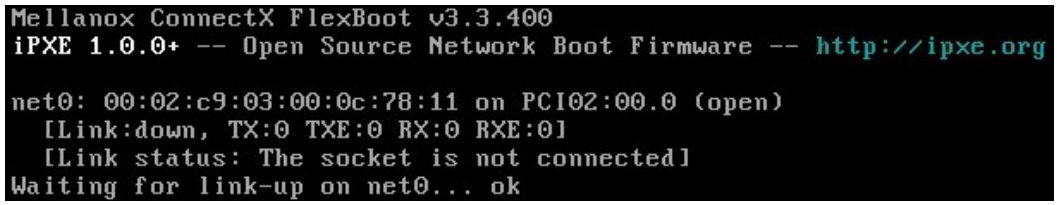
Example with ConnectX-2 QDR device:

```
Image type:      ConnectX
FW Version:      2.9.1000
Rom Info:        type=PXE  version=3.4.142 devid=26428 proto=VPI
Device ID:       26428
Description:     Node  Port1            Port2            Sys image
GUIDs:           0002c9030005cffa 0002c9030005cffb 0002c9030005cffc 0002c9030005cffd
MACs:                                 0002c905cffa     0002c905cffb
Board ID:         (MT_0DD0110009)
VSD:
PSID:            MT_0DD0110009
```

Assuming that FlexBoot is connected via Port 1, then the Port GUID is 00:02:c9:03:00:00:10:39.

Extracting the Port GUID – Method II

An alternative method for obtaining the port GUID involves booting the client machine via Flex-Boot. This requires having a Subnet Manager running on one of the machines in the InfiniBand subnet. The 8 bytes can be captured from the boot session as shown in the figure below.



Placing Client Identifiers in /etc/dhcpd.conf

The following is an excerpt of a `/etc/dhcpd.conf` example file showing the format of representing a client machine for the DHCP server.

```
host host1 {
   next-server 11.4.3.7;
   filename "pxelinux.0";
   fixed-address 11.4.3.130;
```

```
    option dhcp-client-identifier =
        ff:00:00:00:00:00:02:00:00:02:c9:00:00:02:c9:03:00:00:10:39;
}
```

A.5 Subnet Manager – OpenSM

This section applies to ports configured as InfiniBand only.

FlexBoot requires a Subnet Manager to be running on one of the machines in the IB network. OpenSM is part of the *Mellanox OFED for Linux* software package and can be used to accomplish this. Note that OpenSM may be run on the same host running the DHCP server but it is not mandatory. For details on OpenSM, see "OpenSM – Subnet Manager" on page 137.

To use OpenSM caching for large InfiniBand clusters (> 100 nodes), it is recommended to use the OpenSM options described in Section 8.2.1, "opensm Syntax", on page 137.

A.6 BIOS Configuration

The expansion ROM image presents itself to the BIOS as a boot device. As a result, the BIOS will add to the list of boot devices "MLNX FlexBoot <ver>" for a ConnectX device. The priority of this list can be modified through BIOS setup.

A.7 Operation

A.7.1 Prerequisites

- Make sure that your client is connected to the server(s)
- The FlexBoot image is already programmed on the adapter card – see Section A.3
- For InfiniBand ports only: Start the Subnet Manager as described in Section A.5
- The DHCP server should be configured and started (see Section 4.3.3.1, "IPoIB Configuration Based on DHCP", on page 57
- Configure and start at least one of the services iSCSI Target (see Section A.9) and/or TFTP (see Section A.6)

A.7.2 Starting Boot

Boot the client machine and enter BIOS setup to configure “MLNX FlexBoot” to be the first on the boot device priority list – see Section A.6.

On dual-port network adapters, the client first attempts to boot from Port 1. If this fails, it switches to boot from Port 2. Note also that the driver waits up to 90 seconds for each port to come up.

If MLNX FlexBoot/iPXE was selected through BIOS setup, the client will boot from FlexBoot. The client will display FlexBoot attributes, sense the port protocol – Ethernet or InfiniBand. In case of an InfiniBand port, the client will also wait for port configuration by the Subnet Manager.

In case sensing the port protocol fails, the port will be configured as an InfiniBand port.

For ConnectX:

```
Mellanox ConnectX FlexBoot v3.3.400
iPXE 1.0.0+ -- Open Source Network Boot Firmware -- http://ipxe.org

net0: 00:02:c9:03:00:0c:78:11 on PCI02:00.0 (open)
  [Link:down, TX:0 TXE:0 RX:0 RXE:0]
  [Link status: The socket is not connected]
Waiting for link-up on net0... ok
```

After configuring the IB/ETH port, the client attempts connecting to the DHCP server to obtain an IP address and the source location of the kernel/OS to boot from.

For ConnectX (InfiniBand):

```
Mellanox ConnectX FlexBoot v3.3.400
iPXE 1.0.0+ -- Open Source Network Boot Firmware -- http://ipxe.org

net0: 00:02:c9:03:00:0c:78:11 on PCI02:00.0 (open)
  [Link:down, TX:0 TXE:0 RX:0 RXE:0]
  [Link status: The socket is not connected]
Waiting for link-up on net0... ok
DHCP (net0 02:02:c9:0c:78:11).... ok
net0: 11.3.12.2/255.255.255.0
Next server: 11.3.12.121
Filename: pxeilinux.0
Root path: /tftpboot/
tftp://11.3.12.121/pxeilinux.0...
```

Next, FlexBoot attempts to boot as directed by the DHCP server.

A.8 Diskless Machines

Mellanox FlexBoot supports booting diskless machines. To enable using an IB/ETH driver, the `initrd` image must include a device driver module and be configured to load that driver.

This can be achieved by adding the device driver module into the `initrd` image and loading it.

The **'initrd'** image of some Linux distributions such as SuSE Linux Enterprise Server and Red Hat Enterprise Linux, cannot be edited prior or during the installation process.

If you need to install Linux distributions over Flexboot, please replace your **'initrd'** images with the images found at:

www.mellanox.com > Products > Adapter IB/VPI SW > FlexBoot (Download Tab).

A.8.1 Case I: InfiniBand Ports

The IB driver requires loading the following modules in the specified order (see Section A.8.1.1 for an example):

- ib_addr.ko
- ib_core.ko
- ib_mad.ko
- ib_sa.ko
- ib_cm.ko
- ib_uverbs.ko
- ib_ucm.ko
- ib_umad.ko
- iw_cm.ko
- rdma_cm.ko
- rdma_ucm.ko
- mlx4_core.ko
- mlx4_ib.ko
- ib_mthca.ko
- ipoib_helper.ko – this module is *not* required for all OS kernels. Please check the release notes.
- ib_ipoib.ko

A.8.1.1 Example: Adding an IB Driver to initrd (Linux)

Prerequisites

1. The FlexBoot image is already programmed on the HCA card.
2. The DHCP server is installed and configured as described in Section 4.3.3.1, “IPoIB Configuration Based on DHCP”, and is connected to the client machine.
3. An `initrd` file.

4. To add an IB driver into `initrd`, you need to copy the IB modules to the diskless image. Your machine needs to be pre-installed with a *Mellanox OFED for Linux* ISO image that is appropriate for the kernel version the diskless image will run.

Adding the IB Driver to the initrd File

The following procedure modifies critical files used in the boot procedure. It must be executed by users with expertise in the boot process. Improper application of this procedure may prevent the diskless machine from booting.

Step 1. Back up your current `initrd` file.

Step 2. Make a new working directory and change to it.

```
host1$ mkdir /tmp/initrd_ib
host1$ cd /tmp/initrd_ib
```

Step 3. Normally, the `initrd` image is zipped. Extract it using the following command:

```
host1$ gzip -dc <initrd image> | cpio -id
```

The initrd files should now be found under /tmp/initrd_ib

Step 4. Create a directory for the InfiniBand modules and copy them.

```
host1$ mkdir -p /tmp/initrd_ib/lib/modules/ib
host1$ cd /lib/modules/`uname -r`/updates/kernel/drivers
host1$ cp infiniband/core/ib_addr.ko /tmp/initrd_ib/lib/modules/ib
host1$ cp infiniband/core/ib_core.ko /tmp/initrd_ib/lib/modules/ib
host1$ cp infiniband/core/ib_mad.ko /tmp/initrd_ib/lib/modules/ib
host1$ cp infiniband/core/ib_sa.ko /tmp/initrd_ib/lib/modules/ib
host1$ cp infiniband/core/ib_cm.ko /tmp/initrd_ib/lib/modules/ib
host1$ cp infiniband/core/ib_uverbs.ko /tmp/initrd_ib/lib/modules/ib
host1$ cp infiniband/core/ib_ucm.ko /tmp/initrd_ib/lib/modules/ib
host1$ cp infiniband/core/ib_umad.ko /tmp/initrd_ib/lib/modules/ib
host1$ cp infiniband/core/iw_cm.ko /tmp/initrd_ib/lib/modules/ib
host1$ cp infiniband/core/rdma_cm.ko /tmp/initrd_ib/lib/modules/ib
host1$ cp infiniband/core/rdma_ucm.ko /tmp/initrd_ib/lib/modules/ib
host1$ cp net/mlx4/mlx4_core.ko /tmp/initrd_ib/lib/modules/ib
host1$ cp infiniband/hw/mlx4/mlx4_ib.ko /tmp/initrd_ib/lib/modules/ib
host1$ cp infiniband/hw/mthca/ib_mthca.ko /tmp/initrd_ib/lib/modules/ib
host1$ cp infiniband/ulp/ipoib/ipoib_helper.ko /tmp/initrd_ib/lib/modules/ib
host1$ cp infiniband/ulp/ipoib/ib_ipoib.ko /tmp/initrd_ib/lib/modules/ib
```

Step 5. IB requires loading an IPv6 module. If you do not have it in your `initrd`, please add it using the following command:

```
host1$ cp /lib/modules/`uname -r`/kernel/net/ipv6/ipv6.ko \
/tmp/initrd_ib/lib/modules
```

Step 6. To load the modules, you need the `insmod` executable. If you do not have it in your `initrd`, please add it using the following command:

```
host1$ cp /sbin/insmod /tmp/initrd_ib/sbin/
```

Step 7. If you plan to give your IB device a static IP address, then copy `ifconfig`. Otherwise, skip this step.

```
host1$ cp /sbin/ifconfig /tmp/initrd_ib/sbin
```

Step 8. If you plan to obtain an IP address for the IB device through DHCP, then you need to copy the DHCP client which was compiled specifically to support IB;
Otherwise, skip this step.
To continue with this step, DHCP client v3.1.3 needs to be already installed on the machine you are working with.
Copy the DHCP client v3.1.3 file and all the relevant files as described below.

```
host1# cp <path to DHCP client v3.1.3>/dhclient /tmp/initrd_ib/sbin
host1# cp <path to DHCP client v3.1.3>/dhclient-script /tmp/initrd_ib/sbin
host1# mkdir -p /tmp/initrd_ib/var/state/dhcp
host1# touch /tmp/initrd_ib/var/state/dhcp/dhclient.leases
host1# cp /bin/uname /tmp/initrd_ib/bin
host1# cp /usr/bin/expr /tmp/initrd_ib/bin
host1# cp /sbin/ifconfig /tmp/initrd_ib/bin
host1# cp /bin/hostname /tmp/initrd_ib/bin
```

Step 9. Create a configuration file for the DHCP client (as described in Section 4.3.3.1) and place it under `/tmp/initrd_ib/sbin`. The following is an example of such a file (called `dclient.conf`):

dhclient.conf:

```
# The value indicates a hexadecimal number
# For a ConnectX device interface "ib0" {send dhcp-client-identifier
ff:00:00:00:00:00:02:00:00:02:c9:00:00:02:c9:03:00:00:10:39;
}
```

Step 10. Now you can add the commands for loading the copied modules into the file `init`. Edit the file `/tmp/initrd_ib/init` and add the following lines at the point you wish the IB driver to be loaded.

The order of the following commands (for loading modules) is critical.

```
echo "loading ipv6"
/sbin/insmod /lib/modules/ipv6.ko
echo "loading IB driver"
/sbin/insmod /lib/modules/ib/ib_addr.ko
/sbin/insmod /lib/modules/ib/ib_core.ko
/sbin/insmod /lib/modules/ib/ib_mad.ko
/sbin/insmod /lib/modules/ib/ib_sa.ko
/sbin/insmod /lib/modules/ib/ib_cm.ko
/sbin/insmod /lib/modules/ib/ib_uverbs.ko
/sbin/insmod /lib/modules/ib/ib_ucm.ko
/sbin/insmod /lib/modules/ib/ib_umad.ko
/sbin/insmod /lib/modules/ib/iw_cm.ko
```

```
/sbin/insmod /lib/modules/ib/rdma_cm.ko
/sbin/insmod /lib/modules/ib/rdma_ucm.ko
/sbin/insmod /lib/modules/ib/mlx4_core.ko
/sbin/insmod /lib/modules/ib/mlx4_ib.ko
/sbin/insmod /lib/modules/ib/ib_mthca.ko
```

The following command (loading ipoib_helper.ko) is *not* required for all OS kernels. Please check the release notes.

```
/sbin/insmod /lib/modules/ib/ipoib_helper.ko
/sbin/insmod /lib/modules/ib/ib_ipoib.ko
```

Step 11. In case of interoperability issues between iSCSI and Large Receive Offload (LRO), change the last command above as follows to disable LRO:

```
/sbin/insmod /lib/modules/ib/ib_ipoib.ko lro=0
```

Step 12. Now you can assign an IP address to your IB device by adding a call to `ifconfig` or to the DHCP client in the `init` file after loading the modules. If you wish to use the DHCP client, then you need to add a call to the DHCP client in the `init` file after loading the IB modules. For example:

```
/sbin/dhclient -cf /sbin/dhclient.conf ib1
```

Step 13. Save the `init` file.

Step 14. Close `initrd`.

```
host1$ cd /tmp/initrd_ib
host1$ find ./ | cpio -H newc -o > /tmp/new_initrd_ib.img
host1$ gzip /tmp/new_init_ib.img
```

Step 15. At this stage, the modified `initrd` (including the IB driver) is ready and located at `/tmp/new_init_ib.img.gz`. Copy it to the original `initrd` location and rename it properly.

A.8.2 Case II: Ethernet Ports

The Ethernet driver requires loading the following modules in the specified order – see the example below:

- mlx4_core.ko
- mlx4_en.ko

A.8.2.1 Example: Adding an Ethernet Driver to initrd (Linux)

Prerequisites

1. The FlexBoot image is already programmed on the adapter card.
2. The DHCP server is installed and configured as described in Section 4.3.3.1 on page 57, and connected to the client machine.
3. An `initrd` file.

4. To add an Ethernet driver into `initrd`, you need to copy the Ethernet modules to the diskless image. Your machine needs to be pre-installed with a *MLNX_EN Linux Driver* that is appropriate for the kernel version the diskless image will run.

Adding the Ethernet Driver to the initrd File

The following procedure modifies critical files used in the boot procedure. It must be executed by users with expertise in the boot process. Improper application of this procedure may prevent the diskless machine from booting.

Step 1. Back up your current `initrd` file.

Step 2. Make a new working directory and change to it.

```
host1$ mkdir /tmp/initrd_en
host1$ cd /tmp/initrd_en
```

Step 3. Normally, the `initrd` image is zipped. Extract it using the following command:

```
host1$ gzip -dc <initrd image> | cpio -id
```

The initrd files should now be found under /tmp/initrd_en

Step 4. Create a directory for the ConnectX EN modules and copy them.

```
host1$ mkdir -p /tmp/initrd_en/lib/modules/mlnx_en
host1$ cd /lib/modules/`uname -r`/updates/kernel/drivers
host1$ cp net/mlx4/mlx4_core.ko /tmp/initrd_en/lib/modules/mlnx_en
host1$ cp net/mlx4/mlx4_en.ko /tmp/initrd_en/lib/modules/mlnx_en
```

Step 5. To load the modules, you need the `insmod` executable. If you do not have it in your `initrd`, please add it using the following command:

```
host1$ cp /sbin/insmod /tmp/initrd_en/sbin/
```

Step 6. If you plan to give your Ethernet device a static IP address, then copy `ifconfig`. Otherwise, skip this step.

```
host1$ cp /sbin/ifconfig /tmp/initrd_en/sbin
```

Step 7. Now you can add the commands for loading the copied modules into the file `init`. Edit the file `/tmp/initrd_en/init` and add the following lines at the point you wish the Ethernet driver to be loaded.

The order of the following commands (for loading modules) is critical.

```
echo "loading Mellanox ConnectX EN driver"
/sbin/insmod lib/modules/mlnx_en/mlx4_core.ko
/sbin/insmod lib/modules/mlnx_en/mlx4_en.ko
```

Step 8. Now you can assign a static or dynamic IP address to your Mellanox ConnectX EN network interface.

Step 9. Save the `init` file.

Step 10. Close `initrd`.

```
host1$ cd /tmp/initrd_en
host1$ find ./ | cpio -H newc -o > /tmp/new_initrd_en.img
host1$ gzip /tmp/new_init_en.img
```

At this stage, the modified `initrd` (including the Ethernet driver) is ready and located at `/tmp/new_init_ib.img.gz`. Copy it to the original `initrd` location and rename it properly.

A.9 iSCSI Boot

Mellanox FlexBoot enables an iSCSI-boot of an OS located on a remote iSCSI Target. It has a built-in iSCSI Initiator which can connect to the remote iSCSI Target and load from it the kernel and `initrd` (Linux). There are two instances of connection to the remote iSCSI Target: the first is for getting the kernel and `initrd` via FlexBoot, and the second is for loading other parts of the OS via `initrd`.

If you choose to continue loading the OS (after boot) through the HCA device driver, please verify that the initrd image includes the HCA driver as described in Section A.8.

A.9.1 Configuring an iSCSI Target in Linux Environment

Prerequisites

Step 1. Make sure that an iSCSI Target is installed on your server side.

You can download and install an iSCSI Target from the following location: http://sourceforge.net/projects/iscsitarget/files/iscsitarget/

Step 2. Dedicate a partition on your iSCSI Target on which you will later install the operating system

Step 3. Configure your iSCSI Target to work with the partition you dedicated. If, for example, you choose partition /dev/sda5, then edit the iSCSI Target configuration file `/etc/ietd.conf` to include the following line under the iSCSI Target iqn line:

```
Lun 0 Path=/dev/sda5,Type=fileio
```

Example of an iSCSI Target iqn line:

```
Target iqn.2007-08.7.3.4.10:iscsiboot
```

Step 4. Start your iSCSI Target.

Example:

```
host1# /etc/init.d/iscsitarget start
```

Configuring the DHCP Server to Boot From an iSCSI Target

Configure DHCP as described in Section 4.3.3.1, “IPoIB Configuration Based on DHCP”.

Edit your DHCP configuration file (`/etc/dhcpd.conf`) and add the following lines for the machine(s) you wish to boot from the iSCSI target:

```
Filename "";
option root-path "iscsi:iscsi_target_ip::::iscsi_target_iqn";
```

The following is an example for configuring an IB/ETH device to boot from an iSCSI target:

```
host  host1{
filename "";
```

```
# For a ConnectX device with ports configured as InfiniBand, comment out# the following
line
# option dhcp-client-identifier =
ff:00:00:00:00:00:02:00:00:02:c9:00:00:02:c9:03:00:00:10:39;

# For a ConnectX device with ports configured as Ethernet, comment out# the following
line
# hardware ethernet 00:02:c9:00:00:bb;

}
```

Appendix B: SRP Target Driver

The SRP Target driver is designed to work directly on top of OpenFabrics OFED software stacks (http://www.openfabrics.org) or InfiniBand drivers in Linux kernel tree (kernel.org). It also interfaces with Generic SCSI target mid-level driver - SCST (http://scst.sourceforge.net).

By interfacing with an SCST driver, it is possible to work with and support a lot of IO modes on real or virtual devices in the back end.

1. scst_vdisk – fileio and blockio modes. This allows turning software raid volumes, LVM volumes, IDE disks, block devices and normal files into SRP luns
2. NULLIO mode allows measuring the performance without sending IOs to *real* devices

B.1 Prerequisites and Installation

1. SRP targer is part of the OpenFabrics OFED software stacks. Use the latest OFED distribution package to install SRP target.

On distribution default kernels you can run scst_vdisk blockio mode to obtain good performance.

2. Download and install the SCST driver. The supported version is 1.0.1.1.
   a. Download `scst-1.0.1.1.tar.gz` from http://scst.sourceforge.net/downloads.html
   b. Untar `scst-1.0.1.1`

```
$ tar zxvf scst-1.0.1.1.tar.gz
$ cd scst-1.0.1.1
```

   c. Install `scst-1.0.1.1 as follows:`

```
$ make && make install
```

B.2 How-to Run

A. On an SRP Target machine:

1. Please refer to SCST's README for loading scst driver and its dev_handlers drivers (scst_vdisk block or file IO mode, nullio, ...)

Regardless of the mode, you always need to have lun 0 in any group's device list. Then you can have any lun number following lun 0 (it is not required to have the lun numbers in ascending order except that the first lun must always be 0).

Setting SRPT_LOAD=yes in /etc/infiniband/openib.conf is not enough as it only loads the `ib_srpt` module but does *not* load `scst` not its dev_handlers.

The scst_disk module (pass-thru mode) of SCST is not supported by Mellanox OFED.

Example 1: Working with VDISK BLOCKIO mode

(Using the md0 device, sda, and cciss/c1d0)

a. modprobe scst
b. modprobe scst_vdisk
c. echo "open vdisk0 /dev/md0 BLOCKIO" > /proc/scsi_tgt/vdisk/vdisk
d. echo "open vdisk1 /dev/sda BLOCKIO" > /proc/scsi_tgt/vdisk/vdisk
e. echo "open vdisk2 /dev/cciss/c1d0 BLOCKIO" > /proc/scsi_tgt/vdisk/vdisk
f. echo "add vdisk0 0" >/proc/scsi_tgt/groups/Default/devices
g. echo "add vdisk1 1" >/proc/scsi_tgt/groups/Default/devices
h. echo "add vdisk2 2" >/proc/scsi_tgt/groups/Default/devices

Example 2: working with scst_vdisk FILEIO mode

(Using md0 device and file 10G-file)

a. modprobe scst
b. modprobe scst_vdisk
c. echo "open vdisk0 /dev/md0" > /proc/scsi_tgt/vdisk/vdisk
d. echo "open vdisk1 /10G-file" > /proc/scsi_tgt/vdisk/vdisk
e. echo "add vdisk0 0" >/proc/scsi_tgt/groups/Default/devices
f. echo "add vdisk1 1" >/proc/scsi_tgt/groups/Default/devices

2. Run:

```
For all distributions except SLES 11:  > modprobe ib_srpt
For SLES 11:   > modprobe -f ib_srpt
```

For SLES 11, please ignore the following error messages in /var/log/messages when loading ib_srpt to SLES 11 distribution's kernel:

```
...
ib_srpt: no symbol version for scst_unregister
ib_srpt: Unknown symbol scst_unregister
ib_srpt: no symbol version for scst_register
ib_srpt: Unknown symbol scst_register
ib_srpt: no symbol version for scst_unregister_target_template
ib_srpt: Unknown symbol scst_unregister_target_template
...
```

B. On Initiator Machines

On Initiator machines, manually perform the following steps:

1. Run:

```
modprobe ib_srp
```

2. Run:

```
ibsrpdm -c -d /dev/infiniband/umadX
```

(to discover a new SRP target)

```
umad0: port 1 of the first HCA
umad1: port 2 of the first HCA
umad2: port 1 of the second HCA
```

3. echo {new target info} > /sys/class/infiniband_srp/srp-mthca0-1/add_target
4. fdisk -l (will show the newly discovered scsi disks)

Example:

Assume that you use port 1 of first HCA in the system, i.e.: mthca0

```
[root@lab104 ~]# ibsrpdm -c -d /dev/infiniband/umad0
id_ext=0002c90200226cf4,ioc_guid=0002c90200226cf4,
dgid=fe8000000000000002c90200226cf5,pkey=ffff,service_id=0002c90200226cf4
[root@lab104 ~]# echo id_ext=0002c90200226cf4,ioc_guid=0002c90200226cf4,
dgid=fe8000000000000002c90200226cf5,pkey=ffff,service_id=0002c90200226cf4 > /sys/
class/infiniband_srp/srp-mthca0-1/add_target
```

OR

- You can edit /etc/infiniband/openib.conf to load the SRP driver and SRP High Availability (HA) daemon automatically, that is: set “SRP_LOAD=yes” and “SRPHA_ENABLE=yes”
- To set up and use the HA feature, you need the dm-multipath driver and multipath tool
- Please refer to OFED-1.x SRP's user manual for more detailed instructions on how-to enable/use the HA feature

The following is an example of an SRP Target setup file:

```
********************** srpt.sh *******************************
#!/bin/sh
modprobe scst scst_threads=1
modprobe scst_vdisk scst_vdisk_ID=100

echo "open vdisk0 /dev/cciss/c1d0 BLOCKIO" > /proc/scsi_tgt/vdisk/vdisk
echo "open vdisk1 /dev/sdb BLOCKIO" > /proc/scsi_tgt/vdisk/vdisk
echo "open vdisk2 /dev/sdc BLOCKIO" > /proc/scsi_tgt/vdisk/vdisk
echo "open vdisk3 /dev/sdd BLOCKIO" > /proc/scsi_tgt/vdisk/vdisk
echo "add vdisk0 0" > /proc/scsi_tgt/groups/Default/devices
echo "add vdisk1 1" > /proc/scsi_tgt/groups/Default/devices
echo "add vdisk2 2" > /proc/scsi_tgt/groups/Default/devices
echo "add vdisk3 3" > /proc/scsi_tgt/groups/Default/devices

modprobe ib_srpt
```

```
echo "add "mgmt"" > /proc/scsi_tgt/trace_level
echo "add "mgmt_dbg"" > /proc/scsi_tgt/trace_level
echo "add "out_of_mem"" > /proc/scsi_tgt/trace_level
********************** End srpt.sh ****************************
```

B.3 How-to Unload/Shutdown

1. Unload `ib_srpt`

```
$ modprobe -r ib_srpt
```

2. Unload `scst` and its dev_handlers first

```
$ modprobe -r scst_vdisk scst
```

3. Unload `ofed`

```
$ /etc/rc.d/openibd stop
```

Appendix C: mlx4 Module Parameters

In order to set **mlx4** parameters, add the following line(s) to **/etc/modprobe.conf**:

```
options mlx4_core parameter=<value>
```

and/or

```
options mlx4_ib   parameter=<value>
```

and/or

```
options mlx4_en   parameter=<value>
```

The following sections list the available mlx4 parameters.

C.1 mlx4_ib Parameters

sm_guid_assign:	Enable SM alias_GUID assignment if sm_guid_assign > 0 (Default: 1) (int)
dev_assign_str[1]:	Map device function numbers to IB device numbers (e.g.'0000:04:00.0-0,002b:1c:0b.a-1,...'). Hexadecimal digits for the device function (e.g. 002b:1c:0b.a) and decimal for IB device numbers (e.g. 1). Max supported devices - 32 (string)

1. In the current version, this parameter is using decimal number to describe the InfiniBand device and not hexadecimal number as it was in previous versions in order to uniform the mapping of device function numbers to InfiniBand device numbers as defined for other module parameters (e.g. num_vfs and probe_vf).
   For example to map mlx4_15 to device function number 04:00.0 in the current version we use "options mlx4_ib dev_assign_str=04:00.0-15" as opposed to the previous version in which we used "options mlx4_ib dev_assign_str=04:00.0-f"

C.2 mlx4_core Parameters

set_4k_mtu:	(Obsolete) attempt to set 4K MTU to all ConnectX ports (int)
debug_level:	Enable debug tracing if > 0 (int)
msi_x:	0 - don't use MSI-X, 1 - use MSI-X, >1 - limit number of MSI-X irqs to msi_x (non-SRIOV only) (int)
enable_sys_tune:	Tune the cpu's for better performance (default 0) (int)
block_loopback:	Block multicast loopback packets if > 0 (default: 1) (int)
num_vfs:	Either single value (e.g. '5') to define uniform num_vfs value for all devices functions or a string to map device function numbers to their num_vfs values (e.g. '0000:04:00.0-5,002b:1c:0b.a-15'). Hexadecimal digits for the device function (e.g. 002b:1c:0b.a) and decimal for num_vfs value (e.g. 15). (string)
probe_vf:	Either single value (e.g. '3') to define uniform number of VFs to probe by the pf driver for all devices functions or a string to map device function numbers to their probe_vf values (e.g. '0000:04:00.0-3,002b:1c:0b.a-13'). Hexadecimal digits for the device function (e.g. 002b:1c:0b.a) and decimal for probe_vf value (e.g. 13). (string)

log_num_mgm_entry_size:	log mgm size, that defines the num of qp per mcg, for example: 10 gives 248.range: 7 <= log_num_mgm_entry_size <= 12. To activate device managed flow steering when available, set to -1 (int)
high_rate_steer:	Enable steering mode for higher packet rate (default off) (int)
fast_drop:	Enable fast packet drop when no recieve WQEs are posted (int)
enable_64b_cqe_eqe:	Enable 64 byte CQEs/EQEs when the FW supports this if non-zero (default: 1) (int)
log_num_mac:	Log2 max number of MACs per ETH port (1-7) (int)
log_num_vlan:	(Obsolete) Log2 max number of VLANs per ETH port (0-7) (int)
log_mtts_per_seg:	Log2 number of MTT entries per segment (0-7) (default: 0) (int)
port_type_array:	Either pair of values (e.g. '1,2') to define uniform port1/ port2 types configuration for all devices functions or a string to map device function numbers to their pair of port types values (e.g. '0000:04:00.0-1;2,002b:1c:0b.a-1;1'). Valid port types: 1-ib, 2-eth, 3-auto, 4-N/A If only a single port is available, use the N/A port type for port2 (e.g '1,4').
log_num_qp:	log maximum number of QPs per HCA (default: 19) (int)
log_num_srq:	log maximum number of SRQs per HCA (default: 16) (int)
log_rdmarc_per_qp:	log number of RDMARC buffers per QP (default: 4) (int)
log_num_cq:	log maximum number of CQs per HCA (default: 16) (int)
log_num_mcg:	log maximum number of multicast groups per HCA (default: 13) (int)
log_num_mpt:	log maximum number of memory protection table entries per HCA (default: 19) (int)
log_num_mtt:	log maximum number of memory translation table segments per HCA (default: max(20, 2*MTTs for register all of the host memory limited to 30)) (int)
enable_qos:	Enable Quality of Service support in the HCA (default: off) (bool)
internal_err_reset:	Reset device on internal errors if non-zero (default 1, in SRIOV mode default is 0) (int)

C.3 mlx4_en Parameters

inline_thold:	Threshold for using inline data (int) Default and max value is 104 bytes. Saves PCI read operation transaction, packet less then threshold size will be copied to hw buffer directly.
udp_rss:	Enable RSS for incoming UDP traffic (uint) On by default. Once disabled no RSS for incoming UDP traffic will be done.
pfctx:	Priority based Flow Control policy on TX[7:0]. Per priority bit mask (uint)
pfcrx:	Priority based Flow Control policy on RX[7:0]. Per priority bit mask (uint)

Appendix D: mlx5 Module Parameters

The mlx5_ib module supports a single parameter used to select the profile which defines the number of resources supported. The parameter name for selecting the profile is `prof_sel`.

The supported values for profiles are:

- 0 - for medium resources, medium performance
- 1 - for low resources
- 2 - for high performance (int) (default)

Appendix E: Lustre Compilation over MLNX_OFED

This procedure applies to RHEL/SLES OSs only.

- ***To compile Lustre version 2.3.65 and higher:***

```
$ ./configure --with-o2ib=/usr/src/ofa_kernel/default/
$ make rpms
```

- ***To compile older Lustre versions:***

```
$ EXTRA_LNET_INCLUDE="-I/usr/src/ofa_kernel/default/include/ -include /usr/src/
ofa_kernel/default/include/linux/compat-2.6.h" ./configure --with-o2ib=/usr/src/
ofa_kernel/default/
$ EXTRA_LNET_INCLUDE="-I/usr/src/ofa_kernel/default/include/ -include /usr/src/
ofa_kernel/default/include/linux/compat-2.6.h" make rpms
```

For Lustre 2.1.3, due to a duplicate definition of INVALID_UID macro, the following patch must be applied:

```
--- lustre-2.1.3/lustre/include/lustre_cfg.h 2012-09-17 14:26:46.000000000 +0200
+++ lustre-2.1.3/lustre/include/lustre_cfg.h.new 2013-09-07 10:45:07.121772824 +0200
@@ -288,7 +288,9 @@
#include <lustre/lustre_user.h>
+#ifndef INVALID_UID
#define INVALID_UID (-1)
+#endif
```