NVIDIA MLNX_EN Documentation

v23.10-7.1.8.0 LTS

Table of Contents

1 Overview

> This is a long-term support (LTS) release. LTS is the practice of maintaining a software product for an extended period of time (up to three years) to help increase product stability. LTS releases include bug fixes and security patches.

NVIDIA offers a robust and full set of protocol software and driver for Linux with the ConnectX® EN family cards. Designed to provide a high performance support for Enhanced Ethernet with fabric consolidation over TCP/IP based LAN applications. The driver and software in conjunction with the industry's leading ConnectX family of cards achieve full line rate, full duplex of up to 400GbE performance per port.

Further information on this product can be found in the following MLNX_EN documents:

- (23.10-6-LTS6-2023Branch) Release Notes
- (23.10-6-LTS6-2023Branch) User Manual

1.1 Software Download

Please visit nvidia.com/en-us/networking → Products → Software → Ethernet Drivers → NVIDIA EN for Linux

1.2 Document Revision History

For the list of changes made to the User Manual, refer to (23.10-6-LTS6-2023Branch) User Manual Revision History.

For the list of changes made to the Release Notes, refer to Release Notes History.

2 Release Notes

2.1 Release Notes Update History

Version	Date	Description
23.10-7.1.8.0	May 2026	Initial release of this document version. This release introduces Bug Fixes in This Version.

As of MLNX_EN version 5.1-1.0.4.0, the following are no longer supported.

- ConnectX-3
- ConnectX-3 Pro
- Connect-IB
- RDMA experimental verbs libraries (mlnx_lib)

To utilize the above devices/libraries, refer to version 4.9 long-term support (LTS).

Release Notes contain the following sections:

- General Support
- Changes and New Features
- Bug Fixes in This Version
- Known Issues

2.2 Supported NIC Speeds

The Linux Driver operates across all NVIDIA network adapter solutions supporting the following uplinks to servers:

Uplink/Adapter Card	Driver Name	Uplink Speed
BlueField-2	mlx5	• InfiniBand: SDR, FDR, EDR, HDR • Ethernet: 1GbE, 10GbE, 25GbE, 40GbE, 50GbE, 100GbE
BlueField		• InfiniBand: SDR, QDR, FDR, FDR10, EDR • Ethernet: 1GbE, 10GbE, 25GbE, 40GbE, 50GbE, 100GbE
ConnectX-7		• InfiniBand: EDR, HDR100, HDR, NDR200, NDR • Ethernet: 1GbE, 10GbE, 25GbE, 40GbE, 50GbE, 100GbE, 200GbE, 400GbE
ConnectX-6 Lx		• Ethernet: 1GbE, 10GbE, 25GbE, 40GbE, 50GbE
ConnectX-6 Dx		• Ethernet: 10GbE, 25GbE, 40GbE, 50GbE, 100GbE, 200GbE
ConnectX-6		• InfiniBand: SDR, FDR, EDR, HDR • Ethernet: 10GbE, 25GbE, 40GbE, 50GbE, 100GbE, 200GbE
ConnectX-5/ConnectX-5 Ex		• InfiniBand: SDR, QDR, FDR, FDR10, EDR • Ethernet: 1GbE, 10GbE, 25GbE, 40GbE, 50GbE, 100GbE
ConnectX-4 Lx		• Ethernet: 1GbE, 10GbE, 25GbE, 40GbE, 50GbE

Uplink/Adapter Card	Driver Name	Uplink Speed
ConnectX-4		• InfiniBand: SDR, QDR, FDR, FDR10, EDR • Ethernet: 1GbE, 10GbE, 25GbE, 40GbE, 50GbE, 56GbE, 100Gb

Important Notes:

- 50GbE and 100GbE are speed that supports both NRZ and PAM4 modes in Force mode and Auto-Negotiation mode.
- 200GbE is a speed that supports PAM4 mode only.
- 56GbE is an NVIDIA proprietary link speed and can be achieved while connecting an NVIDIA adapter card to NVIDIA SX10XX switch series or when connecting an NVIDIA adapter card to another NVIDIA adapter card.

2.3 Embedded Components

Package	Revision	Licenses
clusterkit	1.11.442-1.2310055	BSD
dpcp	1.1.43-1.2310055	BSD-3-Clause
hcoll	4.8.3228-1.20250521.6f14f256	Proprietary
ibarr	0.1.3-1.2310055	(GPL-2.0 WITH Linux-syscall-note) OR BSD-2-Clause
ibdump	6.0.0-1.2310055	BSD2+GPL2
ibsim	0.12-1.2310055	GPLv2 or BSD
ibutils2	2.1.1-0.21501.MLNX20240610.g840ec16f.2310615	Mellanox Confidential and Proprietary
iser	23.10-OFED.23.10.6.1.5.1	GPLv2
isert	23.10-OFED.23.10.6.1.5.1	GPLv2
kernel-mft	4.26.1-35	Dual BSD/GPL
knem	1.1.4.90mlnx3-OFED.23.10.0.2.1.1	BSD and GPLv2
libvma	9.8.40-1	GPLv2 or BSD
libxlio	3.20.8-1	GPLv2 or BSD
mlnx-en	23.10-6.1.5.0.g2f2078a	GPLv2
mlnx-ethtool	6.4-1.2310055	GPL
mlnx-iproute2	6.4.0-1.2310055	GPL
mlnx-nfsrdma	23.10-OFED.23.10.6.1.5.1	GPLv2
mlnx-nvme	23.10-OFED.23.10.6.1.5.1	GPLv2
mlnx-ofa_kernel	23.10-OFED.23.10.6.1.5.1	GPLv2
mlnx-tools	23.10-0.2310409	GPLv2 or BSD

Package	Revision	Licenses
mlx-steering-dump	1.0.0-0.2310055	GPLv2
mpitests	3.2.21-8418f75.2310055	BSD
mstflint	4.16.1-2.2310055	GPL/BSD
multiperf	3.0-3.0.2310055	BSD 3-Clause, GPL v2 or later
ofed-docs	23.10-OFED.23.10.6.1.5	GPL/BSD
ofed-scripts	23.10-OFED.23.10.6.1.5	GPL/BSD
openmpi	4.1.7a1-1.2310055	BSD
opensm	5.17.2.MLNX20240610.dc7c2998-0.1.2310409	GPLv2 or BSD
openvswitch	2.17.8-1.2310409	ASL 2.0 and LGPLv2+ and SISSL
perftest	23.10.0-0.96.g9729d3d.2310615	BSD 3-Clause, GPL v2 or later
rdma-core	2307mlnx47-1.2310409	GPLv2 or BSD
rshim	2.0.19-0.gbf7f1f2	GPLv2
sharp	3.5.2.MLNX20240610.61bf97b7-1.2310615	Proprietary
sockperf	3.10-0.git5ebd327da983.2310055	BSD
srp	23.10-OFED.23.10.6.1.5.1	GPLv2
ucx	1.16.0-1.2310409	BSD
xpmem	2.7.3-1.2310055	GPLv2 and LGPLv2.1
xpmem-lib	2.7-0.2310055	LGPLv2.1

2.4 General Support

Supported Operating Systems

Operating System	Architecture	Default Kernel Version (Primary)/ Tested with Kernel Version (Community)	OS Support Model	ASAP^2 OVS-Kernel SR-IOV	ASAP^2 OVS-DPDK SR-IOV	NFS over RDMA	NVMe	GPUDirect Storage (GDS)	UCX - CUDA Version
Alma 8.5	x86_64	4.18.0-348.12.2.EL8_5.X86_64	Community	✗	✗	✗	✗	✗	✗
Anolis OS 3.2	x86_64	5.10.134-13.al8.x86_64	Community	✗	✗	✗	✗	✗	✗
Anolis OS 8.6	AArch64	5.10.134+	Primary	✗	✗	✗	✗	✗	✗
	x86_64	5.10.134+	Primary	✗	✗	✗	✗	✗	✗
BCLINUX2 1.10SP2	AArch64	4.19.90-2107.6.0.0098.oe1.bclinux.aarch64	Primary	✗	✗	✗	✓	✗	✗

	x86_64	4.19.90-2107.6.0.0100.oe1.bclinux.x86_64	Primary	✗	✗	✓	✓	✗	✗
CentOS Stream v8	AArch64	4.18.0-553.el8.aarch64	Community	✗	✗	✗	✗	✗	✗
	x86_64	4.18.0-553.el8.x86_64	Community	✗	✗	✗	✗	✗	✗
CentOS Stream v9	AArch64	5.14.0-419.el9.x86_64	Community	✗	✗	✗	✗	✗	✗
	x86_64	5.14.0-419.el9.aarch64	Community	✗	✗	✗	✗	✗	✗
CTYUNOS 2.0	AArch64	4.19.90-2102.2.0.0062.ctl2.aarch64	Primary	✗	✗	✗	✗	✗	✗
	x86_64	4.19.90-2102.2.0.0062.ctl2.x86_64	Primary	✗	✗	✗	✗	✗	✗
CTYUNOS 23.01	AArch64	5.10.0-136.12.0.86.ctl3.aarch64	Primary	✗	✗	✗	✗	✗	✗
	x86_64	5.10.0-136.12.0.86.ctl3.x86_64	Primary	✗	✗	✗	✗	✗	✗
Debian9.13	AArch64	4.9.0-13-arm64	Primary	✗	✗	✗	✗	✗	✗
	x86_64	4.9.0-13-amd64	Primary	✗	✗	✗	✗	✗	✗
Debian10.8	AArch64	4.19.0-14-arm64	Primary	✗	✗	✗	✗	✗	✗
Debian10.9	x86_64	4.19.0-14-amd64	Primary	✗	✗	✗	✗	✗	✗
	x86_64	4.19.0-16-amd64	Primary	✓	✗	✗	✓	✗	✗
Debian10.13	AArch64	4.19.0-21-arm64	Primary	✗	✗	✗	✗	✗	✗
	x86_64	4.19.0-21-amd64	Primary	✗	✗	✗	✗	✗	✗
Debian11.3	AArch64	5.10.0-13-arm64	Primary	✓	✗	✗	✓	✗	✗
	x86_64	5.10.0-13-amd64	Primary	✓	✗	✗	✓	✗	✗
Debian12	AArch64	6.1.0-10-arm64	Primary	✓	✗	✗	✓	✗	✗
	x86_64	6.1.0-10-amd64	Primary	✓	✗	✗	✓	✗	✗
Debian12.5	AArch64	6.1.0-18-arm64	Primary	✓	✗	✗	✓	✗	✗
	x86_64	6.1.0-18-amd64	Primary	✓	✗	✗	✓	✗	✗

EulerOS2.0sp9	AArch64	4.19.90-vhulk2006.2.0.h171.eulerosv2r9.aarch64	Community	✗	✗	✗	✗	✗	✗
	x86_64	4.18.0-147.5.1.0.h269.eulerosv2r9.x86_64	Community	✗	✗	✗	✗	✗	✗
EulerOS2.0sp10	AArch64	4.19.90-vhulk2110.1.0.h860.eulerosv2r10.aarch64	Community	✗	✗	✗	✗	✗	✗
	x86_64	4.18.0-147.5.2.4.h694.eulerosv2r10.x86_64	Community	✗	✗	✗	✗	✗	✗
EulerOS2.0sp11	AArch64	5.10.0-60.18.0.50.h323.eulerosv2r11.aarch64	Primary	✗	✗	✗	✗	✗	✗
	x86_64	5.10.0-60.18.0.50.h323.eulerosv2r11.x86_64	Primary	✗	✗	✗	✗	✗	✗
EulerOS2.0sp12	AArch64	5.10.0-136.12.0.86.h1032.eulerosv2r12.aarch64	Primary	✗	✗	✗	✗	✗	✗
	x86_64	5.10.0-136.12.0.86.h1032.eulerosv2r12.x86_64	Primary	✗	✗	✗	✗	✗	✗
KYLIN10SP2	AArch64	4.19.90-24.4.v2101.ky10.aarch64	Primary	✗	✗	✗	✗	✗	✗
	x86_64	4.19.90-24.4.v2101.ky10.x86_64	Primary	✓	✗	✗	✗	✗	✗
KYLIN10SP3	AArch64	4.19.90-52.15.v2207.ky10.aarch64	Primary	✓	✗	✗	✗	✗	✗
	x86_64	4.19.90-52.15.v2207.ky10.x86_64	Primary	✓	✗	✗	✗	✗	✗
Mariner 2.0	x86_64	5.15.118.1-1.cm2.x86_64	Community	✗	✗	✗	✗	✗	✗
Oracle Linux 7.9	x86_64	5.4.17-2011.6.2.el7uek.x86_64	Primary	✗	✗	✗	✗	✗	✗
Oracle Linux 8.4	x86_64	5.4.17-2102.201.3.el8uek.x86_64	Primary	✗	✗	✗	✗	✗	✗
Oracle Linux 8.6	x86_64	5.4.17-2136.307.3.1.el8uek.x86_64	Primary	✗	✗	✗	✗	✗	✗
Oracle Linux 8.7	x86_64	5.15.0-3.60.5.1.el8uek.x86_64	Primary	✗	✗	✗	✗	✗	✗
Oracle Linux 8.8	x86_64	5.15.0-101.103.2.1.el8uek.x86_64	Primary	✗	✗	✗	✗	✗	✗
Oracle Linux 9.0	x86_64	5.15.0-0.30.19.el9uek.x86_64	Primary	✗	✗	✗	✗	✗	✗

Oracle Linux 9.1	x86_64	5.15.0-3.60.5.1.el9uek.x86_64	Primary	✗	✗	✗	✗	✗	✗
Oracle Linux 9.2	x86_64	5.15.0-101.103.2.1.el9uek.x86_64	Primary	✗	✗	✗	✗	✗	✗
OpenSUSE 15.3	AArch64	-	Community	✗	✗	✗	✗	✗	✗
	x86_64	5.3.18-150300.59.43-DEFAULT	Community	✗	✗	✗	✗	✗	✗
OPENEULER R20.03SP1	AArch64	4.19.90-2012.4.0.0053.OE1.AARCH64	Community	✗	✗	✗	✗	✗	✗
	x86_64	4.19.90-2110.8.0.0119.OE1.X86_64	Community	✗	✗	✗	✗	✗	✗
OPENEULER R20.03SP3	AArch64	4.19.90-2112.8.0.0131.oe1.aarch64	Primary	✗	✗	✗	✗	✗	✗
	x86_64	4.19.90-2112.8.0.0131.oe1.x86_64	Primary	✗	✗	✗	✗	✗	✗
OPENEULER R22.03	AArch64	5.10.0-60.18.0.50.oe2203.aarch64	Primary	✗	✗	✗	✗	✗	✗
	x86_64	5.10.0-60.18.0.50.oe2203.x86_64	Primary	✗	✗	✗	✗	✗	✗
Photon OS 3.0	x86_64	4.19.225-3.ph3	Community	✗	✗	✗	✗	✗	✗
RHEL/ CentOS7.2	x86_64	3.10.0-327.el7.x86_64	Primary	✗	✗	✗	✓	✗	12.2
RHEL/ CentOS7.4	x86_64	3.10.0-693.el7.x86_64	Primary	✓	✓	✗	✓	✗	12.2
RHEL/ CentOS7.6	x86_64	3.10.0-957.el7.x86_64	Primary	✓	✓	✓	✓	✗	12.2
RHEL/ CentOS7.6 alternate	aarch64	4.14.0-115.el7a.aarch64	Community	✓	✓	✗	✗	✗	✗
RHEL/ CentOS7.7	x86_64	3.10.0-1062.el7.x86_64	Primary	✓	✓	✗	✗	✗	12.2
RHEL/ CentOS7.8	x86_64	3.10.0-1127.el7.x86_64	Primary	✓	✓	✓	✓	✗	12.2
RHEL/ CentOS7.9	x86_64	3.10.0-1160.el7.x86_64	Primary	✓	✓	✓	✓	✗	12.2
RHEL/ CentOS8.0	AArch64	4.18.0-80.el8.aarch64	Primary	✓	✓	✗	✗	✓	12.2
	x86_64	4.18.0-80.el8.x86_64	Primary	✓	✓	✗	✗	✓	12.2
RHEL/ CentOS8.1	AArch64	4.18.0-147.el8.aarch64	Primary	✓	✓	✗	✗	✓	12.2
	x86_64	4.18.0-147.el8.x86_64	Primary	✓	✓	✗	✗	✓	12.2

RHEL/ CentOS8.2	AArch64	4.18.0-193.el8.aarch64	Primary	✓	✓	✗	✓	✓	12.2
	x86_64	4.18.0-193.el8.x86_64	Primary	✓	✓	✓	✓	✓	12.2
RHEL/ CentOS8.3	AArch64	4.18.0-240.el8.aarch64	Primary	✓	✓	✗	✗	✓	12.2
	x86_64	4.18.0-240.el8.x86_64	Primary	✓	✓	✗	✗	✓	12.2
RHEL/ CentOS8.4	AArch64	4.18.0-305.el8.aarch64	Primary	✓	✓	✗	✓	✓	12.2
	x86_64	4.18.0-305.el8.x86_64	Primary	✓	✓	✓	✓	✓	12.2
RHEL/ CentOS/ Rocky8.5	AArch64	4.18.0-348.el8.aarch64	Primary	✓	✓	✗	✓	✓	12.2
	x86_64	4.18.0-348.el8.x86_64	Primary	✓	✓	✓	✓	✓	12.2
RHEL/ Rocky8.6	AArch64	AArch644.18.0-372.41.1.el8_6.aarch64	Primary	✓	✓	✗	✓	✓	12.2
	x86_64	4.18.0-372.41.1.el8_6.x86_64	Primary	✓	✓	✓	✓	✓	12.2
RHEL/ Rocky8.7	AArch64	4.18.0-425.14.1.el8_7.aarch64	Primary	✓	✗	✗	✓	✓	12.2
	x86_64	4.18.0-425.14.1.el8_7.x86_64	Primary	✓	✗	✓	✓	✓	12.2
RHEL/ Rocky8.8	AArch64	4.18.0-477.10.1.el8_8.aarch64	Primary	✓	✗	✗	✓	✓	12.2
	x86_64	4.18.0-477.10.1.el8_8.x86_64	Primary	✓	✗	✓	✓	✓	12.2
RHEL/ Rocky8.9	AArch64	4.18.0-513.5.1.el8_9.aarch64	Primary	✓	✗	✗	✓	✓	12.2
	x86_64	4.18.0-513.5.1.el8_9.x86_64	Primary	✓	✗	✓	✓	✓	12.2
RHEL/ Rocky8.10	AArch64	4.18.0-553.el8_10.aarch64	Primary	✓	✗	✗	✓	✓	12.2
	x86_64	4.18.0-553.el8_10.x86_64	Primary	✓	✗	✓	✓	✓	12.2
RHEL/ Rocky9.0	AArch64	5.14.0-70.46.1.el9_0.aarch64	Primary	✓	✗	✗	✓	✓	12.2
	x86_64	5.14.0-70.46.1.el9_0.x86_64	Primary	✓	✗	✓	✓	✓	12.2
RHEL/ Rocky9.1	AArch64	5.14.0-162.19.1.el9_1.aarch64	Primary	✓	✗	✗	✓	✓	12.2
	x86_64	5.14.0-162.19.1.el9_1.x86_64	Primary	✓	✗	✓	✓	✓	12.2
RHEL/ Rocky9.2	AArch64	5.14.0-284.11.1.el9_2.aarch64	Primary	✓	✗	✗	✓	✓	12.2

	x86_64	5.14.0-284.11.1.el9_2.x86_64	Primary	✓	✗	✓	✓	✓	12.2
RHEL/ Rocky9.3	AArch64	5.14.0-362.8.1.el9_3.aarch64	Primary	✓	✗	✗	✓	✓	12.2
	x86_64	5.14.0-362.8.1.el9_3.x86_64	Primary	✓	✗	✓	✓	✓	12.2
RHEL/ Rocky9.4	AArch64	5.14.0-427.13.1.el9_4.aarch64	Primary	✓	✗	✗	✓	✓	12.2
	x86_64	5.14.0-427.13.1.el9_4.x86_64	Primary	✓	✗	✓	✓	✓	12.2
SLES12.1SP2	x86_64	4.4.21-69-default	Community	✗	✗	✗	✗	✗	✗
SLES12SP3	x86_64	4.4.73-5-default	Community	✗	✗	✗	✗	✗	✗
SLES12SP4	AArch64	4.12.14-94.41-default	Community	✓	✗	✗	✓	✗	✗
	x86_64	4.12.14-94.41-default	Community	✓	✗	✓	✓	✗	✗
SLES12SP5	AArch64	4.12.14-120-default	Primary	✓	✗	✗	✓	✗	✗
	x86_64	4.12.14-120-default	Primary	✓	✗	✓	✓	✗	✗
SLES15SP2	AArch64	5.3.18-22-default	Primary	✓	✓	✗	✓	✗	✗
	x86_64	5.3.18-22-default	Primary	✓	✓	✓	✓	✗	✗
SLES15SP3	AArch64	5.3.18-57-default	Primary	✓	✗	✗	✓	✗	✗
	x86_64	5.3.18-57-default	Primary	✓	✗	✓	✓	✗	✗
SLES15SP4	AArch64	5.14.21-150400.22-default	Primary	✓	✗	✗	✓	✗	✗
	x86_64	5.14.21-150400.22-default	Primary	✓	✗	✓	✓	✗	✗
SLES15SP5	AArch64	5.14.21-150500.53-default	Primary	✓	✗	✗	✓	✗	✗
	x86_64	5.14.21-150500.53-default	Primary	✓	✗	✓	✓	✗	✗
SLES15SP6	x86_64	6.4.0-150600.21-default	Primary	✓	✗	✓	✓	✗	✗
Ubuntu16.04	x86_64	4.4.0-21-generic	Community	✗	✗	✗	✗	✗	✗
Ubuntu18.04	AArch64	4.15.0-20-generic	Primary	✓	✓	✗	✓	✓	11.6
	x86_64	4.15.0-20-generic	Primary	✓	✓	✓	✓	✓	11.6

Ubuntu20.04	AArch64	5.4.0-26-generic	Primary	✓	✓	✗	✓	✓	12.2
	x86_64	5.4.0-26-generic	Primary	✓	✓	✓	✓	✓	12.2
Ubuntu22.04	AArch64	5.15.0-25-generic	Primary	✓	✓	✓	✓	✓	12.2
	x86_64	5.15.0-25-generic	Primary	✓	✓	✓	✓	✓	12.2
Ubuntu23.04	x86_64	6.2.0-20-generic	Primary	✓	✗	✓	✓	✗	✗
Ubuntu23.10	x86_64	6.5.0-5-generic	Primary	✓	✗	✗	✗	✗	✗
UOS20.1020	AArch64	4.19.90-2109.1.0.0108.up2.uel20.aarch64	Primary	✗	✗	✗	✗	✗	✗
	x86_64	4.19.90-2109.1.0.0108.up2.uel20.x86_64	Primary	✗	✗	✗	✗	✗	✗
UOS20.1040	AArch64	4.19.0-arm64-server	Primary	✗	✗	✗	✗	✗	✗
	x86_64	4.19.0-server-amd64	Primary	✗	✗	✗	✗	✗	✗
Citrix XenServer Host8.2	x86_64	4.19.0+1	Primary	✗	✗	✗	✗	✗	✗
Kernel 6.6	AArch64	6.6	Primary	✓	✗	✗	✓	✗	✗
	x86_64	6.6	Primary	✓	✗	✓	✓	✗	✗

⚠ 32 bit platforms are no longer supported in MLNX_EN.

2.4.1 Upgrade/Downgrade Matrix

This section reflects which versions were tested and verified for upgrade and downgrade.

Target Version	Versions Verified for Upgrade/ Downgrade	Release Type	Release Date
23.10-7.1.8.0 LTS (December 2025)	23.10-6.1.6.1 • MLNX_OFED and DOCA-OFED Profile	LTS-U6	December 2025
	23.10-0.5.5.0 - MLNX_OFED and DOCA-OFED Profile	GA-LTS-U0	October 2023
	5.8-7.0.6.1	LTS-U7	June 2025
	5.4-3.7.5.0	GA-LTS-Update	November 2022

2.4.2 MLNX_OFED Version Interoperability

This section reflects which versions were tested and verified for multi-version environments.

Target Version	Verified OFED Version Interoperability	Release Type	Release Date
23.10-7.1.8.0 LTS (December 2025)	23.10-6.1.6.1 • MLNX_OFED and DOCA-OFED Profile	LTS-U6	December 2025
	5.8-7.0.6.1	LTS-U7	June 2025

2.4.3 Supported NIC Firmware Versions

> ⚠ As of version 5.1, ConnectX-3, ConnectX-3 Pro or Connect-IB adapter cards are no longer supported. To work with a version that supports these adapter cards, please refer to version 4.9 long-term support (LTS).

This current version is tested with the following NVIDIA adapter card firmware versions:

Adapter Card	Bundled Firmware Version
BlueField®-2	24.39.5172
ConnectX-7	28.39.5172
ConnectX-6 Lx	26.39.5172
ConnectX-6 Dx	22.39.5172
ConnectX-6	20.39.5172
ConnectX-5/ConnectX-5 Ex	16.35.4506
BlueField	18.33.1048
ConnectX-4	12.28.2006
ConnectX-4 Lx	14.32.1010

For the official firmware versions, please see https://www.nvidia.com/en-us/networking/ → Support → Support → Firmware Download.

2.5 Changes and New Features

2.5.1 New Features

The following are the new features and changes that were added in this version. The supported adapter cards are specified as follows:

Supported Cards	Description
All HCAs	Supported in the following adapter cards **unless specifically stated otherwise:** ConnectX-4/ConnectX-4 Lx/ConnectX-5/ConnectX-6/ConnectX-6 Dx/ConnectX-6 Lx/ConnectX-7/BlueField-2
ConnectX-6 Dx and above	Supported in the following adapter cards **unless specifically stated otherwise:** ConnectX-6 Dx/ConnectX-6 Lx/ConnectX-7/BlueField-2
ConnectX-6 and above	Supported in the following adapter cards **unless specifically stated otherwise:** ConnectX-6/ConnectX-6 Dx/ConnectX-6 Lx/ConnectX-7/BlueField-2
ConnectX-5 and above	Supported in the following adapter cards **unless specifically stated otherwise:** ConnectX-5/ConnectX-6/ConnectX-6 Dx/ConnectX-6 Lx/ConnectX-7/BlueField-2
ConnectX-4 and above	Supported in the following adapter cards **unless specifically stated otherwise:** ConnectX-4/ConnectX-4 Lx/ConnectX-5/ConnectX-6/ConnectX-6 Dx/ConnectX-6 Lx/ConnectX-7/BlueField-2

Feature/Change	Description
23.10-7.1.8.0	
General	
Embedded Components	Updated the versions of the following embedded component: • MFT • ConnectX adapter cards firmware For further information, see Embedded Components section.
General	Bug fixes

For a list of features from previous versions, see Release Notes Change Log History section. For additional information on the new features, please refer to the MLNX_EN User Manual.

2.5.2 Customer Affecting Changes

2.5.2.1 Changes in This Release

This section provides a list of changes that took place in the current version and break compatibility/interface, discontinue support for features and/or OS versions, etc.

Introduced in Version	Description
23.10-6.1.6.1	XenServer 7.1 OS will not be supported in MLNX_OFED v23.10-6.x.x.x. The last update release that supports this OS is 23.10-5.1.4.0.

2.5.2.2 Changes Planned for Future Releases

This section provides a list of changes that will take place in a future version of the product and will break compatibility/interface, discontinue support for features and/or OS versions, etc.

Planned for Version	Description
N/A	N/A

2.5.2.3 Changes in Earlier Releases

This section provides a list of changes that took place throughout the past two major releases that broke compatibility/interface, discontinued support for features and/or OS versions, etc.

For an archive of all changes, please refer to the Release Notes History section.

Introduced in Version	Description	Customer Impact and Recommendation
N/A	N/A	N/A

2.5.2.4 Discontinued Features

List of features which are supported in previous generations of hardware devices.

N/A

2.5.3 Unsupported Features

This section provides a list of features that are not supported by the software.

- Soft-RoCE
- RDMA experimental verbs library (mlnx_lib)
- CIFS (Common Internet File System) module installation

2.6 Bug Fixes in This Version

Below are the bugs fixed in this version. For a list of fixes previous version, see Bug Fixes History.

Internal Reference Number	Description
4921892 / 4913956	**Description:** Fixed an issue that occurred when CONFIG_UBSAN=y and the user provided an invalid value, resulting in the following UBSAN error:
	Keywords: CONFIG_UBSAN
	Discovered in Release: 23.10-6.1.6.1
	Fixed in Release: 23.10-7.1.8.0
4858694 / 4858587	**Description:** Fixed soft lockup warnings that could occur while releasing large UMEMs by periodically yielding the CPU during the release flow.
	Keywords: Soft lockup warnings
	Discovered in Release: 23.10-5.1.4.0
	Fixed in Release: 23.10-7.1.8.0

2.7 Known Issues

The following is a list of general limitations and known issues of the current version of the release.

Internal Ref. Number	Issue
3546668	**Description:** On 64k page size systems, applications that open a large number of RDMA resources (UARs/QPs/CQs etc.) might face errors creating those resources due to a PCI BAR size limitation.
	Keywords: PCI BAR size limitation
	Workaround: It is recommended to increase the BAR size via mlxconfig to allow enough space for the allocation of all the needed RDMA resources.
	Discovered in Release: 23.10-1.1.9.0
3678715	**Description:** When attempting to restart drivers using openIbd service while the nvme_rdma module is loaded, the process may fail. This behavior is intentional, as unloading nvme_rdma during the driver restart can lead to connectivity issues in other applications within the setup.
	Keywords: openIbd service, nvme_rdma module
	Workaround: Manually unload the nvme_rdma module before performing the driver restart. This can be achieved using the `modprobe -r nvme_rdma` command.
	Discovered in Release: 23.10-1.1.9.0
3676223	**Description:** When using kernel version 4.12 or above, it is advised to run echo 0 > /sys/bus/pci/devices/0000\:08\:00.0/sriov_drivers_autoprobe to avoid VF probing
	Keywords: VF probing
	Workaround: N/A
	Discovered in Release: 23.10-1.1.9.0
3682658	**Description:** While using the RDMA-CM user application and the AF_IB parameter, the kernel uses only the first byte of the private data to set the CMA version. In such scenario, any user data written to this byte will be overwritten.
	Keywords: RDMA-CM user application, AF_IB, private data
	Workaround: Do not use AF_IB for application's private data.
	Discovered in Release: 23.10-0.5.5.0
3640082	**Description:** A potential null pointer dereference might occur due to a missing update in the PCI subsystem code when creating the maximum number of VFs. All kernel versions lacking the following fix are impacted: "PCI: Avoid enabling PCI atomics on VFs."
	Keywords: Maximal VF number
	Workaround: N/A
	Discovered in Release: 23.10-0.5.5.0
3653417	**Description:** When offloading IPsec policy rules while in legacy mode there are two options: 1. Software steering - The software stack will handle the task, and no device offload will take place. 2. Changing the steering mode to firmware steering will return unsupported.
	Keywords: IPsec, legacy mode

Internal Ref. Number	Issue
	Workaround: Perform a devlink reload after changing the steering mode.
	Discovered in Release: 23.10-0.5.5.0
3612274	**Description:** Currently, either IPsec offload or TC offload for a specific interface is allowed. The offloading TC rule to an interface will fail if an IPSec rule is already offloaded on it, and vice-versa.
	Keywords: IPsec offload, TC offload
	Workaround: N/A
	Discovered in Release: 23.10-0.5.5.0
3596126	**Description:** OVS mirroring of both egress and ingress together with modified TTL is not supported by Connectx-5 cards, and may cause packets checksum issues and errors in the dmesg command.
	Keywords: OVS mirroring, Connectx-5
	Workaround: N/A
	Discovered in Release: 23.10-0.5.5.0
3538463	**Description:** A Kernel ABI problem in Sles15SP4 may lead to issues during driver start. This impacts kernels starting from version 5.14.21-150400.24.11.1 up to version 5.14.21-150400.24.63.1 (July 2022 to May 2023), inclusive. For more information, see https://www.suse.com/support/kb/doc/?id=000021137.
	Keywords: Kernel ABI, Sles15SP4, driver start
	Workaround: Upgrade to a kernel version newer than 5.14.21-150400.24.63.1 (May 2023).
	Discovered in Release: 23.10-0.5.5.0
3637252	**Description:** When running over REHL7.6 with excessive RDMA/RoCE workload, kernel warnings may be triggered.
	Keywords: REHL7.6, RDMA, RoCE
	Workaround: N/A
	Discovered in Release: 23.10-0.5.5.0

Internal Ref. Number	Issue
3046655	**Description:** A package manager upgrade with zypper (on an SLES system) may prompt a question about vendor change from "Mellanox Technologies" to "OpenFabrics".
	Keywords: Installation, SLES
	Workaround: Either accept the prompted change, or add the /etc/zypp/vendors.d/mlnx_ofed file with the following content: [main] vendors = Mellanox,OpenFabrics
	Discovered in Release: 23.07-0.5.0.0
3392477	**Description:** The ConnectX-7 firmware embedded in this MLNX_OFED version cannot be burnt using the MLNX_OFED installer script.

Internal Ref. Number	Issue
	Keywords: ConnectX-7, MLNX_OFED installer script
	Workaround: Please download and install the dedicated firmware from the web https://network.nvidia.com/support/firmware/connectx7ib/
	Discovered in Release: 23.07-0.5.0.0
3532756	**Description:** The kernel may crash when restarting the driver while IP sec rules are configured.
	Keywords: IP sec
	Workaround: Flush the IP sec configuration before reloading the driver: ip xfrm state flush ip xfrm policy flush
	Discovered in Release: 23.07-0.5.0.0
3472979	**Description:** When a large number of virtual functions are present, the output of the `"ip link show"` command may be truncated.
	Keywords: virtual functions, ip link show
	Workaround: N/A
	Discovered in Release: 23.07-0.5.0.0
3413938	**Description:** When using the mlnx-sf script, creating and deleting an SF with the same ID number in a stressful manner may cause the setup to hang due to a race between the create and delete commands.
	Keywords: Hang; mlnx-sf
	Workaround: N/A
	Discovered in Release: 23.07-0.5.0.0
3461572	**Description:** Configuring Multiport Eswitch LAG mode can be performed only via devlink from this release onwards. The compat sysfs should not be used to configure mpesw LAG.
	Keywords: Multiport Eswitch, compat sysfs, mpesw LAG
	Workaround: N/A
	Discovered in Release: 23.07-0.5.0.0
3464337	**Description:** Simultaneously adding or removing TC rules while operating on kernel version 6.3 could potentially result in stability issues.
	Keywords: ASAP, rules, TC
	Workaround: Make sure the following fix is part of the kernel: https://lore.kernel.org/netdev/20230504181616.2834983-3-vladbu@nvidia.com/T/
	Discovered in Release: 23.07-0.5.0.0
3469484	**Description:** Mirror and connection tracking (CT) offload actions are not supported simultaneously if the kernel version does not support hardware miss to TC actions. Thus, when performing a CT offload test, the actual number of offloaded connections may be lower than expected.
	Keywords: ASAP, CT offload

Internal Ref. Number	Issue
	Workaround: Make sure to have the following offending commit in the tree: net/sched: act_ct: offload UDP NEW connections Make sure to to have https://www.spinics.net/lists/stable-commits/msg303536.html in the kernel tree to fix this issue.
	Discovered in Release: 23.07-0.5.0.0
3473331	**Description:** When performing a CT offload test, the actual number of offloaded connections may be lower than expected.
	Keywords: ASAP, CT offload
	Workaround: N/A
	Discovered in Release: 23.07-0.5.0.0
3499413	**Description:** Due to the following kernel issue, under heavy load, some connections may not be offloaded, leading to performance issues: "net/sched: act_ct: offload UDP NEW connections."
	Keywords: ASAP, CT offload
	Workaround: N/A
	Discovered in Release: 23.07-0.5.0.0

Internal Ref. Number	Issue
3360710	**Description:** Configuring PFC in parallel to buffer size and prio2buffer commands may lead to misalignment between firmware and software in regards to receiving buffer ownership.
	Keywords: NetDev, PFC, Buffer Size, prio2buffer
	Workaround: First, configure PFC on all ports, and then perform other needed QoS (i.e., buffer_size or prio2buffer) configurations accordingly.
	Discovered in Release: 23.04-0.5.3.3
3413879	**Description:** OpenSM may not be started automatically if chkconfig was not installed before OpenSM is installed. Note, however, that chkconfig will fail to install if the directory (rather than symbolic link to directory) /etc/init.d already exists (e.g., from a previous installation of MLNX_OFED).
	Keywords: Installation, OpenSM, chkconfig
	Workaround: Install chkconfig before installing MLNX_OFED. If installing it fails, make sure /etc/init.d does not exist at the time of installing it.
	Discovered in Release: 23.04-0.5.3.3
3424596	**Description:** On SLES 15.4, installing MLNX_OFED using a package repository (with zypper) may trigger an error message about missing dependency for 'librte_eal.so.20.0()(64bit)' . This is because the inbox package libdpdk-20_0 is being uninstalled as it is incompatible with the MLNX_OFED rdma-core packages.
	Keywords: Installation, SLES 15.4
	Workaround: Uninstall the relevant packages: 'zypper uninstall libdpdk-20_0' before installing MLNX_OFED. This will also remove the inbox openvswitch package.
	Discovered in Release: 23.04-0.5.3.3

Internal Ref. Number	Issue
3433416	**Description:** On systems that were installed with MLNX_OFED 5.9 or older and include a CUDA package (ucx-cuda / hcoll-cuda), an upgrade to MLNX_OFED 23.04 using the package manager ("yum") method will fail. This is because MLNX_OFED up to 5.9 is built with CUDA 11. MLNX_OFED 23.04 is built with CUDA 12 and those CUDA versions are incompatible.
	Keywords: Installation, CUDA, yum
	Workaround: Remove CUDA packages included with OFED (ucx-cuda, hcoll-cuda) before upgrading. This will allow to upgrade MLNX_OFED regardless of CUDA version installed. To install them later, CUDA 12 must be installed on the system.
	Discovered in Release: 23.04-0.5.3.3
3420831	**Description:** mlx-steering-dump is not supported on systems in which Python3 is not the default.
	Keywords: mlx-steering-dump, Python3
	Workaround: N/A
	Discovered in Release: 23.04-0.5.3.3
3351989	**Description:** If the underlying persistent device name exceeds 15 characters in length, the operating system will not be able to perform renaming (i.e., the device name will remain "eth<digit>").
	Keywords: Persistant Interface Names
	Workaround: Add the --copy-ifnames-udev flag to the OFED installation command. Note that this flag is only applicable if the persistent name provided by the kernel, without the 'np<digit>' suffix, is 15 characters or fewer.
	Discovered in Release: 23.04-0.5.3.3

3 User Manual

3.1 Introduction

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

This document provides information about MLNX_EN Linux driver, and instructions on how to install the driver on ConnectX network adapter solutions supporting the following uplinks to servers:

Uplink/NICs	Driver Name	Uplink Speed
BlueField-2	mlx5	• InfiniBand: SDR, FDR, EDR, HDR • Ethernet: 1GbE, 10GbE, 25GbE, 40GbE, 50GbE[2], 100GbE[2]
BlueField		• InfiniBand: SDR, QDR, FDR, FDR10, EDR • Ethernet: 1GbE, 10GbE, 25GbE, 40GbE, 50GbE, 100GbE
ConnectX-6 Dx		• Ethernet: 10GbE, 25GbE, 40GbE, 50GbE[2], 100GbE[2], 200GbE[2]
ConnectX-6 Lx		• Ethernet: 1GbE, 10GbE, 25GbE, 40GbE, 50GbE[2]
ConnectX-6		• InfiniBand: SDR, FDR, EDR, HDR • Ethernet: 10GbE, 25GbE, 40GbE, 50GbE[2], 100GbE[2], 200GbE[2]
ConnectX-5/ConnectX-5 Ex		• InfiniBand: SDR, QDR, FDR, FDR10, EDR • Ethernet: 1GbE, 10GbE, 25GbE, 40GbE, 50GbE, 100GbE
ConnectX-4 Lx		• Ethernet: 1GbE, 10GbE, 25GbE, 40GbE, 50GbE
ConnectX-4		• InfiniBand: SDR, QDR, FDR, FDR10, EDR • Ethernet: 1GbE, 10GbE, 25GbE, 40GbE, 50GbE, 56GbE[1], 100GbE

1. 56GbE is an NVIDIA proprietary link speed and can be achieved while connecting an NVIDIA adapter card to
   NVIDIA SX10XX switch series or when connecting an NVIDIA adapter card to another NVIDIA adapter card.
2. Supports both NRZ and PAM4 modes.

MLNX_EN driver release exposes the following capabilities:

- Single/Dual port
- Multiple Rx and Tx queues
- Rx steering mode: Receive Core Affinity (RCA)
- MSI-X or INTx
- Adaptive interrupt moderation

- HW Tx/Rx checksum calculation
- Large Send Offload (i.e., TCP Segmentation Offload)
- Large Receive Offload
- Multi-core NAPI support
- VLAN Tx/Rx acceleration (HW VLAN stripping/insertion)
- Ethtool support
- Net device statistics
- SR-IOV support
- Flow steering
- Ethernet Time Stamping

3.1.1 Package Contents

3.1.1.1 Package Images

MLNX_EN is provided as an ISO image or as a tarball per Linux distribution and CPU architecture that includes source code and binary RPMs, firmware and utilities. The ISO image contains an installation script (called install) that performs the necessary steps to accomplish the following:

- Discover the currently installed kernel
- Uninstall any previously installed MLNX_OFED/MLNX_EN packages
- Install the MLNX_EN binary RPMs (if they are available for the current kernel)
- Identify the currently installed HCAs and perform the required firmware updates

3.1.1.2 Software Components

MLNX_EN contains the following software components:

Components	Description
mlx5 driver	mlx5 is the low level driver implementation for the ConnectX-4 adapters. ConnectX-4 operates as a VPI adapter.
mlx5_core	Acts as a library of common functions (e.g. initializing the device after reset) required by the ConnectX-4 adapter cards.
mlx4 driver	mlx4 is the low level driver implementation for the ConnectX adapters. The ConnectX can operate as an InfiniBand adapter and as an Ethernet NIC. To accommodate the two flavors, the driver is split into modules: mlx4_core, mlx4_en, and mlx4_ib. Note: mlx4_ib is not part of this package.
mstflint	An application to burn a firmware binary image.
Software modules	Source code for all software modules (for use under conditions mentioned in the modules' LICENSE files)

3.1.1.3 Firmware

The ISO image includes the following firmware item:

- Firmware images (.bin format wrapped in the mlxfwmanager tool) for ConnectX-4 and and above network adapters

3.1.1.4 Directory Structure

The tarball image of MLNX_EN contains the following files and directories:

- install—the MLNX_EN installation script
- uninstall.sh—the MLNX_EN un-installation script
- RPMS/—directory of binary RPMs for a specific CPU architecture
- src/—directory of the OFED source tarball
- mlnx_add_kernel_support.sh—a script required to rebuild MLNX_EN for customized kernel version on supported Linux distribution

3.1.2 Module Parameters

3.1.2.1 mlx5_core Module Parameters

The mlx5_core module supports a single parameter used to select the profile which defines the number of resources supported.

prof_sel	The parameter name for selecting the profile. The supported values for profiles are: • 0—for medium resources, medium performance • 1—for low resources • 2—for high performance (int) (default)
guids	charp
node_guid	guids configuration. This module parameter will be obsolete!
debug_mask	debug_mask: 1 = dump cmd data, 2 = dump cmd exec time, 3 = both. Default=0 (uint)
probe_vf	probe VFs or not, 0 = not probe, 1 = probe. Default = 1 (bool)
num_of_groups	Controls the number of large groups in the FDB flow table. Default=4; Range=1-1024

3.1.3 Devlink Parameters

The following parameters, supported in mlx4 driver only, can be changed using the Devlink user interface:

Parameter	Description	Parameter Type
internal_error_reset	Enables resetting the device on internal errors	Generic
max_macs	Max number of MACs per ETH port	Generic
region_snapshot_enable	Enables capturing region snapshots	Generic
enable_64b_cqe_eqe	Enables 64 byte CQEs/EQEs when supported by FW	Driver-specific

Parameter	Description	Parameter Type
enable_4k_uar	Enables using 4K UAR	Driver-specific

3.2 Installation

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

The chapter contains the following sections:

3.2.1 Software Dependencies

MLNX_EN driver cannot coexist with OFED software on the same device. Therefore, when installing MLNX_EN, all OFED packages should be removed (by running the `install` script).

3.2.2 Downloading the Drivers

1. Verify that the system has a NVIDIA network adapter (HCA/NIC) installed.
   The following example shows a system with an installed NVIDIA HCA:

```
# lspci -v | grep Mellanox
86:00.0 Network controller [0207]: Mellanox Technologies MT27620 Family
        Subsystem: Mellanox Technologies Device 0014
86:00.1 Network controller [0207]: Mellanox Technologies MT27620 Family
        Subsystem: Mellanox Technologies Device 0014
```

   Note: For ConnectX-5 Socket Direct adapters, use ibdev2netdev to display the installed card and the mapping of logical ports to physical ports. Example:

```
[root@gen-l-vrt-203 ~]# ibdev2netdev -v | grep -i MCX556M-ECAT-S25
0000:84:00.0 mlx5_10 (MT4119 - MCX556M-ECAT-S25SN) CX556M - ConnectX-5 QSFP28 fw 16.22.0228 port 1 (DOWN )
==> p2p1 (Down)
0000:84:00.1 mlx5_11 (MT4119 - MCX556M-ECAT-S25SN) CX556M - ConnectX-5 QSFP28 fw 16.22.0228 port 1 (DOWN )
==> p2p2 (Down)
0000:05:00.0 mlx5_2 (MT4119 - MCX556M-ECAT-S25SN) CX556M - ConnectX-5 QSFP28 fw 16.22.0228 port 1 (DOWN )
==> p5p1 (Down)
0000:05:00.1 mlx5_3 (MT4119 - MCX556M-ECAT-S25SN) CX556M - ConnectX-5 QSFP28 fw 16.22.0228 port 1 (DOWN )
==> p5p2 (Down)
```

   - Each PCI card of ConnectX-5 Socket Direct has a different PCI address. In the output example above, the first two rows indicate that one card is installed in a PCI slot with PCI Bus address 84 (hexadecimal), and PCI Device Number 00, and PCI Function Number 0 and 1. RoCE assigned mlx5_10 as the logical port,

which is the same as netdevice p2p1, and both are mapped to physical port of PCI function 0000:84:00.0.
- RoCE logical port mlx5_2 of the second PCI card (PCI Bus address 05) and netdevice p5p1 are mapped to physical port of PCI function 0000:05:00.0, which is the same physical port of PCI function 0000:84:00.0.
MT4119 is the PCI Device ID of the ConnectX-5 adapters family.

For more details, please refer to ConnectX-5 Socket Direct Hardware User Manual, available at nvidia.com/en-us/networking/.

2. Download the ISO image to your host.
The image name has the format mlnx-en-<ver>-<OS label-<CPU arch>.iso. It can also be downloaded from nvidia.com/en-us/networking/ → Products → Software → Ethernet Drivers.
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.

3.2.3 Installing MLNX_EN

3.2.3.1 Installation Script

The `install` installation script performs the following:

- Discovers the currently installed kernel
- Uninstalls any previously installed MLNX_EN package
- Installs the MLNX_EN binary (if they are available for the current kernel)
- Identifies the currently installed Ethernet network adapters and automatically upgrades the firmware

⚠ If the driver detects unsupported cards on your system, it will abort the installation procedure. To avoid this, make sure to add `--skip-unsupported-devices-check` flag during installation.

3.2.3.2 Installation Modes

mlnx_en installer supports 2 modes of installation. The install script selects the mode of driver installation depending on the running OS/kernel version.

- Kernel Module Packaging (KMP) mode, where the source rpm is rebuilt for each installed flavor of the kernel. This mode is used for RedHat and SUSE distributions.
- Non KMP installation mode, where the sources are rebuilt with the running kernel. This mode is used for vanilla kernels.
- By default, the package will install drivers supporting Ethernet only. In addition, the package will include the following new installation options:
    - Full VMA support which can be installed using the installation option “-vma”.

- Infrastructure to run DPDK using the installation option “-dpdk”.
  Notes:
    - DPDK itself is not included in the package. Users would still need to install DPDK separately after the MLNX_EN installation is completed.
    - Installing VMA or DPDK infrastructure will allow users to run RoCE.

Installation Results

- For Ethernet only installation mode:
    - The kernel modules are installed under:
        - /lib/modules/`uname -r`/updates on SLES and Fedora Distributions
        - /lib/modules/`uname -r`/extra/mlnx-en on RHEL and other RedHat like Distributions
        - /lib/modules/`uname -r`/updates/dkms/ on Ubuntu
    - The kernel module sources are placed under: `/usr/src/mlnx-en-<ver>/`
- For VPI installation mode:
    - 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
    - The kernel module sources are placed under: `/usr/src/ofa_kernel-<ver>/`

3.2.3.3 Installation Procedure

This section describes the installation procedure of MLNX_EN on NVIDIA adapter cards.

1. Log into the installation machine as root.
2. Mount the ISO image on your machine.

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

3. Run the installation script.

```
/mnt/install
```

4. Case A: If the installation script has performed a firmware update on your network adapter, you need to either restart the driver or reboot your system before the firmware update can take effect. Refer to the table below to find the appropriate action for your specific card.

Action \ Adapter	Driver Restart	Standard Reboot (Soft Reset)	Cold Reboot (Hard Reset)
Standard ConnectX-4/ ConnectX-4 Lx or higher	-	+	-
Adapters with Multi-Host Support	-	-	+
Socket Direct Cards	-	-	+

Case B: If the installations script has not performed a firmware upgrade on your network adapter, restart the driver by running: `# /etc/init.d/mlnx-en.d restart`

The result is a new net-device appearing in the 'ifconfig -a' output.

3.2.3.4 Additional Installation Procedures

3.2.3.4.1 Installing MLNX_EN Using YUM

This type of installation is applicable to RedHat/OL, Fedora, XenServer operating systems.

3.2.3.4.1.1 Setting up MLNX_EN YUM Repository

The package consists of two folders that can be set up as a repository:

- "RPMS_ETH" - provides the Ethernet only installation mode
- "RPMS" - provides the RDMA support installation mode

1. Log into the installation machine as `root`.
2. Mount the ISO image on your machine and copy its content to a shared location in your network.

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

   You can download the image from nvidia.com/en-us/networking/.com → Products → Software → Ethernet Drivers.
3. Download and install the NVIDIA GPG-KEY:
   The key can be downloaded via the following link: http://www.mellanox.com/downloads/ofed/RPM-GPG-KEY-Mellanox
   Example:

   ```
   # wget http://www.mellanox.com/downloads/ofed/RPM-GPG-KEY-Mellanox
   --2014-04-20 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
   2014-04-20 13:52:30 (247 MB/s) - ?RPM-GPG-KEY-Mellanox? saved [1354/1354]
   ```
4. Install the key.
   Example:

   ```
   # sudo rpm --import RPM-GPG-KEY-Mellanox
   warning: rpmts_HdrFromFdno: Header V3 DSA/SHA1 Signature, key ID 6224c050: NOKEY
   Importing GPG key 0x6224C050:
   Userid: "Mellanox Technologies (Mellanox Technologies - Signing Key v2) <support@mellanox.com>"
   From : /repos/MLNX_EN/RPM-GPG-KEY-Mellanox
   Is this ok [y/N]:
   ```
5. Check that the key was successfully imported.
   Example:

   ```
   # rpm -q gpg-pubkey --qf '%{NAME}-%{VERSION}-%{RELEASE}\t%{SUMMARY}\n' | grep Mellanox
   gpg-pubkey-a9e4b643-520791ba gpg(Mellanox Technologies <support@mellanox.com>)
   Rev 3.30
   Mellanox Technologies 45
   ```

6. Create a YUM repository configuration file called “/etc/yum.repos.d/mlnx_en.repo” with the following content:

```
[mlnx_en]
name=MLNX_EN Repository
baseurl=file:///<path to extracted MLNX_EN package>/<RPMS FOLDER NAME>
enabled=1
gpgkey=file:///<path to the downloaded key RPM-GPG-KEY-Mellanox>
gpgcheck=1
```

> ⚠ Replace <RPMS FOLDER NAME> with “RPMS_ETH” or “RPMS” depending on the desired installation mode (Ethernet only or RDMA).

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 subscription-manager to
register.
repo id                                                              repo name
status
mlnx_en                                                              MLNX_EN Repository
```

3.2.3.4.1.2 Installing MLNX_EN Using the YUM Tool

After setting up the YUM repository for MLNX_EN package, install one of the following metadata packages:

- In case you set up the “RPMS_ETH” folder as the repository (for Ethernet only mode), install:

```
# yum install mlnx-en-eth-only
```

- In case you set up the “RPMS” folder as the repository (for RDMA mode), install either:

```
# yum install mlnx-en-vma
```

Or

```
# yum install mlnx-en-dpdk
```

Please note the following:

MLNX_EN provides kernel module RPM packages with KMP support for RHEL and SLES. For other operating systems, kernel module RPM packages are provided only for the operating system’s default kernel. In this case, the group RPM packages have the supported kernel version in their packages name.
If you have an operating systems different than RHEL or SLES, or you have installed a kernel that is not supported by default in MLNX_EN, you can use the mlnx_add_kernel_support.sh script to build MLNX_EN for your kernel.
The script will automatically build the matching group RPM packages for your kernel so that you can still install MLNX_EN via YUM.
Please note that the resulting MLNX_EN repository will contain unsigned RPMs. Therefore, you should set 'gpgcheck=0' in the repository configuration file.

Installing MLNX_EN using the YUM tool does not automatically update the firmware.
To update the firmware to the version included in MLNX_EN package, you can either:

1. Run:

```
# yum install mlnx-fw-updater
```

   OR
2. Update the firmware to the latest version available on NVIDIA website as described in Updating Firmware After Installation section.

3.2.3.4.2 Installing MLNX_EN Using apt-get

This type of installation is applicable to Debian and Ubuntu operating systems.

3.2.3.4.2.1 Setting up MLNX_EN apt-get Repository

The package consists of two folders that can be set up as a repository:

- “DEBS_ETH” - provides the Ethernet only installation mode.
- “RPMS” - provides the RDMA support installation mode.

1. Log into the installation machine as root.
2. Extract the MLNX_EN package on a shared location in your network.
   You can download it from nvidia.com/en-us/networking/.com → Products → Software → Ethernet Drivers.
3. Create an apt-get repository configuration file called "/etc/apt/sources.list.d/mlnx_en.list" with the following content:

```
deb file:/<path to extracted MLNX_EN package>/<DEBS FOLDER NAME> ./
```

   Replace <DEBS FOLDER NAME> with “DEBS_ETH” or “DEBS” depending on the desired installation mode (Ethernet only or RDMA).
4. Download and install the NVIDIA GPG-KEY.
   Example:

```
# wget -qO - http://www.mellanox.com/downloads/ofed/RPM-GPG-KEY-Mellanox | sudo apt-key add -
```

5. Verify that the key was successfully imported.
   Example:

```
# apt-key list
pub 1024D/A9E4B643 2013-08-11
uid Mellanox Technologies <support@mellanox.com>
sub 1024g/09FCC269 2013-08-11
```

6. Update the apt-get cache.

```
# sudo apt-get update
```

3.2.3.4.2.2 Installing MLNX_EN Using the apt-get Tool

After setting up the apt-get repository for MLNX_EN package, install one of the following metadata packages:

- In case you set up the “DEBS_ETH” folder as the repository (for Ethernet only mode), install:

```
# apt-get install mlnx-en-eth-only
```

- In case you set up the “DEBS” folder as the repository (for RDMA mode), install either:

```
# apt-get install mlnx-en-vma
```

OR

```
# apt-get install mlnx-en-dpdk
```

Installing MLNX_EN using the apt-get tool does not automatically update the firmware. To update the firmware to the version included in MLNX_EN package, you can either:

1. Run:

```
# apt-get install mlnx-fw-updater
```

Or

2. Update the firmware to the latest version available on NVIDIA website as described in Updating Firmware After Installation section.

3.2.3.4.2.3 Installation Using Repositories

NVIDIA Legacy Libraries can also be installed using the operating system's standard package manager (yum, apt-get, etc.).

- For RPM based operating systems, follow the steps in “Installing MLNX_EN Using YUM” section, using the directory “MLNX_LIBS" instead of “UPSTREAM_LIBS” when creating the “/etc/yum.repos.d/mlnx_ofed.repo” file.
- For Debian based operating systems, follow the steps in “Installing MLNX_EN Using apt-get” section using the directory “MLNX_LIBS” instead of “UPSTREAM_LIBS” when creating the “/etc/apt/sources.list.d/mlnx_ofed.list” file.

Finally, for both RPM and Debian based OSs, install the new metadata package called “mlnx-ofed-dpdk-upstream-libs”, which will install both the user space and kernel packages.
If you wish to install only the user space packages, make sure to install the metadata package called “mlnx-ofed-dpdk-upstream-libs-user-only”.

3.2.4 Uninstall

3.2.4.1 Uninstalling MLNX_EN Using the YUM and apt-get Tools

Use the script `/usr/sbin/mlnx_en_uninstall.sh` to uninstall MLNX_EN package.

3.2.5 Updating Firmware After Installation

The firmware can be updated using one of the following methods.

3.2.5.1 Updating the Device Online

To update the device online on the machine from the NVIDIA site, use the following command line:

```
mlxfwmanager --online -u -d <device>
```

Example:

```
# mlxfwmanager --online -u -d 0000:81:00.0
Querying Mellanox devices firmware ...

Device #1:
----------

  Device Type:      ConnectX6DX
  Part Number:      MCX623106AN-CDA_Ax
  Description:      ConnectX-6 Dx EN adapter card; 100GbE; Dual-port QSFP56; PCIe 4.0/3.0 x16;
  PSID:             MT_0000000359
  PCI Device Name:  0000:81:00.0
  Base GUID:        1c34da030080284a
  Base MAC:         1c34da80284a
  Versions:         Current        Available
     FW             22.28.1034     22.28.1002
     PXE            3.6.0101       3.6.0101
     UEFI           14.21.0016     14.21.0016

Status:             Update required
---------
Found 1 device(s) requiring firmware update. Please use -u flag to perform the update.
```

3.2.5.2 Updating the Device Manually

When running the `install` script with the '`--without-fw-update`' option or using an OEM card that you now wish to (manually) update firmware on your adapter card(s), perform the steps below. The following steps are also appropriate to burn newer firmware that was downloaded from the website (i.e., nvidia.com/en-us/networking/ → Support → Support → Firmware Download).

1. Get the device's PSID.

   ```
   mlxfwmanager_pci | grep PSID
   PSID:            MT_1210110019
   ```

2. Download the firmware BIN file from the website or the OEM website.
3. Burn the firmware.

   ```
   mlxfwmanager_pci -i <fw_file.bin>
   ```

4. Reboot the device once the firmware burning is completed.

3.2.5.3 Updating the Device Firmware Automatically Upon System Boot

Firmware can be automatically updated upon system boot.
The firmware update package (mlnx-fw-updater) is installed in the "/opt/mellanox/mlnx-fw-updater" folder, and the openibd service script can invoke the firmware update process if requested

on boot.
If the firmware is updated, the following message will be printed to the system's standard logging file:

```
fw_updater: Firmware was updated. Please reboot your system for the changes to take effect.
```

Otherwise, the following message will be printed:

```
fw_updater: Didn't detect new devices with old firmware.
```

Please note that this feature is disabled by default. To enable the automatic firmware update upon system boot, set the following parameter to "yes" "RUN_FW_UPDATER_ONBOOT=yes" in the openibd service configuration file "/etc/infiniband/openib.conf".
You can opt to exclude a list of devices from the automatic firmware update procedure. To do so, edit the configurations file "/opt/mellanox/mlnx-fw-updater/mlnx-fw-updater.conf" and provide a comma separated list of PCI devices to exclude from the firmware update.
Example:

```
MLNX_EXCLUDE_DEVICES="00:05.0,00:07.0"
```

3.2.6 Ethernet Driver Usage and Configuration

To assign an IP address to the interface, run:

```
#> ifconfig eth<x> <ip>
```

Note: 'x' is the OS assigned interface number.

To check driver and device information:

```
#> ethtool -i eth<x>
```

Example:

```
#> ethtool -i eth2
driver: mlx4_en
version: 2.1.8 (Oct 06 2013)
firmware-version: 2.30.3110
bus-info: 0000:1a:00.0
```

To query stateless offload status:

```
#> ethtool -k eth<x>
```

To set stateless offload status:

```
#> ethtool -K eth<x> [rx on|off] [tx on|off] [sg on|off] [tso on|off] [lro on|off]
```

To query interrupt coalescing settings:

```
#> ethtool -c eth<x>
```

To enable/disable adaptive interrupt moderation:

```
#>ethtool -C eth<x> adaptive-rx on|off
```

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

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

```
#> ethtool -C eth<x> [pkt-rate-low N] [pkt-rate-high N] [rx-usecs-low N] [rx-usecs-high N]
```

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

To set interrupt coalescing settings when adaptive moderation is disabled:

```
#> ethtool -C eth<x> [rx-usecs N] [rx-frames N]
```

usec settings correspond to the time to wait after the *last* packet is sent/received before triggering an interrupt.

➢ *To query ring size values:*

```
#> ethtool -g eth<x>
```

➢ *To modify rings size:*

```
#> ethtool -G eth<x> [rx <N>] [tx <N>]
```

➢ *To obtain additional device statistics:*

```
#> ethtool -S eth<x>
```

The driver defaults to the following parameters:

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

Some of these values can be changed using module parameters, which can be displayed by running:

```
#> modinfo mlx5_en
```

To set non-default values to module parameters, add to the /etc/modprobe.conf file:

```
"options mlx5_en <param_name>=<value> <param_name>=<value> ..."
```

Values of all parameters can be observed in /sys/module/mlx5_en/parameters/.

3.2.7 Performance Tuning

Depending on the application of the user's system, it may be necessary to modify the default configuration of network adapters based on the ConnectX® adapters. In case tuning is required, please refer to the Performance Tuning for NVIDIA Adapters Community post.

3.3 Features Overview and Configuration

> ⚠ It is recommended to enable the "above 4G decoding" BIOS setting for features that require a large amount of PCIe resources (e.g., SR-IOV with numerous VFs, PCIe Emulated Switch, Large BAR Requests).

The chapter contains the following sections:

3.3.1 Ethernet Network

The chapter contains the following sections:

3.3.1.1 Ethernet Interface

3.3.1.1.1 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 help 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 Queue Pair (QP) attribute given in the QP context. Multiple QPs may be associated with the same counter set. If multiple QPs share the same counter, the counter value will represent the cumulative total.

3.3.1.1.1.1 RoCE Counters

- RoCE counters are available only through sysfs located under:
  - # /sys/class/infiniband/<device>/ports/*/hw_counters/
  - # /sys/class/infiniband/<device>/hw_counters/
  - # /sys/class/infiniband/<device>/ports/*/counters/

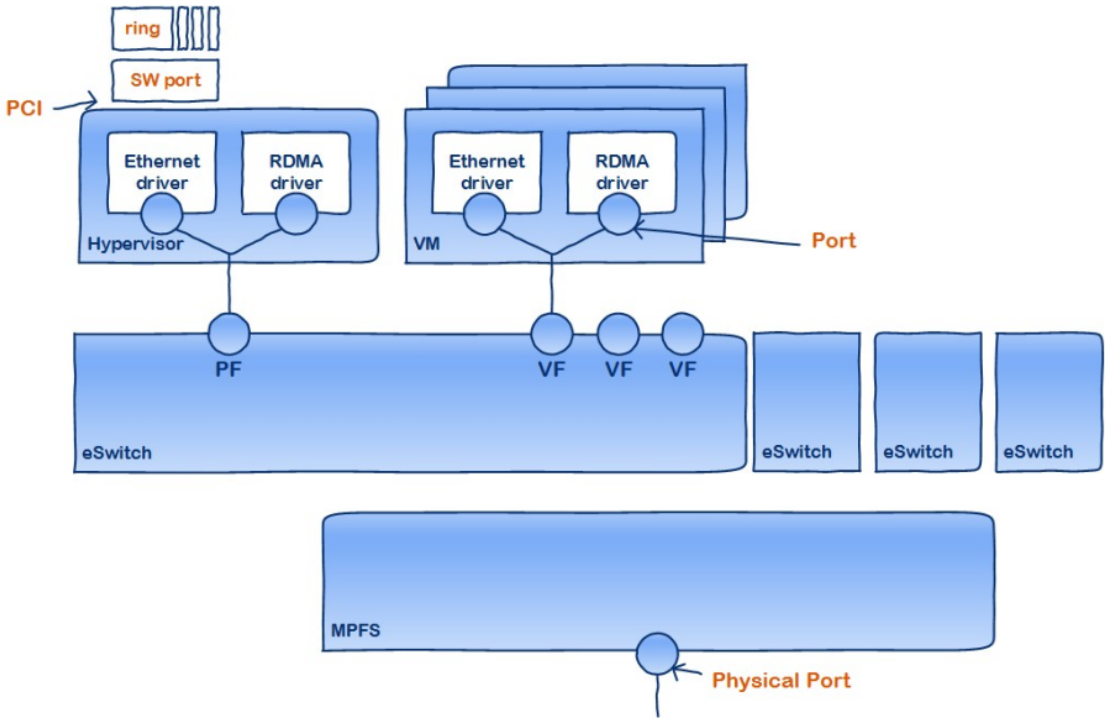
For mlx5 port and RoCE counters, refer to the Understanding mlx5 Linux Counters Community post.

3.3.1.1.1.2 SR-IOV Counters

Physical Function can also read Virtual Functions' port counters through sysfs located under # /sys/class/net/<interface_name>/device/sriov/<index>/stats/

3.3.1.1.1.3 ethtool Counters

The ethtool counters are counted in different places, according to which they are divided into groups. Each counters group may also have different counter types.



For the full list of supported ethtool counters, refer to the Understanding mlx5 ethtool Counters community post.

3.3.1.1.2 Persistent Naming

To avoid network interface renaming after boot or driver restart, set the desired constant interface name in the "/etc/udev/rules.d/70-persistent-net.rules" file.

- Example for Ethernet interfaces:

```
PCI device 15b3:1019 (mlx5_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"
```

3.3.1.1.3 Interrupt Request (IRQ) Naming

Once IRQs are allocated by the driver, they are named `mlx5_comp<x>@pci:<pci_addr>`. The IRQ name is constant and is not affected by the interface state.
The mlx5_core driver allocates all IRQs during loading time to support the maximum possible number of channels. Once the driver is up, no further IRQs are freed or allocated. Changing the number of working channels does not re-allocate or free the IRQs.

3.3.1.2 Quality of Service (QoS)

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 stage process. The TC is assigned with the QoS attributes and the different flows behave accordingly.

3.3.1.2.1 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 Queue Pairs (QPs)
- Raw Ethernet QP - Application use VERBs API to transmit using a Raw Ethernet QP

3.3.1.2.2 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 in the following conditions:
    a. If the underlying device is a VLAN device, egress_map is used controlled by the vconfig command. This is per VLAN mapping.

b. If the underlying device is not a VLAN device, the mapping is done in the driver.

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

3.3.1.2.3 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 the case of a VLAN device where the parent real device is used for the purpose of this mapping
    - If the underlying device is a VLAN device, and the parent real device was not used for the mapping, the VLAN device's egress_map is used
4. 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.

3.3.1.2.4 Map Priorities with set_egress_map

For RoCE old kernels that do not support set_egress_map, use the tc_wrap script to map between sk_prio and UP. Use tc_wrap with option -u. For example:

```
tc_wrap -i <ethX> -u <skprio2up mapping>
```

3.3.1.2.5 Quality of Service Properties

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

- Strict Priority
- Enhanced Transmission Selection (ETS)
- Rate Limit
- Trust State
- Receive Buffer
- DCBX Control Mode

3.3.1.2.5.1 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 the highest priority, TC 0 is lowest). It also has an absolute priority over nonstrict 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.
Nonstrict priority TCs will be considered last to transmit.
This property is extremely useful for low latency low bandwidth traffic that needs to get immediate service when it exists, but is not of high volume to starve other transmitters in the system.

3.3.1.2.5.2 Enhanced Transmission Selection (ETS)

Enhanced Transmission Selection standard (ETS) exploits the time periods in which the offered load of a particular Traffic Class (TC) is less than its minimum allocated bandwidth by allowing the difference to be available to other traffic classes.
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 minimum 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.
ETS is configured using the mlnx_qos tool (mlnx_qos) which allows you to:

- Assign a transmission algorithm to each TC (strict or ETS)
- Set minimal BW guarantee to ETS TCs
  Usage:

```
mlnx_qos -i \[options\]
```

3.3.1.2.5.3 Rate Limit

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

3.3.1.2.5.4 Trust State

Trust state enables prioritizing sent/received packets based on packet fields.
The default trust state is PCP. Ethernet packets are prioritized based on the value of the field (PCP/DSCP).

For further information on how to configure Trust mode, please refer to HowTo Configure Trust State on NVIDIA Adapters community post.

> ⚠ Setting the Trust State mode shall be done before enabling SR-IOV in order to propagate the Trust State to the VFs.

3.3.1.2.5.5 Receive Buffer

By default, the receive buffer configuration is controlled automatically. Users can override the receive buffer size and receive buffer's xon and xoff thresholds using mlnx_qos tool.
For further information, please refer to HowTo Tune the Receive buffers on NVIDIA Adapters community post.

3.3.1.2.5.6 DCBX Control Mode

DCBX settings, such as "ETS" and "strict priority" can be controlled by firmware or software. When DCBX is controlled by firmware, changes of QoS settings cannot be done by the software. The DCBX control mode is configured using the mlnx_qos -d os/fw command.
For further information on how to configure the DCBX control mode, please refer to mlnx_qos community post.

3.3.1.2.6 Quality of Service Tools

3.3.1.2.6.1 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
- Set DCBX control mode
- Set cable length
- Set trust state

> ⚠ For an unlimited ratelimit, set the ratelimit to 0.

Usage

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

Options

--version	Show the program's version number and exit
-h, --help	Show this help message and exit
-f LIST, --pfc=LIST	Set priority flow control for each priority. LIST is a comma separated value for each priority starting from 0 to 7. Example: 0,0,0,0,1,1,1,1 enable PFC on TC4-7
-p LIST, --prio_tc=LIST	Maps UPs to TCs. LIST is 8 comma-separated 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 separated algorithm names for each TC. Possible algorithms: strict, ets and vendor. Example: vendor,strict,ets,ets,ets,ets,ets,ets sets TC0 to vendor, TC1 to strict, TC2-7 to ets
-t LIST, --tcbw=LIST	Set the minimally guaranteed %BW for ETS TCs. LIST is comma-separated 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 10,0,90,0,0,0,0,0will 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-separated Gbps limit for each TC. Example: 1,8,8 will limit TC0 to 1Gbps, and TC1,TC2 to 8 Gbps each
-d DCBX, --dcbx=DCBX	Set dcbx mode to firmware controlled(fw) or OS controlled(os). Note, when in OS mode, mlnx_qos should not be used in parallel with other dcbx tools, such as lldptool
--trust=TRUST	set priority trust state to pcp or dscp
--dscp2prio=DSCP2PRIO	Set/del a (dscp,prio) mapping. Example 'set,30,2' maps dscp 30 to priority 2. 'del,30,2' resets the dscp 30 mapping back to the default setting priority 0
--cable_len=CABLE_LEN	Set cable_len for buffer's xoff and xon thresholds
-i INTF, --interface=INTF	Interface name
-a	Show all interface's TCs

Get Current Configuration

```
ofed_scripts/utils/mlnx_qos -i ens1f0
DCBX mode: OS controlled
Priority trust state: dscp
dscp2prio mapping:
        prio:0 dscp:07,06,05,04,03,02,01,00,
        prio:1 dscp:15,14,13,12,11,10,09,08,
        prio:2 dscp:23,22,21,20,19,18,17,16,
        prio:3 dscp:31,30,29,28,27,26,25,24,
        prio:4 dscp:39,38,37,36,35,34,33,32,
        prio:5 dscp:47,46,45,44,43,42,41,40,
        prio:6 dscp:55,54,53,52,51,50,49,48,
```

```
        prio:7 dscp:63,62,61,60,59,58,57,56,
Cable len: 7
PFC configuration:
        priority 0 1 2 3 4 5 6 7
        enabled 0 0 0 0 0 0 0 0
tc: 0 ratelimit: unlimited, tsa: vendor
         priority: 1
tc: 1 ratelimit: unlimited, tsa: vendor
         priority: 0
tc: 2 ratelimit: unlimited, tsa: vendor
         priority: 2
tc: 3 ratelimit: unlimited, tsa: vendor
         priority: 3
tc: 4 ratelimit: unlimited, tsa: vendor
         priority: 4
tc: 5 ratelimit: unlimited, tsa: vendor
         priority: 5
tc: 6 ratelimit: unlimited, tsa: vendor
         priority: 6
tc: 7 ratelimit: unlimited, tsa: vendor
         priority: 7
```

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

```
# mlnx_qos -i <interface> -p 0,1,2 -r 3,4,2
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:  3
         up:  4
         up:  5
         up:  6
         up:  7
tc: 1 ratelimit: 4 Gbps, tsa: strict
         up:  1
tc: 2 ratelimit: 2 Gbps, tsa: strict
         up:  2
```

ConfigureQoS. Map UP0,7 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 <interface> -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
```

tc and tc_wrap.py

The tc tool is used to create 8 Traffic Classes (TCs).
The tool will either use the sysfs (/sys/class/net/<ethX>/qos/tc_num) or the tc tool to create the TCs.

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 priority for RoCE. LIST is <=16 comma separated priority. index of element is sk_prio
-i INTF, --interface=INTF	Interface name

Example
Run:

```
tc_wrap.py -i enp139s0
```

Output:

```
Tarrfic classes are set to 8

UP  0
    skprio: 0 (vlan 5)
UP  1
    skprio: 1 (vlan 5)
UP  2
    skprio: 2 (vlan 5 tos: 8)
UP  3
    skprio: 3 (vlan 5)
UP  4
    skprio: 4 (vlan 5 tos: 24)
UP  5
    skprio: 5 (vlan 5)
UP  6
    skprio: 6 (vlan 5 tos: 16)
UP  7
    skprio: 7 (vlan 5)
```

3.3.1.2.6.2 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 version 2.5 < = X
- tc_wrap.py (package: ofed-scripts) requires python version 2.5 < = X

3.3.1.3 Quantized Congestion Notification (QCN)

Congestion control is used to reduce packet drops in lossy environments and mitigate congestion spreading and resulting victim flows in lossless environments.
The Quantized Congestion Notification (QCN) IEEE standard (802.1Qau) provides congestion control for long-lived flows in limited bandwidth-delay product Ethernet networks. It is part of the IEEE Data Center Bridging (DCB) protocol suite, which also includes ETS, PFC, and DCBX. QCN in conducted at L2, and is targeted for hardware implementations. QCN applies to all Ethernet packets and all transports, and both the host and switch behavior is detailed in the standard.
QCN user interface allows the user to configure QCN activity. QCN configuration and retrieval of information is done by the mlnx_qcn tool. The command interface provides the user with a set of changeable attributes, and with information regarding QCN's counters and statistics. All parameters and statistics are defined per port and priority. QCN command interface is available if and only the hardware supports it.

3.3.1.3.1 QCN Tool - mlnx_qcn

This tool is supported on ConnectX-3 and ConnectX-3 Pro NICs only.

mlnx_qcn is a tool used to configure QCN attributes of the local host. It communicates directly with the driver thus does not require setting up a DCBX daemon on the system.

The mlnx_qcn enables the user to:

- Inspect the current QCN configurations for a certain port sorted by priority
- Inspect the current QCN statistics and counters for a certain port sorted by priority
- Set values of chosen QCN parameters

Usage

```
mlnx_qcn -i <interface> \[options\]
```

Options

--version	Show program's version number and exit
-h, --help	Show this help message and exit
-i INTF, --interface=INTF	Interface name
-g TYPE, --get_type=TYPE	Type of information to get statistics/parameters
--rpg_enable=RPG_ENABLE_LIST	Set value of rpg_enable according to priority, use spaces between values and -1 for unknown values.
--rppp_max_rps=RPPP_MAX_RPS_LIST	Set value of rppp_max_rps according to priority, use spaces between values and -1 for unknown values.
--rpg_time_reset=RPG_TIME_RESET_LIST	Set value of rpg_time_reset according to priority, use spaces between values and -1 for unknown values.
--rpg_byte_reset=RPG_BYTE_RESET_LIST	Set value of rpg_byte_reset according to priority, use spaces between values and -1 for unknown values.

--rpg_threshold=RPG_THRESHOLD_LIST	Set value of rpg_threshold according to priority, use spaces between values and -1 for unknown values.
--rpg_max_rate=RPG_MAX_RATE_LIST	Set value of rpg_max_rate according to priority, use spaces between values and -1 for unknown values.
--rpg_ai_rate=RPG_AI_RATE_LIST	Set value of rpg_ai_rate according to priority, use spaces between values and -1 for unknown values.
--rpg_hai_rate=RPG_HAI_RATE_LIST	Set value of rpg_hai_rate according to priority, use spaces between values and -1 for unknown values.
--rpg_gd=RPG_GD_LIST	Set value of rpg_gd according to priority, use spaces between values and -1 for unknown values.
--rpg_min_dec_fac=RPG_MIN_DEC_FAC_LIST	Set value of rpg_min_dec_fac according to priority, use spaces between values and -1 for unknown values.
--rpg_min_rate=RPG_MIN_RATE_LIST	Set value of rpg_min_rate according to priority, use spaces between values and -1 for unknown values.
--cndd_state_machine=CNDD_STATE_MACHINE_LIST	Set value of cndd_state_machine according to priority, use spaces between values and -1 for unknown values.

To get QCN current configuration sorted by priority, run:

```
mlnx_qcn -i eth2 -g parameters
```

To show QCN's statistics sorted by priority, run:

```
mlnx_qcn -i eth2 -g statistics
```

Output example of running mlnx_qcn -i eth2 -g parameters:

```
priority 0:
rpg_enable: 0
rppp_max_rps: 1000
rpg_time_reset: 1464
rpg_byte_reset: 150000
rpg_threshold: 5
rpg_max_rate: 40000
rpg_ai_rate: 10
rpg_hai_rate: 50
rpg_gd: 8
rpg_min_dec_fac: 2
rpg_min_rate: 10
cndd_state_machine: 0
priority 1:
rpg_enable: 0
rppp_max_rps: 1000
rpg_time_reset: 1464
rpg_byte_reset: 150000
rpg_threshold: 5
rpg_max_rate: 40000
```

```
rpg_ai_rate: 10
rpg_hai_rate: 50
rpg_gd: 8
rpg_min_dec_fac: 2
rpg_min_rate: 10
cndd_state_machine: 0
.............................
.............................
priority 7:
rpg_enable: 0
rppp_max_rps: 1000
rpg_time_reset: 1464
rpg_byte_reset: 150000
rpg_threshold: 5
rpg_max_rate: 40000
rpg_ai_rate: 10
rpg_hai_rate: 50
rpg_gd: 8
rpg_min_dec_fac: 2
rpg_min_rate: 10
cndd_state_machine: 0
```

3.3.1.3.2 Setting QCN Configuration

Setting QCN parameters requires updating its value for each priority. '-1' indicates no change in the current value.

Example for setting 'rpg_enable' in order to enable QCN for priorities 3, 5, 6:

```
mlnx_qcn -i eth2 --rpg_enable=-1 -1 -1 1 -1 1 1 -1
```

Example for setting 'rpg_hai_rate' for priorities 1, 6, 7:

```
mlnx_qcn -i eth2 --rpg_hai_rate=60 -1 -1 -1 -1 -1 60 60
```

3.3.1.4 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, auto-negotiation and flow control for Ethernet devices
- Control checksum offload and other hardware offload features
- Control DMA ring sizes and interrupt moderation
- Flash device firmware using a .mfa2 image

Ethtool Supported Options

Options	Description
ethtool --set-priv-flags eth<x> <priv flag> <on/off>	Enables/disables driver feature matching the given private flag.
ethtool --show-priv-flags eth<x>	Shows driver private flags and their states (ON/OFF).
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 -c eth<x>	Queries interrupt coalescing settings.

Options	Description
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.
ethtool -C eth<x> [rx-usecs N] [rx-frames N]	Sets the interrupt coalescing setting. rx-frames will be enforced immediately, rx-usecs will be enforced only when 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 -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> adaptive-tx on|off	**Note**: Supported by mlx5e for ConnectX-4 and above adapter cards. Enables/disables adaptive interrupt moderation. By default, the driver uses adaptive interrupt moderation for the transmit path, which adjusts the moderation parameters (time/frames) to the traffic pattern.
ethtool -g eth<x>	Queries the ring size values.
ethtool -G eth<x> [rx <N>] [tx <N>]	Modifies the ring size.
ethtool -i eth<x>	Checks driver and device information. For example: driver: mlx5_core version: 5.1-0.4.0 firmware-version: 4.6.4046 (MT_QEMU000000) expansion-rom-version: bus-info: 0000:07:00.0 supports-statistics: yes supports-test: yes supports-eeprom-access: no supports-register-dump: no supports-priv-flags: yes
ethtool -k eth<x>	Queries the stateless offload status.

Options	Description
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] [rxvlan on|off] [txvlan on|off] [ntuple on/off] [rxhash on/off] [rx-all on/off] [rx-fcs 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. Hardware VLAN insertion Offload (txvlan): When enabled, the sent VLAN tag will be inserted into the packet by the hardware. **Note**: LRO will be done whenever possible. Otherwise GRO will be done. Generic Receive Offload (GRO) is available throughout all kernels. Hardware VLAN Striping Offload (rxvlan): When enabled received VLAN traffic will be stripped from the VLAN tag by the hardware. RX FCS (rx-fcs): Keeps FCS field in the received packets.Sets the stateless offload status. RX FCS validation (rx-all): Ignores FCS validation on the received packets.
ethtool -l eth<x>	Shows the number of channels.
ethtool -L eth<x> [rx <N>] [tx <N>]	Sets the number of channels. **Notes**: • This also resets the RSS table to its default distribution, which is uniform across the cores on the NUMA (non-uniform memory access) node that is closer to the NIC. • For ConnectX®-4 cards, use ethtool -L eth<x> combined <N> to set both RX and TX channels.
ethtool -m|--dump-module-eeprom eth<x> [raw on|off] [hex on|off] [offset N] [length N]	Queries/decodes the cable module eeprom information.
ethtool -p|--identify DEVNAME	Enables visual identification of the port by LED blinking [TIME-IN-SECONDS].
ethtool -p|--identify eth<x> <LED duration>	Allows users to identify interface's physical port by turning the ports LED on for a number of seconds. **Note**: The limit for the LED duration is 65535 seconds.
ethtool -S eth<x>	Obtains additional device statistics.

<table>
<tr><th>Options</th><th>Description</th></tr>
<tr><td>ethtool -s eth<x> advertise <N> autoneg on</td><td>Changes the advertised link modes to requested link modes <N>
To check the link modes' hex values, run <code><man ethtool></code> and to check the supported link modes, run <code>ethtool eth<x></code>
For advertising new link modes, make sure to configure the entire bitmap as follows:
<table><tr><td>200GAUI-4 / 200GBASE-CR4/KR4</td><td>0x7c000000000000000</td></tr><tr><td>100GAUI-2 / 100GBASE-CR2 / KR2</td><td>0x3E00000000000000</td></tr><tr><td>CAUI-4 / 100GBASE-CR4 / KR4</td><td>0xF000000000</td></tr><tr><td>50GAUI-1 / LAUI-1/ 50GBASE-CR / KR</td><td>0x1F0000000000000</td></tr><tr><td>50GAUI-2 / LAUI-2/ 50GBASE-CR2/KR2</td><td>0x10C00000000</td></tr><tr><td>XLAUI-4/XLPPI-4 // 40G</td><td>0x7800000</td></tr><tr><td>25GAUI-1/ 25GBASE-CR / KR</td><td>0x380000000</td></tr><tr><td>XFI / XAUI-1 // 10G</td><td>0x7C0000181000</td></tr><tr><td>5GBASE-R</td><td>0x1000000000000</td></tr><tr><td>2.5GBASE-X / 2.5GMII</td><td>0x820000000000</td></tr><tr><td>1000BASE-X / SGMII</td><td>0x20000020020</td></tr></table>
Notes:
Both previous and new link modes configurations are supported, however, they must be run separately.
Any link mode configuration on Kernels below v5.1 and ConnectX-6 HCAs will result in the advertisement of the full capabilities.
<code><autoneg on></code> only sends a hint to the driver that the user wants to modify advertised link modes and not speed.</td></tr>
<tr><td>ethtool -s eth<x> msglvl [N]</td><td>Changes the current driver message level.</td></tr>
<tr><td>ethtool -s eth<x> speed <SPEED> autoneg off</td><td>Changes the link speed to requested <SPEED>. To check the supported speeds, run <code>ethtool eth<x></code>.
Note: <autoneg off> does not set autoneg OFF, it only hints the driver to set a specific speed.</td></tr>
<tr><td>ethtool -t eth<x></td><td>Performs a self-diagnostics test.</td></tr>
<tr><td>ethtool -T eth<x></td><td>Shows time stamping capabilities</td></tr>
<tr><td>ethtool -x eth<x></td><td>Retrieves the receive flow hash indirection table.</td></tr>
<tr><td>ethtool -X eth<x> equal a b c...</td><td>Sets the receive flow hash indirection table.
Note: The RSS table configuration is reset whenever the number of channels is modified (using ethtool -L command).</td></tr>
<tr><td>ethtool --show-fec eth<x></td><td>Queries current Forward Error Correction (FEC) encoding in case FEC is supported.
Note: An output of "baser" implies Firecode encoding.</td></tr>
<tr><td>ethtool --set-fec eth<x> encoding auto|off|rs|baser</td><td>Configures Forward Error Correction (FEC).
Note: 'baser' encoding applies to the Firecode encoding, and 'auto' regards the HCA's default.</td></tr>
<tr><td>ethtool -f|--flash <devname> FILE [N]</td><td>Flash firmware image on the device using the specified .mfa2 file (FILE). By default, the command flashes all the regions on the device unless a region number (N) is specified.</td></tr>
</table>

3.3.1.5 Checksum Offload

The following Receive IP/L4 Checksum Offload modes are supported.

- CHECKSUM_UNNECESSARY: When this mode is used, the driver indicates to the Linux Networking Stack that the hardware successfully validated the IP and L4 checksum so the Linux Networking Stack does not need to deal with IP/L4 Checksum validation.
- CHECKSUM_COMPLETE: When this mode is used, the driver still reports to the OS the calculated by hardware checksum value. This allows accelerating checksum validation in Linux Networking Stack, since it does not have to calculate the whole checksum including payload by itself.
- CHECKSUM_NONE: When this mode is used, the driver indicates to the Linux Networking Stack that it must calculate and validate the IP/L4 checksum.

3.3.1.6 Ignore Frame Check Sequence (FCS) Errors

Upon receiving packets, the packets go through a checksum validation process for the FCS field. If the validation fails, the received packets are dropped.
When FCS is enabled (disabled by default), the device does not validate the FCS field even if the field is invalid.
It is not recommended to enable FCS.
For further information on how to enable/disable FCS, please refer to ethtool option rx-fcs on/off.

3.3.1.7 RDMA over Converged Ethernet (RoCE)

Remote Direct Memory Access (RDMA) is the remote memory management capability that allows server-to-server data movement directly between application memory without any CPU involvement. RDMA over Converged Ethernet (RoCE) is a mechanism to provide this efficient data transfer with very low latencies on lossless Ethernet networks. With advances in data center convergence over reliable Ethernet, ConnectX® Ethernet adapter cards family with RoCE uses the proven and efficient RDMA transport to provide the platform for deploying RDMA technology in mainstream data center application at 10GigE and 40GigE link-speed. ConnectX® Ethernet adapter cards family with its hardware offload support takes advantage of this efficient RDMA transport (InfiniBand) services over Ethernet to deliver ultra-low latency for performance-critical and transaction-intensive applications such as financial, database, storage, and content delivery networks.
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 the 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. Therefore, APM (another type of High Availability and part of the InfiniBand protocol) is not supported

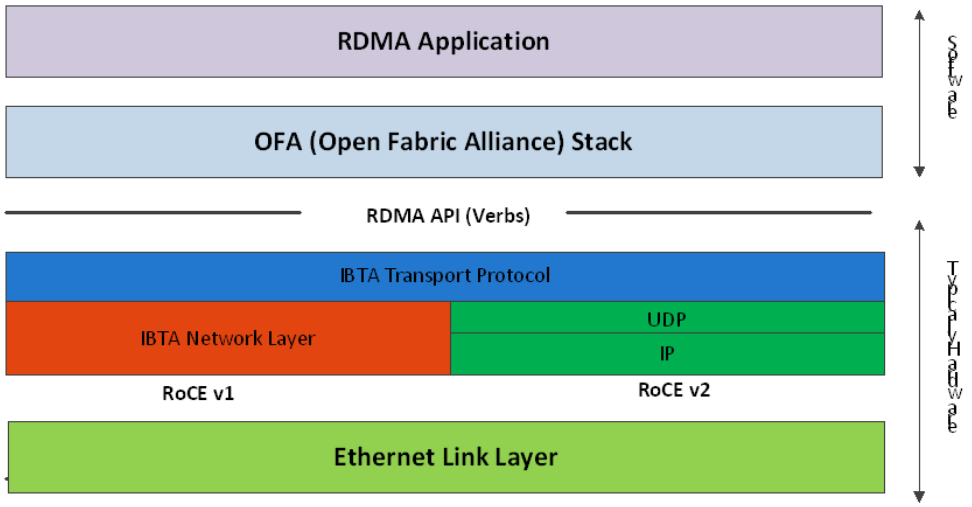
- Since the SM is not present, querying a path is impossible. Therefore, the path record structure must be filled with relevant values before establishing a connection. Hence, it is recommended working with RDMA-CM to establish a connection as it takes care of filling the path record structure
- VLAN tagged Ethernet frames carry a 3-bit priority field. The value of this field is derived from the IB SL field by taking the 3 least significant bits of the SL field
- RoCE traffic is not shown in the associated Ethernet device's counters since it is offloaded by the hardware and does not go through Ethernet network driver. RoCE traffic is counted in the same place where InfiniBand traffic is counted; /sys/class/infiniband/<device>/ports/<port number>/counters/

3.3.1.7.1 RoCE Modes

RoCE encapsulates IB transport in one of the following Ethernet packets:

- RoCEv1 - dedicated ether type (0x8915)
- RoCEv2 - UDP and dedicated UDP port (4791)

RoCEv1 and RoCEv2 Protocol Stack



3.3.1.7.1.1 RoCEv1

RoCE v1 protocol is defined as RDMA over Ethernet header (as shown in the figure above). It uses ethertype 0x8915 and can be used with or without the VLAN tag. The regular Ethernet MTU applies on the RoCE frame.

3.3.1.7.1.2 RoCEv2

A straightforward extension of the RoCE protocol enables traffic to operate in IP layer 3 environments. This capability is obtained via a simple modification of the RoCE packet format. Instead of the GRH used in RoCE, IP routable RoCE packets carry an IP header which allows traversal

of IP L3 Routers and a UDP header (RoCEv2 only) that serves as a stateless encapsulation layer for the RDMA Transport Protocol Packets over IP.

The proposed RoCEv2 packets use a well-known UDP destination port value that unequivocally distinguishes the datagram. Similar to other protocols that use UDP encapsulation, the UDP source port field is used to carry an opaque flow-identifier that allows network devices to implement packet forwarding optimizations (e.g. ECMP) while staying agnostic to the specifics of the protocol header format.
Furthermore, since this change exclusively affects the packet format on the wire, and due to the fact that with RDMA semantics packets are generated and consumed below the AP, applications can seamlessly operate over any form of RDMA service, in a completely transparent way.

Both RoCEv1 and RoCEv2 are supported by default; the driver associates all GID indexes to RoCEv1 and RoCEv2, thus, a single entry for each RoCE version.

For further information, please refer to Recommended Network Configuration Examples For RoCE Deployment Community post.

3.3.1.7.2 GID Table Population

GID table entries are created whenever an IP address is configured on one of the Ethernet devices of the NIC's ports. Each entry in the GID table for RoCE ports has the following fields:

- GID value
- GID type
- Network device

The GID table is occupied with two GIDs, both with the same GID value but with different types. The network device in an entry is the Ethernet device with the IP address that GID is associated with. The GID format can be of 2 types; IPv4 and IPv6. IPv4 GID is an IPv4-mapped IPv6 address, while IPv6 GID is the IPv6 address itself. Layer 3 header for packets associated with IPv4 GIDs will be IPv4 (for RoCEv2) and IPv6/GRH for packets associated with IPv6 GIDs and IPv4 GIDs for RoCEv1.

GID Table in sysfs

GID table is exposed to userspace via sysfs

- GID values can be read from:

```
/sys/class/infiniband/{device}/ports/{port}/gids/{index}
```

- GID type can be read from:

```
/sys/class/infiniband/{device}/ports/{port}/gid_attrs/types/{index}
```

- GID net_device can be read from:

```
/sys/class/infiniband/{device}/ports/{port}/gid_attrs/ndevs/{index}
```

3.3.1.7.2.1 Setting the RoCE Mode for a Queue Pair (QP)

Setting RoCE mode for devices that support two RoCE modes is different for RC/UC QPs (connected QP types) and UD QP.

To modify an RC/UC QP (connected QP) from INIT to RTR, an Address Vector (AV) must be given. The AV, among other attributes, should specify the index of the port's GID table for the source GID of the QP. The GID type in that index will be used to set the RoCE type of the QP.

3.3.1.7.2.2 Setting RoCE Mode of RDMA_CM Applications

RDMA_CM interface requires only the active side of the peer to pass the IP address of the passive side. The RDMA_CM decides upon the source GID to be used and obtains it from the GID table. Since more than one instance of the GID value is possible, the lookup should be also according to the GID type. The type to use for the lookup is defined as a global value of the RDMA_CM module. Changing the value of the GID type for the GID table lookups is done using the cma_roce_mode script.

- *To print the current RoCE mode for a device port:*

```
cma_roce_mode -d <dev> -p <port>
```

- *To set the RoCE mode for a device port:*

```
cma_roce_mode -d <dev> -p <port> -m <1|2>
```

3.3.1.7.2.3 GID Table Example

The following is an example of the GID table.

DEV	PORT	INDEX	GID	IPv4	Type	Netdev
mlx5_0	1	0	fe80:0000:0000:0000:ba59:9fff:fe1a:e3ea		v1	p4p1
mlx5_0	1	1	fe80:0000:0000:0000:ba59:9fff:fe1a:e3ea		v2	p4p1
mlx5_0	1	2	0000:0000:0000:0000:0000:ffff:0a0a:0a01	10.10.10.1	v1	p4p1
mlx5_0	1	3	0000:0000:0000:0000:0000:ffff:0a0a:0a01	10.10.10.1	v2	p4p1
mlx5_1	1	0	fe80:0000:0000:0000:ba59:9fff:fe1a:e3eb		v1	p4p2
mlx5_1	1	1	fe80:0000:0000:0000:ba59:9fff:fe1a:e3eb		v2	p4p2

where:

- Entries on port 1 index 0/1 are the default GIDs, one for each supported RoCE type
- Entries on port 1 index 2/3 belong to IP address 192.168.1.70 on eth1
- Entries on port 1 index 4/5 belong to IP address 193.168.1.70 on eth1.100
- Packets from a QP that is associated with these GID indexes will have a VLAN header (VID=100)
- Entries on port 1 index 6/7 are IPv6 GID. Packets from a QP that is associated with these GID indexes will have an IPv6 header

3.3.1.7.3 RoCE Lossless Ethernet Configuration

In order to function reliably, RoCE requires a form of flow control. While it is possible to use global flow control, this is normally undesirable, for performance reasons.
The normal and optimal way to use RoCE is to use Priority Flow Control (PFC). To use PFC, it must be enabled on all endpoints and switches in the flow path.

3.3.1.7.3.1 Configuring SwitchX® Based Switch System

To enable RoCE, the SwitchX should be configured as follows:

- Ports facing the host should be configured as access ports, and either use global pause or Port Control Protocol (PCP) for priority flow control
- Ports facing the network should be configured as trunk ports, and use Port Control Protocol (PCP) for priority flow control
- For further information on how to configure SwitchX, please refer to SwitchX User Manual

3.3.1.7.4 Installing and Loading the Driver

To install and load the driver:

1. Install MLNX_OFED (See Installation section for further details).
   RoCE is installed as part of mlx5 and other modules upon driver's installation.

   > The list of the modules that will be loaded automatically upon boot can be found in the configuration file /etc/infiniband/openib.conf.

2. Query for the device's information. Example:

```
ibv_devinfo MLNX_OFED_LINUX-5.0-2.1.8.0:
```

3. Display the existing MLNX_OFED version.

```
ofed_info -s
hca_id: mlx5_0
        transport:                      InfiniBand (0)
        fw_ver:                         16.28.0578
        node_guid:                      ec0d:9a03:0044:3764
        sys_image_guid:                 ec0d:9a03:0044:3764
        vendor_id:                      0x02c9
        vendor_part_id:                 4121
        hw_ver:                         0x0
        board_id:                       MT_0000000009
        phys_port_cnt:                  1
                port:   1
                        state:                  PORT_ACTIVE (4)
                        max_mtu:                4096 (5)
                        active_mtu:             1024 (3)
                        sm_lid:                 0
                        port_lid:               0
                        port_lmc:               0x00
                        link_layer:             Ethernet
```

Output Notes:

The port's state is: Ethernet is in PORT_ACTIVE state	The port state can also be obtained by running the following command: # cat /sys/class/infiniband/mlx5_0/ports/1/state 4: ACTIVE

link_layer parameter shows that port 1 is Ethernet	The link_layer can also be obtained by running the following command: `# cat /sys/class/infiniband/mlx5_0/ports/1/link_layer Ethernet`
The fw_ver parameter shows that the firmware version is 16.28.0578.	The firmware version can also be obtained by running the following command: `# cat /sys/class/infiniband/mlx5_0/fw_ver 16.28.0578`

3.3.1.7.4.1 Associating InfiniBand Ports to Ethernet Ports

The mlx5_ib driver holds a reference to the net device for getting notifications about the state of the port, as well as using the mlx5_core driver to resolve IP addresses to MAC that are required for address vector creation. However, RoCE traffic does not go through the mlx5_core driver; it is completely offloaded by the hardware.

```
# ibdev2netdev
mlx5_0 port 1 <===> eth2
#
```

3.3.1.7.4.2 Configuring an IP Address to the netdev Interface

To configure an IP address to the netdev interface:

1. Configure an IP address to the netdev interface on both sides of the link.

```
# ifconfig eth2 20.4.3.220
# ifconfig eth2
eth2      Link encap:Ethernet HWaddr 00:02:C9:08:E8:11
          inet addr:20.4.3.220 Bcast:20.255.255.255 Mask:255.0.0.0
          UP BROADCAST MULTICAST MTU:1500 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:1000
          RX bytes:0 (0.0 b) TX bytes:0 (0.0 b)
```

2. Make sure that ping is working.

```
ping 20.4.3.219
PING 20.4.3.219 (20.4.3.219) 56(84) bytes of data.
64 bytes from 20.4.3.219: icmp_seq=1 ttl=64 time=0.873 ms
64 bytes from 20.4.3.219: icmp_seq=2 ttl=64 time=0.198 ms
64 bytes from 20.4.3.219: icmp_seq=3 ttl=64 time=0.167 ms
20.4.3.219 ping statistics —
3 packets transmitted, 3 received, 0% packet loss, time 2000ms rtt min/avg/max/mdev = 0.167/0.412/0.873/
0.326 ms
```

3.3.1.7.4.3 Adding VLANs

To add VLANs:

1. Make sure that the 8021.q module is loaded.

```
modprobe 8021q
```

2. Add VLAN.

```
# vconfig add eth2 7
Added VLAN with VID == 7 to IF -:eth2:-
#
```

3. Configure an IP address.

```
ifconfig eth2.7 7.4.3.220
```

3.3.1.7.4.4 Defining Ethernet Priority (PCP in 802.1q Headers)

1. Define Ethernet priority on the server.

```
# ibv_rc_pingpong -g 1 -i 2 -l 4
local address: LID 0x0000, QPN 0x1c004f, PSN 0x9daf6c, GID fe80::202:c900:708:e799
remote address: LID 0x0000, QPN 0x1c004f, PSN 0xb0a49b, GID fe80::202:c900:708:e811
8192000 bytes in 0.01 seconds = 4840.89 Mbit/sec
1000 iters in 0.01 seconds = 13.54 usec/iter
```

2. Define Ethernet priority on the client.

```
# ibv_rc_pingpong -g 1 -i 2 -l 4 sw419
local address: LID 0x0000, QPN 0x1c004f, PSN 0xb0a49b, GID fe80::202:c900:708:e811
remote address: LID 0x0000, QPN 0x1c004f, PSN 0x9daf6c, GID fe80::202:c900:708:e799
8192000 bytes in 0.01 seconds = 4855.96 Mbit/sec
1000 iters in 0.01 seconds = 13.50 usec/iter
```

3.3.1.7.4.5 Using rdma_cm Tests

1. Use rdma_cm test on the server.

```
# ucmatose
cmatose: starting server
initiating data transfers
completing sends
receiving data transfers
data transfers complete
cmatose: disconnecting
disconnected
test complete
return status 0
#
```

2. Use rdma_cm test on the client.

```
# ucmatose -s 20.4.3.219
cmatose: starting client
cmatose: connecting
receiving data transfers
sending replies
data transfers complete
test complete
return status 0
#
```

This server-client run is without PCP or VLAN because the IP address used does not belong to a VLAN interface. If you specify a VLAN IP address, then the traffic should go over VLAN.

3.3.1.7.5 Type Of Service (ToS)

3.3.1.7.5.1 Overview

The TOS field for rdma_cm sockets can be set using the rdma_set_option() API, just as it is set for regular sockets. If a TOS is not set, the default value (0) is used. Within the rdma_cm kernel driver, the TOS field is converted into an SL field. The conversion formula is as follows:

- SL = TOS >> 5 (e.g., take the 3 most significant bits of the TOS field)

In the hardware driver, the SL field is converted into PCP by the following formula:

- PCP = SL & 7 (take the 3 least significant bits of the TOS field)

SL affects the PCP only when the traffic goes over tagged VLAN frames.

3.3.1.7.5.2 DSCP

A new entry has been added to the RDMA-CM configfs that allows users to select default TOS for RDMA-CM QPs. This is useful for users that want to control the TOS field without changing their code. Other applications that set the TOS explicitly using the rdma_set_option API will continue to work as expected to override the configfs value.
For further information about DSCP marking, refer to HowTo Set Egress ToS/DSCP on RDMA CM QPs Community post.

3.3.1.7.6 RoCE LAG

RoCE LAG is a feature meant for mimicking Ethernet bonding for IB devices and is available for dual port cards only.
This feature is supported on kernel versions 4.9 and above.

RoCE LAG mode is entered when both Ethernet interfaces are configured as a bond in one of the following modes:

- active-backup (mode 1)
- balance-xor (mode 2)
- 802.3ad (LACP) (mode 4)

Any change of bonding configuration that negates one of the above rules (i.e, bonding mode is not 1, 2 or 4, or both Ethernet interfaces that belong to the same card are not the only slaves of the bond interface), will result in exiting RoCE LAG mode and the return to normal IB device per port configuration.
Once RoCE LAG is enabled, instead of having two IB devices; mlx5_0 and mlx5_1, there will be one device named mlx5_bond_0.
For information on how to configure RoCE LAG, refer to HowTo Configure RoCE over LAG (ConnectX-4/ConnectX-5/ConnectX-6) Community post.

3.3.1.7.7 Disabling RoCE

By default, RoCE is enabled on all mlx5 devices. When RoCE is enabled, all traffic to UDP port 4791 is treated as RoCE traffic by the device.

In case you are only interested in Ethernet (no RDMA) and wish to enable forwarding of traffic to this port, you can disable RoCE through sysfs:

```
echo <0|1> > /sys/devices/{pci-bus-address}/roce_enable
```

Once RoCE is disabled, only Ethernet traffic will be supported. Therefore, there will be no GID tables and only Raw Ethernet QPs will be supported.

The current RoCE state can be queried by sysfs:

```
cat /sys/devices/{pci-bus-address}/roce_enable
```

3.3.1.7.8 Enabling/Disabling RoCE on VMs via VFs

By default, when configuring VFs on the hypervisor, all VFs will be enabled with RoCE. This means they require more OS memory (from the VM). In case you are only interested in Ethernet (no RDMA) on the VM, and you wish to save the VM memory, you can disable RoCE on the VF from the hypervisor. In addition, by disabling RoCE, a VM can have the capability of utilizing the RoCE UDP port (4791) for standard UDP traffic.

For details on how to enable/disable RoCE on a VF, refer to HowTo Enable/Disable RoCE on VMs via VFs Community post.

3.3.1.7.9 Force DSCP

This feature enables setting a global traffic_class value for all RC QPs, or setting a specific traffic class based on several matching criteria.

Usage

- To set a single global traffic class to be applied to all QPs, write the desired global traffic_class value to /sys/class/infiniband/<dev>/tc/<port>/traffic_class.
  Note the following:
  - Negative values indicate that the feature is disabled. traffic_class value can be set using ibv_modify_qp()
  - Valid values range between 0 - 255

The ToS field is 8 bits, while the DSCP field is 6 bits. To set a DSCP value of X, you need to multiply this value by 4 (SHIFT 2). For example, to set DSCP value of 24, set the ToS bit to 96 (24x4=96).

- To set multiple traffic class values based on source and/or destination IPs, write the desired rule to /sys/class/infiniband/<dev>/tc/<port>/traffic_class. For example:

  ```
  echo "tclass=16,src_ip=1.1.1.2,dst_ip=1.1.1.0/24" > /sys/class/infiniband/mlx5_0/tc/1/traffic_class
  ```

  Note: Adding "tclass" prefix to tclass value is optional.

In the example above, traffic class 16 will be set to any QP with source IP 1.1.1.2 and destination IP 1.1.1.0/24.
Note that when setting a specific traffic class, the following rule precedence will apply:

- If a global traffic class value is set, it will be applied to all QPs

- If no global traffic class value is set, and there is a rule with matching source and destination IPs applicable to at least one QP, it will be applied
- Rules only with matching source and/or destination IPs have no defined precedence over other rules with matching source and/or destination IPs

Notes:

- A mask can be provided when using destination IPv4 addresses
- The rule precedence is not affected by the order in which rules are inserted
- Overlapping rules are entirely up to the administrator.
- "tclass=-1" will remove the rule from the database

3.3.1.7.10 Force Time to Live (TTL)

This feature enables setting a global TTL value for all RC QPs.
Write the desired TTL value to /sys/class/infiniband/<dev>/tc/<port>/ttl. Valid values range between 0 - 255

3.3.1.8 Flow Control

3.3.1.8.1 Priority Flow Control (PFC)

Priority Flow Control (PFC) IEEE 802.1Qbb applies pause functionality to specific classes of traffic on the Ethernet link. For example, PFC can provide lossless service for the RoCE traffic and best-effort service for the standard Ethernet traffic. PFC can provide different levels of service to specific classes of Ethernet traffic (using IEEE 802.1p traffic classes).

3.3.1.8.1.1 Configuring PFC on ConnectX-4 and above

1. Enable PFC on the desired priority:

   ```
   mlnx_qos -i <ethX> --pfc <0/1>,<0/1>,<0/1>,<0/1>,<0/1>,<0/1>,<0/1>,<0/1>
   ```

   Example (Priority=4):

   ```
   mlnx_qos -i eth1 --pfc 0,0,0,0,1,0,0,0
   ```

2. Create a VLAN interface:

   ```
   vconfig add <ethX> <VLAN_ID>
   ```

   Example (VLAN_ID=5):

   ```
   vconfig add eth1 5
   ```

3. Set egress mapping:
   a. For Ethernet traffic:

      ```
      vconfig set_egress_map <vlan_einterface> <skprio> <up>
      ```

      Example (skprio=3, up=5):

```
vconfig set_egress_map eth1.5 3 5
```

4. Create 8 Traffic Classes (TCs):

```
tc_wrap.py -i <interface>
```

5. Enable PFC on the switch.
   For information on how to enable PFC on your respective switch, please refer to Switch FC/PFC Configuration sections in the RDMA/RoCE Solutions Community page.

3.3.1.8.1.2 PFC Configuration Using LLDP DCBX

PFC Configuration on Hosts

PFC Auto-Configuration Using LLDP Tool in the OS

1. Start lldpad daemon on host.

```
lldpad -d Or
service lldpad start
```

2. Send lldpad packets to the switch.

```
lldptool set-lldp -i <ethX>  adminStatus=rxtx  ;
lldptool -T -i <ethX>   -V  sysName enableTx=yes;
lldptool -T -i <ethX>   -V  portDesc enableTx=yes ;
lldptool -T -i <ethX>   -V  sysDesc enableTx=yes
lldptool -T -i <ethX>   -V sysCap enableTx=yess
lldptool -T -i <ethX>   -V mngAddr enableTx=yess
lldptool -T -i <ethX>   -V PFC enableTx=yes;
lldptool -T -I <ethX>   -V CEE-DCBX enableTx=yes;
```

3. Set the PFC parameters.

   - For the CEE protocol, use dcbtool:

```
dcbtool sc <ethX> pfc pfcup:<xxxxxxxx>
```

   Example:

```
dcbtool sc eth6 pfc pfcup:01110001
```

   where:

[pfcup:xxxxxxx]	Enables/disables priority flow control. From left to right (priorities 0-7) - x can be equal to either 0 or 1. 1 indicates that the priority is configured to transmit priority pause.

   - For IEEE protocol, use lldptool:

```
lldptool -T -i <ethX> -V PFC enabled=x,x,x,x,x,x,x,x
```

   Example:

```
lldptool -T -i eth2 -V PFC enabled=1,2,4
```

   where:

enabled	Displays or sets the priorities with PFC enabled. The set attribute takes a comma-separated list of priorities to enable, or the string none to disable all priorities.

PFC Auto-Configuration Using LLDP in the Firmware (for mlx5 driver)

There are two ways to configure PFC and ETS on the server:

1. Local Configuration - Configuring each server manually.
2. Remote Configuration - Configuring PFC and ETS on the switch, after which the switch will pass the configuration to the server using LLDP DCBX TLVs.
   There are two ways to implement the remote configuration using mlx5 driver:
   a. Configuring the adapter firmware to enable DCBX.
   b. Configuring the host to enable DCBX.

For further information on how to auto-configure PFC using LLDP in the firmware, refer to the HowTo Auto-Config PFC and ETS on ConnectX-4 via LLDP DCBX Community post.

PFC Configuration on Switches

1. In order to enable DCBX, LLDP should first be enabled:

```
switch (config) # lldp
show lldp interfaces ethernet remote
```

2. Add DCBX to the list of supported TLVs per required interface.
   For IEEE DCBX:

```
switch (config) # interface 1/1
switch (config interface ethernet 1/1) # lldp tlv-select dcbx
```

   For CEE DCBX:

```
switch (config) # interface 1/1
switch (config interface ethernet 1/1) # lldp tlv-select dcbx-cee
```

3. [Optional] Application Priority can be configured on the switch, with the required ethertype and priority. For example, IP packet, priority 1:

```
switch (config) # dcb application-priority 0x8100 1
```

4. Make sure PFC is enabled on the host (for enabling PFC on the host, refer to PFC Configuration on Hosts section above). Once it is enabled, it will be passed in the LLDP TLVs.
5. Enable PFC with the desired priority on the Ethernet port.

```
dcb priority-flow-control enable force
dcb priority-flow-control priority <priority> enable
interface ethernet <port> dcb priority-flow-control mode on force
```

   Example - Enabling PFC with priority 3 on port 1/1:

```
dcb priority-flow-control enable force
dcb priority-flow-control priority 3 enable
interface ethernet 1/1 dcb priority-flow-control mode on force
```

Priority Counters

Several ingress and egress counters per priority are supported. Run ethtool -S to get the full list of port counters.

ConnectX-4 Counters

- Rx and Tx Counters:
  - Packets
  - Bytes
  - Octets
  - Frames
  - Pause
  - Pause frames
  - Pause Duration
  - Pause Transition

ConnectX-4 Example

```
# ethtool -S eth35 | grep prio4
     prio4_rx_octets: 62147780800
     prio4_rx_frames: 14885696
     prio4_tx_octets: 0
     prio4_tx_frames: 0
     prio4_rx_pause: 0
     prio4_rx_pause_duration: 0
     prio4_tx_pause: 26832
     prio4_tx_pause_duration: 14508
     prio4_rx_pause_transition: 0
```

Note: The Pause counters in ConnectX-4 are visible via ethtool only for priorities on which PFC is enabled.

3.3.1.8.1.3 PFC Storm Prevention

PFC storm prevention enables toggling between default and auto modes.
The stall prevention timeout is configured to 8 seconds by default. Auto mode sets the stall prevention timeout to be 100 msec.
The feature can be controlled using sysfs in the following directory: /sys/class/net/eth*/settings/pfc_stall_prevention

- To query the PFC stall prevention mode:

  ```
  cat /sys/class/net/eth*/settings/pfc_stall_prevention
  ```

  Example

  ```
  $ cat /sys/class/net/ens6/settings/pfc_stall_prevention
  default
  ```

- To configure the PFC stall prevention mode:

  ```
  Echo "auto"/"default" > /sys/class/net/eth*/settings/pfc_stall_prevention
  ```

The following two counters were added to the ethtool -S:

- `tx_Pause_storm_warning_events` - when the device is stalled for a period longer than a pre-configured watermark, the counter increases, allowing the debug utility an insight into current device status.
- `tx_pause_storm_error_events` - when the device is stalled for a period longer than a pre-configured timeout, the pause transmission is disabled, and the counter increase.

3.3.1.8.2 Dropless Receive Queue (RQ)

Dropless RQ feature enables the driver to notify the FW when SW receive queues are overloaded. This scenario takes place when the handling of SW receive queue is slower than the handling of the HW receive queues.
When this feature is enabled, a packet that is received while the receive queue is full will not be immediately dropped. The FW will accumulate these packets assuming posting of new WQEs will resume shortly. If received WQEs are not posted after a certain period of time, out_of_buffer counter will increase, indicating that the packet has been dropped.
This feature is disabled by default. In order to activate it, ensure that Flow Control feature is also enabled.

To enable the feature, run:

```
ethtool --set-priv-flags ens6 dropless_rq on
```

To get the feature state, run:

```
ethtool --show-priv-flags DEVNAME
```

Output example:

```
Private flags for DEVNAME:
rx_cqe_moder   : on
rx_cqe_compress: off
sniffer        : off
dropless_rq    : off
hw_lro         : off
```

To disable the feature, run:

```
ethtool --set-priv-flags ens6 dropless_rq off
```

3.3.1.9 Explicit Congestion Notification (ECN)

ECN is an extension to the IP protocol. It allows reliable communication by notifying all ends of communication when congestion occurs. This is done without dropping packets.
Please note that this feature requires all nodes in the path (nodes, routers etc) between the communicating nodes to support ECN to ensure reliable communication. ECN is marked as 2 bits in the traffic control IP header. This ECN implementation refers to RoCE v2.

3.3.1.9.1 Enabling ECN

To enable ECN on the hosts:

1. Enable ECN in sysfs.

```
/sys/class/net/<interface>/<protocol>/ecn_<protocol>_enable =1
```

2. Query the attribute.

```
cat /sys/class/net/<interface>/ecn/<protocol>/params/<requested attribute>
```

3. Modify the attribute.

```
echo <value> /sys/class/net/<interface>/ecn/<protocol>/params/<requested attribute>
```

ECN supports the following algorithms:

- r_roce_ecn_rp - Reaction point
- r_roce_ecn_np - Notification point

Each algorithm has a set of relevant parameters and statistics, which are defined per device, per port, per priority.

To query whether ECN is enabled per Priority X:

```
cat /sys/class/net/<interface>/ecn/<protocol>/enable/X
```

To read ECN configurable parameters:

```
cat /sys/class/net/<interface>/ecn/<protocol>/requested attributes
```

➢ *To enable ECN for each priority per protocol:*

```
echo 1 > /sys/class/net/<interface>/ecn/<protocol>/enable/X
```

➢ *To modify ECN configurable parameters:*

```
echo <value> > /sys/class/net/<interface>/ecn/<protocol>/requested attributes
```

where:

- X: priority {0..7}
- protocol: roce_rp / roce_np
- requested attributes: Next Slide for each protocol.

3.3.1.10 RSS Support

3.3.1.10.1 RSS Hash Function

The device has the ability to use XOR as the RSS distribution function, instead of the default Toplitz function.
The XOR function can be better distributed among driver's receive queues in a small number of streams, where it distributes each TCP/UDP stream to a different queue. provides the following option to change the working RSS hash function from Toplitz to XOR, and vice-versa:

Through sysfs, located at: /sys/class/net/eth*/settings/hfunc.

➢ *To query the operational and supported hash functions:*

```
cat /sys/class/net/eth*/settings/hfunc
```

Example:

```
cat /sys/class/net/eth2/settings/hfunc
Operational hfunc: toeplitz
Supported hfuncs: xor toeplitz
```

➢ *To change the operational hash function:*

```
echo xor > /sys/class/net/eth*/settings/hfunc
```

3.3.1.10.1.1 RSS Verbs Support

Receive Side Scaling (RSS) technology allows spreading incoming traffic between different receive descriptor queues. Assigning each queue to different CPU cores allows better load balancing of the incoming traffic and improves performance.
This technology was extended to user space by the verbs layer and can be used for RAW ETH QP.

3.3.1.10.1.2 RSS Flow Steering

Steering rules classify incoming packets and deliver a specific traffic type (e.g. TCP/UDP, IP only) or a specific flow to "RX Hash" QP. "RX Hash" QP is responsible for spreading the traffic it handles between the Receive Work Queues using RX hash and Indirection Table. The Receive Work Queue can point to different CQs that can be associated with different CPU cores.

3.3.1.10.1.3 Verbs

The below verbs should be used to achieve this task in both control and data path. Details per verb should be referenced from its man page.

- ibv_create_wq, ibv_modify_wq, ibv_destory_wq
- ibv_create_rwq_ind_table, ibv_destroy_rwq_ind_table
- ibv_create_qp_ex with specific RX configuration to create the "RX hash" QP

3.3.1.11 Time-Stamping

3.3.1.11.1 Time-Stamping Service

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.

3.3.1.11.1.1 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 into 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 dev*ice:*

Admin privileged user can enable/disable time stamping through calling ioctl (sock, SIOCSH-WTSTAMP, &ifreq) with the 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.
```

3.3.1.11.1.2 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 prepended down to and including the link layer, the scm_time-stamping 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.

⚠ On ConnectX-4 and above adapter cards, when time-stamping is enabled, RX CQE compression is disabled (features are mutually exclusive).

3.3.1.11.1.3 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: 1
Hardware Transmit Timestamp Modes:
off                              (HWTSTAMP_TX_OFF)
on                               (HWTSTAMP_TX_ON)

Hardware Receive Filter Modes:
```

```
none                                (HWTSTAMP_FILTER_NONE)
all                                 (HWTSTAMP_FILTER_ALL)
```

For more details on PTP Hardware Clock, please refer to: https://www.kernel.org/doc/Documentation/ptp/ptp.txt

3.3.1.11.1.4 Steering PTP Traffic to Single RX Ring

As a result of Receive Side Steering (RSS) PTP traffic coming to UDP ports 319 and 320, it may reach the user space application in an out of order manner. In order to prevent this, PTP traffic needs to be steered to single RX ring using ethtool.

Example:

```
# ethtool -u ens7
8 RX rings available
Total 0 rules
# ethtool -U ens7 flow-type udp4 dst-port 319 action 0 loc 1
# ethtool -U ens7 flow-type udp4 dst-port 320 action 0 loc 0
# ethtool -u ens7
8 RX rings available
Total 2 rules
Filter: 0
Rule Type: UDP over IPv4
Src IP addr: 0.0.0.0 mask: 255.255.255.255
Dest IP addr: 0.0.0.0 mask: 255.255.255.255
TOS: 0x0 mask: 0xff
Src port: 0 mask: 0xffff
Dest port: 320 mask: 0x0
Action: Direct to queue 0
Filter: 1
Rule Type: UDP over IPv4
Src IP addr: 0.0.0.0 mask: 255.255.255.255
Dest IP addr: 0.0.0.0 mask: 255.255.255.255
TOS: 0x0 mask: 0xff
Src port: 0 mask: 0xffff
Dest port: 319 mask: 0x0
Action: Direct to queue 0
```

3.3.1.11.1.5 Tx Port Time-Stamping

Transmitted packet time-stamping accuracy can be improved when using a timestamp generated at the port level instead of a timestamp generated upon CQE creation. Tx port time-stamping better reflects the actual time of a packet's transmission.

Normal Send queues (SQs) are open with CQE time-stamp support. When this feature is enabled, the driver is expected to open extra Tx port time-stamped SQ per traffic class (TC).

The stream must meet the following conditions in order to be transmitted through a Tx port time-stamped SQ.

1. SKBTX_HW_TSTAMP flag was set at tx_flag (SO_TIMESTAMPING was set via setsockopt() or similarly)
2. Packet type is:
    a. Non-IP, with EtherType of PTP over IEEE 802.3 (0x88f7)
       or
    b. UDP over IPv4/IPv6

This feature is disabled by default in order to avoid extra SQ memory allocations. The feature can be enabled or disabled using the following command.

```
ethtool --set-priv-flags <ifs-name> tx_port_ts on / off
```

3.3.1.11.1.6 PTP Cyc2time Hardware Translation Offload

This feature is supported on ConnectX-6 Dx and above adapter cards only.

Overview

Device timestamp can be in one of two modes: real time or free running internal time.
In free running internal time mode, the device clock is not editable in any way. Driver and/or user space must adjust it to the real-time nanosecond values.
In real time mode, the hardware clock device can be adjusted and can provide timestamps which are already translated into real-time nanoseconds.

Both modes are global per device. Once a mode is set, all clock-related features (such as PPS, CQE TS, PCIe bar, etc) will work with the chosen clock mode only.

Free running internal time is the default mode configured in the hardware. The driver will modify the hardware real time clock based on PTP daemon clock adjustments.

Only physical functions are allowed to modify the hardware real-time clock, so PTP daemon adjustments from VFs will be treated as NOP. In case more than one physical function tries to modify the hardware real-time clock, the device will select one of the functions as its designated clock provider. All other input will also be treated as a NOP. The designated clock provide can be replaced by the device if no new adjustments have been received from the current provider after some period.

Timestamp Format

CQE hardware timestamp format for ConnectX-6 Dx and ConnectX-6 Lx NICs is 64 bit, as follows.

{32bit sec, 32 bit nsec}

Configuration

In order to enable the feature, set REAL_TIME_CLOCK_ENABLE in NV_CONFIG via mlxconfig and restart the driver.

Limitations

- Administrator must restart the driver and perform a FW reset for the configuration to take effect. Otherwise, mismatch between HW and driver timestamp mode might occur.
- Once real time mode is activated on a given device (see configuration section), version 5.3 or newer must run on all device functions. Any older driver running on a device function at this configuration will fail to open any traffic queues (RDMA or ETH), hence becoming dysfunctional.
- In real time mode, all device functions must be PTP-synchronized by a single clock domain—do not use multiple GMs for different functions on the same device.
- Regarding hardware clock ownership, the hardware is configured only from a single elected function; other function settings are ignored by the device. There is no indication as to which function is the hardware-clock's owner. After an internal timeout without modifying the

hardware clock, a function losses the hardware-clock's ownership and is open to be grasped by any of the functions.
- All PFs/VFs within the same device must sync to the same 1588 master clock. If multiple masters are used, the device will use a single elected function. This might lead to wrong clock representation by device, wrong 1588 TLVs and hiccups on replacement of elected function.
- This feature is supported on ConnectX-6 Dx and above adapter cards only.

3.3.1.11.2 RoCE Time-Stamping

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 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.

3.3.1.11.2.1 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 ibv_query_device_ex.

For further information, please see ibv_query_device_ex manual page.

3.3.1.11.2.2 Creating a 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.

For further information, please see ibv_create_cq_ex manual page.

> ⚠ Time Stamping in not available when CQE zipping is used.

3.3.1.11.2.3 Querying the Hardware Time

Querying the hardware for time is done via the ibv_query_rt_values_ex verb. For example:

For further information, please see ibv_query_rt_values_ex manual page.

3.3.1.11.3 One Pulse Per Second (1PPS)

1PPS is a time synchronization feature that allows the adapter to be able to send or receive 1 pulse per second on a dedicated pin on the adapter card using an SMA connector (SubMiniature version A). Only one pin is supported and could be configured as 1PPS in or 1PPS out.
For further information, refer to HowTo Test 1PPS on NVIDIA Adapters Community post.

3.3.1.12 Flow Steering

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 different terminology from the flow attribute (ibv_flow_attr), defined by a combination of specifications (struct ibv_flow_spec_*).

3.3.1.12.1 Flow Steering Support

All flow steering features are enabled in the supported adapter cards.

3.3.1.12.2 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 a higher priority.
In addition to the domain, there is a 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 to 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_eth
      struct ibv_flow_spec_ipv4
      struct ibv_flow_spec_tcp_udp
      struct ibv_flow_spec_ipv6
      ```

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

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

    Input parameters:
    `ibv_destroy_flow` requires struct `ibv_low` 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.

3.3.1.12.3 Ethtool

Ethtool domain is used to attach an RX ring, specifically its QP to a specified flow. Please refer to the most recent ethtool man page 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 flow-type ether dst 00:11:22:33:44:55 vlan 45 m 0xf000 loc 5 action 2
  All packets that contain the above destination MAC address and specific VLAN are steered into ring 2. Please pay attention to the VLAN's mask 0xf000. It is required in order to add such a rule.
- 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).

3.3.1.12.4 Accelerated Receive Flow Steering (aRFS)

Receive Flow Steering (RFS) and Accelerated Receive Flow Steering (aRFS) are kernel features currently available in most distributions. For RFS, packets are forwarded based on the location of the application consuming the packet. aRFS boosts the speed of RFS by adding support for the hardware. By usingaRFS(unlike RFS), the packets are directed to a CPU that is local to the thread running the application.

aRFSis 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 aRFS mechanism to use the flow steering infrastructure to support the aRFS logic by implementing the ndo_rx_flow_steer, which, in turn, calls the underlying flow steering mechanism with the aRFS domain.

To configure RFS:

Configure the RFS flow table entries (globally and per core).
Note: The functionality remains disabled until explicitly configured (by default it is 0).

- The number of entries in the global flow table is set as follows:

  /proc/sys/net/core/rps_sock_flow_entries

- The number of entries in the per-queue flow table are set as follows:

/sys/class/net/<dev>/queues/rx-<n>/rps_flow_cnt

Example:

```
# echo 32768 > /proc/sys/net/core/rps_sock_flow_entries
# NUM_CHANNELS=`ethtool -l ens6 | grep "Combined:" | tail -1 | awk '{print $2}'`
# for f in `seq 0 $((NUM_CHANNELS-1))`; do echo 32768 > /sys/class/net/ens6/queues/rx-$f/rps_flow_cnt; done
```

To Configure aRFS:

The aRFS feature requires explicit configuration in order to enable it. Enabling the aRFS requires enabling the 'ntuple' flag via the ethtool.
For example, to enable ntuple for eth0, run:

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

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

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

3.3.1.12.5 Flow Steering Dump Tool

The mlx_fs_dump is a python tool that prints the steering rules in a readable manner. Python v2.7 or above, as well as pip, anytree and termcolor libraries are required to be installed on the host.

Running example:

```
./ofed_scripts/utils/mlx_fs_dump -d /dev/mst/mt4115_pciconf0
FT: 9 (level: 0x18, type: NIC_RX)
+-- FG: 0x15 (MISC)
    |-- FTE: 0x0 (FWD) to (TIR:0x7e) out.ethtype:IPv4 out.ip_prot:UDP out.udp_dport:0x140
    +-- FTE: 0x1 (FWD) to (TIR:0x7e) out.ethtype:IPv4 out.ip_prot:UDP out.udp_dport:0x13f
...
```

For further information on the mlx_fs_dump tool, please refer to mlx_fs_dump Community post.

3.3.1.13 Wake-on-LAN (WoL)

Wake-on-LAN (WoL) is a technology that allows a network professional to remotely power on a computer or to wake it up from sleep mode.

- To enable WoL:

```
ethtool -s <interface> wol g
```

- To get WoL:

```
ethtool <interface> | grep Wake-on Wake-on: g
```

Where:
"`g`" is the magic packet activity.

3.3.1.14 Hardware Accelerated 802.1ad VLAN (Q-in-Q Tunneling)

Q-in-Q tunneling allows the user to create a Layer 2 Ethernet connection between two servers. The user can segregate a different VLAN traffic on a link or bundle different VLANs into a single VLAN. Q-in-Q tunneling adds a service VLAN tag before the user's 802.1Q VLAN tags.
For Q-in-Q support in virtualized environments (SR-IOV), please refer to "Q-in-Q Encapsulation per VF in Linux (VST)".

To enable device support for accelerated 802.1ad VLAN:

1. Turn on the new ethtool private flag "phv-bit" (disabled by default).

```
$ ethtool --set-priv-flags eth1 phv-bit on
```

Enabling this flag sets the phv_en port capability.

2. Change the interface device features by turning on the ethtool device feature "tx-vlan- stag-hw-insert" (disabled by default).

```
$ ethtool -K eth1 tx-vlan-stag-hw-insert on
```

Once the private flag and the ethtool device feature are set, the device will be ready for 802.1ad VLAN acceleration.

The "phv-bit" private flag setting is available for the Physical Function (PF) only. The Virtual Function (VF) can use the VLAN acceleration by setting the "tx-vlan-stag-hw-insert" parameter only if the private flag "phv-bit" is enabled by the PF. If the PF enables/disables the "phv-bit" flag after the VF driver is up, the configuration will take place only after the VF driver is restarted.

3.3.1.15 VLAN Stripping in Linux Verbs

This capability is now accessible from userspace using the verbs.

VLAN stripping adds access to the device's ability to offload the Customer VLAN (cVLAN) header stripping from an incoming packet, thus achieving acceleration of VLAN handing in receive flow.

It is configured per WQ option. You can either enable it upon creation or modify it later using the appropriate verbs (`ibv_create_wq` / `ibv_modify_wq`).

3.3.1.16 Offloaded Traffic Sniffer

> To be able to activate this feature, make sure libpcap library v1.9 or above is installed on your setup.
>
> To download libpcap, please visit https://www.tcpdump.org/.

Offloaded Traffic Sniffer allows bypass kernel traffic (such as RoCE, VMA, and DPDK) to be captured by existing packet analyzer, such as tcpdump.

To capture the interface's bypass kernel traffic, run tcpdump on the RDMA device.

For examples on how to dump RDMA traffic using the Inbox tcpdump tool for ConnectX-4 adapter cards and above, click here.

> Note that enabling Offloaded Traffic Sniffer can cause bypass kernel traffic speed degradation.

> In case you do not wish to install libpcap on your setup, you can use docker to run the tcpdump. For further information, please see https://hub.docker.com/r/mellanox/tcpdump-rdma.

3.3.1.17 Dump Configuration

This feature helps dumping driver and firmware configuration using ethtool. It creates a backup of the configuration files into a specified dump file.

3.3.1.17.1 Dump Parameters (Bitmap Flag)

The following bitmap parameters are used to set the type of dump:

Bitmap Parameters

Value	Description
1	MST dump
2	Ring dump (Software context information for SQs, EQs, RQs, CQs)
3	MST dump + Ring dump (1+2)
4	Clear this parameter

3.3.1.17.2 Configuration

In order to configure this feature, follow the steps below:

1. Set the dump bitmap parameter by running -W (uppercase) with the desired bitmap parameter value (see Bitmap Parameters table above). In the following example, the bitmap parameter value is 3.

```
ethtool -W ens1f0 3
```

2. Dump the file by running -w (lowercase) with the desired configuration file name.

```
ethtool -w ens1f0 data /tmp/dump.bin
```

3. [Optional] To get the bitmap parameter value, version and size of the dump, run the command above without the file name.

```
ethtool -w ens1f0
flag: 3, version: 1, length: 4312
```

4. To open the dump file, run:

```
mlnx_dump_parser -f /tmp/dump.bin -m mst_dump_demo.txt -r ring_dump_demo.txt
Version: 1 Flag: 3 Number of blocks: 123 Length 327584
MCION module number: 0 status: | present |
DRIVER VERSION: 1-23 (03 Mar 2015)
DEVICE NAME 0000:81:00.0:ens1f0
Parsing Complete!
```

where:

-f	For the file to be parsed (the file that was just created)
-m	For the mst dump file
-r	For the ring dump file

For further information, refer to HowTo Dump Driver Configuration (via ethtool) Community post.
Output:

```
# mlnx_dump_parser -f /tmp/dump.bin -m mst_dump_demo.txt -r ring_dump_demo.txt
Version: 1 Flag: 3 Number of blocks: 123 Length 327584
MCION module number: 0 status: | present |
DRIVER VERSION: 1-23 (03 Mar 2015)
DEVICE NAME 0000:81:00.0:ens1f0
Parsing Complete!
```

5. Open the files.
    a. The MST dump file will look as follows. In order to analyze it, contact NVIDIA Support.

```
cat mst_dump_demo.txt
0x00000000 0x01002000
0x00000004 0x00000000
0x00000008 0x00000000
0x0000000c 0x00000000
0x00000010 0x00000000
0x00000014 0x00000000
0x00000018 0x00000000
...
```

    b. The Ring dump file can help developers debug ring-related issues, and it looks as follows:

```
# cat ring_dump_demo.txt
SQ TYPE: 3, WQN: 102, PI: 0, CI: 0, STRIDE: 6, SIZE: 1024...
SQ TYPE: 3, WQN: 102, PI: 0, CI: 0, STRIDE: 6, SIZE: 1024, WQE_NUM: 65536, GROUP_IP: 0
CQ TYPE: 5, WQN: 20, PI: 0, CI: 0, STRIDE: 6, SIZE: 1024, WQE_NUM: 1024, GROUP_IP: 0
```

```
RQ TYPE: 4, WQN: 103, PI: 15, CI: 0, STRIDE: 5, SIZE: 16, WQE_NUM: 512, GROUP_IP: 0
CQ TYPE: 5, WQN: 21, PI: 0, CI: 0, STRIDE: 6, SIZE: 16384, WQE_NUM: 16384, GROUP_IP: 0
EQ TYPE: 6, CI: 1, SIZE: 0, IRQN: 109, EQN: 19, NENT: 2048, MASK: 0, INDEX: 0, GROUP_ID: 0
SQ TYPE: 3, WQN: 106, PI: 0, CI: 0, STRIDE: 6, SIZE: 1024, WQE_NUM: 65536, GROUP_IP: 1
CQ TYPE: 5, WQN: 23, PI: 0, CI: 0, STRIDE: 6, SIZE: 1024, WQE_NUM: 1024, GROUP_IP: 1
RQ TYPE: 4, WQN: 107, PI: 15, CI: 0, STRIDE: 5, SIZE: 16, WQE_NUM: 512, GROUP_IP: 1
CQ TYPE: 5, WQN: 24, PI: 0, CI: 0, STRIDE: 6, SIZE: 16384, WQE_NUM: 16384, GROUP_IP: 1
EQ TYPE: 6, CI: 1, SIZE: 0, IRQN: 110, EQN: 20, NENT: 2048, MASK: 0, INDEX: 1, GROUP_ID: 1
SQ TYPE: 3, WQN: 110, PI: 0, CI: 0, STRIDE: 6, SIZE: 1024, WQE_NUM: 65536, GROUP_IP: 2
CQ TYPE: 5, WQN: 26, PI: 0, CI: 0, STRIDE: 6, SIZE: 1024, WQE_NUM: 1024, GROUP_IP: 2
RQ TYPE: 4, WQN: 111, PI: 15, CI: 0, STRIDE: 5, SIZE: 16, WQE_NUM: 512, GROUP_IP: 2
CQ TYPE: 5, WQN: 27, PI: 0, CI: 0, STRIDE: 6, SIZE: 16384, WQE_NUM: 16384, GROUP_IP: 2
...
```

3.3.1.18 Local Loopback Disable

Local Loopback Disable feature allows users to force the disablement of local loopback on the virtual port (vport). This disables both unicast and multicast loopback in the hardware.

To enable Local Loopback Disable, run the following command:

```
echo 1 > /sys/class/net/<ifname>/settings/force_local_lb_disable"
```

To disable Local Loopback Disable, run the following command:

```
echo 0 > /sys/class/net/<ifname>/settings/force_local_lb_disable"
```

When turned off, the driver configures the loopback mode according to its own logic.

3.3.1.19 Kernel Transport Layer Security (kTLS) Offloads

This feature is supported on ConnectX-6 Dx crypto cards only.

3.3.1.19.1 Overview

Transport Layer Security (TLS) is a widely-deployed protocol used for securing TCP connections on the Internet. TLS is also a required feature for HTTP/2, the latest web standard. Kernel implementation of TLS (kTLS) provides new opportunities for offloading the protocol into the hardware.

TLS data-path offload allows the NIC to accelerate encryption, decryption and authentication of AES-GCM. TLS offload handles data as it goes through the device without storing any data, but only updating context. If the packet cannot be encrypted/decrypted by the device, then a software fallback handles the packet.

3.3.1.19.2 Establishing a kTLS Connection

To avoid unnecessary complexity in the kernel, the TLS handshake is kept in the user space. A full TLS connection using the socket is done using the following scheme:

1. Call `connect()` or `accept()` on a standard TCP file descriptor.
2. Use a user space TLS library to complete a handshake.
3. Create a new KTLS socket file descriptor.
4. Extract the TLS Initialization Vectors (IVs), session keys, and sequence IDs from the TLS library. Use the `setsockopt` function on the kTLS file descriptor (FD) to pass them to the kernel.
5. Use standard `read()`, `write()`, `sendfile()` and `splice()` system calls on the KTLS FD.

Drivers can offer Tx and Rx packet encryption/decryption offload from the kernel into the NIC hardware. Upon receipt of a non-data TLS message (a control message), the kTLS socket returns an error, and the message is left on the original TCP socket instead. The kTLS socket is automatically unattached. Transfer of control back to the original encrypted FD is done by calling `getsockopt` to receive the current sequence numbers, and inserting them into the TLS library.

3.3.1.19.3 Kernel Support

For support in the kernel, make sure the following flags are set as follows.

- CONFIG_TLS=y
- CONFIG_TLS_DEVICE=y | m

> ⚠ For kTLS Tx device offloads with OFED drivers, kernel TLS module (kernel/net/tls) must be aligned to kernel v5.3 and above.
>
> For kTLS Rx device offloads with OFED drivers, kernel TLS module (kernel/net/tls) must be aligned to kernel v5.9 and above.

3.3.1.19.4 Configuring kTLS Offloads

➤ *To enable kTLS Tx offload, run:*

```
ethtool -K <ifs> tls-hw-tx-offload on
```

➤ *To enable kTLS Rx offload, run:*

```
ethtool -K <ifs> tls-hw-rx-offload on
```

For further information on TLS offloads, please visit the following kernel documentation:

- https://www.kernel.org/doc/html/latest/networking/tls-offload.html
- https://www.kernel.org/doc/html/latest/networking/tls.html#kernel-tls

3.3.1.19.5 OpenSSL with kTLS Offload

OpenSSL version 3.0.0 or above is required to support kTLS TX/RX offloads.

Supported OpenSSL version is available to download from distro packages, or can be downloaded and compiled from the OpenSSL github.

3.3.1.20 IPsec Crypto Offload

This feature is supported on crypto-enabled products of BlueField-2 DPUs, and ConnectX-6 Dx and ConnectX-7 adapters (but not of ConnectX-6 or ConnectX-6 Lx).

Newer/future crypto-enabled DPU and adapter product generations should also support the feature, unless explicitly stated in their documentation.

For NVIDIA BlueField-2 DPUs and ConnectX-6 Dx adapters Only: If your target application will utilize bandwidth of100Gb/s or higher, where a substantial part of the bandwidth will be allocated for IPsec traffic, please refer to the NVIDIA BlueField-2 DPUs Product Release Notes or NVIDIA ConnectX-6 Dx Adapters Product Release Notes document to learn about a potential bandwidth limitation. To access the relevant product release notes, please contact your NVIDIA sales representative.

3.3.1.20.1 Overview and Configuration

IPsec crypto offload feature, also known as IPsec inline offload or IPsec aware offload feature enables the user to offload IPsec crypto encryption and decryption operations to the hardware.

Note that the hardware implementation only supports AES-GCM encryption scheme.

To enable the feature, support in both kernel and adapter firmware is required.

- For support in the kernel, make sure the following flags are set as follows.

```
CONFIG_XFRM_OFFLOAD=y
CONFIG_INET_ESP_OFFLOAD=m
CONFIG_INET6_ESP_OFFLOAD=m
```

  Note: These flags are enabled by default in RedHat 8 and Ubuntu 18.04.

- For support in the firmware, make sure the below string is found in the dmesg.

```
mlx5e: IPSec ESP acceleration enabled
```

3.3.1.20.2 Configuring Security Associations for IPsec Offloads

To program the inline offload security associations (SA), add the option "offload dev <netdev interface> dir out/in" in the "ip xfrm state" command for transmitting and receiving SA.

Transmit inline offload SA xfrm command example:

```
sudo ip xfrm state add src 192.168.1.64/24 dst 192.168.1.65/24 proto esp spi 0x46dc6204 reqid 0x46dc6204 mode
transport aead 'rfc4106(gcm(aes))' 0x60bd6c3eafba371a46411830fd56c53af93883261ed1fb26767820ff493f43ba35b0dcca 128
 offload dev p4p1 dir out sel src 192.168.1.64 dst 192.168.1.65
```

Receive inline offload SA xfrm command example:

```
sudo ip xfrm state add src 192.168.1.65/24 dst 192.168.1.64/24 proto esp spi 0xaea0846c reqid 0xaea0846c mode
transport aead 'rfc4106(gcm(aes))' 0x81d5c3167c912c1dd50dab0cb4b6d815b6ace8844304db362215a258cd19deda8f89deda 128
 offload dev p4p1 dir in sel src 192.168.1.65 dst 192.168.1.64
```

3.3.1.20.2.1 Setting xfrm Policies Example

First server:

```
+ sudo ip xfrm state add src 192.168.1.64/24 dst 192.168.1.65/24 proto esp spi 0x28f39549 reqid 0x28f39549 mode
transport aead 'rfc4106(gcm(aes))' 0x492e8ffe718a95a00c1893ea61afc64997f4732848ccfe6ea07db483175cb18de9ae411a 128
 offload dev enp4s0 dir out sel src 192.168.1.64 dst 192.168.1.65
+ sudo ip xfrm state add src 192.168.1.65/24 dst 192.168.1.64/24 proto esp spi 0x622a73b4 reqid 0x622a73b4 mode
transport aead 'rfc4106(gcm(aes))' 0x093bfee2212802d626716815f862da31bcc7d9c44cfe3ab8049e7604b2feb1254869d25b 128
 offload dev enp4s0 dir in sel src 192.168.1.65 dst 192.168.1.64
+ sudo ip xfrm policy add src 192.168.1.64 dst 192.168.1.65 dir out tmpl src 192.168.1.64/24 dst 192.168.1.65/24
 proto esp reqid 0x28f39549 mode transport
+ sudo ip xfrm policy add src 192.168.1.65 dst 192.168.1.64 dir in tmpl src 192.168.1.65/24 dst 192.168.1.64/24
 proto esp reqid 0x622a73b4 mode transport
+ sudo ip xfrm policy add src 192.168.1.65 dst 192.168.1.64 dir fwd tmpl src 192.168.1.65/24 dst 192.168.1.64/24
 proto esp reqid 0x622a73b4 mode transport
```

Second server:

```
+ ssh -A -t root@l-csi-0921d /bin/bash
+ set -e
+ '[' 0 == 1 ']'
+ sudo ip xfrm state add src 192.168.1.64/24 dst 192.168.1.65/24 proto esp spi 0x28f39549 reqid 0x28f39549 mode
transport aead 'rfc4106(gcm(aes))' 0x492e8ffe718a95a00c1893ea61afc64997f4732848ccfe6ea07db483175cb18de9ae411a 128
 offload dev enp4s0 dir in sel src 192.168.1.64 dst 192.168.1.65
+ sudo ip xfrm state add src 192.168.1.65/24 dst 192.168.1.64/24 proto esp spi 0x622a73b4 reqid 0x622a73b4 mode
transport aead 'rfc4106(gcm(aes))' 0x093bfee2212802d626716815f862da31bcc7d9c44cfe3ab8049e7604b2feb1254869d25b 128
 offload dev enp4s0 dir out sel src 192.168.1.65 dst 192.168.1.64
+ sudo ip xfrm policy add src 192.168.1.65 dst 192.168.1.64 dir out tmpl src 192.168.1.65/24 dst 192.168.1.64/24
 proto esp reqid 0x622a73b4 mode transport
+ sudo ip xfrm policy add src 192.168.1.64 dst 192.168.1.65 dir in tmpl src 192.168.1.64/24 dst 192.168.1.65/24
 proto esp reqid 0x28f39549 mode transport
+ sudo ip xfrm policy add src 192.168.1.64 dst 192.168.1.65 dir fwd tmpl src 192.168.1.64/24 dst 192.168.1.65/24
 proto esp reqid 0x28f39549 mode transport
+ echo 'IPSec tunnel configured successfully'
```

3.3.1.21 IPsec Full Offload

This feature is supported on crypto-enabled products of BlueField-2 DPUs, as well as on ConnectX-6 Dx, ConnectX-6 Lx and ConnectX-7 adapter cards. Note that it is not supported on ConnectX-6 cards.

Newer/future crypto-enabled DPU and adapter product generations should also support this feature, unless explicitly stated otherwise in their documentation.

⚠ When using NVIDIA® BlueField®-2 DPUs and NVIDIA® ConnectX®-6 Dx adapters only: If your target application utilizes 100Gb/s or a higher bandwidth, where a substantial part of the bandwidth is allocated for IPsec traffic, please refer to the relevant DPU or adapter card Product Release Notes to learn about a potential bandwidth limitation. To access the Release Notes, visit https://docs.nvidia.com/networking/, or contact your NVIDIA sales representative.

⚠ ConnectX-6 Dx adapters only support Full Offload: Encrypted Overlay (where a Hypervisor controls IPsec offload - See for example OVS IPsec - https://docs.openvswitch.org/en/latest/tutorials/ipsec/) in a Linux OS with NVIDIA drivers.

⚠ This feature requires Linux kernel v6.6, or higher.

This feature is designed to enable IPsec full offload in switchdev mode. The `ip-xfrm` command is used to configure IPsec states and policies, and it is similar to legacy mode configuration. However, there are several limitations to the use of full offload in this mode:

1. Only IPsec Transport Mode and Tunnel Mode are supported.
2. The first IPsec TX state/policy is not allowed to be offloaded if any offloaded TC rule exists, and the same applies for the first RX state/policy. More specifically, IPsec RX/TX tables must be created before offloading any TC rule. For this reason, it is a common practice to configure IPsec rules before adding any TC rule.

Following is an example for IPsec configuration with a VXLan tunnel:

- Enable switchdev mode:

```
echo 1 > /sys/class/net/$PF0 /device/sriov_numvfs
echo 0000:08:00.2 > /sys/bus/pci/drivers/mlx5_core/unbind
devlink dev param set pci/0000:08:00.0 name flow_steering_mode value dmfs cmode runtime
devlink dev eswitch set pci/0000:08:00.0 mode switchdev
echo 0000:08:00.2 > /sys/bus/pci/drivers/mlx5_core/bind
```

- Configure PF/VF/REP netdevices, and place a VF in a namespace:

```
ifconfig $PF $LOCAL_TUN/16 up
ip l set dev $PF mtu 2000

ifconfig $REP up
ip netns add ns0
ip link set dev $VF netns ns0
ip netns exec ns0 ifconfig $VF $IP/16 up
```

- Configure IPsec states and policies:

```
ip xfrm state add src $LOCAL_TUN/16 dst $REMOTE_IP/16 proto esp spi 0xb29ed314 reqid 0xb29ed314 mode
transport aead 'rfc4106(gcm(aes))' 0x20f01f80a26f633d85617465686c32552c92c42f 128 offload packet dev $PF
dir out sel src $LOCAL_TUN/16 dst $REMOTE_IP/16 flag esn replay-window 64
ip xfrm state add src $REMOTE_IP/16 dst $LOCAL_TUN/16 proto esp spi 0xc35aa26e reqid 0xc35aa26e mode
transport aead 'rfc4106(gcm(aes))' 0x6cb228189b4c6e82e66e46920a2cde39187de4ba 128 offload packet dev $PF
dir in sel src $REMOTE_IP/16 dst $LOCAL_TUN/16 flag esn replay-window 64

ip xfrm policy add src $LOCAL_TUN dst $REMOTE_IP offload packet dev $PF dir out tmpl src $LOCAL_TUN/16 dst
$REMOTE_IP/16 proto esp reqid 0xb29ed314 mode transport priority 12
ip xfrm policy add src $REMOTE_IP dst $LOCAL_TUN offload packet dev $PF dir in tmpl src $REMOTE_IP/16 dst
$LOCAL_TUN/16 proto esp reqid 0xc35aa26e mode transport priority 12
```

- Configure Openvswitch:

```
ovs-vsctl add-br br-ovs
ovs-vsctl add-port br-ovs $REP
ovs-vsctl add-port br-ovs vxlan1 -- set interface vxlan1 type=vxlan options:local_ip=$LOCAL_TUN
options:remote_ip=$REMOTE_IP options:key=$VXLAN_ID options:dst_port=4789
```

3.3.1.21.1 IPsec Full Offload for RDMA Traffic

This IPsec Full Offload for RDMA Traffic option provides a significant performance improvement compared to the software IPsec counterpart, and enables the use of IPsec over RoCE packets, which are outside the network stack and cannot be used without full hardware offload. As a result, users can leverage the benefits of the IPsec protocol with RoCE V2, even when using SR-IOV VFs.

The configuration steps for this feature should be identical to the steps mentioned above, but if this feature is supported, the traffic that will be sent can also be RoCEV2 IPsec traffic.

To configure this feature:

1. Configure an SR-IOV VF normally, and add its OVS/TC rules.
2. Enable IPsec over VF. For more information, please see IPsec Functionality.
3. Configure IPsec policies and states on the relevant VF net device. This should be identical to the software configuration of IPsec rules, which can be done using one of the following implementation options:

Command	Offload Request Parameter
iproute2 ip xfrm	offload packet
libreswan	nic-offload=packet
strongswan	

For this feature to work, switchdev mode and dmfs steering mode must be enabled.

- The following is a full minimalistic configuration example using iproute, whereas PF0 is the netdevice PF, F0_REP is the VF representor, and NIC is the VF netdevice to configure IPsec over:

```
1.  echo 1 > /sys/class/net/$PF0 /device/sriov_numvfs
2.  echo 0000:08:00.2 > /sys/bus/pci/drivers/mlx5_core/unbind
3.  devlink dev eswitch set pci/0000:08:00.0 mode switchdev
4.  devlink dev param set pci/0000:08:00.0 name flow_steering_mode value dmfs cmode runtime
5.  devlink port function set pci/0000:08:00.0/1 ipsec_packet enable
6.  echo 0000:08:00.2 > /sys/bus/pci/drivers/mlx5_core/bind
7.  tc qdisc add dev $PF0 ingress
tc qdisc add dev $VF0_REP ingress
tc filter add dev $PF0 parent ffff: protocol 802.1q chain 0 flower vlan_id 10 vlan_ethtype 802.1q cvlan_id
5 action vlan pop action vlan pop  action mirred egress redirect dev $VF0_REP

tc filter add dev $VF0_REP parent ffff: protocol all chain 0 flower action vlan push protocol 802.1q id 5
 action vlan push protocol 802.1q id 10 action mirred egress redirect dev $PF0

8.  ifconfig $PF0 $PF_IP/24 up
ifconfig $NIC $LOC_IP/$SUB_NET up
ip link set dev $VF_REP up
9.  ip  xfrm state flush
ip  xfrm policy flush
```

- Configure ipsec states and policies:

```
#states
ip  -4 xfrm state add src $LOC_IP/$SUB_NET dst $REMOTE_IP/$SUB_NET proto esp spi 1000 reqid 10000 aead
'rfc4106(gcm(aes))' 0x010203047aeaca3f87d060a12f4a4487d5a5c335 128 mode transport sel src $LOC_IP dst
$REMOTE_IP offload packet dev $NIC dir out
ip  -4 xfrm state add src $REMOTE_IP/$SUB_NET dst $LOC_IP/$SUB_NET proto esp spi 1001 reqid 10001 aead
'rfc4106(gcm(aes))' 0x010203047aeaca3f87d060a12f4a4487d5a5c335 128 mode transport sel src $REMOTE_IP dst
$LOC_IP offload packet dev $NIC dir in
#policies
ip  -4 xfrm policy add src $LOC_IP dst $REMOTE_IP offload packet dev $NIC dir out tmpl src $LOC_IP/$SUB_NET
dst $REMOTE_IP/$SUB_NET proto esp reqid 10000 mode transport
ip  -4 xfrm policy add src $REMOTE_IP dst $LOC_IP offload packet dev $NIC dir in tmpl src $REMOTE_IP/
$SUB_NET dst $LOC_IP/$SUB_NET proto esp reqid 10001 mode transport
ip  -4 xfrm policy add src $REMOTE_IP dst $LOC_IP dir fwd tmpl src $REMOTE_IP/$SUB_NET dst $LOC_IP/$SUB_NET
proto esp reqid 10001 mode transport
```

Note that the configuration above is for one side only, yet IPsec must be configured for both sides in order for them to communicate properly. The configuration for the other side should be almost identical, but Step 9 would be configured in an asymmetrical way, meaning the first policy would look the following, and all other states/policies would be adjusted accordingly:

```
ip -4 xfrm state add src $LOC_IP/$SUB_NET dst $REMOTE_IP/$SUB_NET proto esp spi 1001 reqid 10001 aead
'rfc4106(gcm(aes))' 0x010203047aeaca3f87d060a12f4a4487d5a5c335 128 mode transport sel src $LOC_IP dst $REMOTE_IP
offload packet dev $NIC dir out
```

Once this step is completed, you can send any RoCE traffic of your choice between the two machines with configured IPsec. For example, `ibv_rc_pingpong -g 3 -d VF_device :` on one side, and `ibv_rc_pingpong -g 3 -d VF_device $IP_OF_OTHER_SIDE :` on the other side.

Finally, you can verify that the traffic was encrypted using IPsec by using the ipsec counters:

```
ethtool -S VF_NETDEV | grep ipsec
```

3.3.1.22 MACsec Full Offload

MACsec Full offload feature, also known as MACsec inline Full offload, enables the user to offload MACsec crypto encryption and decryption, MACsec headers encapsulation and decapsulation, and Anti replay operations to the hardware.

⚠ Hardware implementation supports GCM-AES & GCM-AES-XPN encryption schemes and is supported with ConnectX-7 onwards.

⚠ MACsec introduced in MOFED v5.9 requires a minimal Kernel version of 6.1.

To enable the feature, support in both kernel and adapter firmware is required.

For support in the kernel, make sure the following flags are set as follows:

- CONFIG_MACSEC=y
- CONFIG_MLX5_EN_MACSEC=y

For support in firmware use the following version:

- xx.34.0364 and up

3.3.1.22.1 Configurations

3.3.1.22.1.1 IProute2 Configuration

Configuring Physical Interface

Client side:

- ip address flush <physical_device>
- ip address add <client_physical_device_ip> dev <physical interface>
- ip link set dev <physical_device>up

Server side:

- ip address flush <physical_device>
- ip address add <server_physical_device_ip> dev <physical interface>
- ip link set dev <physical_device>up

Add MACsec Device

Client side:

- ip link add link <physical_device> <macsec_device> type macsec sci <client_sci> client on

Server side:

- ip link add link <physical_device> <macsec_device> type macsec sci <client_sci> client on

Offload MACsec Device

Client side:

- ip macsec offload <macsec_device> mac

Server side:

- ip macsec offload <macsec_device> mac

Add MACsec rules:

Client side:

- ip macsec add <macsec_device> tx sa <sa_num>pn <inital_packet_number>on key <client_key_id> <client_key>
- ip macsec add <macsec_device> rx sci <server_sci> on
- ip macsec add <macsec_device> rx sci <server_sci>sa <sa_num> pn <inital_packet_number> on key <server_key_id> <server_key>

Server side:

- ip macsec add <macsec_device> tx sa <sa_num>pn <inital_packet_number>on key <server_key_id> <server_key>
- ip macsec add <macsec_device> rx sci <client_sci> on

- ip macsec add <macsec_device> rx sci <client_sci>sa <sa_num> pn <inital_packet_number> on key <client_key_id> <client_key>

Configure MACsec Device IPs:

Client side:

- ip address flush <macsec_device>
- ip address add <client_macsec_device_ip> dev <macsec_device>
- ip link set dev <macsec_device> up

Server side:

- ip address flush <macsec_device>
- ip address add <server_macsec_device_ip> dev <macsec_device>
- ip link set dev <macsec_device> up

3.3.1.22.1.2 Configuration Example

Client side:

- ip address flush enp8s0f0
- ip address add 1.1.1.1/24 dev enp8s0f0
- ip link set dev enp8s0f0 up
- ip link add link enp8s0f0 macsec0 type macsec sci 1 encrypt on
- ip macsec offload macsec0 mac
- ip macsec add macsec0 tx sa 0 pn 1 on key 00 dffafc8d7b9a43d5b9a3dfbbf6a30c16
- ip macsec add macsec0 rx sci 2 on
- ip macsec add macsec0 rx sci 2 sa 0 pn 1 on key 00 ead3664f508eb06c40ac7104cdae4ce5
- ip address flush macsec0
- ip address add 2.2.2.1/24 dev macsec0
- ip link set dev macsec0 up

Server side:

- ip link del macsec0
- ip address flush enp8s0f0
- ip address add 1.1.1.2/24 dev enp8s0f0
- ip link set dev enp8s0f0 up
- ip link add link enp8s0f0 macsec0 type macsec sci 2 encrypt on
- ip macsec offload macsec0 mac
- ip macsec add macsec0 tx sa 0 pn 1 on key 00 ead3664f508eb06c40ac7104cdae4ce5
- ip macsec add macsec0 rx sci 1 on
- ip macsec add macsec0 rx sci 1 sa 0 pn 1 on key 00 dffafc8d7b9a43d5b9a3dfbbf6a30c16
- ip address flush macsec0
- ip address add 2.2.2.2/24 dev macsec0
- ip link set dev macsec0 up

- Use: "ip macsec show" command to check configuration

- To make sure traffic is offloaded, check MACsec counters: "`ethtool -S <physical_device> | grep macsec`"

Additional Resources

Linux Manual page: linux_manual

3.3.2 Virtualization

The chapter contains the following sections:

3.3.2.1 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. NVIDIA adapters are capable of exposing up to 127 virtual instances (Virtual Functions (VFs) for each port in the NVIDIA ConnectX® family cards. These virtual functions can then be provisioned separately. Each VF can be seen as an additional 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 ConnectX® VPI adapter cards.

3.3.2.1.1 System Requirements

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

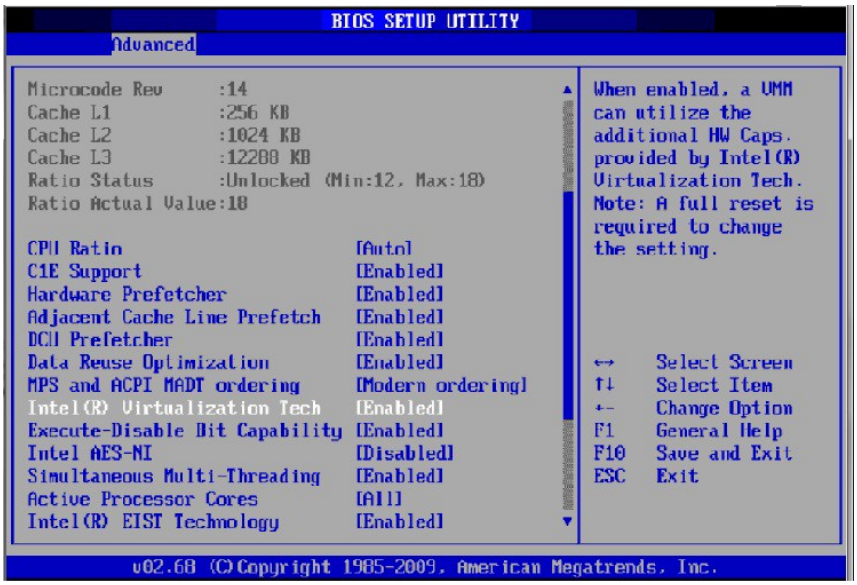
- MLNX_EN Driver
- A server/blade with an SR-IOV-capable motherboard BIOS
- Hypervisor that supports SR-IOV such as: Red Hat Enterprise Linux Server Version 6
- NVIDIA ConnectX® VPI Adapter Card family with SR-IOV capability

3.3.2.1.2 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:

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

2. Enable "Intel Virtualization Technology".



3. Install a hypervisor that supports SR-IOV.
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 (4.x.x)
        root (hd0,0)
        kernel /vmlinuz-4.x.x ro root=/dev/VolGroup00/LogVol00 rhgb quiet
        intel_iommu=on        initrd /initrd-4.x.x.img
```

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

Some OSs use /boot/grub2/grub.cfg file. If your server uses such file, please edit this file instead (add “ `intel_iommu=on` ” for the relevant menu entry at the end of the line that starts with "linux16").

3.3.2.1.3 Configuring SR-IOV (Ethernet)

To set SR-IOV in Ethernet mode, refer to HowTo Configure SR-IOV for ConnectX-4/ConnectX- 5/ ConnectX-6 with KVM (Ethernet) Community Post.

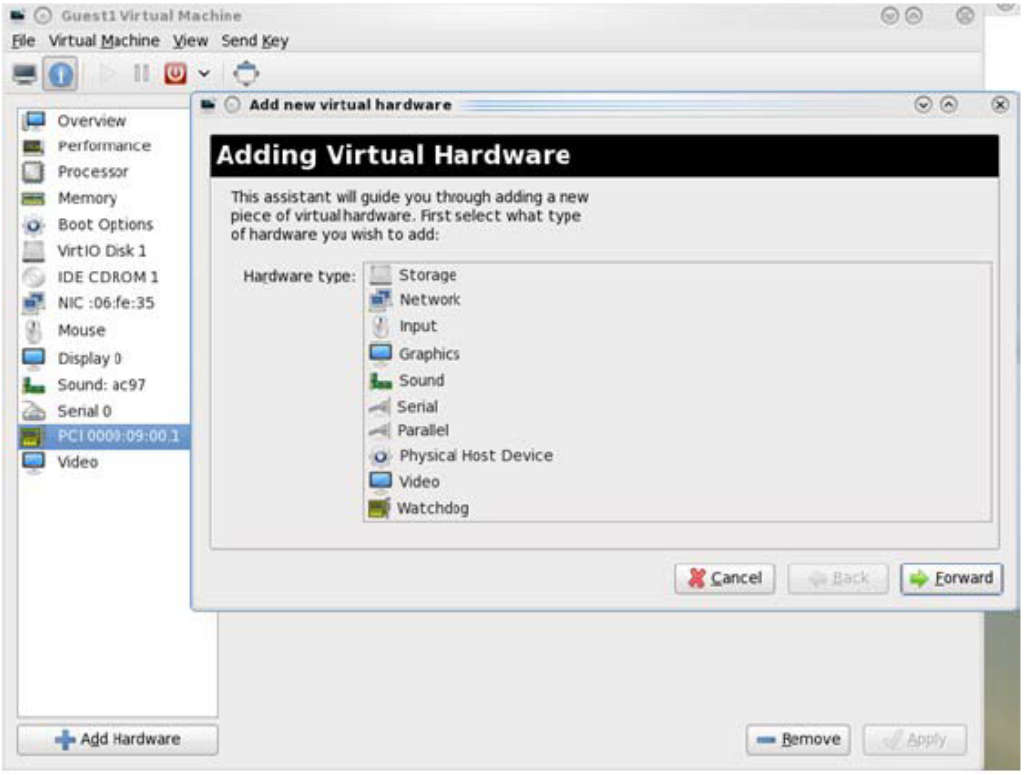
3.3.2.1.4 Additional SR-IOV Configurations

3.3.2.1.4.1 Assigning a Virtual Function to a Virtual Machine

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

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

1. Run the virt-manager.
2. Double click on the virtual machine and open its Properties.
3. Go to Details → Add hardware → PCI host device.



4. Choose a NVIDIA virtual function according to its PCI device (e.g., 00:03.1)
5. If the Virtual Machine is up reboot it, otherwise start it.
6. Log into the virtual machine and verify that it recognizes the NVIDIA card. Run:

```
lspci | grep Mellanox
```

Example:

```
lspci | grep Mellanox
01:00.0 Infiniband controller: Mellanox Technologies MT28800 Family [ConnectX-5 Ex]
```

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.

3.3.2.1.4.3 Ethernet Virtual Function Configuration when Running SR-IOV

SR-IOV Virtual function configuration can be done through Hypervisor iprout2/netlink tool, if present. Otherwise, it can be done via sysfs.

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

sysfs configuration (ConnectX-4):

/sys/class/net/enp8s0f0/device/sriov/[VF]

+-- [VF]
| +-- config
| +-- link_state
| +-- mac
| +-- mac_list
| +-- max_tx_rate
| +-- min_tx_rate
| +-- spoofcheck
| +-- stats
| +-- trunk
| +-- trust
| +-- vlan
```

VLAN 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.
The default behavior is VGT.

To configure VF VST mode, run:

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

where:

- NUM = 0..max-vf-num
- vlan_id = 0..4095
- qos = 0..7

For example:

- ip link set dev eth2 vf 2 vlan 10 qos 3 - sets VST mode for VF #2 belonging to PF eth2, with vlan_id = 10 and qos = 3
- ip link set dev eth2 vf 2 vlan 0 - sets mode for VF 2 back to VGT

Additional Ethernet VF Configuration Options

- Guest MAC configuration - by default, guest MAC addresses are configured to be all zeroes. 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 and ConnectX-4 guests, which do not generate random MACs, the administrator 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]
```

- Guest Link State

```
ip link set dev <PF device> vf <UM> state [enable| disable| auto]
```

Virtual Function Statistics

Virtual function statistics can be queried via sysfs:

```
cat /sys/class/infiniband/mlx5_2/device/sriov/2/stats tx_packets : 5011
tx_bytes : 4450870
tx_dropped : 0
rx_packets : 5003
rx_bytes : 4450222
rx_broadcast : 0
rx_multicast : 0
tx_broadcast : 0
tx_multicast : 8
rx_dropped : 0
```

Mapping VFs to Ports

To view the VFs mapping to ports:

Use the ip link tool v2.6.34~3 and above.

```
ip link
```

Output:

```
61: p1p1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/ether 00:02:c9:f1:72:e0 brd ff:ff:ff:ff:ff:ff
    vf 0 MAC 00:00:00:00:00:00, vlan 4095, spoof checking off, link-state auto
    vf 37 MAC 00:00:00:00:00:00, vlan 4095, spoof checking off, link-state auto
    vf 38 MAC ff:ff:ff:ff:ff:ff, vlan 65535, spoof checking off, link-state disable
    vf 39 MAC ff:ff:ff:ff:ff:ff, vlan 65535, spoof checking off, link-state disable
```

When a MAC is ff:ff:ff:ff:ff:ff, the VF is not assigned to the port of the net device it is listed under.

In the example above, vf38 is not assigned to the same port as p1p1, in contrast to vf0.
However, even VFs that are not assigned to the net device, could be used to set and change its settings. For example, the following is a valid command to change the spoof check:

```
ip link set dev p1p1 vf 38 spoofchk on
```

This command will affect only the vf38. The changes can be seen in ip link on the net device that this device is assigned to.

RoCE 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.

3.3.2.1.4.4 Virtual Guest Tagging (VGT+)

VGT+ is an advanced mode of Virtual Guest Tagging (VGT), in which a VF is allowed to tag its own packets as in VGT, but is still subject to an administrative VLAN trunk policy. The policy determines which VLAN IDs are allowed to be transmitted or received. The policy does not determine the user priority, which is left unchanged.
Packets can be sent in one of the following modes: when the VF is allowed to send/receive untagged and priority tagged traffic and when it is not. No default VLAN is defined for VGT+ port. The send packets are passed to the eSwitch only if they match the set, and the received packets are forwarded to the VF only if they match the set.

Configuration

> When working in SR-IOV, the default operating mode is VGT.

To enable VGT+ mode:

Set the corresponding port/VF (in the example below port eth5, VF0) range of allowed VLANs.

```
echo "<add> <start_vid> <end_vid>" > /sys/class/net/eth5/device/sriov/0/trunk
```

Examples:

- Adding VLAN ID range (4-15) to trunk:

  ```
  echo add 4 15 > /sys/class/net/eth5/device/sriov/0/trunk
  ```

- Adding a single VLAN ID to trunk:

  ```
  echo add 17 17 > /sys/class/net/eth5/device/sriov/0/trunk
  ```

Note: When VLAN ID = 0, it indicates that untagged and priority-tagged traffics are allowed

To disable VGT+ mode, make sure to remove all VLANs.

```
echo rem 0 4095 > /sys/class/net/eth5/device/sriov/0/trunk
```

To remove selected VLANs.

- Remove VLAN ID range (4-15) from trunk:

```
echo rem 4 15 > /sys/class/net/eth5/device/sriov/0/trunk
```

- Remove a single VLAN ID from trunk:

```
echo rem 17 17 > /sys/class/net/eth5/device/sriov/0/trunk
```

3.3.2.1.4.5 SR-IOV Advanced Security Features

SR-IOV MAC Anti-Spoofing

Normally, MAC addresses are unique identifiers assigned to network interfaces, and they are fixed addresses that cannot be changed. MAC address spoofing is a technique for altering the MAC address to serve different purposes. Some of the cases in which a MAC address is altered can be legal, while others can be illegal and abuse security mechanisms or disguises a possible attacker.
The SR-IOV MAC address anti-spoofing feature, also known as MAC Spoof Check provides protection against malicious VM MAC address forging. If the network administrator assigns a MAC address to a VF (through the hypervisor) and enables spoof check on it, this will limit the end user to send traffic only from the assigned MAC address of that VF.

MAC Anti-Spoofing Configuration

MAC anti-spoofing is disabled by default.

In the configuration example below, the VM is located on VF-0 and has the following MAC address: 11:22:33:44:55:66.
There are two ways to enable or disable MAC anti-spoofing:

1. Use the standard IP link commands - available from Kernel 3.10 and above.
    a. To enable MAC anti-spoofing, run:

```
ip link set ens785f1 vf 0 spoofchk on
```

    b. To disable MAC anti-spoofing, run:

```
ip link set ens785f1 vf 0 spoofchk off
```

2. Specify echo "ON" or "OFF" to the file located under /sys/class/net/<ifname / device/sriov/<VF index>/spoofcheck.
    a. To enable MAC anti-spoofing, run:

```
echo "ON" > /sys/class/net/ens785f1/vf/0/spoofchk
```

    b. To disable MAC anti-spoofing, run:

```
echo "OFF" > /sys/class/net/ens785f1/vf/0/spoofchk
```

⚠ This configuration is non-persistent and does not survive driver restart.

Limit and Bandwidth Share Per VF

This feature enables rate limiting traffic per VF in SR-IOV mode. For details on how to configure rate limit per VF for ConnectX-4 and above adapter cards, please refer to HowTo Configure Rate Limit per VF for ConnectX-4/ConnectX-5/ConnectX-6 Community post.

Limit Bandwidth per Group of VFs

VFs Rate Limit for vSwitch (OVS) feature allows users to join available VFs into groups and set a rate limitation on each group. Rate limitation on a VF group ensures that the total Tx bandwidth that the VFs in this group get (altogether combined) will not exceed the given value.
With this feature, a VF can still be configured with an individual rate limit as in the past (under /sys/class/net/<ifname>/device/sriov/<vf_num>/max_tx_rate). However, the actual bandwidth limit on the VF will eventually be determined considering the VF group limitation and how many VFs are in the same group.
For example: 2 VFs (0 and 1) are attached to group 3.

Case 1: The rate limitation on the group is set to 20G. Rate limit of each VF is 15G
Result: Each VF will have a rate limit of 10G

Case 2: Group's max rate limitation is still set to 20G. VF 0 is configured to 30G limit, while VF 1 is configured to 5G rate limit
Result: VF 0 will have 15G de-facto. VF 1 will have 5G

The rule of thumb is that the group's bandwidth is distributed evenly between the number of VFs in the group. If there are leftovers, they will be assigned to VFs whose individual rate limit has not been met yet.

VFs Rate Limit Feature Configuration

1. When VF rate group is supported by FW, the driver will create a new hierarchy in the SRI-OV sysfs named "groups" (/sys/class/net/<ifname>/device/sriov/groups/). It will contain all the info and the configurations allowed for VF groups.
2. All VFs are placed in group 0 by default since it is the only existing group following the initial driver start. It would be the only group available under /sys/class/net/<ifname>/device/sriov/groups/
3. The VF can be moved to a different group by writing to the group file -> echo $GROUP_ID > /sys/class/net/<ifname>/device/sriov/<vf_id>/group

4. The group IDs allowed are 0-255
5. Only when there is at least 1 VF in a group, there will be a group configuration available under /sys/class/net/<ifname>/device/sriov/groups/ (Except for group 0, which is always available even when it's empty).
6. Once the group is created (by moving at least 1 VF to that group), users can configure the group's rate limit. For example:
    a. echo 10000 > /sys/class/net/<ifname>/device/sriov/5/max_tx_rate – setting individual rate limitation of VF 5 to 10G (Optional)
    b. echo 7 > /sys/class/net/<ifname>/device/sriov/5/group – moving VF 5 to group 7
    c. echo 5000 > /sys/class/net/<ifname>/device/sriov/groups/7/max_tx_rate – setting group 7 with rate limitation of 5G
    d. When running traffic via VF 5 now, it will be limited to 5G because of the group rate limit even though the VF itself is limited to 10G
    e. echo 3 > /sys/class/net/<ifname>/device/sriov/5/group – moving VF 5 to group 3
    f. Group 7 will now disappear from /sys/class/net/<ifname>/device/sriov/groups since there are 0 VFs in it. Group 3 will now appear. Since there's no rate limit on group 3, VF 5 can transmit at 10G (thanks to its individual configuration)

Notes:

- You can see to which group the VF belongs to in the 'stats' sysfs (cat /sys/class/net/<ifname>/device/sriov/<vf_num>/stats)
- You can see the current rate limit and number of attached VFs to a group in the group's 'config' sysfs (cat /sys/class/net/<ifname>/device/sriov/groups/<group_id>/config)

Bandwidth Guarantee per Group of VFs

Bandwidth guarantee (minimum BW) can be set on a group of VFs to ensure this group is able to transmit at least the amount of bandwidth specified on the wire.

Note the following:

- The minimum BW settings on VF groups determine how the groups share the total BW between themselves. It does not impact an individual VF's rate settings.
- The total minimum BW that is set on the VF groups should not exceed the total line rate. Otherwise, results are unexpected.
- It is still possible to set minimum BW on the individual VFs inside the group. This will determine how the VFs share the group's minimum BW between themselves. The total minimum BW of the VF member should not exceed the minimum BW of the group.

For instruction on how to create groups of VFs, see Limit Bandwidth per Group of VFs above.

Example

With a 40Gb link speed, assuming 4 groups and default group 0 have been created:

```
echo 20000 > /sys/class/net/<ifname>/device/sriov/group/1/min_tx_rate
echo 5000 > /sys/class/net/<ifname>/device/sriov/group/2/min_tx_rate
echo 15000 > /sys/class/net/<ifname>/device/sriov/group/3/min_tx_rate
```

```
Group 0(default) : 0 - No BW guarantee is configured.
Group 1 : 20000 - This is the maximum min rate among groups
Group 2 : 5000 which is 25% of the maximum min rate
```

```
Group 3 : 15000 which is 75% of the maximum min rate
Group 4 : 0 - No BW guarantee is configured.
```

Assuming there are VFs attempting to transmit in full line rate in all groups, the results would look like: In which case, the minimum BW allocation would be:

```
Group0 - Will have no BW to use since no BW guarantee was set on it while other groups do have such settings.
Group1 - Will transmit at 20Gb/s
Group2 - Will transmit at 5Gb/s
Group3 - Will transmit at 15Gb/s
Group4 - Will have no BW to use since no BW guarantee was set on it while other groups do have such settings.
```

Privileged VFs

In case a malicious driver is running over one of the VFs, and in case that VF's permissions are not restricted, this may open security holes. However, VFs can be marked as trusted and can thus receive an exclusive subset of physical function privileges or permissions. For example, in case of allowing all VFs, rather than specific VFs, to enter a promiscuous mode as a privilege, this will enable malicious users to sniff and monitor the entire physical port for incoming traffic, including traffic targeting other VFs, which is considered a severe security hole.

Privileged VFs Configuration

In the configuration example below, the VM is located on VF-0 and has the following MAC address: 11:22:33:44:55:66.
There are two ways to enable or disable trust:

1. Use the standard IP link commands - available from Kernel 4.5 and above.
    a. To enable trust for a specific VF, run:

    ```
    ip link set ens785f1 vf 0 trust on
    ```

    b. To disable trust for a specific VF, run:

    ```
    ip link set ens785f1 vf 0 trust off
    ```

2. Specify echo "ON" or "OFF" to the file located under /sys/class/net/<ETH_IF_NAME> / device/ sriov/<VF index>/trust.
    a. To enable trust for a specific VF, run:

    ```
    echo "ON" > /sys/class/net/ens785f1/device/sriov/0/trust
    ```

    b. To disable trust for a specific VF, run:

    ```
    echo "OFF" > /sys/class/net/ens785f1/device/sriov/0/trust
    ```

Probed VFs

Probing Virtual Functions (VFs) after SR-IOV is enabled might consume the adapter cards' resources. Therefore, it is recommended not to enable probing of VFs when no monitoring of the VM is needed. VF probing can be disabled in two ways, depending on the kernel version installed on your server:

1. If the kernel version installed is v4.12 or above, it is recommended to use the PCI sysfs interface `sriov_drivers_autoprobe`. For more information, see linux-next branch.

2. If the kernel version installed is older than v4.12, it is recommended to use the mlx5_core module parameter probe_vf with driver version 4.1 or above.

Example:

```
echo 0 > /sys/module/mlx5_core/parameters/probe_vf
```

For more information on how to probe VFs, see HowTo Configure and Probe VFs on mlx5 Drivers Community post.

3.3.2.1.4.6 VF Promiscuous Rx Modes

VF Promiscuous Mode

VFs can enter a promiscuous mode that enables receiving the unmatched traffic and all the multicast traffic that reaches the physical port in addition to the traffic originally targeted to the VF. The unmatched traffic is any traffic's DMAC that does not match any of the VFs' or PFs' MAC addresses.
Note: Only privileged/trusted VFs can enter the VF promiscuous mode.

To set the promiscuous mode on for a VF, run:

```
ifconfig eth2 promisc
```

To exit the promiscuous mode, run:

```
ifconfig eth2 -promisc
```

VF All-Multi Mode

VFs can enter an all-multi mode that enables receiving all the multicast traffic sent from/to the other functions on the same physical port in addition to the traffic originally targeted to the VF.
Note: Only privileged/trusted VFs can enter the all-multi RX mode.

To set the all-multi mode on for a VF, run:

```
ifconfig eth2 allmulti
```

To exit the all-multi mode, run:

```
#ifconfig eth2 -allmulti
```

3.3.2.1.5 Uninstalling the SR-IOV Driver

To uninstall SR-IOV driver, perform the following:

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 that stopping the driver when there are VMs that use the VFs, will cause machine to hang.
2. Run the script below. Please be aware, uninstalling the driver deletes the entire driver's file, but does not unload the driver.

```
[root@sw1022 ~]# /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 compat-dapl compat-dapl-devel dapl dapl-devel dapl-devel-static dapl-
utils srptools infiniband-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
```

3. Restart the server.

3.3.2.2 Enabling Paravirtualization

To enable Paravirtualization:

> The example below works on RHEL7.* without a Network Manager.

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
```

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
```

3. Restart the service network.
4. Attach a bridge 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
…
```

3.3.2.3 VXLAN Hardware Stateless Offloads

VXLAN technology provides scalability and security challenges solutions. It requires extension of the traditional stateless offloads to avoid performance drop. ConnectX family cards offer the following stateless offloads for a VXLAN packet, similar to the ones offered to non-encapsulated packets. VXLAN protocol encapsulates its packets using outer UDP header.

Available hardware stateless offloads:

- Checksum generation (Inner IP and Inner TCP/UDP)
- Checksum validation (Inner IP and Inner TCP/UDP)
- TSO support for inner TCP packets
- RSS distribution according to inner packets attributes
- Receive queue selection - inner frames may be steered to specific QPs

3.3.2.3.1 Enabling VXLAN Hardware Stateless Offloads

VXLAN offload is enabled by default for ConnectX-4 family devices running the minimum required firmware version and a kernel version that includes VXLAN support.

To confirm if the current setup supports VXLAN, run:

```
ethtool -k $DEV | grep udp_tnl
```

Example:

```
ethtool -k ens1f0 | grep udp_tnl
tx-udp_tnl-segmentation: on
```

ConnectX-4 family devices support configuring multiple UDP ports for VXLAN offload. Ports can be added to the device by configuring a VXLAN device from the OS command line using the "ip" command.

Note: If you configure multiple UDP ports for offload and exceed the total number of ports supported by hardware, then those additional ports will still function properly, but will not benefit from any of the stateless offloads.

Example:

```
ip link add vxlan0 type vxlan id 10 group 239.0.0.10 ttl 10 dev ens1f0 dstport 4789
ip addr add 192.168.4.7/24 dev vxlan0
ip link set up vxlan0
```

Note: dstport' parameters are not supported in Ubuntu 14.4.

The VXLAN ports can be removed by deleting the VXLAN interfaces.

Example:

```
ip link delete vxlan0
```

3.3.2.3.2 Important Note

VXLAN tunneling adds 50 bytes (14-eth + 20-ip + 8-udp + 8-vxlan) to the VM Ethernet frame. Please verify that either the MTU of the NIC who sends the packets, e.g. the VM virtio-net NIC or the host side veth device or the uplink takes into account the tunneling overhead. Meaning, the MTU of the sending NIC has to be decremented by 50 bytes (e.g 1450 instead of 1500), or the uplink NIC MTU has to be incremented by 50 bytes (e.g 1550 instead of 1500)

3.3.2.4 Q-in-Q Encapsulation per VF in Linux (VST)

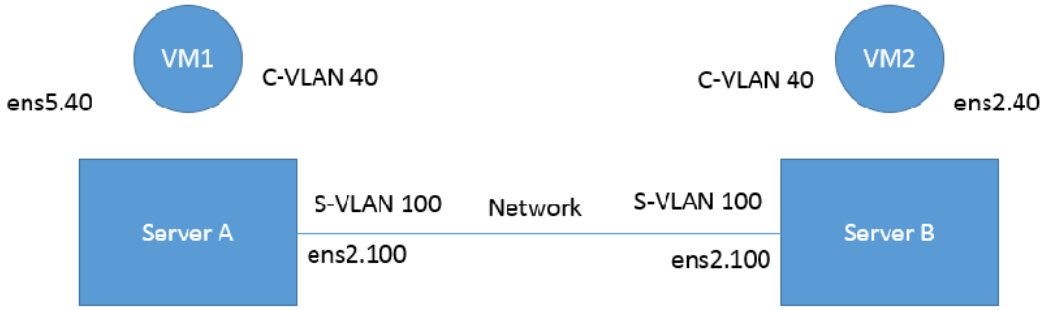
This feature is supported on ConnectX-5 and ConnectX-6 adapter cards only.

ConnectX-4 and ConnectX-4 Lx adapter cards support 802.1Q double-tagging (C-tag stacking on C-tag), refer to "802.1Q Double-Tagging" section.

This section describes the configuration of IEEE 802.1ad QinQ VLAN tag (S-VLAN) to the hypervisor per Virtual Function (VF). The Virtual Machine (VM) attached to the VF (via SR- IOV) can send traffic with or without C-VLAN. Once a VF is configured to VST QinQ encapsulation (VST QinQ), the adapter's hardware will insert S-VLAN to any packet from the VF to the physical port. On the receive side, the adapter hardware will strip the S-VLAN from any packet coming from the wire to that VF.

3.3.2.4.1 Setup

The setup assumes there are two servers equipped with ConnectX-5/ConnectX-6 adapter cards.



3.3.2.4.2 Prerequisites

- Kernel must be of v3.10 or higher, or custom/inbox kernel must support vlan-stag
- Firmware version 16/20.21.0458 or higher must be installed for ConnectX-5/ConnectX-6 HCAs
- The server should be enabled in SR-IOV and the VF should be attached to a VM on the hypervisor.
    - In order to configure SR-IOV in Ethernet mode for ConnectX-5/ConnectX-6 adapter cards, please refer to "Configuring SR-IOV for ConnectX-4/ConnectX-5 (Ethernet)" section. In the following configuration example, the VM is attached to VF0.
- Network Considerations - the network switches may require increasing the MTU (to support 1522 MTU size) on the relevant switch ports.

3.3.2.4.3 Configuring Q-in-Q Encapsulation per Virtual Function for ConnectX-5/ConnectX-6

1. Add the required S-VLAN (QinQ) tag (on the hypervisor) per port per VF. There are two ways to add the S-VLAN:
    a. By using sysfs:

```
echo '100:0:802.1ad' > /sys/class/net/ens1f0/device/sriov/0/vlan
```

    b. By using the ip link command (available only when using the latest Kernel version):

```
ip link set dev ens1f0 vf 0 vlan 100 proto 802.1ad
```

    Check the configuration using the ip link show command:

```
# ip link show ens1f0
 ens1f0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT qlen 1000
    link/ether ec:0d:9a:44:37:84 brd ff:ff:ff:ff:ff:ff
    vf 0 MAC 00:00:00:00:00:00, vlan 100, vlan protocol 802.1ad, spoof checking off, link-state
auto, trust off
    vf 1 MAC 00:00:00:00:00:00, spoof checking off, link-state auto, trust off
    vf 2 MAC 00:00:00:00:00:00, spoof checking off, link-state auto, trust off
    vf 3 MAC 00:00:00:00:00:00, spoof checking off, link-state auto, trust off
    vf 4 MAC 00:00:00:00:00:00, spoof checking off, link-state auto, trust off
```

2. **Optional:** Add S-VLAN priority. Use the qos parameter in the ip link command (or sysfs):

```
ip link set dev ens1f0 vf 0 vlan 100 qos 3 proto 802.1ad
```

Check the configuration using the ip link show command:

```
# ip link show ens1f0
ens1f0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT qlen 1000
    link/ether ec:0d:9a:44:37:84 brd ff:ff:ff:ff:ff:ff
    vf 0 MAC 00:00:00:00:00:00, vlan 100, qos 3, vlan protocol 802.1ad, spoof checking off, link-state
auto, trust off
    vf 1 MAC 00:00:00:00:00:00, spoof checking off, link-state auto, trust off
    vf 2 MAC 00:00:00:00:00:00, spoof checking off, link-state auto, trust off
    vf 3 MAC 00:00:00:00:00:00, spoof checking off, link-state auto, trust off
    vf 4 MAC 00:00:00:00:00:00, spoof checking off, link-state auto, trust off
```

3. Create a VLAN interface on the VM and add an IP address.

```
ip link add link ens5 ens5.40 type vlan protocol 802.1q id 40
ip addr add 42.134.135.7/16 brd 42.134.255.255 dev ens5.40
ip link set dev ens5.40 up
```

4. To verify the setup, run ping between the two VMs and open Wireshark or tcpdump to capture the packet.

3.3.2.5 802.1Q Double-Tagging

This section describes the configuration of 802.1Q double-tagging support to the hypervisor per Virtual Function (VF). The Virtual Machine (VM) attached to the VF (via SR-IOV) can send traffic with or without C-VLAN. Once a VF is configured to VST encapsulation, the adapter's hardware will insert C-VLAN to any packet from the VF to the physical port. On the receive side, the adapter hardware will strip the C-VLAN from any packet coming from the wire to that VF.

3.3.2.5.1 Configuring 802.1Q Double-Tagging per Virtual Function

1. Add the required C-VLAN tag (on the hypervisor) per port per VF. There are two ways to add the C-VLAN:
    a. By using sysfs:

```
echo '100:0:802.1q' > /sys/class/net/ens1f0/device/sriov/0/vlan
```

    b. By using the ip link command (available only when using the latest Kernel version):

```
ip link set dev ens1f0 vf 0 vlan 100
```

Check the configuration using the ip link show command:

```
# ip link show ens1f0
 ens1f0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT qlen 1000
    link/ether ec:0d:9a:44:37:84 brd ff:ff:ff:ff:ff:ff
    vf 0 MAC 00:00:00:00:00:00, vlan 100, spoof checking off, link-state auto, trust off
    vf 1 MAC 00:00:00:00:00:00, spoof checking off, link-state auto, trust off
    vf 2 MAC 00:00:00:00:00:00, spoof checking off, link-state auto, trust off
    vf 3 MAC 00:00:00:00:00:00, spoof checking off, link-state auto, trust off
    vf 4 MAC 00:00:00:00:00:00, spoof checking off, link-state auto, trust off
```

2. Create a VLAN interface on the VM and add an IP address.

```
# ip link add link ens5 ens5.40 type vlan protocol 802.1q id 40
# ip addr add 42.134.135.7/16 brd 42.134.255.255 dev ens5.40
# ip link set dev ens5.40 up
```

3. To verify the setup, run ping between the two VMs and open Wireshark or tcpdump to capture the packet.

3.3.2.6 Scalable Functions

Scalable function is a lightweight function that has a parent PCI function on which it is deployed. Scalable functions are useful for containers where netdevice and RDMA devices of a scalable function can be assigned to a container. This way, the container can get complete offload capabilities of an eswitch, isolation and dedicated accelerated network device. For Step-by-Step Configuration instructions, follow the User Guide here.

3.3.3 Resiliency

The chapter contains the following sections:

- Reset Flow

3.3.3.1 Reset Flow

Reset Flow is activated by default. Once a "fatal device" error is recognized, both the HCA and the software are reset, the ULPs and user application are notified about it, and a recovery process is performed once the event is raised.

Currently, a reset flow can be triggered by a firmware assert with Recover Flow Request (RFR) only. Firmware RFR support should be enabled explicitly using mlxconfig commands.

To query the current value, run:

```
mlxconfig -d /dev/mst/mt4115_pciconf0 query | grep SW_RECOVERY_ON_ERRORS
```

To enable RFR bit support, run:

```
mlxconfig -d /dev/mst/mt4115_pciconf0 set SW_RECOVERY_ON_ERRORS=true
```

3.3.3.1.1 Kernel ULPs

Once a "fatal device" error is recognized, an IB_EVENT_DEVICE_FATAL event is created, ULPs are notified about the incident, and outstanding WQEs are simulated to be returned with "flush in error"

message to enable each ULP to close its resources and not get stuck via calling its "remove_one" callback as part of "Reset Flow".
Once the unload part is terminated, each ULP is called with its "`add_one`" callback, its resources are re-initialized and it is re-activated.

3.3.3.1.2 SR-IOV

If the Physical Function recognizes the error, it notifies all the VFs about it by marking their communication channel with that information, consequently, all the VFs and the PF are reset.
If the VF encounters an error, only that VF is reset, whereas the PF and other VFs continue to work unaffected.

3.3.3.1.3 Forcing the VF to Reset

If an outside "reset" is forced by using the PCI sysfs entry for a VF, a reset is executed on that VF once it runs any command over its communication channel.
For example, the below command can be used on a hypervisor to reset a VF defined by 0000:04:00.1:

```
echo 1 >/sys/bus/pci/devices/0000:04:00.1/reset
```

3.3.3.1.4 Extended Error Handling (EEH)

Extended Error Handling (EEH) is a PowerPC mechanism that encapsulates AER, thus exposing AER events to the operating system as EEH events.
The behavior of ULPs and user space applications is identical to the behavior of AER.

3.3.3.1.5 CRDUMP

CRDUMP feature allows for taking an automatic snapshot of the device CR-Space in case the device's FW/HW fails to function properly.

Snapshots Triggers:

The snapshot is triggered after firmware detects a critical issue, requiring a recovery flow.

This snapshot can later be investigated and analyzed to track the root cause of the failure.
Currently, only the first snapshot is stored, and is exposed using a temporary virtual file. The virtual file is cleared upon driver reset.
When a critical event is detected, a message indicating CRDUMP collection will be printed to the Linux log. User should then back up the file pointed to in the printed message. The file location format is: /proc/driver/mlx5_core/crdump/<pci address>

Snapshot should be copied by Linux standard tool for future investigation.

3.3.3.1.6 Firmware Tracer

This mechanism allows for the device's FW/HW to log important events into the event tracing system (/sys/kernel/debug/tracing) without requiring any NVIDIA tool.

> ⚠ To be able to use this feature, trace points must be enabled in the kernel.

This feature is enabled by default, and can be controlled using sysfs commands.

To disable the feature:

```
echo 0 > /sys/kernel/debug/tracing/events/mlx5/fw_tracer/enable
```

To enable the feature:

```
echo 1 > /sys/kernel/debug/tracing/events/mlx5/fw_tracer/enable
```

To view FW traces using vim text editor:

```
vim /sys/kernel/debug/tracing/trace
```

3.3.4 Docker Containers

On Linux, Docker uses resource isolation of the Linux kernel, to allow independent "containers" to run within a single Linux kernel instance.
Docker containers are supported on MLNX-EN using Docker runtime. Virtual RoCE and InfiniBand devices are supported using SR-IOV mode.

Currently, RDMA/RoCE devices are supported in the modes listed in the following table.

Linux Containers Networking Modes

Orchestration and Clustering Tool	Version	Networking Mode	Link Layer	Virtualization Mode
Docker	Docker Engine 17.03 or higher	SR-IOV using sriov-plugin along with docker run wrapper tool	InfiniBand and Ethernet	SR-IOV

Orchestration and Clustering Tool	Version	Networking Mode	Link Layer	Virtualization Mode
Kubernetes	Kubernetes 1.10.3 or higher	SR-IOV using device plugin, and using SR- IOV CNI plugin	InfiniBand and Ethernet	SR-IOV
		VXLAN using IPoIB bridge	InfiniBand	Shared HCA

3.3.4.1 Docker Using SR-IOV

In this mode, Docker engine is used to run containers along with SR-IOV networking plugin. To isolate the virtual devices, docker_rdma_sriov tool should be used. This mode is applicable to both InfiniBand and Ethernet link layers.
To obtain the plugin, visit: hub.docker.com/r/rdma/sriov-plugin
To install the docker_rdma_sriov tool, use the container tools installer available via hub.docker.com/r/rdma/container_tools_installer
For instructions on how to use Docker with SR-IOV, refer to Docker RDMA SRIOV Networking with ConnectX4/ConnectX5/ConnectX6 Community post.

3.3.4.2 Kubernetes Using SR-IOV

In order to use RDMA in Kubernetes environment with SR-IOV networking mode, two main components are required:

1. RDMA device plugin - this plugin allows for exposing RDMA devices in a Pod
2. SR-IOV CNI plugin - this plugin provisions VF net device in a Pod

When used in SR-IOV mode, this plugin enables SR-IOV and performs necessary configuration including setting GUID, MAC, privilege mode, and Trust mode.
The plugin also allocates the VF devices when Pods are scheduled and requested by Kubernetes framework.

3.3.4.3 Kubernetes with Shared HCA

One RDMA device (HCA) can be shared among multiple Pods running in a Kubernetes worker nodes. User defined networks are created using VXLAN or VETH networking devices. RDMA device (HCA) can be shared among multiple Pods running in a Kubernetes worker nodes.

3.3.5 Fast Driver Unload

This feature enables optimizing mlx5 driver teardown time in shutdown and kexec flows.

The fast driver unload is disabled by default. To enable it, the `prof_sel` module parameter of mlx5_core module should be set to 3.

3.3.6 OVS Offload Using ASAP² Direct

3.3.6.1 Overview

> ⚠ Supported on ConnectX-5 and above adapter cards.

Open vSwitch (OVS) allows Virtual Machines (VMs) to communicate with each other and with the outside world. OVS traditionally resides in the hypervisor and switching is based on twelve tuple matching on flows. The OVS software based solution is CPU intensive, affecting system performance and preventing full utilization of the available bandwidth.
NVIDIA Accelerated Switching And Packet Processing (ASAP^2) technology allows OVS offloading by handling OVS data-plane in ConnectX-5 onwards NIC hardware (Embedded Switch or eSwitch) while maintaining OVS control-plane unmodified. As a result, we observe significantly higher OVS performance without the associated CPU load.

As of v5.0, OVS-DPDK became part of MLNX_EN package. OVS-DPDK supports ASAP^2 just as the OVS-Kernel (Traffic Control (TC) kernel-based solution) does, yet with a different set of features.

The traditional ASAP^2 hardware data plane is built over SR-IOV virtual functions (VFs), so that the VF is passed through directly to the VM, with the NVIDIA driver running within the VM. An alternate approach that is also supported is vDPA (vhost Data Path Acceleration). vDPA allows the connection to the VM to be established using VirtIO, so that the data-plane is built between the SR-IOV VF and the standard VirtIO driver within the VM, while the control-plane is managed on the host by the vDPA application. Two flavors of vDPA are supported, Software vDPA; and Hardware vDPA. Software vDPA management functionality is embedded into OVS-DPDK, while Hardware vDPA uses a standalone application for management, and can be run with both OVS-Kernel and OVS-DPDK. For further information, please see sections VirtIO Acceleration through VF Relay (Software vDPA) and VirtIO Acceleration through Hardware vDPA.

3.3.6.2 Installing OVS-Kernel ASAP² Packages

Install the required packages. For the complete solution, you need to install supporting MLNX_EN (v4.4 and above), iproute2, and openvswitch packages.

3.3.6.3 Installing OVS-DPDK ASAP² Packages

Run:

```
./install --ovs-dpdk -upstream-libs
```

3.3.6.4 Setting Up SR-IOV

> ⚠ Note that this section applies to both OVS-DPDK and OVS-Kernel similarly.

To set up SR-IOV:

1. Choose the desired card.
   The example below shows a dual-ported ConnectX-5 card (device ID 0x1017) and a single SR-IOV VF (Virtual Function, device ID 0x1018).
   In SR-IOV terms, the card itself is referred to as the PF (Physical Function).

```
# lspci -nn | grep Mellanox

0a:00.0 Ethernet controller [0200]: Mellanox Technologies MT27800 Family [ConnectX-5] [15b3:1017]
0a:00.1 Ethernet controller [0200]: Mellanox Technologies MT27800 Family [ConnectX-5] [15b3:1017]

0a:00.2 Ethernet controller [0200]: Mellanox Technologies MT27800 Family [ConnectX-5 Virtual Function]
[15b3:1018]
```

   ⚠ Enabling SR-IOV and creating VFs is done by the firmware upon admin directive as explained in Step 5 below.

2. Identify the NVIDIA NICs and locate net-devices which are on the NIC PCI BDF.

```
# ls -l /sys/class/net/ | grep 04:00

lrwxrwxrwx 1 root root 0 Mar 27 16:58 enp4s0f0 -> ../../devices/pci0000:00/0000:00:03.0/0000:04:00.0/net/
enp4s0f0
lrwxrwxrwx 1 root root 0 Mar 27 16:58 enp4s0f1 -> ../../devices/pci0000:00/0000:00:03.0/0000:04:00.1/net/
enp4s0f1
lrwxrwxrwx 1 root root 0 Mar 27 16:58 eth0 -> ../../devices/pci0000:00/0000:00:03.0/0000:04:00.2/net/eth0
lrwxrwxrwx 1 root root 0 Mar 27 16:58 eth1 -> ../../devices/pci0000:00/0000:00:03.0/0000:04:00.3/net/eth1
```

   The PF NIC for port #1 is enp4s0f0, and the rest of the commands will be issued on it.
3. Check the firmware version.
   Make sure the firmware versions installed are as state in the Release Notes document.

```
# ethtool -i enp4s0f0 | head -5
driver: mlx5_core
version: 5.0-5
firmware-version: 16.21.0338
expansion-rom-version:
bus-info: 0000:04:00.0
```

4. Make sure SR-IOV is enabled on the system (server, card).
   Make sure SR-IOV is enabled by the server BIOS, and by the firmware with up to N VFs, where N is the number of VFs required for your environment. Refer to "NVIDIA Firmware Tools" below for more details.

```
# cat /sys/class/net/enp4s0f0/device/sriov_totalvfs
4
```

5. Turn ON SR-IOV on the PF device.

```
# echo 2 > /sys/class/net/enp4s0f0/device/sriov_numvfs
```

6. Provision the VF MAC addresses using the IP tool.

```
# ip link set enp4s0f0 vf 0 mac e4:11:22:33:44:50
# ip link set enp4s0f0 vf 1 mac e4:11:22:33:44:51
```

7. Verify the VF MAC addresses were provisioned correctly and SR-IOV was turned ON.

```
# cat /sys/class/net/enp4s0f0/device/sriov_numvfs
2

# ip link show dev enp4s0f0
256: enp4s0f0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq master ovs-system state UP mode DEFAULT
group default qlen 1000
    link/ether e4:1d:2d:60:95:a0 brd ff:ff:ff:ff:ff:ff
    vf 0 MAC e4:11:22:33:44:50, spoof checking off, link-state auto
```

```
    vf 1 MAC e4:11:22:33:44:51, spoof checking off, link-state auto
```

In the example above, the maximum number of possible VFs supported by the firmware is 4 and only 2 are enabled.

8. Provision the PCI VF devices to VMs using PCI Pass-Through or any other preferred virt tool of choice, e.g virt-manager.

For further information on SR-IOV, refer to HowTo Configure SR-IOV for ConnectX-4/ConnectX-5/ ConnectX-6 with KVM (Ethernet).

3.3.6.5 OVS Hardware Offloads Configuration

3.3.6.5.1 OVS-Kernel Hardware Offloads

3.3.6.5.1.1 SwitchDev Configuration

1. Unbind the VFs.

```
echo 0000:04:00.2 > /sys/bus/pci/drivers/mlx5_core/unbind
echo 0000:04:00.3 > /sys/bus/pci/drivers/mlx5_core/unbind
```

⚠ VMs with attached VFs must be powered off to be able to unbind the VFs.

2. Change the eSwitch mode from Legacy to SwitchDev on the PF device.
   This will also create the VF representor netdevices in the host OS.

```
# devlink dev eswitch set pci/0000:3b:00.0 mode switchdev
```

⚠ Before changing the mode, make sure that all VFs are unbound.

⚠ To go back to SR-IOV legacy mode, run:
`# devlink dev eswitch set pci/0000:3b:00.0 mode legacy`
This will also remove the VF representor netdevices.

On old OSs or kernels that do not support Devlink, moving to SwitchDev mode can be done using sysfs.

```
# echo switchdev > /sys/class/net/enp4s0f0/compat/devlink/mode
```

3. At this stage, VF representors have been created. To map representor to its VF, make sure to obtain the representor's switchid and portname from:

```
# ip -d link show eth4
41: enp0s8f0_1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default
 qlen 1000
    link/ether ba:e6:21:37:bc:d4 brd ff:ff:ff:ff:ff:ff promiscuity 0 addrgenmode eui64 numtxqueues 10
 numrxqueues 10 gso_max_size 65536 gso_max_segs 65535 portname pf0vf1 switchid f4ab580003a1420c
```

switchid - used to map representor to device, both device PFs have the same switchid.
portname - used to map representor to PF and VF, value returned is pfXvfY, where X is the PF number and Y is the number of VF.
On old kernels, switchid and portname can be acquired through sysfs:

4. Bind the VFs.

```
echo 0000:04:00.2 > /sys/bus/pci/drivers/mlx5_core/bind
echo 0000:04:00.3 > /sys/bus/pci/drivers/mlx5_core/bind
```

3.3.6.5.1.2 SwitchDev Performance Tuning

SwitchDev performance can be further improved by tuning it.

Steering Mode

OVS-kernel supports two steering modes for rules insertion into hardware.

1. SMFS – Software Managed Flow Steering (as of MLNX_OFED v5.1, this is the default mode)
   Rules are inserted directly to the hardware by the software (driver). This mode is optimized for rules insertion.
2. DMFS – Device Managed Flow Steering
   Rules insertion is done using firmware commands. This mode is optimized for throughput with a small amount of rules in the system.
   The mode can be controlled via sysfs or devlink API in kernels that support it:

```
Sysfs:
# echo smfs > /sys/class/net/<PF netdev>/compat/devlink/steering_mode

Devlink:
# devlink dev param set pci/0000:00:08.0 name flow_steering_mode value "smfs" cmode runtime

Replace smfs param with dmfs for device managed flow steering
```

Notes:

- The mode should be set prior to moving to SwitchDev, by echoing to the sysfs or invoking the devlink command.
- Only when moving to SwitchDev will the driver use the mode set by the previous step.
- Mode cannot be changed after moving to SwitchDev.
- The steering mode is applicable for SwitchDev mode only, meaning it does not affect legacy SR-IOV or other configurations.

Troubleshooting SMFS

mlx5 debugfs was extended to support presenting Software Steering resources:
dr_domain including it's tables, matchers and rules.
The interface is read-only.

While dump is being created, new steering rules cannot be inserted/deleted.
The steering information is dumped in the CSV form with the following format:
`<object_type>,<object_ID>, <object_info>,...,<object_info>`

This data can be read at the following path: `/sys/kernel/debug/mlx5/<BDF>/steering/fdb/<domain_handle>`

Example:

```
# cat /sys/kernel/debug/mlx5/0000:82:00.0/steering/fdb/dmn_000018644
3100,0x55caa4621c50,0xee802,4,65533
3101,0x55caa4621c50,0xe0100008
```

You can then use the steering dump parser to make the output more human readable.
The parser can be found in the following public GitHub repository: https://github.com/Mellanox/mlx_steering_dump

vPort Match Mode

OVS-kernel support two modes that define how the rules on match on vport.

1. Metadata – rules match on metadata instead of vport number (default mode).
   This mode is needed in order to support SR-IOV Live migration and Dual port RoCE features.
   Matching on Metadata can have a performance impact.
2. Legacy – rules match on vport number.
   In this mode, performance can be higher in comparison to Metadata. It can still be used only if none of the above features (SR-IOV Live migration and Dual port RoCE) is enabled/used.
   The mode can be controlled via sysfs:

   ```
   Set Legacy:
   # echo legacy > /sys/class/net/<PF netdev>/compat/devlink/vport_match_mode

   Set metadata:
   Devlink:
   # echo metadata > /sys/class/net/<PF netdev>/compat/devlink/vport_match_mode
   ```

   Note: This mode should be set prior to moving to SwitchDev, by echoing to the sysfs.

Flow Table Large Group Number

Offloaded flows, including Connection Tracking, are added to Virtual Switch Forwarding Data Base (FDB) flow tables. FDB tables have a set of flow groups, where each flow group saves the same traffic pattern flows. E.g, for connection tracking offloaded flow, TCP and UDP are different traffic patterns which will end up in two different flow groups.

A flow group has a limited size to save flow entries. As default, the driver has 15 big FDB flow groups. Each of these big flow groups can save 4M / (15 + 1) = 256k different 5-tuple flow entries at most. For scenarios with more than 15 traffic patterns, the driver provides a module parameter (num_of_groups) to allow customization and performance tuning.

The mode can be controlled via module param or devlink API for kernels that support it:

```
Module param:
# echo <num_of_groups> > /sys/module/mlx5_core/parameters/num_of_groups

Devlink:
# devlink dev param set pci/0000:82:00.0 name fdb_large_groups \
      cmode driverinit value 20
```

Notes:

- In MLNX_OFED v5.1, the default value was changed from 4 to 15.

- The change takes effect immediately if there is no flow inside the FDB table (no traffic running and all offloaded flows are aged out). And it can be dynamically changed without reloading the driver.
  If there are still offloaded flows residual when changing this parameter, it will only take effect after all flows have aged out.

3.3.6.5.1.3 Open vSwitch Configuration

Open vSwitch configuration is a simple OVS bridge configuration with SwitchDev.

1. Run the openvswitch service.

```
# systemctl start openvswitch
```

2. Create an OVS bridge (here it's named ovs-sriov).

```
# ovs-vsctl add-br ovs-sriov
```

3. Enable hardware offload (disabled by default).

```
# ovs-vsctl set Open_vSwitch . other_config:hw-offload=true
```

4. Restart the openvswitch service. This step is required for HW offload changes to take effect.

```
# systemctl restart openvswitch
```

> HW offload policy can also be changed by setting the tc-policy using one on the following values:
>
> * none - adds a TC rule to both the software and the hardware (default)
>
> * skip_sw - adds a TC rule only to the hardware
>
> * skip_hw - adds a TC rule only to the software
>
> The above change is used for debug purposes.

5. Add the PF and the VF representor netdevices as OVS ports.

```
# ovs-vsctl add-port ovs-sriov enp4s0f0
# ovs-vsctl add-port ovs-sriov enp4s0f0_0
# ovs-vsctl add-port ovs-sriov enp4s0f0_1
```

Make sure to bring up the PF and representor netdevices.

```
# ip link set dev enp4s0f0 up
# ip link set dev enp4s0f0_0 up
# ip link set dev enp4s0f0_1 up
```

The PF represents the uplink (wire).

```
# ovs-dpctl show
system@ovs-system:
        lookups: hit:0 missed:192 lost:1
        flows: 2
        masks: hit:384 total:2 hit/pkt:2.00
        port 0: ovs-system (internal)
        port 1: ovs-sriov (internal)
        port 2: enp4s0f0
        port 3: enp4s0f0_0
```

```
        port 4: enp4s0f0_1
```

6. Run traffic from the VFs and observe the rules added to the OVS data-path.

```
# ovs-dpctl dump-flows

recirc_id(0),in_port(3),eth(src=e4:11:22:33:44:50,dst=e4:1d:2d:a5:f3:9d),
eth_type(0x0800),ipv4(frag=no), packets:33, bytes:3234, used:1.196s, actions:2

recirc_id(0),in_port(2),eth(src=e4:1d:2d:a5:f3:9d,dst=e4:11:22:33:44:50),
eth_type(0x0800),ipv4(frag=no), packets:34, bytes:3332, used:1.196s, actions:3
```

In the example above, the ping was initiated from VF0 (OVS port 3) to the outer node (OVS port 2), where the VF MAC is e4:11:22:33:44:50 and the outer node MAC is e4:1d:2d:a5:f3:9d As shown above, two OVS rules were added, one in each direction.
Note that you can also verify offloaded packets by adding type=offloaded to the command. For example:

```
# ovs-appctl dpctl/dump-flows type=offloaded
```

3.3.6.5.1.4 Open vSwitch Performance Tuning

Flow Aging

The aging timeout of OVS is given in ms and can be controlled using the following command.

```
# ovs-vsctl set Open_vSwitch . other_config:max-idle=30000
```

TC Policy

Specifies the policy used with HW offloading.

- none - adds a TC rule to both the software and the hardware (default)
- skip_sw - adds a TC rule only to the hardware
- skip_hw - adds a TC rule only to the software

Example:

```
# ovs-vsctl set Open_vSwitch . other_config:tc-policy=skip_sw
```

Note: TC policy should only be used for debugging purposes.

Max-Revalidator

Specifies the maximum time (in ms) that revalidator threads will wait for kernel statistics before executing flow revalidation.

```
# ovs-vsctl set Open_vSwitch . other_config:max-revalidator=10000
```

n-handler-threads

Specifies the number of threads for software datapaths to use for handling new flows.
The default value is the number of online CPU cores minus the number of revalidators.

```
# ovs-vsctl set Open_vSwitch . other_config:n-handler-threads=4
```

n-revalidator-threads

Specifies the number of threads for software datapaths to use for revalidating flows in the datapath.

```
# ovs-vsctl set Open_vSwitch . other_config:n-revalidator-threads=4
```

vlan-limit

Limits the number of VLAN headers that can be matched to the specified number.

```
# ovs-vsctl set Open_vSwitch . other_config:vlan-limit=2
```

3.3.6.5.1.5 Basic TC Rules Configuration

Offloading rules can also be added directly, and not only through OVS, using the tc utility.

To create an offloading rule using TC:

1. Create an ingress qdisc (queueing discipline) for each interface that you wish to add rules into.

   ```
   # tc qdisc add dev enp4s0f0 ingress
   # tc qdisc add dev enp4s0f0_0 ingress
   # tc qdisc add dev enp4s0f0_1 ingress
   ```

2. Add TC rules using flower classifier in the following format.

   ```
   # tc filter add dev NETDEVICE ingress protocol PROTOCOL prio PRIORITY \
   [chain CHAIN] flower [ MATCH_LIST ] [ action ACTION_SPEC ]
   ```

   Note: List of supported matches (specifications) and actions can be found in Classification Fields (Matches) section.

3. Dump the existing tc rules using flower classifier in the following format.

   ```
   # tc [ -s ] filter show dev NETDEVICE ingress
   ```

3.3.6.5.1.6 SR-IOV VF LAG

SR-IOV VF LAG allows the NIC's physical functions (PFs) to get the rules that the OVS will try to offload to the bond net-device, and to offload them to the hardware e-switch. Bond modes supported are:

- Active-Backup
- XOR
- LACP

SR-IOV VF LAG enables complete offload of the LAG functionality to the hardware. The bonding creates a single bonded PF port. Packets from up-link can arrive from any of the physical ports, and will be forwarded to the bond device.

When hardware offload is used, packets from both ports can be forwarded to any of the VFs. Traffic from the VF can be forwarded to both ports according to the bonding state. Meaning, when in

active-backup mode, only one PF is up, and traffic from any VF will go through this PF. When in XOR or LACP mode, if both PFs are up, traffic from any VF will split between these two PFs.

SR-IOV VF LAG Configuration on ASAP^2

To enable SR-IOV VF LAG, both physical functions of the NIC should first be configured to SR-IOV SwitchDev mode, and only afterwards bond the up-link representors.

The example below shows the creation of bond interface on two PFs:

1. Load bonding device and enslave the up-link representor (currently PF) net-device devices.

```
modprobe bonding mode=802.3ad
Ifup bond0 (make sure ifcfg file is present with desired bond configuration)
ip link set enp4s0f0 master bond0
ip link set enp4s0f1 master bond0
```

2. Add the VF representor net-devices as OVS ports. If tunneling is not used, add the bond device as well.

```
ovs-vsctl add-port ovs-sriov bond0
ovs-vsctl add-port ovs-sriov enp4s0f0_0
ovs-vsctl add-port ovs-sriov enp4s0f1_0
```

3. Make sure to bring up the PF and the representor netdevices.

```
ip link set dev bond0 up
ip link set dev enp4s0f0_0 up
ip link set dev enp4s0f1_0 up
```

Once SR-IOV VF LAG is configured, all VFs of the two PFs will become part of the bond, and will behave as described above.

Limitations

- In VF LAG mode, outgoing traffic in load balanced mode is according to the origin ring, thus, half of the rings will be coupled with port 1 and half with port 2. All the traffic on the same ring will be sent from the same port.
- VF LAG configuration is not supported when the NUM_OF_VFS configured in mlxconfig is higher than 64.

Using TC with VF LAG

Both rules can be added using either of the following.

1. Shared block (supported from kernel 4.16 and RHEL/CentOS 7.7 and above).

```
# tc qdisc add dev bond0 ingress_block 22 ingress
# tc qdisc add dev ens4p0 ingress_block 22 ingress
# tc qdisc add dev ens4p1 ingress_block 22 ingress
```

   a. Add drop rule.

```
# tc filter add block 22 protocol arp parent ffff: prio 3 \
    flower \
        dst_mac e4:11:22:11:4a:51 \
        action drop
```

b. Add redirect rule from bond to representor.

```
# tc filter add block 22 protocol arp parent ffff: prio 3 \
        flower \
        dst_mac e4:11:22:11:4a:50 \
        action mirred egress redirect dev ens4f0_0
```

c. Add redirect rule from representor to bond.

```
# tc filter add dev ens4f0_0 protocol arp parent ffff: prio 3 \
    flower \
        dst_mac ec:0d:9a:8a:28:42 \
        action mirred egress redirect dev bond0
```

2. Without shared block (supported from kernel 4.15 and below).
   a. Add redirect rule from bond to representor.

```
# tc filter add dev bond0 protocol arp parent ffff: prio 1 \
        flower \
        dst_mac e4:11:22:11:4a:50 \
        action mirred egress redirect dev ens4f0_0
```

   b. Add redirect rule from representor to bond.

```
# tc filter add dev ens4f0_0 protocol arp parent ffff: prio 3 \
        flower \
        dst_mac ec:0d:9a:8a:28:42 \
        action mirred egress redirect dev bond0
```

3.3.6.5.1.7 Classification Fields (Matches)

OVS-Kernel supports multiple classification fields which packets can fully or partially match.

Ethernet Layer 2

- Destination MAC
- Source MAC
- Ethertype

Supported on all kernels.

In OVS dump flows:

```
skb_priority(0/0),skb_mark(0/0),in_port(eth6),eth(src=00:02:10:40:10:0d
,dst=68:54:ed:00:af:de),eth_type(0x8100), packets:1981, bytes:206024, used:0.440s, dp:tc, actions:eth7
```

Using TC rules:

```
tc filter add dev $rep parent ffff: protocol arp pref 1 \
flower \
dst_mac e4:1d:2d:5d:25:35 \
src_mac e4:1d:2d:5d:25:34 \
action mirred egress redirect dev $NIC
```

IPv4/IPv6

- Source address
- Destination address
- Protocol
  - TCP/UDP/ICMP/ICMPv6

- TOS
- TTL (HLIMIT)

Supported on all kernels.

In OVS dump flows:

```
Ipv4:
ipv4(src=0.0.0.0/0.0.0.0,dst=0.0.0.0/0.0.0.0,proto=17,tos=0/0,ttl=0/0,frag=no)
Ipv6:
ipv6(src=::/::,dst=1:1:1::3:1040:1008,label=0/0,proto=58,tclass=0/0x3,hlimit=64),
```

Using TC rules:

```
IPv4:
tc filter add dev $rep parent ffff: protocol ip pref 1 \
flower \
dst_ip 1.1.1.1 \
src_ip 1.1.1.2 \
ip_proto TCP \
ip_tos 0x3 \
ip_ttl 63 \
action mirred egress redirect dev $NIC


IPv6:
tc filter add dev $rep parent ffff: protocol ipv6 pref 1 \
flower \
dst_ip 1:1:1::3:1040:1009 \
src_ip 1:1:1::3:1040:1008 \
ip_proto TCP \
ip_tos 0x3 \
ip_ttl 63\
action mirred egress redirect dev $NIC
```

TCP/UDP Source and Destination ports & TCP Flags

- TCP/UDP source and destinations ports
- TCP flags

Supported kernels are kernel > 4.13 and RHEL > 7.5

In OVS dump flows:

```
TCP: tcp(src=0/0,dst=32768/0x8000),
UDP: udp(src=0/0,dst=32768/0x8000),
TCP flags: tcp_flags(0/0)
```

Using TC rules:

```
tc filter add dev $rep parent ffff: protocol ip pref 1 \
flower \
ip_proto TCP \
dst_port 100 \
src_port 500 \
tcp_flags 0x4/0x7 \
action mirred egress redirect dev $NIC
```

VLAN

- ID
- Priority
- Inner vlan ID and Priority

Supported kernels: All (QinQ: kernel 4.19 and higher, and RHEL 7.7 and higher)

In OVS dump flows:

```
eth_type(0x8100),vlan(vid=2347,pcp=0),
```

Using TC rules:

```
tc filter add dev $rep parent ffff: protocol 802.1Q pref 1 \
                    flower \
                    vlan_ethtype 0x800 \
                    vlan_id 100 \
                    vlan_prio 0 \
             action mirred egress redirect dev $NIC
QinQ:
tc filter add dev $rep parent ffff: protocol 802.1Q pref 1 \
                    flower \
                    vlan_ethtype 0x8100  \
                    vlan_id 100 \
                    vlan_prio 0 \
                    cvlan_id 20 \
                    cvlan_prio 0 \
                    cvlan_ethtype 0x800 \
             action mirred egress redirect dev $NIC
```

Tunnel

- ID (Key)
- Source IP address
- Destination IP address
- Destination port
- TOS (supported from kernel 4.19 and above & RHEL 7.7 and above)
- TTL (support from kernel 4.19 and above & RHEL 7.7 and above)
- Tunnel options (Geneve)

Supported kernels:

- VXLAN: All
- GRE: Kernel > 5.0, RHEL 7.7 and above
- Geneve: Kernel > 5.0, RHEL 7.7 and above

In OVS dump flows:

```
tunnel(tun_id=0x5,src=121.9.1.1,dst=131.10.1.1,ttl=0/0,tp_dst=4789,flags(+key))
```

Using TC rules:

```
# tc filter add dev $rep protocol 802.1Q parent ffff: pref 1
flower \
vlan_ethtype 0x800 \
vlan_id 100 \
vlan_prio 0 \
action mirred egress redirect dev $NIC
QinQ:
# tc filter add dev vxlan100 protocol ip parent ffff: \
                flower \
                          skip_sw \
                          dst_mac e4:11:22:11:4a:51 \
                          src_mac e4+:11:22:11:4a:50 \
                          enc_src_ip 20.1.11.1 \
                          enc_dst_ip 20.1.12.1 \
                          enc_key_id 100 \
                          enc_dst_port 4789 \
                action tunnel_key unset \
                action mirred egress redirect dev ens4f0_0
```

3.3.6.5.1.8 Supported Actions

Forward

Forward action allows for packet redirection:

- From VF to wire
- Wire to VF
- VF to VF

Supported on all kernels.

In OVS dump flows:

```
skb_priority(0/0),skb_mark(0/0),in_port(eth6),eth(src=00:02:10:40:10:0d
,dst=68:54:ed:00:af:de),eth_type(0x8100), packets:1981, bytes:206024, used:0.440s, dp:tc, actions:eth7
```

Using TC rules:

```
tc filter add dev $rep parent ffff: protocol arp pref 1 \
                        flower \
                        dst_mac e4:1d:2d:5d:25:35 \
                        src_mac e4:1d:2d:5d:25:34 \
                    action mirred egress redirect dev $NIC
```

Drop

Drop action allows to drop incoming packets.

Supported on all kernels.

In OVS dump flows:

```
skb_priority(0/0),skb_mark(0/0),in_port(eth6),eth(src=00:02:10:40:10:0d
,dst=68:54:ed:00:af:de),eth_type(0x8100), packets:1981, bytes:206024, used:0.440s, dp:tc, actions:drop
```

Using TC rules:

```
tc filter add dev $rep parent ffff: protocol arp pref 1 \
                            flower \
                            dst_mac e4:1d:2d:5d:25:35 \
                            src_mac e4:1d:2d:5d:25:34 \
                    action drop
```

Statistics

By default, each flow collects the following statistics:

- Packets - number of packets which hit the flow
- Bytes - total number of bytes which hit the flow
- Last used - the amount of time passed since last packet hit the flow

Supported on all kernels.

In OVS dump flows:

```
skb_priority(0/0),skb_mark(0/0),in_port(eth6),eth(src=00:02:10:40:10:0d
,dst=68:54:ed:00:af:de),eth_type(0x8100), packets:1981, bytes:206024, used:0.440s, dp:tc, actions:drop
```

Using TC rules:

```
#tc -s filter show dev $rep ingress

filter protocol ip pref 2 flower chain 0
filter protocol ip pref 2 flower chain 0 handle 0x2
```

```
eth_type ipv4
ip_proto tcp
src_ip 192.168.140.100
src_port 80
skip_sw
in_hw
    action order 1: mirred (Egress Redirect to device p0v11_r) stolen
    index 34 ref 1 bind 1 installed 144 sec used 0 sec
    Action statistics:
    Sent 388344 bytes 2942 pkt (dropped 0, overlimits 0 requeues 0)
    backlog 0b 0p requeues 0
```

Tunnels (Encapsulation/Decapsulation)

OVS-kernel supports offload of tunnels using encapsulation and decapsulation actions.

- Encapsulation – pushing of tunnel header is supported on Tx
- Decapsulation – popping of tunnel header is supported on Rx

Supported Tunnels:

- VXLAN (IPv4/IPv6) – supported on all Kernels
- GRE (IPv4/IPv6) – supported on kernel 5.0 and above & RHEL 7.6 and above
- Geneve (IPv4/IPv6) - supported on kernel 5.0 and above & RHEL 7.6 and above

OVS configuration:
In case of offloading tunnel, the PF/bond should not be added as a port in the OVS datapath. It should rather be assigned with the IP address to be used for encapsulation.
The example below shows two hosts (PFs) with IPs 1.1.1.177 and 1.1.1.75, where the PF device on both hosts is enp4s0f0, and the VXLAN tunnel is set with VNID 98:

- On the first host:

```
# ip addr add 1.1.1.177/24 dev enp4s0f1

# ovs-vsctl add-port ovs-sriov vxlan0 -- set interface vxlan0 type=vxlan
options:local_ip=1.1.1.177 options:remote_ip=1.1.1.75 options:key=98
```

- On the second host:

```
# ip addr add 1.1.1.75/24 dev enp4s0f1

# ovs-vsctl add-port ovs-sriov vxlan0 -- set interface vxlan0 type=vxlan
options:local_ip=1.1.1.75 options:remote_ip=1.1.1.177 options:key=98


•   for GRE IPv4 tunnel need use type=gre
•   for GRE IPv6 tunnel need use type=ip6gre
•   for GENEVE tunnel need use type=geneve
```

When encapsulating guest traffic, the VF's device MTU must be reduced to allow the host/HW to add the encap headers without fragmenting the resulted packet. As such, the VF's MTU must be lowered by 50 bytes from the uplink MTU for IPv4 and 70 bytes for IPv6.

Tunnel offload using TC rules:

```
Encapsulation:
# tc filter add dev ens4f0_0 protocol 0x806 parent ffff: \
                flower \
                       skip_sw \
                       dst_mac e4:11:22:11:4a:51 \
                       src_mac e4:11:22:11:4a:50 \
                action tunnel_key set \
                src_ip 20.1.12.1 \
```

```
                dst_ip 20.1.11.1 \
                id 100 \
                action mirred egress redirect dev vxlan100

Decapsulation:
# tc filter add dev vxlan100 protocol 0x806 parent ffff: \
                flower \
                        skip_sw \
                        dst_mac e4:11:22:11:4a:51 \
                        src_mac e4:11:22:11:4a:50 \
                        enc_src_ip 20.1.11.1 \
                        enc_dst_ip 20.1.12.1 \
                        enc_key_id 100 \
                        enc_dst_port 4789 \
                action tunnel_key unset \
                  action mirred egress redirect dev ens4f0_0
```

VLAN Push/Pop

OVS-kernel supports offload of vlan header push/pop actions.

- Push—pushing of VLAN header is supported on Tx
- Pop—popping of tunnel header is supported on Rx

> Starting with ConnectX-6 Dx hardware models and above, pushing of VLAN header is also supported on Rx, and popping of VLAN header is also supported on Tx.

OVS Configuration

Add a tag=$TAG section for the OVS command line that adds the representor ports. For example, VLAN ID 52 is being used here.

```
# ovs-vsctl add-port ovs-sriov enp4s0f0
# ovs-vsctl add-port ovs-sriov enp4s0f0_0 tag=52
# ovs-vsctl add-port ovs-sriov enp4s0f0_1 tag=52
```

The PF port should not have a VLAN attached. This will cause OVS to add VLAN push/pop actions when managing traffic for these VFs.

Dump Flow Example

```
recirc_id(0),in_port(3),eth(src=e4:11:22:33:44:50,dst=00:02:c9:e9:bb:b2),eth_type(0x0800),ipv4(frag=no), \
packets:0, bytes:0, used:never, actions:push_vlan(vid=52,pcp=0),2

recirc_id(0),in_port(2),eth(src=00:02:c9:e9:bb:b2,dst=e4:11:22:33:44:50),eth_type(0x8100), \
vlan(vid=52,pcp=0),encap(eth_type(0x0800),ipv4(frag=no)), packets:0, bytes:0, used:never, actions:pop_vlan,3
```

VLAN Offload using TC Rules Example

```
# tc filter add dev ens4f0_0 protocol ip parent ffff: \
                flower \
                        skip_sw \
                        dst_mac e4:11:22:11:4a:51 \
                        src_mac e4:11:22:11:4a:50 \
                action vlan push id 100 \
                action mirred egress redirect dev ens4f0

# tc filter add dev ens4f0 protocol 802.1Q parent ffff: \
                flower \
                        skip_sw \
                        dst_mac e4:11:22:11:4a:51 \
                        src_mac e4:11:22:11:4a:50 \
                        vlan_ethtype 0x800 \
                        vlan_id 100 \
                        vlan_prio 0 \
                action vlan pop \
                action mirred egress redirect dev ens4f0_0
```

TC Configuration for ConnectX-6 Dx and Above

Example of VLAN Offloading with popping header on Tx and pushing on Rx using TC Rules:

```
# tc filter add dev ens4f0_0 ingress protocol 802.1Q parent ffff: \
        flower \
                vlan_id 100 \
        action vlan pop \
        action tunnel_key set \
                src_ip 4.4.4.1 \
                dst_ip 4.4.4.2 \
                dst_port 4789 \
                id 42 \
        action mirred egress redirect dev vxlan0

# tc filter add dev vxlan0 ingress protocol all parent ffff: \
        flower \
                enc_dst_ip 4.4.4.1 \
                enc_src_ip 4.4.4.2 \
                enc_dst_port 4789 \
                enc_key_id 42 \
        action tunnel_key unset \
        action vlan push id 100 \
        action mirred egress redirect dev ens4f0_0
```

Header Rewrite

This action allows for modifying packet fields.

Ethernet Layer 2

- Destination MAC
- Source MAC

Supported kernels: Kernel 4.14 and above & RHEL 7.5 and above

In OVS dump flows:

```
skb_priority(0/0),skb_mark(0/0),in_port(eth6),eth(src=00:02:10:40:10:0d
,dst=68:54:ed:00:af:de),eth_type(0x8100), packets:1981, bytes:206024, used:0.440s, dp:tc, actions:
set(eth(src=68:54:ed:00:f4:ab,dst=fa:16:3e:dd:69:c4)),eth7
```

Using TC rules:

```
tc filter add dev $rep parent ffff: protocol arp pref 1 \
                            flower \
                            dst_mac e4:1d:2d:5d:25:35 \
                            src_mac e4:1d:2d:5d:25:34 \
                    action pedit ex \
                    munge eth dst set 20:22:33:44:55:66 \
                    munge eth src set aa:ba:cc:dd:ee:fe \
                    action mirred egress redirect dev $NIC
```

IPv4/IPv6

- Source address
- Destination address
- Protocol
- TOS
- TTL (HLIMIT)

Supported kernels: Kernel 4.14 and above & RHEL 7.5 and above

In OVS dump flows:

```
Ipv4:
    set(eth(src=de:e8:ef:27:5e:45,dst=00:00:01:01:01:01)),
    set(ipv4(src=10.10.0.111,dst=10.20.0.122,ttl=63))
Ipv6:
    set(ipv6(dst=2001:1:6::92eb:fcbe:f1c8,hlimit=63)),
```

Using TC rules:

```
IPv4:
tc filter add dev $rep parent ffff: protocol ip pref 1 \
                            flower \
                            dst_ip 1.1.1.1 \
                            src_ip 1.1.1.2 \
                            ip_proto TCP \
                            ip_tos 0x3 \
                            ip_ttl 63 \
                      pedit ex \
                      munge ip src set 2.2.2.1 \
                      munge ip dst set 2.2.2.2 \
                      munge ip tos set 0 \
                      munge ip ttl dec \
                      action mirred egress redirect dev $NIC


IPv6:
tc filter add dev $rep parent ffff: protocol ipv6 pref 1 \
                            flower \
                            dst_ip 1:1:1::3:1040:1009 \
                            src_ip 1:1:1::3:1040:1008 \
                            ip_proto tcp \
                            ip_tos 0x3 \
                            ip_ttl 63\
                      pedit ex \
                      munge ipv6 src set 2:2:2::3:1040:1009 \
                      munge ipv6 dst set 2:2:2::3:1040:1008 \
                      munge ipv6 hlimit dec \
                      action mirred egress redirect dev $NIC
```

IPv4 and IPv6 header rewrite is only supported with match on UDP/TCP/ICMP protocols.

TCP/UDP Source and Destination Ports

- TCP/UDP source and destinations ports

Supported kernels: kernel > 4.16 & RHEL > 7.6

In OVS dump flows:

```
TCP:

        set(tcp(src= 32768/0xffff,dst=32768/0xffff)),
UDP:

        set(udp(src= 32768/0xffff,dst=32768/0xffff)),
```

Using TC rules:

```
TCP:

        tc filter add dev $rep parent ffff: protocol ip pref 1 \
                            flower \
                            dst_ip 1.1.1.1 \
                            src_ip 1.1.1.2 \
                            ip_proto tcp \
                            ip_tos 0x3 \
                            ip_ttl 63 \
                      pedit ex \
                      pedit ex munge ip tcp sport set 200
                      pedit ex munge ip tcp dport set 200
                      action mirred egress redirect dev $NIC

UDP:
        tc filter add dev $rep parent ffff: protocol ip pref 1 \
                            flower \
                            dst_ip 1.1.1.1 \
                            src_ip 1.1.1.2 \
                            ip_proto udp \
                            ip_tos 0x3 \
                            ip_ttl 63 \
                      pedit ex \
                      pedit ex munge ip udp sport set 200
                      pedit ex munge ip udp dport set 200
                      action mirred egress redirect dev $NIC
```

VLAN

- ID

Supported on all kernels.

In OVS dump flows:

```
Set(vlan(vid=2347,pcp=0/0)),
```

Using TC rules:

```
tc filter add dev $rep parent ffff: protocol 802.1Q pref 1 \
                    flower \
                    vlan_ethtype 0x800 \
                    vlan_id 100 \
                    vlan_prio 0 \
             action vlan modify id 11 pipe
             action mirred egress redirect dev $NIC
```

Connection Tracking

The TC connection tracking action performs connection tracking lookup by sending the packet to netfilter conntrack module. Newly added connections may be associated, via the ct commit action, with a 32 bit mark, 128 bit label and source/destination NAT values.

The following example allows ingress tcp traffic from the uplink representor to vf1_rep, while assuring that egress traffic from vf1_rep is only allowed on established connections. In addition, mark and source IP NAT is applied.

In OVS dump flows:

```
ct(zone=2,nat)
ct_state(+est+trk)
actions:ct(commit,zone=2,mark=0x4/0xffffffff,nat(src=5.5.5.5))
```

Using TC rules:

```
# tc filter add dev $uplink_rep ingress chain 0 prio 1 proto ip \
                    flower \
                    ip_proto tcp   \
                    ct_state -trk \
                action ct zone 2 nat pipe
                action goto chain 2
  # tc filter add dev $uplink_rep ingress chain 2 prio 1 proto ip \
                     flower \
                     ct_state +trk+new \
                action ct zone 2 commit mark 0xbb nat src addr 5.5.5.7 pipe \
                action mirred egress redirect dev $vf1_rep
 # tc filter add dev $uplink_rep ingress chain 2 prio 1 proto ip \
                    flower \
                    ct_zone 2 \
                    ct_mark 0xbb \
                    ct_state +trk+est \
                 action mirred egress redirect dev $vf1_rep

  #Setup filters on $vf1_rep, allowing only established connections of zone 2   through, and reverse nat (dst nat
in this case)
 # tc filter add dev $vf1_rep ingress chain 0 prio 1 proto ip \
                    flower \
                    ip_proto tcp \
                    ct_state -trk \
                action ct zone 2 nat pipe \
                action goto chain 1
  # tc filter add dev $vf1_rep ingress chain 1 prio 1 proto ip \
                    flower \
                    ct_zone 2 \
                    ct_mark 0xbb \
                    ct_state +trk+est \
                action mirred egress redirect dev eth0
```

Connection Tracking Performance Tuning

- Max offloaded connections—specifies the limit on the number of offloaded connections.
  Example:

```
# devlink dev param set pci/${pci_dev} name ct_max_offloaded_conns value $max cmode runtime
```

- Allow mixed NAT/non-NAT CT—allows offloading of the following scenario:

```
•   cookie=0x0, duration=21.843s, table=0, n_packets=4838718, n_bytes=241958846, ct_state=-
trk,ip,in_port=enp8s0f0 actions=ct(table=1,zone=2)
•   cookie=0x0, duration=21.823s, table=1, n_packets=15363, n_bytes=773526,
ct_state=+new+trk,ip,in_port=enp8s0f0 actions=ct(commit,zone=2,nat(dst=11.11.11.11)),output:"enp8s0f0_1"
•   cookie=0x0, duration=21.806s, table=1, n_packets=4767594, n_bytes=238401190,
ct_state=+est+trk,ip,in_port=enp8s0f0 actions=ct(zone=2,nat),output:"enp8s0f0_1"
```

  Example:

```
# echo enable > /sys/class/net/<device>/compat/devlink/ct_action_on_nat_conns
```

Forward to Chain (TC Only)

TC interface supports adding flows on different chains. Only chain 0 is accessed by default. Access to the other chains requires usafe of the goto action.

In this example, a flow is created on chain 1 without any match and redirect to wire.

The second flow is created on chain 0 and match on source MAC and action goto chain 1.

This example simulates simple MAC spoofing.

```
#tc filter add dev $rep parent ffff: protocol all chain 1 pref 1 \
                    flower \
                action mirred egress redirect dev $NIC

#tc filter add dev $rep parent ffff: protocol all chain 1 pref 1 \
                    flower \
                    src_mac aa:bb:cc:aa:bb:cc
                action goto chain 1
```

3.3.6.5.1.9 Port Mirroring (Flow Based VF Traffic Mirroring for ASAP²)

Unlike para-virtual configurations, when the VM traffic is offloaded to the hardware via SR-IOV VF, the host side Admin cannot snoop the traffic (e.g. for monitoring).

ASAP² uses the existing mirroring support in OVS and TC along with the enhancement to the offloading logic in the driver to allow mirroring the VF traffic to another VF.

The mirrored VF can be used to run traffic analyzer (tcpdump, wireshark, etc) and observe the traffic of the VF being mirrored.

The example below shows the creation of port mirror on the following configuration:

```
# ovs-vsctl show
  09d8a574-9c39-465c-9f16-47d81c12f88a
      Bridge br-vxlan
                  Port "enp4s0f0_1"
                     Interface "enp4s0f0_1"
                  Port "vxlan0"
                      Interface "vxlan0"
                                  type: vxlan
                                  options: {key="100", remote_ip="192.168.1.14"}
                  Port "enp4s0f0_0"
                      Interface "enp4s0f0_0"
                  Port "enp4s0f0_2"
                      Interface "enp4s0f0_2"
```

```
                Port br-vxlan
                    Interface br-vxlan
                                type: internal
    ovs_version: "2.14.1"
```

- To set enp4s0f0_0 as the mirror port, and mirror all of the traffic:

```
# ovs-vsctl -- --id=@p get port enp4s0f0_0 \
            -- --id=@m create mirror name=m0 select-all=true output-port=@p \
            -- set bridge br-vxlan mirrors=@m
```

- To set enp4s0f0_0 as the mirror port, and only mirror the traffic, the destination is enp4s0f0_1:

```
# ovs-vsctl -- --id=@p1 get port enp4s0f0_0 \
            -- --id=@p2 get port enp4s0f0_1 \
            -- --id=@m create mirror name=m0 select-dst-port=@p2 output-port=@p1 \
            -- set bridge br-vxlan mirrors=@m
```

- To set enp4s0f0_0 as the mirror port, and only mirror the traffic the source is enp4s0f0_1:

```
# ovs-vsctl -- --id=@p1 get port enp4s0f0_0 \
            -- --id=@p2 get port enp4s0f0_1 \
            -- --id=@m create mirror name=m0 select-src-port=@p2 output-port=@p1 \
            -- set bridge br-vxlan mirrors=@m
```

- To set enp4s0f0_0 as the mirror port and mirror, all the traffic on enp4s0f0_1:

```
# ovs-vsctl -- --id=@p1 get port enp4s0f0_0 \
      -- --id=@p2 get port enp4s0f0_1 \
      -- --id=@m create mirror name=m0 select-dst-port=@p2 select-src-port=@p2 output-port=@p1 \
      -- set bridge br-vxlan mirrors=@m
```

To clear the mirror port:

```
 # ovs-vsctl clear bridge br-vxlan mirrors
```

Mirroring using TC:

```
Mirror to VF
tc filter add dev $rep parent ffff: protocol arp pref 1 \
                        flower \
                        dst_mac e4:1d:2d:5d:25:35 \
                        src_mac e4:1d:2d:5d:25:34 \
                        action mirred egress mirror dev $mirror_rep pipe \
                        action mirred egress redirect dev $NIC

Mirror to tunnel:
tc filter add dev $rep parent ffff: protocol arp pref 1 \
                        flower \
                        dst_mac e4:1d:2d:5d:25:35 \
                        src_mac e4:1d:2d:5d:25:34 \
                action tunnel_key set \
                src_ip 1.1.1.1 \
                dst_ip 1.1.1.2 \
                dst_port 4789 \
                id 768 \
                pipe \
                action mirred egress mirror dev vxlan100 pipe \
                action mirred egress redirect dev $NIC
```

3.3.6.5.1.10 Forward to Multiple Destinations

Forward to up 32 destinations (representors and tunnels) is supported using TC.

Example 1: forward to 32 VFs.

```
tc filter add dev $NIC parent ffff: protocol arp pref 1 \
                        flower \
                        dst_mac e4:1d:2d:5d:25:35 \
                        src_mac e4:1d:2d:5d:25:34 \
                        action mirred egress mirror dev $rep0 pipe \
                        action mirred egress mirror dev $rep1 pipe \
...
                        action mirred egress mirror dev $rep30 pipe \
                        action mirred egress redirect dev $rep31
```

Example 2: forward to 16 tunnels.

```
tc filter add dev $rep parent ffff: protocol arp pref 1 \
                        flower \
                        dst_mac e4:1d:2d:5d:25:35 \
                        src_mac e4:1d:2d:5d:25:34 \
                        action tunnel_key set src_ip $ip_src dst_ip $ip_dst \
dst_port 4789 id 0 nocsum \
pipe action mirred egress mirror dev vxlan0 pipe \
                        action tunnel_key set src_ip $ip_src dst_ip $ip_dst \
dst_port 4789 id 1 nocsum \
pipe action mirred egress mirror dev vxlan0 pipe \
...
                        action tunnel_key set src_ip $ip_src dst_ip $ip_dst \
dst_port 4789 id 15 nocsum \
pipe action mirred egress redirect dev vxlan0
```

- TC supports up to 32 actions
- If header rewrite is used, then all destinations should have the same header rewrite
- If VLAN push/pop is used, then all destinations should have the same VLAN ID and actions

3.3.6.5.1.11 sFLOW

This feature allows for monitoring traffic sent between two VMs on the same host using an sFlow collector.

The example below assumes the environment is configured as described below.

```
# ovs-vsctl show
  09d8a574-9c39-465c-9f16-47d81c12f88a
      Bridge br-vxlan
                Port "enp4s0f0_1"
                  Interface "enp4s0f0_1"
                Port "vxlan0"
                  Interface "vxlan0"
                              type: vxlan
                              options: {key="100", remote_ip="192.168.1.14"}
                Port "enp4s0f0_0"
                  Interface "enp4s0f0_0"
                Port "enp4s0f0_2"
                  Interface "enp4s0f0_2"
                Port br-vxlan
                  Interface br-vxlan
                              type: internal
      ovs_version: "2.14.1"
```

To sample all traffic over the OVS bridge:

```
# ovs-vsctl -- --id=@sflow create sflow agent=\"$SFLOW_AGENT\" \
             target=\"$SFLOW_TARGET:$SFLOW_PORT\" header=$SFLOW_HEADER \
             sampling=$SFLOW_SAMPLING polling=10 \
             -- set bridge br-vxlan sflow=@sflow
```

Parameter	Description
SFLOW_AGENT	Indicates that the sFlow agent should send traffic from SFLOW_AGENT's IP address

SFLOW_TARGET	Remote IP address of the sFLOW collector
SFLOW_HEADER	Size of packet header to sample (in bytes)
SFLOW_SAMPLING	Sample rate

To clear the sFLOW configuration:

```
# ovs-vsctl clear bridge br-vxlan sflow
```

To list the sFLOW configuration:

```
# ovs-vsctl list sflow
```

sFLOW using TC:

```
Sample to VF
tc filter add dev $rep parent ffff: protocol arp pref 1 \
                        flower \
                        dst_mac e4:1d:2d:5d:25:35 \
                        src_mac e4:1d:2d:5d:25:34 \
                        action sample rate 10 group 5 trunc 96 \
                        action mirred egress redirect dev $NIC
```

Userspace application is needed in order to process to sampled packet from the kernel.
Example: https://github.com/Mellanox/libpsample

3.3.6.5.1.12 Rate Limit

OVS-kernel supports offload of VF rate limit using OVS configuration and TC.

The example below sets a rate limit to the VF related to representor eth0 to 10Mbps.

```
OVS:
 # ovs-vsctl set interface eth0 ingress_policing_rate=10000

tc:
 # tc_filter add dev eth0 root prio 1 protocol ip matchall skip_sw action police rate 10mbit burst 20k
```

3.3.6.5.1.13 Kernel Requirements

This kernel config should be enabled in order to support switchdev offload.

- CONFIG_NET_ACT_CSUM - needed for action csum
- CONFIG_NET_ACT_PEDIT - needed for header rewrite
- CONFIG_NET_ACT_MIRRED - needed for basic forward
- CONFIG_NET_ACT_CT - needed for connection tracking (supported from kernel 5.6)
- CONFIG_NET_ACT_VLAN - needed for action vlan push/pop
- CONFIG_NET_ACT_GACT
- CONFIG_NET_CLS_FLOWER
- CONFIG_NET_CLS_ACT
- CONFIG_NET_SWITCHDEV
- CONFIG_NET_TC_SKB_EXT - needed for connection tracking (supported from kernel 5.6)

- CONFIG_NET_ACT_CT - needed for connection tracking (supported from kernel 5.6)
- CONFIG_NFT_FLOW_OFFLOAD
- CONFIG_NET_ACT_TUNNEL_KEY
- CONFIG_NF_FLOW_TABLE - needed for connection tracking (supported from kernel 5.6)
- CONFIG_SKB_EXTENSIONS - needed for connection tracking (supported from kernel 5.6)
- CONFIG_NET_CLS_MATCHALL
- CONFIG_NET_ACT_POLICE
- CONFIG_MLX5_ESWITCH

3.3.6.5.1.14 VF Metering

OVS-kernel supports offloading of VF metering (TX and RX) using sysfs. Metering of number of packets per second (PPS) and bytes per second (BPS) is supported.

The example bellow sets Rx meter on VF 0 with value 10Mbps BPS.

```
echo 10000000 > /sys/class/net/enp4s0f0/device/sriov/0/meters/rx/bps/rate
echo 65536 > /sys/class/net/enp4s0f0/device/sriov/0/meters/rx/bps/burst
```

The example bellow sets Tx meter on VF 0 with value 1000 PPS.

```
echo 1000 > /sys/class/net/enp4s0f0/device/sriov/0/meters/tx/pps/rate
echo 100 > /sys/class/net/enp4s0f0/device/sriov/0/meters/tx/pps/burst
```

Both rate and burst must not be zero and burst may need to be adjusted according to the requirements.

The following counters can be used to query the number dropped packet/bytes:

```
#cat /sys/class/net/enp8s0f0/device/sriov/0/meters/rx/pps/packets_dropped
#cat /sys/class/net/enp8s0f0/device/sriov/0/meters/rx/pps/bytes_dropped
#cat /sys/class/net/enp8s0f0/device/sriov/0/meters/rx/bps/packets_dropped
#cat /sys/class/net/enp8s0f0/device/sriov/0/meters/rx/bps/bytes_dropped
#cat /sys/class/net/enp8s0f0/device/sriov/0/meters/tx/pps/packets_dropped
#cat /sys/class/net/enp8s0f0/device/sriov/0/meters/tx/pps/bytes_dropped
#cat /sys/class/net/enp8s0f0/device/sriov/0/meters/tx/bps/packets_dropped
#cat /sys/class/net/enp8s0f0/device/sriov/0/meters/tx/bps/bytes_dropped
```

3.3.6.5.1.15 Representor Metering

Metering for uplink and VF representors traffic support has been added.

Traffic going to a representor device can be a result of a miss in the embedded switch (eSwitch) FDB tables. This means that a packet which arrived from that representor into the eSwitch was not matched against the existing rules in the hardware FDB tables and needs to be forwarded to software to be handled there and is, therefore, forwarded to the originating representor device driver.

The meter allows to configure the max rate [packets/sec] and max burst [packets] for traffic going to the representor driver. Any traffic exceeding values provided by the user will be dropped in hardware. There are statistics that show number of dropped packets.

The configuration of a representors metering is done via a new sysfs called `miss_rl_cfg`.

- Full path of the miss_rl_cfg parameter: / `sys/class/net//rep_config/miss_rl_cfg`
- Usage: `echo "<rate> <burst>" > /sys/class/net//rep_config/miss_rl_cfg`. Rate is the max rate of packets allowed for this representor (in packets/sec units) and burst is the max burst size allowed for this representor (in packets units). Both values must be specified. The default is 0 for both, meaning unlimited rate and burst.

To view the amount of packets and bytes that were dropped due to traffic exceeding the user-provided rate and burst, two read-only sysfs for statistics are exposed.

- `/sys/class/net//rep_config/miss_rl_dropped_bytes` counts how many FDB-miss bytes were dropped due to reaching the miss limits
- `/sys/class/net//rep_config/miss_rl_dropped_packets` counts how many FDB-miss packets were dropped due to reaching the miss limits

3.3.6.5.1.16 Open vSwitch Metering

There are two types of meters, kpps (kilobits per second) and pktps(packets per second), which are described in Meter Syntax of OpenFlow 1.3+ Switch Meter Table Commands. OVS-Kernel supports offloading them both.

The example below is to offload a kpps meter. Please follow the steps after doing basic configurations as described in section 5.1.3.

1. Create OVS meter with a target rate.

```
ovs-ofctl -O OpenFlow13 add-meter ovs-sriov meter=1,kbps,band=type=drop,rate=204800
```

2. Delete the default rule.

```
ovs-ofctl del-flows ovs-sriov
```

3. Configure OpenFlow rules. Here VF bandwidth on the receiving side will be limited by the rate configured in step 1.

```
  ovs-ofctl -O OpenFlow13 add-flow ovs-sriov 'ip,dl_dst=e4:11:22:33:44:50,actions=
meter:1,output:enp4s0f0_0'
  ovs-ofctl -O OpenFlow13 add-flow ovs-sriov 'ip,dl_src=e4:11:22:33:44:50,actions= output:enp4s0f0'
  ovs-ofctl -O OpenFlow13 add-flow ovs-sriov 'arp,actions=normal'
```

4. Run iperf server and be ready to receive UDP traffic. On the outer node, run iperf client to send UDP traffic to this VF. After traffic starts, check the offloaded meter rule.

```
 ovs-appctl dpctl/dump-flows --names type=offloaded

recirc_id(0),in_port(enp4s0f0),eth(dst=e4:11:22:33:44:50),eth_type(0x0800),ipv4(frag=no), packets:11626587,
bytes:17625889188, used:0.470s, actions:meter(0),enp4s0f0_0
```

In order to verify metering, iperf client should set the target bandwidth with a number which is larger than the meter rate configured. Then it will be visible that packets are received with the limited rate on the server side and the extra packets are dropped by hardware.

3.3.6.5.1.17 Multiport eSwitch Mode

The multiport eSwitch mode allows to add rules on a VF representor with an action forwarding the packet to the physical port of the physical function. This can be used to implement failover or forward packets based on external information such the cost of the route.

1. To configure this more, the nvconig parameter LAG_RESOURCE_ALLOCATION must be set.
2. After the driver loads, configure multiport eSwitch for each PF where enp8s0f0 and enp8s0f1 represent the netdevices for the PFs.

   ```
   echo multiport_esw > /sys/class/net/enp8s0f0/compat/devlink/lag_port_select_mode
   echo multiport_esw > /sys/class/net/enp8s0f1/compat/devlink/lag_port_select_mode
   ```

   The mode becomes operational after entering switchdev mode on both PFs.

Rule example:

```
tc filter add dev enp8s0f0_0 prot ip root flower dst_ip 7.7.7.7 action mirred egress redirect dev enp8s0f1
```

3.3.6.5.2 OVS-DPDK Hardware Offloads

3.3.6.5.2.1 OVS-DPDK Hardware Offloads Configuration

To configure OVS-DPDK HW offloads:

1. Unbind the VFs.

   ```
   echo 0000:04:00.2 > /sys/bus/pci/drivers/mlx5_core/unbind
   echo 0000:04:00.3 > /sys/bus/pci/drivers/mlx5_core/unbind
   ```

   Note: VMs with attached VFs must be powered off to be able to unbind the VFs.
2. Change the e-switch mode from Legacy to SwitchDev on the PF device (make sure all VFs are unbound). This will also create the VF representor netdevices in the host OS.

   ```
   echo switchdev > /sys/class/net/enp4s0f0/compat/devlink/mode
   ```

   To revert to SR-IOV Legacy mode:

   ```
   echo legacy > /sys/class/net/enp4s0f0/compat/devlink/mode
   ```

   Note that running this command will also result in the removal of the VF representor netdevices.
3. Bind the VFs.

   ```
   echo 0000:04:00.2 > /sys/bus/pci/drivers/mlx5_core/bind
   echo 0000:04:00.3 > /sys/bus/pci/drivers/mlx5_core/bind
   ```
4. Run the Open vSwitch service.

   ```
   systemctl start openvswitch
   ```
5. Enable hardware offload (disabled by default).

```
ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-init=true
ovs-vsctl set Open_vSwitch . other_config:hw-offload=true
```

6. Configure the DPDK white list.

```
ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-extra="-a
0000:01:00.0,representor=[0],dv_flow_en=1,dv_esw_en=1,dv_xmeta_en=1"
```

⚠ Representor=[0-N]

7. Restart the Open vSwitch service. This step is required for HW offload changes to take effect.

```
systemctl restart openvswitch
```

8. Create OVS-DPDK bridge.

```
ovs-vsctl --no-wait add-br br0-ovs -- set bridge br0-ovs datapath_type=netdev
```

9. Add PF to OVS.

```
ovs-vsctl add-port br0-ovs pf -- set Interface pf type=dpdk options:dpdk-devargs=0000:88:00.0
```

10. Add representor to OVS.

```
ovs-vsctl add-port br0-ovs representor -- set Interface representor type=dpdk options:dpdk-
devargs=0000:88:00.0,representor=[0]
```

⚠ Representor=[0-N]

3.3.6.5.2.2 Offloading VXLAN Encapsulation/Decapsulation Actions

vSwitch in userspace rather than kernel-based Open vSwitch requires an additional bridge. The purpose of this bridge is to allow use of the kernel network stack for routing and ARP resolution.

The datapath needs to look-up the routing table and ARP table to prepare the tunnel header and transmit data to the output port.

Configuring VXLAN Encap/Decap Offloads

⚠ The configuration is done with:

- PF on 0000:03:00.0 PCI and MAC 98:03:9b:cc:21:e8
- Local IP 56.56.67.1 - br-phy interface will be configured to this IP
- Remote IP 56.56.68.1

To configure OVS-DPDK VXLAN:

1. Create a br-phy bridge.

```
ovs-vsctl add-br br-phy -- set Bridge br-phy datapath_type=netdev -- br-set-external-id br-phy bridge-id
br-phy -- set bridge br-phy fail-mode=standalone other_config:hwaddr=98:03:9b:cc:21:e8
```

2. Attach PF interface to br-phy bridge.

```
ovs-vsctl add-port br-phy p0 -- set Interface p0 type=dpdk options:dpdk-devargs=0000:03:00.0
```

3. Configure IP to the bridge.

```
ip addr add 56.56.67.1/24 dev br-phy
```

4. Create a br-ovs bridge.

```
ovs-vsctl add-br br-ovs -- set Bridge br-ovs datapath_type=netdev -- br-set-external-id br-ovs bridge-id
br-ovs -- set bridge br-ovs fail-mode=standalone
```

5. Attach representor to br-ovs.

```
ovs-vsctl add-port br-ovs pf0vf0 -- set Interface pf0vf0 type=dpdk options:dpdk-
devargs=0000:03:00.0,representor=[0]
```

6. Add a port for the VXLAN tunnel.

```
ovs-vsctl add-port ovs-sriov vxlan0 -- set interface vxlan0 type=vxlan options:local_ip=56.56.67.1
 options:remote_ip=56.56.68.1 options:key=45 options:dst_port=4789
```

3.3.6.5.2.3 Connection Tracking Offload

Connection tracking enables stateful packet processing by keeping a record of currently open connections.
OVS flows using connection tracking can be accelerated using advanced Network Interface Cards (NICs) by offloading established connections.

To view offloaded connections, run:

```
ovs-appctl dpctl/offload-stats-show
```

3.3.6.5.2.4 SR-IOV VF LAG

To configure OVS-DPDK SR-IOV VF LAG:

1. Enable SR-IOV on the NICs.

```
mlxconfig -d <PCI> set SRIOV_EN=1
```

2. Allocate the desired number of VFs per port.

```
echo $n > /sys/class/net/<net name>/device/sriov_numvfs
```

3. Unbind all VFs.

```
echo <VF PCI> >/sys/bus/pci/drivers/mlx5_core/unbind
```

4. Change both NICs' mode to SwitchDev.

```
devlink dev eswitch set pci/<PCI> mode switchdev
```

5. Create Linux bonding using kernel modules.

```
modprobe bonding mode=<desired mode>
```

Note: Other bonding parameters can be added here. The supported Bond modes are: Active-Backup, XOR and LACP.

6. Bring all PFs and VFs down.

```
ip link set <PF/VF> down
```

7. Attach both PFs to the bond.

```
ip link set <PF> master bond0
```

8. To work with VF-LAG with OVS-DPDK, add the bond master (PF) to the bridge.

```
ovs-vsctl add-port br-phy p0 -- set Interface p0 type=dpdk options:dpdk-devargs=0000:03:00.0  options:dpdk-
lsc-interrupt=true
```

9. Add representor $N of PF0 or PF1 to a bridge.

```
ovs-vsctl add-port br-phy rep$N -- set Interface rep$N type=dpdk options:dpdk-devargs=<PF0
PCI>,representor=pf0vf$N

OR

ovs-vsctl add-port br-phy rep$N -- set Interface rep$N type=dpdk options:dpdk-devargs=<PF0
PCI>,representor=pf1vf$N
```

3.3.6.5.2.5 VirtIO Acceleration through VF Relay (Software & Hardware vDPA)

⚠ Hardware vDPA is supported on ConnectX-6 Dx, ConnectX-6 Lx & BlueField-2 cards and above only.

⚠ Hardware vDPA is enabled by default. In case your hardware does not support vDPA, the driver will fall back to Software vDPA.

To check which vDPA mode is activated on your driver, run: `ovs-ofctl -O OpenFlow14 dump-ports br0-ovs` and look for `hw-mode` flag.

⚠ This feature has not been accepted to the OVS-DPDK Upstream yet, making its API subject to change.

In user space, there are two main approaches for communicating with a guest (VM), either through SR-IOV, or through virtIO.

Phy ports (SR-IOV) allow working with port representor, which is attached to the OVS and a matching VF is given with pass-through to the guest. HW rules can process packets from up-link and direct

them to the VF without going through SW (OVS). Therefore, using SR-IOV achieves the best performance.

However, SR-IOV architecture requires the guest to use a driver specific to the underlying HW. Specific HW driver has two main drawbacks:

1. Breaks virtualization in some sense (guest is aware of the HW). It can also limit the type of images supported.
2. Gives less natural support for live migration.

Using virtIO port solves both problems. However, it reduces performance and causes loss of some functionalities, such as, for some HW offloads, working directly with virtIO. To solve this conflict, a new netdev type- dpdkvdpa has been created. The new netdev is similar to the regular DPDK netdev, yet introduces several additional functionalities.

dpdkvdpa translates between phy port to virtIO port. It takes packets from the Rx queue and sends them to the suitable Tx queue, and allows transfer of packets from virtIO guest (VM) to a VF, and vice-versa, benefitting from both SR-IOV and virtIO.

To add vDPA port:

```
ovs-vsctl add-port br0 vdpa0 -- set Interface vdpa0 type=dpdkvdpa \
options:vdpa-socket-path=<sock path> \
options:vdpa-accelerator-devargs=<vf pci id> \
options:dpdk-devargs=<pf pci id>,representor=[id] \
options: vdpa-max-queues =<num queues> \
options: vdpa-sw=<true/false>
```

Note: vdpa-max-queues is an optional field. When the user wants to configure 32 vDPA ports, the maximum queues number is limited to 8.

vDPA Configuration in OVS-DPDK Mode

Prior to configuring vDPA in OVS-DPDK mode, follow the steps below.

1. Generate the VF.

```
echo 0 > /sys/class/net/enp175s0f0/device/sriov_numvfs
echo 4 > /sys/class/net/enp175s0f0/device/sriov_numvfs
```

2. Unbind each VF.

```
echo <pci> > /sys/bus/pci/drivers/mlx5_core/unbind
```

3. Switch to SwitchDev mode.

```
echo switchdev >> /sys/class/net/enp175s0f0/compat/devlink/mode
```

4. Bind each VF.

```
echo <pci> > /sys/bus/pci/drivers/mlx5_core/bind
```

5. Initialize OVS with:

```
ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-init=true
ovs-vsctl --no-wait set Open_vSwitch . other_config:hw-offload=true
```

To configure vDPA in OVS-DPDK mode on ConnectX-5 cards and above:

1. Open vSwitch configuration.

```
ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-extra="-a
0000:01:00.0,representor=[0],dv_flow_en=1,dv_esw_en=1,dv_xmeta_en=1"
/usr/share/openvswitch/scripts/ovs-ctl restart
```

2. Create OVS-DPDK bridge.

```
ovs-vsctl add-br br0-ovs -- set bridge br0-ovs datapath_type=netdev
ovs-vsctl add-port br0-ovs pf -- set Interface pf type=dpdk options:dpdk-devargs=0000:01:00.0
```

3. Create vDPA port as part of the OVS-DPDK bridge.

```
ovs-vsctl add-port br0-ovs vdpa0 -- set Interface vdpa0 type=dpdkvdpa options:vdpa-socket-path=/var/run/
virtio-forwarder/sock0 options:vdpa-accelerator-devargs=0000:01:00.2 options:dpdk-
devargs=0000:01:00.0,representor=[0] options: vdpa-max-queues=8
```

To configure vDPA in OVS-DPDK mode on BlueField cards:

Set the bridge with the software or hardware vDPA port:

- On the ARM side:
  Create the OVS-DPDK bridge.

```
ovs-vsctl add-br br0-ovs -- set bridge br0-ovs datapath_type=netdev
ovs-vsctl add-port br0-ovs pf -- set Interface pf type=dpdk options:dpdk-devargs=0000:af:00.0
ovs-vsctl add-port br0-ovs rep-- set Interface rep type=dpdk options:dpdk-
devargs=0000:af:00.0,representor=[0]
```

- On the host side:
  Create the OVS-DPDK bridge.

```
ovs-vsctl add-br br1-ovs -- set bridge br1-ovs datapath_type=netdev protocols=OpenFlow14
ovs-vsctl add-port br0-ovs vdpa0 -- set Interface vdpa0 type=dpdkvdpa options:vdpa-socket-path=/var/run/
virtio-forwarder/sock0 options:vdpa-accelerator-devargs=0000:af:00.2
```

  Note: To configure SW vDPA, add "`options:vdpa-sw=true"` to the end of the command.

Software vDPA Configuration in OVS-Kernel Mode

SW vDPA can also be used in configurations where the HW offload is done through TC and not DPDK.

1. Open vSwitch configuration.

```
ovs-vsctl set Open_vSwitch . other_config:dpdk-extra="-a
0000:01:00.0,representor=[0],dv_flow_en=1,dv_esw_en=0,idv_xmeta_en=0,isolated_mode=1"
/usr/share/openvswitch/scripts/ovs-ctl restart
```

2. Create OVS-DPDK bridge.

```
ovs-vsctl add-br br0-ovs -- set bridge br0-ovs datapath_type=netdev
```

3. Create vDPA port as part of the OVS-DPDK bridge.

```
ovs-vsctl add-port br0-ovs vdpa0 -- set Interface vdpa0 type=dpdkvdpa options:vdpa-socket-path=/var/run/
virtio-forwarder/sock0 options:vdpa-accelerator-devargs=0000:01:00.2 options:dpdk-
devargs=0000:01:00.0,representor=[0] options: vdpa-max-queues=8
```

4. Create Kernel bridge.

```
ovs-vsctl add-br br-kernel
```

5. Add representors to Kernel bridge.

```
ovs-vsctl add-port br-kernel enp1s0f0_0
ovs-vsctl add-port br-kernel enp1s0f0
```

3.3.6.5.2.6 Large MTU/Jumbo Frame Configuration

To configure MTU/jumbo frames:

1. Verify that the Kernel version on the VM is 4.14 or above.

```
cat /etc/redhat-release
```

2. Set the MTU on both physical interfaces in the host.

```
ifconfig ens4f0 mtu 9216
```

3. Send a large size packet and verify that it is sent and received correctly.

```
tcpdump -i ens4f0 -nev icmp &
ping 11.100.126.1 -s 9188 -M do -c 1
```

4. Enable host_mtu in xml, and add the following values to xml.

```
host_mtu=9216,csum=on,guest_csum=on,host_tso4=on,host_tso6=on
```

   Example:

```
<qemu:commandline>
<qemu:arg value='-chardev'/>
<qemu:arg value='socket,id=charnet1,path=/tmp/sock0,server'/>
<qemu:arg value='-netdev'/>
<qemu:arg value='vhost-user,chardev=charnet1,queues=16,id=hostnet1'/>
<qemu:arg value='-device'/>
<qemu:arg value='virtio-net-
pci,mq=on,vectors=34,netdev=hostnet1,id=net1,mac=00:21:21:24:02:01,bus=pci.0,addr=0xC,page-per-
vq=on,rx_queue_size=1024,tx_queue_size=1024,host_mtu=9216,csum=on,guest_csum=on,host_tso4=on,host_tso6=on'/
>
</qemu:commandline>
```

5. Add mtu_request=9216 option to the OvS ports inside the container and restart the OVS:

```
ovs-vsctl add-port br0-ovs pf -- set Interface pf type=dpdk options:dpdk-devargs=0000:c4:00.0
 mtu_request=9216
```

   OR:

```
ovs-vsctl add-port br0-ovs vdpa0 -- set Interface vdpa0 type=dpdkvdpa options:vdpa-socket-path=/tmp/sock0
options:vdpa-accelerator-devargs=0000:c4:00.2 options:dpdk-devargs=0000:c4:00.0,representor=[0]
mtu_request=9216
/usr/share/openvswitch/scripts/ovs-ctl restart
```

6. Start the VM and configure the MTU on the VM.

```
ifconfig eth0 11.100.124.2/16 up
ifconfig eth0 mtu 9216
ping 11.100.126.1 -s 9188 -M do -c1
```

3.3.6.5.2.7 E2E Cache

> ⚠ This feature is at beta level.

OVS offload rules are based on a multi-table architecture. E2E cache feature enables merging the multi-table flow matches and actions into one joint flow.

This improves connection tracking performance by using a single-table when exact match is detected.

- *To set the E2E cache size (default = 4k):*

```
ovs-vsctl set open_vswitch . other_config:e2e-size=<size>
systemctl restart openvswitch
```

  Note: Make sure to restart the openvswitch service in order for the configuration to take effect.

- *To enable/disable E2E cache (default = disabled) :*

```
ovs-vsctl set open_vswitch . other_config:e2e-enable=<true/false>
systemctl restart openvswitch
```

  Note: Make sure to restart the openvswitch service in order for the configuration to take effect.

- *To run E2E cache statistics:*

```
ovs-appctl dpctl/dump-e2e-stats
```

- *To run E2E cache flows:*

```
ovs-appctl dpctl/dump-e2e-flows
```

3.3.6.5.2.8 Geneve Encapsulation/Decapsulation

Geneve tunneling offload feature support includes matching on extension header.

To configure OVS-DPDK Geneve encap/decap:

1. Create a br-phy bridge.

```
ovs-vsctl --may-exist add-br br-phy -- set Bridge br-phy datapath_type=netdev -- br-set-external-id br-phy
bridge-id br-phy -- set bridge br-phy fail-mode=standalone
```

2. Attach PF interface to br-phy bridge.

```
ovs-vsctl add-port br-phy pf -- set Interface pf type=dpdk options:dpdk-devargs=<PF PCI>
```

3. Configure IP to the bridge.

```
ifconfig br-phy <$local_ip_1> up
```

4. Create a br-int bridge.

```
ovs-vsctl --may-exist add-br br-int -- set Bridge br-int datapath_type=netdev -- br-set-external-id br-int
 bridge-id br-int -- set bridge br-int fail-mode=standalone
```

5. Attach representor to br-int.

```
ovs-vsctl add-port br-int rep$x -- set Interface rep$x type=dpdk options:dpdk-devargs=<PF
PCI>,representor=[$x]
```

6. Add a port for the GENEVE tunnel.

```
ovs-vsctl add-port br-int geneve0 -- set interface geneve0 type=geneve options:key=<VNI>
options:remote_ip=<$remote_ip_1> options:local_ip=<$local_ip_1>
```

3.3.6.5.2.9 Parallel Offloads

OVS-DPDK supports parallel insertion and deletion of offloads (flow & CT). While multiple threads are supported, by default only one is used.

To configure multiple threads:

```
ovs-vsctl set Open_vSwitch . other_config:n-offload-threads=3
```

Make sure to restart the openvswitch service in order for the configuration to take effect.

```
systemctl restart openvswitch
```

⚠ For more information, see the OvS user manual.

3.3.6.5.2.10 sFlow

This feature allows for monitoring traffic sent between two VMs on the same host using an sFlow collector.

To sample all traffic over the OVS bridge, run the following:

```
# ovs-vsctl -- --id=@sflow create sflow agent=\"$SFLOW_AGENT\" \
             target=\"$SFLOW_TARGET:$SFLOW_HEADER\" header=$SFLOW_HEADER \
             sampling=$SFLOW_SAMPLING polling=10 \
             -- set bridge  sflow=@sflow
```

Parameter	Description
SFLOW_AGENT	Indicates that the sFlow agent should send traffic from SFLOW_AGENT's IP address
SFLOW_TARGET	Remote IP address of the sFLOW collector
SFLOW_PORT	Remote IP destination port of the sFlow collector
SFLOW_HEADER	Size of packet header to sample (in bytes)
SFLOW_SAMPLING	Sample rate

To clear the sFLOW configuration, runt he following:

```
# ovs-vsctl clear bridge br-vxlan mirrors
```

Currently sFlow for OVS-DPDK is supported without CT.

3.3.6.5.2.11 CT CT NAT

To enable ct-ct-nat offloads in OvS-DPDK, execute the following command (deafult value is false):

```
ovs-vsctl set open_vswitch . other_config:ct-action-on-nat-conns=true
```

If disabled, ct-ct-nat configurations will not be fully offloaded, improving connection offloading rate for
other cases (ct and ct-nat).

If enabled, ct-ct-nat configurations will be fully offloaded but ct and ct-nat offloading
will be slower to be created.

3.3.6.5.2.12 OpenFlow Meters (OpenFlow13+):

OpenFlow meters in OVS are implemented according to RFC 2697 (Single Rate Three Color Marker—srTCM).

- The srTCM meters an IP packet stream and marks its packets either green, yellow, or red. The color is decided on a Committed Information Rate (CIR) and two associated burst sizes, Committed Burst Size (CBS), and Excess Burst Size (EBS).
- A packet is marked green if it does not exceed the CBS, yellow if it exceeds the CBS but not the EBS, and red otherwise.
- The volume of green packets should never be smaller than the CIR.

To configure a meter in OVS:

1. Create a meter over a certain bridge:

   a.
   ```
   ovs-ofctl -O openflow13 add-meter $bridge
   meter=$id,$pktps/$kbps,band=type=drop,rate=$rate,[burst,burst_size=$burst_size]
   ```

   b. Parameters:

Parameter	Description
bridge	Name of the bridge on which the meter will be applied.
id	Unique meter ID (32 bits) which will be used as an identifier for the meter.
pktps/kbps	Indication if the meter should work according to packets-per-second or kilobits-per-second.
rate	Rate of pktps/kbps of allowed data transmission.
burst	If set, enables burst support for meter bands through the “burst_size” parameter.

Parameter	Description
burst_size	If burst is specified for the meter entry, configures the maximum burst allowed for the band in kilobits/packets, depending on whether kbps or pktps was specified. If unspecified, the switch is free to select some reasonable value depending on its configuration. Currently, if burst was not specified, the burst_size parameter is set as the “rate”.

2. Add the meter to a certain OpenFlow rule. For example:

```
ovs-ofctl -O openflow13 add-flow $bridge "table=0,actions=meter:$id,normal“
```

3. View the meter statistics:

```
ovs-ofctl -O openflow13 meter-stats $bridge meter=$id
```

4. For more information, refer to openvswitch documentation http://www.openvswitch.org/support/dist-docs/ovs-ofctl.8.txt

3.3.6.6 VirtIO Acceleration through Hardware vDPA

3.3.6.6.1 Hardware vDPA Installation

Hardware vDPA requires QEMU v2.12 (or with upstream 6.1.0) and DPDK v20.11 as minimal versions.

To install QEMU:

1. Clone the sources:

```
git clone https://git.qemu.org/git/qemu.git
cd qemu
git checkout v2.12
```

2. Build QEMU:

```
mkdir bin
cd bin
../configure --target-list=x86_64-softmmu --enable-kvm
make -j24
```

To install DPDK:

1. Clone the sources:

```
git clone git://dpdk.org/dpdk
cd dpdk
git checkout v20.11
```

2. Install dependencies (if needed):

```
yum install cmake gcc libnl3-devel libudev-devel make pkgconfig valgrind-devel pandoc libibverbs libmlx5
libmnl-devel -y
```

3. Configure DPDK:

```
export RTE_SDK=$PWD
make config T=x86_64-native-linuxapp-gcc
cd build
sed -i 's/\(CONFIG_RTE_LIBRTE_MLX5_PMD=\)n/\1y/g' .config
sed -i 's/\(CONFIG_RTE_LIBRTE_MLX5_VDPA_PMD=\)n/\1y/g' .config
```

4. Build DPDK:

```
make -j
```

5. Build the vDPA application:

```
cd $RTE_SDK/examples/vdpa/
make -j
```

3.3.6.6.2 Hardware vDPA Configuration

To configure huge pages:

```
mkdir -p /hugepages
mount -t hugetlbfs hugetlbfs /hugepages
echo <more> > /sys/devices/system/node/node0/hugepages/hugepages-1048576kB/nr_hugepages
echo <more> > /sys/devices/system/node/node1/hugepages/hugepages-1048576kB/nr_hugepages
```

To configure a vDPA VirtIO interface in an existing VM's xml file (using libvirt):

1. Open the VM's configuration xml for editing:

```
virsh edit <domain name>
```

2. Modify/add the following:
    a. Change the top line to:

    ```
    <domain type='kvm' xmlns:qemu='http://libvirt.org/schemas/domain/qemu/1.0'>
    ```

    b. Assign a memory amount and use 1GB page size for hugepages (size must be the same as used for the vDPA application), so that the memory configuration looks like the following.

    ```
    <memory unit='KiB'>4194304</memory>
    <currentMemory unit='KiB'>4194304</currentMemory>
    <memoryBacking>
      <hugepages>
        <page size='1048576' unit='KiB'/>
      </hugepages>
    </memoryBacking>
    ```

    c. Assign an amount of CPUs for the VM CPU configuration, so that the `vcpu` and `cputune` configuration looks like the following.

    ```
    <vcpu placement='static'>5</vcpu>
    <cputune>
      <vcpupin vcpu='0' cpuset='14'/>
      <vcpupin vcpu='1' cpuset='16'/>
      <vcpupin vcpu='2' cpuset='18'/>
      <vcpupin vcpu='3' cpuset='20'/>
      <vcpupin vcpu='4' cpuset='22'/>
    </cputune>
    ```

    d. Set the memory access for the CPUs to be shared, so that the `cpu` configuration looks like the following.

```
<cpu mode='custom' match='exact' check='partial'>
  <model fallback='allow'>Skylake-Server-IBRS</model>
  <numa>
    <cell id='0' cpus='0-4' memory='8388608' unit='KiB' memAccess='shared'/>
  </numa>
</cpu>
```

e. Set the emulator in use to be the one built in step "2. Build QEMU" above, so that the emulator configuration looks as follows.

```
<emulator><path to qemu executable></emulator>
```

f. Add a virtio interface using qemu command line argument entries, so that the new interface snippet looks as follows.

```
<qemu:commandline>
  <qemu:arg value='-chardev'/>
  <qemu:arg value='socket,id=charnet1,path=/tmp/sock-virtio0'/>
  <qemu:arg value='-netdev'/>
  <qemu:arg value='vhost-user,chardev=charnet1,queues=16,id=hostnet1'/>
  <qemu:arg value='-device'/>
  <qemu:arg value='virtio-net-
pci,mq=on,vectors=6,netdev=hostnet1,id=net1,mac=e4:11:c6:d3:45:f2,bus=pci.0,addr=0x6,
   page-per-vq=on,rx_queue_size=1024,tx_queue_size=1024'/>
</qemu:commandline>
```

Note: In this snippet, the vhostuser socket file path, the amount of queues, the MAC and the PCI slot of the VirtIO device can be configured.

3.3.6.6.3 Running Hardware vDPA

⚠ Hardware vDPA supports SwitchDev mode only.

Create the ASAP^2 environment:

1. Create the VFs.
2. Enter switchdev mode.
3. Set up OVS.

***Run** the vDPA application.*

```
cd $RTE_SDK/examples/vdpa/build
./vdpa -w <VF PCI BDF>,class=vdpa --log-level=pmd,info -- -i
```

Create a vDPA port via the vDPA application CLI.

```
create /tmp/sock-virtio0 <PCI DEVICE BDF>
```

Note: The vhostuser socket file path must be the one used when configuring the VM.

Start the VM.

```
virsh start <domain name>
```

For further information on the vDPA application, please visit: https://doc.dpdk.org/guides/sample_app_ug/vdpa.html.

3.3.6.7 Bridge Offload

- Bridge offload is supported on ConnectX-6 Dx NIC
- Bridge offload is supported SwitchDev mode only
- Bridge offload is supported from kernel version 5.15

A Linux bridge is in-kernel software network switch (based on and implements subset of IEEE 802.1D standard) that is used to connect Ethernet segments together in a protocol-independent way. Packets are forwarded based on L2 Ethernet header addresses.

mlx5 provides capabilities to offload bridge data-plane unicast packet forwarding and VLAN management to hardware.

3.3.6.7.1 Basic Configuration

1. Initialize the ASAP^2 environment:
    a. Create the VFs.
    b. Enter switchdev mode.
2. Create a bridge and add mlx5 representors to bridge:

```
ip link add name bridge0 type bridge
ip link set enp8s0f0_0 master bridge0
```

3.3.6.7.2 Configuring VLAN

1. Enable VLAN filtering on the bridge.

```
ip link set bridge0 type bridge vlan_filtering 1
```

2. Configure port VLAN matching (trunk mode). In this configuration, only packets with specified VID are allowed.

```
bridge vlan add dev enp8s0f0_0 vid 2
```

3. Configure port VLAN tagging (access mode). In this configuration VLAN header in pushed/popped on reception/transmission on port.

```
bridge vlan add dev enp8s0f0_0 vid 2 pvid untagged
```

3.3.6.7.3 VF LAG Support

Bridge supports offloading on bond net device that is fully initialized with mlx5 uplink representors and is in single (shared) FDB LAG mode. Details about initialization of LAG are provided in SR-IOV VF LAG section, above.

Add bonding net device to bridge.

```
ip link set bond0 master bridge0
```

For further information on interacting with Linux bridge via iproute2 bridge tool, please consult man page (man 8 bridge).

3.3.6.8 Appendix: NVIDIA Firmware Tools

Download and install the MFT package corresponding to your computer's operating system. You would need the kernel-devel or kernel-headers RPM before the tools are built and installed.
The package is available at nvidia.com/en-us/networking/ → Products → Software → Firmware Tools.

1. Start the mst driver.

```
# mst start
Starting MST (Mellanox Software Tools) driver set
Loading MST PCI module - Success
Loading MST PCI configuration module - Success
Create devices
```

2. Show the devices status.

```
ST modules:
------------
    MST PCI module loaded
    MST PCI configuration module loaded

PCI devices:
------------
DEVICE_TYPE             MST                           PCI      RDMA NET        NUMA
ConnectX4lx(rev:0)        /dev/mst/mt4117_pciconf0.1    04:00.1        net-enp4s0f1  NA
ConnectX4lx(rev:0)        /dev/mst/mt4117_pciconf0      04:00.0        net-enp4s0f0  NA


# mlxconfig -d /dev/mst/mt4117_pciconf0 q | head -16

Device #1:
----------

Device type:    ConnectX4lx
PCI device:     /dev/mst/mt4117_pciconf0

Configurations:                                 Current
         SRIOV_EN                               True(1)
         NUM_OF_VFS                             8
         PF_LOG_BAR_SIZE                        5
         VF_LOG_BAR_SIZE                        5
         NUM_PF_MSIX                            63
         NUM_VF_MSIX                            11
         LINK_TYPE_P1                           ETH(2)
         LINK_TYPE_P2                           ETH(2)
```

3. Make sure your configuration is as follows:
   * SR-IOV is enabled (SRIOV_EN=1)
   * The number of enabled VFs is enough for your environment (NUM_OF_VFS=N)
   * The port's link type is Ethernet (LINK_TYPE_P1/2=2) when applicable

   If this is not the case, use mlxconfig to enable that, as follows:
   a. Enable SR-IOV.

```
# mlxconfig -d /dev/mst/mt4115_pciconf0 s SRIOV_EN=1
```

   b. Set the number of required VFs.

```
# mlxconfig -d /dev/mst/mt4115_pciconf0 s NUM_OF_VFS=8
```

   c. Set the link type to Ethernet.

```
# mlxconfig -d /dev/mst/mt4115_pciconf0 s LINK_TYPE_P1=2
# mlxconfig -d /dev/mst/mt4115_pciconf0 s LINK_TYPE_P2=2
```

4. Conduct a cold reboot (or a firmware reset).

```
# mlxfwreset -d /dev/mst/mt4115_pciconf0 reset
```

5. Query the firmware to make sure everything is set correctly.

```
# mlxconfig -d /dev/mst/mt4115_pciconf0 q
```

3.4 Troubleshooting

You may be able to easily resolve the issues described in this section. If a problem persists and you are unable to resolve it yourself, please contact your NVIDIA representative or NVIDIA Support at networking-support@nvidia.com.

.

The chapter contains the following sections:

3.4.1 General Issues

Issue	Cause	Solution
The system panics when it is booted with a failed adapter installed.	Malfunction hardware component	1. Remove the failed adapter. 2. Reboot the system.
NVIDIA adapter is not identified as a PCI device.	PCI slot or adapter PCI connector dysfunctionality	1. Run lspci. 2. Reseat the adapter in its PCI slot or insert the adapter to a different PCI slot. If the PCI slot confirmed to be functional, the adapter should be replaced.
NVIDIA adapters are not installed in the system.	Misidentification of the NVIDIA adapter installed	Run the command below and check NVIDIA's MAC to identify the NVIDIA adapter installed. `lspci | grep Mellanox' or 'lspci -d 15b3:` **Note**: NVIDIA MACs start with: 00:02:C9:xx:xx:xx, 00:25:8B:xx:xx:xx or F4:52:14:xx:xx:xx"
Insufficient memory to be used by udev upon OS boot.	udev is designed to fork() new process for each event it receives so it could han- dle many events in parallel, and each udev instance consumes some RAM memory.	Limit the udev instances running simultaneously per boot by adding udev.children-max=<number> to the kernel command line in grub.

Issue	Cause	Solution
Operating system running from root file system located on a remote storage (over NVIDIA devices), hang during reboot/shutdown (errors such as “No such file or directory” will appear).	The mlnx-en.d service script is called using the ‘stop’ option by the operating system. This option unloads the driver stack. Therefore, the OS root file system dis- appears before the reboot/ shutdown procedure is completed, leaving the OS in a hang state.	Disable the openibd ‘stop’ option by setting 'ALLOW_STOP=no' in /etc/ mlnx-en.conf configuration file.

3.4.2 Ethernet Related Issues

Issue	Cause	Solution
Ethernet interfaces renaming fails leaving them with names such as renameXY.	Invalid udev rules.	Review the udev rules inside the "/etc/ udev/rules.d/70-persistent-net.rules" file. Modify the rules such that every rule is unique to the target interface, by adding correct unique attribute values to each interface, such as dev_id, dev_port and KERNELS or address). Example of valid udev rules: `SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{dev_id}=="0x0", ATTR{type}=="1", KERNEL=="eth*", ATTR{dev_port}=="0", KER- NELS=="0000:08:00.0", NAME="eth4" SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{dev_id}=="0x0", ATTR{type}=="1", KERNEL=="eth*", ATTR{dev_port}=="1", KER- NELS=="0000:08:00.0", NAME="eth5"`
No link.	Misconfiguration of the switch port or using a cable not supporting link rate.	• Ensure the switch port is not down • Ensure the switch port rate is configured to the same rate as the adapter's port

Degraded performance is measured when having a mixed rate environment (10GbE, 40GbE and 56GbE).	Sending traffic from a node with a higher rate to a node with lower rate.	Enable Flow Control on both switch ports and nodes:
		• On the server side run:
		`ethtool -A <interface> rx on tx on`
		• On the switch side run the following command on the relevant interface:
		`send on force and receive on force`

No link with break-out cable.	Misuse of the break-out cable or misconfiguration of the switch's split ports	• Use supported ports on the switch with proper configuration. For further information, please refer to the MLNX_OS User Manual. • Make sure the QSFP breakout cable side is connected to the SwitchX.

3.4.3 Installation Related Issues

3.4.3.1 Application Binary Interface (ABI) Incompatibility with MLNX_EN Kernel Modules

⚠ This section is relevant for RedHat and SLES distributions only.

3.4.3.1.1 Overview

MLNX_EN package for RedHat comes with RPMs that support KMP (weak-modules), meaning that when a new errata kernel is installed, compatibility links will be created under the weak-updates directory for the new kernel. Those links allow using the existing MLNX_EN kernel modules without the need for recompilation. However, at times, the ABI of the new kernel may not be compatible with the MLNX_EN modules, which will prevent loading them. In this case, the MLNX_EN modules must be rebuilt against the new kernel.

3.4.3.1.2 Detecting ABI Incompatibility with MLNX_EN Modules

When MLNX_EN modules are not compatible with a new kernel from a new OS or errata kernel, no links will be created under the weak-updates directory for the new kernel, causing the driver load to fail. Checking for the existence of needed module links under weak-updates directory can be done by reloading the MLNX_EN modules. If one or more modules are missing, the driver reload will fail with an error message.

Example:

```
********************************************************************************
# /etc/init.d/mlnx-en.d restart
Unloading HCA driver:                                      [  OK  ]
Loading HCA driver and Access Layer:                       [  OK  ]
Module rdma_cm belong to kernel which is not a part of MLNX[FAILED]kipping...
Loading rdma_ucm                                           [FAILED]
********************************************************************************
```

3.4.3.1.2.1 Resolving ABI Incompatibility with MLNX_EN Modules

In order to fix ABI incompatibility with MLNX_EN modules, the modules should be recompiled against the new kernel, using the mlnx_add_kernel_support.sh script, available in MLNX_EN installation image.
There are two ways to recompile the MLNX_EN modules:

1. Local recompilation and installation on one server.
   Run the `install` command to recompile the kernel modules and reinstall the whole MLNX_EN on the server. Mount MLNX_EN ISO image or extract the TGZ file:

   ```
   # cd <MLNX_EN dir>
   # ./install --skip-distro-check --add-kernel-support --kmp --force
   ```

   Notes:
   - The `--kmp` flag will enable rebuilding RPMs with KMP (weak-updates) support for the new

kernel. Therefore, in the next OS/kernel update, the same modules can be used with the new kernel (assuming that the ABI compatibility was not broken again).
- The command above will rebuild only the kernel RPMs (using mlnx_add_kernel_support.sh), and will save the resulting MLNX_EN package under /tmp and start installing it automatically. This package can be used for installation on other servers using regular `install` command or yum.

2. Preparing a new image on one server and deploying it on the cluster.
    a. Use the `mlnx_add_kernel_support.sh` script directly only to rebuild the kernel RPMs (without running any installations) on one server. Mount MLNX_EN ISO image or extract the TGZ file:

```
# cd <MLNX_EN dir>
# ./mlnx_add_kernel_support.sh -m $PWD --kmp -y
```

Note: This command will save the resulting MLNX_EN package under /tmp.
Example:

```
********************************************************************************
# cd /tmp/MLNX_EN_LINUX-5.2-2.2.0.0-DB-rhel7.8-x86_64
# ./mlnx_add_kernel_support.sh -m $PWD --kmp -y
Note: This program will create mlnx-en TGZ for rhel7.8 under /tmp directory.
See log file /tmp/mlnx_iso.28286_logs/mlnx_ofed_iso.28286.log

Checking if all needed packages are installed...
Building mlnx-en RPMS . Please wait...

Creating metadata-rpms for 3.10.0-1127.el7.x86_64 ...
WARNING: If you are going to configure this package as a repository, then please note
WARNING: that it contains unsigned rpms, therefore, you need to disable the gpgcheck
WARNING: by setting 'gpgcheck=0' in the repository conf file.
Created /tmp/mlnx-en-5.3-1.0.0.1-rhel7.8-x86_64-ext.tgz
********************************************************************************
```

    b. Install the newly created MLNX_EN package on the cluster:
    Option 1: Copy the package to the servers and install it using the `install` script.
    Option 2: Deploy the MLNX_EN package using YUM (for YUM installation instructions, refer to Installing MLNX_EN Using YUM section):
        i. Extract the resulting MLNX_EN image and copy it to a shared NFS location.
        ii. Create a YUM repository configuration.
        iii. Install the new MLNX_EN kernel RPMs on the servers: `# yum update`
    Example:

```
********************************************************************************
...
...
================================================================================================================
=====================
Package                 Arch     Version
Repository  Size
================================================================================================================
=====================
Updating:
epel-release            noarch
7-7                                                                 epel        14 k
kmod-iser               x86_64   1.8.0-OFED.3.3.1.0.0.1.gf583963.201606210906.rhel7u1
mlnx_ofed   35 k
kmod-isert              x86_64   1.0-OFED.3.3.1.0.0.1.gf583963.201606210906.rhel7u1
mlnx_ofed   32 k
kmod-kernel-mft-mlnx    x86_64   4.4.0-1.201606210906.rhel7u1
mlnx_ofed   10 k
kmod-knem-mlnx          x86_64   1.1.2.90mlnx1-OFED.3.3.0.0.1.0.3.1.ga04469b.201606210906.rhel7u1
mlnx_ofed   22 k
kmod-mlnx-ofa_kernel    x86_64   3.3-OFED.3.3.1.0.0.1.gf583963.201606210906.rhel7u1
mlnx_ofed  1.4 M
kmod-srp                x86_64   1.6.0-OFED.3.3.1.0.0.1.gf583963.201606210906.rhel7u1
mlnx_ofed   39 k

Transaction Summary
================================================================================================================
====================
Upgrade  7 Packages
...
...
```

```
*******************************************************************************
```

Note: The MLNX_EN user-space packages will not change; only the kernel RPMs will be updated. However, “YUM update” can also update other inbox packages (not related to OFED). In order to install the MLNX_EN kernel RPMs only, make sure to run:

```
# yum install mlnx-en-kernel-only
```

Note: `mlnx-en-kernel-only` is a metadata RPM that requires the MLNX_EN kernel RPMs only.

c. Verify that the driver can be reloaded:

```
# /etc/init.d/mlnx-en.d restart
```

3.4.4 Performance Related Issues

Issue	Cause	Solution
The driver works but the transmit and/or receive data rates are not optimal.	-	These recommendations may assist with gaining immediate improvement: 1. Confirm PCI link negotiated uses its maximum capability 2. Stop the IRQ Balancer service: `/etc/init.d/irq_balancer stop` 3. Start mlnx_affinity service: `mlnx_affinity start` For best performance practices, please refer to the "Performance Tuning Guide for NVIDIA Network Adapters".
Out of the box throughput performance in Ubuntu14.04 is not optimal and may achieve results below the line rate in 40GE link speed.	IRQ affinity is not set properly by the irq_balancer	For additional performance tuning, please refer to Performance Tuning Guide.

3.4.5 SR-IOV Related Issues

Issue	Cause	Solution
When assigning a VF to a VM the following message is reported on the screen: `PCI-assgine: error: requires KVM support`	SR-IOV and virtualization are not enabled in the BIOS.	1. Verify they are both enabled in the BIOS 2. Add to the GRUB configuration file to the following kernel parameter: "intel_immun=on" (see “Setting Up SR-IOV” section).

3.4.6 OVS Offload Using ASAP2 Direct Related Issues

Issue	Cause	Solution(s)
Traffic is not offloaded	OVS uses TC flower classifier to add offloading rules to both the software and the hardware. • TC flower classifier fails to add a rule. • A rule was added to the TC flower classifier but failed to be added to the firmware.	• Check for system error in dmesg or the system logging facility like journalctl • Check OVS logs for errors • Dump the rules using the TC command line For example: Dump rules on a specific interface `# tc filter show dev ens4f0 parent ffff:`

4 Common Abbreviations and Related Documents

Common Abbreviations and Acronyms

Abbreviation/ Acronym	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
RoCE	RDMA over Converged Ethernet
SL	Service Level
SRP	SCSI RDMA Protocol
MPI	Message Passing Interface
QoS	Quality of Service
ULP	Upper Layer 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*.

Term	Description
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	An 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
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
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 the agency of the master SM
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 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 Interconnect (VPI)	An NVIDIA technology that allows NVIDIA channel adapter devices (ConnectX®) to simultaneously connect to an InfiniBand subnet and a 10GigE subnet (each subnet connects to one of the adapter ports)

Related Documentation

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
Firmware Release Notes for NVIDIA adapter devices	See the Release Notes relevant to your adapter device
MFT User Manual and Release Notes	NVIDIA Firmware Tools (MFT) User Manual and Release Notes documents

Document Name	Description
WinOF User Manual	Mellanox WinOF User Manual describes the installation, configuration, and operation of NVIDIA Windows driver
VMA User Manual	NVIDIA VMA User Manual describes the installation, configuration, and operation of NVIDIA VMA driver

5 Documentation History

5.1 Release Notes History

5.1.1 Changes and New Features History

This section includes history of changes and new feature of three major (GA) releases back. For older versions' history, please refer to their dedicated release notes.

Supported Cards	Description
All HCAs	Supported in the following adapter cards **unless specifically stated otherwise:** ConnectX-4 / ConnectX -4 Lx / ConnectX-5 / ConnectX-6 / ConnectX-6 Dx / ConnectX-6 Lx / ConnectX-7 / BlueField-2
ConnectX-6 Dx and above	Supported in the following adapter cards **unless specifically stated otherwise:** ConnectX-6 Dx / ConnectX-6 Lx / ConnectX-7 / BlueField-2
ConnectX-6 and above	Supported in the following adapter cards **unless specifically stated otherwise:** ConnectX-6 / ConnectX-6 Dx / ConnectX-6 Lx / ConnectX-7 / BlueField-2
ConnectX-5 and above	Supported in the following adapter cards **unless specifically stated otherwise:** ConnectX-5 / ConnectX-6 / ConnectX-6 Dx / ConnectX-6 Lx / ConnectX-7 / BlueField-2
ConnectX-4 and above	Supported in the following adapter cards **unless specifically stated otherwise:** ConnectX-4 / ConnectX -4 Lx / ConnectX-5 / ConnectX-6 / ConnectX-6 Dx / ConnectX-6 Lx / ConnectX-7 / BlueField-2

Feature/Change	Description
23.10-5.1.4.0	
General	
General	Bug fixes

Feature/Change	Description
23.10-4.0.9.1	
General	
Embedded Components	Updated the versions of the following embedded component: • MFT • ConnectX adapter cards firmware For further information, see Embedded Components section.
General	Bug fixes

Feature/Change	Description
23.10-3.2.2.0	
Operating Systems	Added support for the following OSes: • RHEL 8.10 • RHEL 9.4 • SLES15-SP6 • Debian 12.5 For further information, see Supported Operating Systems section
Embedded Components	Updated the versions of the following embedded component: • MFT • ConnectX adapter cards firmware For further information, see Embedded Components section.
Bridge Offload	Enabled Bridge Offload in mlx5_core by default. A new option was added to the configure of mlnx_ofa_kernel to disable it: `"--without-bridge-offload"`. To use this option from `"mlnxofedinstall --add-kernel-support"` (kernel modules rebuilding), add the following to the command line: `"--kernel-extra-args --without-bridge-offload"`.
General	Bug fixes

5.9-0.5.6.0	
ASAP^2 Features	
Linux Bridge VLAN Filtering of 802.1 Q Packets	[ConnectX-6 Dx] Extended mlx5 Linux bridge VLAN offload to support packets tagged with 802.1 Q VLAN ethertype.
Offloading sFlow Sampling Rules	[ConnectX-5 and above] Added support for sFlow sampling rules offloads. sFlow is an industry standard technology for monitoring high speed switched networks. Open vSwitch integrated sFlow to extend the visibility into virtual servers, ensuring data center visibility and control.
Core Features	
Configuring Shared Buffer Size	[ConnectX-6 Dx and above] Enabled user to control shared buffer size and configuration, implicitly. As with each port buffer command the user triggers, the shared buffer configuration will be updated accordingly by the driver.
Control SF Class	[All HCAs] Added support for Control SF Class. Each PCI, PF, VF, SF function, by default, has netdevice, RDMA, and vdpa-net devices always enabled. This feature enables the user to control which device functionality to enable/disable. **Note:** Requires kernel 5.18 or higher.
NetDev Features	
Support RSS over XSK Queues	[All HCAs] Use default RSS functionality to spread traffic across different XSK queues instead of having to provide explicit steering rules.
TLS TIS Pool	[TLS-Enabled Devices] Per-connection hardware TIS objects is used to maintain the device TLS TX context. Use a SW TIS pool for recycling the TIS objects instead of destroying/creating them. This reduces the interaction with the device via the FW command interface, which increases the TLS connection rate.

RDMA Features	
Expand Rep Counters	[ConnectX-5 and above] Adding RDMA traffic-only counters for rep devices. These counters can now be read from host with ethtool or from sysfs and not only from the cointainer.
UMR QP Recilency	[ConnectX-5 and above] Added a recovery flow for the driver's UMR logic so that other UMR requests can be proccessed after the error UMR was dropped and the UMR QP was reset. Previously, a faulty UMR request would have moved the QP to error state and disable any option to continue issuing UMRs.
General	Bug fixes

Feature/ Change	Description
5.8-1.1.2.1	
General	Bug fixes
5.8-1.0.1.1	
Remove Dependency Between SR-IOV and eSwitch Mode	[All HCAs] Removed dependency between SR-IOV and eSwitch mode. Currently, there are three eSwitch modes: none, legacy, and switchdev (non of which are the default mode). When disabling SR-IOV, the current eSwitch mode will be changed to none. This feature removes eSwitch mode none and also removes dependency between SR-IOV and eSwitch mode.
DevLink Parallel Command	[All HCAs] Added support for running DevLink commands in parallel on different DevLink devices is possible. For example, burning firmware on a few cards on the same host in parallel using DevLink API is now possible.
Graceful Shutdown of Parent and Page Supplier	[All HCAs] Set default graceful period values for functions based on their type. ECPFs will get graceful period of 3 minutes, PFs get 1 minute, and VFs/SFs get 30 seconds.
N Pulses Per Second (NPPS)	[ConnectX-6 Dx and above] Enhanced NPPS to allow setting a pulse period higher than 1 pulse per second and to allow setting the pulse width. If the width is unset, the driver implicitly sets it to half the given period (the width should be less than the pulse period). In this release, the pulse duration ranges between 65536 NS-524288 NS.
Remote Invalidate Option for MKeys	[All HCAs] Added support for the option to enable remote invalidation when creating a new mkey. This way the rkey for a memory region can be changed frequently.
GPUDirect Over DMA-BUF	[All HCAs] Added support for GPUDirect support over dma-buf. As such, using the new mechanism nv_peer_mem is no longer required. The following is required for dma-buf support: • Linux kernel version 5.12 or later • OpenRM version 515 or later Perftest support was added as well: Default option in perftest is without dmabuf. To run with this option, add --use_cuda_dmabuf in addition to use_cuda flag.
General	Bug fixes

Feature/Change	Description
23.10-2.1.3.1	
General	Bug fixes

Feature/ Change	Description
5.7-1.0.2.0	
Support Representor Metering Over SFs	[ConnectX-6 Dx and above and BlueField-2] Extended the support of representor metering from supporting only VFs representor to also supporting SFs representor.
Exposing Error Counters on a VPort Manager	[ConnectX-4 and above] Added support for exposing error counters on a VPort manager function for all other VPorts. These counters can be used to detect malicious users who are exploiting flows that can slow the device. The counters are exposed through debugfs under: /sys/kernel/debug/mlx5/esw/<func>/ vnic_diag/
Memory Consumption Minimization	[ConnectX-4 and above] Added support for providing knobs which enable users to minimize memory consumption of mlx5 functions (PF/VF/SF).
XDP Support for Uplink Representors	[ConnectX-5 and ConnectX-6 Dx) Added XDP support for uplink representors in switchdev mode.
Resiliency to tx_port_ts	[ConnectX-6 Dx and above] Added resiliency to the tx_port_ts feature. private-flag may be enabled via ethtool tx_port_ts which provides a more accurate time-stamp. In very rare cases, the said time-stamp was lost, leading to losing the synchronization altogether. This feature allows for fast recovery and allows to quickly regain synchronization.
Database of Devlink Health Asserts	[ConnectX-4 and above] Health buffer now contains more debug information like the epoch time in sec of the error and the error's severity. The print to dmesg is done with the debug level corresponding to the error's severity. This allows the user to use dmesg attribute: dmesg --level to focus on different severity levels of firmware errors.
Expose FEC Counters via Ethtool	[ConnectX-5 and above] Exposed the following FEC (forward error detection) counters: ETHTOOL_A_FEC_STAT_CORRECTED • fc_fec_corrected_blocks_laneX • rs_fec_corrected_blocks ETHTOOL_A_FEC_STAT_UNCORR • fc_fec_uncorrectable_blocks_laneX • rs_fec_uncorrectable_blocks ETHTOOL_A_FEC_STAT_CORR_BITS • phy_corrected_bits Command: ethtool -I show-fec <ifc>
Application Device Queues	[ConnectX-4 and above] Added driver-level support for Application Device Queues. This feature allows partition defining over the RX/TX queues into groups and isolates traffic of different applications. This mainly improves predictability and tail latency.

Feature/ Change	Description
5.7-1.0.2.0	
Reinjection of Packets Into Kernel	[All HCAs] Added support for a new software steering action, mlx5dv_dr_action_create_dest_root_table(). This action can be used to forward packets back into a level 0 table. As a table with level 0 is the kernel owned table, this will result in injecting packets to the kernel steering pipeline.
DCT LAG	[ConnectX-6 Dx and above] Added firmware support to allow explicit port selection based on steering and not QP affinity. Functionality: 1. Use LAG Hash Mode for the HCA with two ports, if supported. 2. Keep port affinity function in LAG Hash Mode if it supports bypass select flow table in non-SwitchDev mode.
AES-XTS in RDMA	Added support for plaintext AES-XTS DEKs.
General	Bug fixes

5.1.1.1 Customer Affecting Changes

Feature/Change	Description
23.10-1.1.9.0	
Lightweight Local SFs	Following the addition of the Lightweight Local SFs feature in version 23.07, in order to configure the scalable-functions, follow the revised instructions as detailed in the Step-by-Step Guide. **Note:** "Step 2.9 - Set all SF specific device parameters" is now mandatory for local SFs.

23.10-0.5.5.0	
Customer Affecting Change	**Description**
Debugfs Directory Path Change	The debugfs directory of each interface can now be found under: /sys/kernel/debug/mlx5//, and not directly under the root of the debugfs filesystem (/sys/debug/kernel).
Deprecation of OFED Public Power PC Installation	Starting from this release, MLNX_OFED releases for Power PC are no longer available for download from the public Download Center web page. Instead, you can find it on the following page: https://network.nvidia.com/support/firmware/ibm-systemp/.
Pre-notification: Deprecation of Older Operating Systems	Starting from next release, MLNX_OFED releases will no longer support operating systems with kernels below v4.18. This includes the following systems: • Rhel7.x • Debian9.13 • Sles12.x • Xenserver7.1
Customer Affecting Change	**Description**

23.07-0.5.1.2	
Creating a QKEY with an MSB Set	To allow non-privileged users to create a QKEY with an MSB set, a new module parameter was added. For details, please see "QKEY Mitigation in the Kernel" under New Features.
23.07-0.5.0.0	
IRQ Naming	IRQ renaming is no longer done when bringing the interface up/down. The IRQ name is now constant and is not affected by the interface state.
RPM Packages Verification Key	RPM_GPG-KEY-Mellanox (or its variants) is no longer the public key that verifies RPM packages of MLNX_OFED. Instead, the RPM_GPG-KEY-Mellanox file on the top-level directory of the ISO should be used.
Hairpin sysfs Support	Hairpin sysfs support was restricted to physical and virtual functions only.
mlx5_core node_guid Module Parameter	Removed a non-functional mlx5_core node_guid module parameter.
OpenSM Init	Starting from this release, the opensm init service moves from init.d (`/etc/init.d/opensmd start`) to systemd (`# service opensmd start`).
Apt Signing Key	Starting from this release, the public key that signed the apt repository of MLNX_OFED is included in the ISO in a format that can be used directly by the apt for repository signatures verification.
IPoIB ULP Mode Deprecation	Starting from this release, MLNX_OFED supports IPoIB enhanced mode only. The ability to switch back to ULP mode using ipoib_enhanced module parameter is not supported. For more information about the enhanced mode, please refer to the OFED user manual, example: Enhanced IP over InfiniBand.
Pre-notification: Deprecation of OFED Public Power PC Installation	Starting from next release, MLNX_OFED releases for Power PC will no longer be available for download from the public Download Center web page.

Customer Affecting Change	Description
23.07-0.5.0.0	
Creating a QKEY with an MSB Set	To allow non-privileged users to create a QKEY with an MSB set, a new module parameter was added. For details, please see "QKEY Mitigation in the Kernel" under New Features.
IRQ Naming	IRQ renaming is no longer done when bringing the interface up/down. The IRQ name is now constant and is not affected by the interface state.
RPM Packages Verification Key	RPM_GPG-KEY-Mellanox (or its variants) is no longer the public key that verifies RPM packages of MLNX_OFED. Instead, the RPM_GPG-KEY-Mellanox file on the top-level directory of the ISO should be used.
Hairpin sysfs Support	Hairpin sysfs support was restricted to physical and virtual functions only.
mlx5_core node_guid Module Parameter	Removed a non-functional mlx5_core node_guid module parameter.
OpenSM Init	Starting from this release, the opensm init service moves from init.d (`/etc/init.d/opensmd start`) to systemd (`# service opensmd start`).

Customer Affecting Change	Description
23.07-0.5.0.0	
Apt Signing Key	Starting from this release, the public key that signed the apt repository of MLNX_OFED is included in the ISO in a format that can be used directly by the apt for repository signatures verification.
IPoIB ULP Mode Deprecation	Starting from this release, MLNX_OFED supports IPoIB enhanced mode only. The ability to switch back to ULP mode using ipoib_enhanced module parameter is not supported. For more information about the enhanced mode, please refer to the OFED user manual, example: Enhanced IP over InfiniBand.
Pre-notification: Deprecation of OFED Public Power PC Installation	Starting from next release, MLNX_OFED releases for Power PC will no longer be available for download from the public Download Center web page.

Customer Affecting Change	Description
23.04-0.5.3.3	
Netdev Interface Configuration is not Preserved During Reload/ Reset/ Recovery	As of OFED 23.04, during reset/reload/recovery flows, the netdev interface is destroyed and re-created (rather than just suspended). As a result, the netdev interface configuration is not preserved, and must be re-applied. The way to do this is to use proper network-scripts and/or udev rules files to configure network interface parameters. These are automatically triggered whenever a netdev interface is added, regardless of whether it was added due to a user-initiated operation or an automatic failure recovery operation. Thus, no special processing is required to re-apply the network interface configuration parameters following a reset/reload/recovery operation – it is performed automatically.
Prenotification : Deprecation of OFED Public Power PC Installation	Starting next release, MLNX_OFED releases for Power PC will no longer be available for download from the public Download Center web page.
Prenotification : ULP Mode Deprecation	Starting next release, MLNX_OFED will support IPoIB enhanced mode only. The ability to switch back to ULP mode using ipoib_enhanced module param will not be supported. For more information about the enhanced mode, please refer to OFED user manual, example: Enhanced IP over InfiniBand
Installation, ISO, RedHat	In order to address RHEL kernel symbol changes, ISO images for the following operating systems are built with the updated kernel versions as follows: OS Name Old Kernel New Kernel rhel8.6-aarch64 4.18.0-372.9.1.el8_6.aarch64 4.18.0-372.41.1.el8_6.aarch64 rhel8.6-ppc64le 4.18.0-372.0.1.el8_6.ppc64le 4.18.0-372.41.1.el8_6.ppc64le rhel8.6-x86_64 4.18.0-372.9.1.el8_6.x86_64 4.18.0-372.41.1.el8_6.x86_64 rhel8.7-aarch64 4.18.0-425.3.1.el8.aarch64 4.18.0-425.14.1.el8_7.aarch64 rhel8.7-ppc64le 4.18.0-425.3.1.el8.ppc64le 4.18.0-425.14.1.el8_7.ppc64le rhel8.7-x86_64 4.18.0-425.3.1.el8.x86_64 4.18.0-425.14.1.el8_7.x86_64 rhel9.0-aarch64 5.14.0-70.13.1.el9_0.aarch64 5.14.0-70.46.1.el9_0.aarch64 rhel9.0-ppc64le 5.14.0-70.13.1.el9_0.ppc64le 5.14.0-70.46.1.el9_0.ppc64le rhel9.0-x86_64 5.14.0-70.13.1.el9_0.x86_64 5.14.0-70.46.1.el9_0.x86_64 rhel9.1-aarch64 5.14.0-162.6.1.el9_1.aarch64 5.14.0-162.19.1.el9_1.aarch64 rhel9.1-ppc64le 5.14.0-162.6.1.el9_1.ppc64le 5.14.0-162.19.1.el9_1.ppc64le rhel9.1-x86_64 5.14.0-162.6.1.el9_1.x86_64 5.14.0-162.19.1.el9_1.x86_64 This change comes to support RedHat updated kernels without the need to add --add-kernel-support during OFED installation.

Customer Affecting Change	Description
23.04-0.5.3.3	
Power Setups on UCX/HPC-X	UCX/HPC-X no longer supports Power setups.
NEO-Host	Starting from this release, OFED will discontinue the provision of NEO-Host. NEO-Host can be manually downloaded and installed using the following guide: https://docs.nvidia.com/networking/display/NEOSDKv26/Installation+and+Initial+Configuration#InstallationandInitialConfiguration-DownloadingtheMellanoxNEOSDKSoftware
dapl	Starting from this release, OFED will discontinue the provision of dapl.
Signing Key for SLES15 sp4 and sp5	As of version 23.04, the builds for SLES15 sp4 and sp5 are being singed with a newer signing key. The corresponding public key can be downloaded from https://www.mellanox.com/downloads/ofed/nv_nbu_kernel_signing_key_pub.der instead of https://www.mellanox.com/downloads/ofed/mlnx_signing_key_pub.der.
dump_pr SM Plugin	Starting from this release, OFED will discontinue the provision of dump_pr subnet manager plugin.
mpi-selector	Starting from this release, OFED will discontinue the provision of mpi-selector.
OpenSM Init	Starting 23.07 release, opensm init service will move from init.d to systemd.

Customer Affecting Change	Description
5.9-0.5.6.0	
Deprecation, LAG Mode via Sysfs	Setting LAG mode via Sysfs is going to be deprecated in a future release. Instead, LAG Hash mode will be used by default, similar to upstream behavior.
LAG Configuration, PCI Error	From version 5.9, LAG configuration will be lost in case driver incurs a PCI error. Make sure to reconfigure the bond after driver completes the recovery from the PCI error. In releases prior to 5.9, in case of PCI error (EEH injections on PPC setup), the driver recovers LAG bond and reconfigures it automatically in case it what configured before the appearance of the error.

Customer Affecting Change	Description
5.7-1.0.2.0	
Multi-Block Encryption	Multi-block encryption is currently unsupported, due to a hardware limitation.

Feature / Change	Description
5.6-2.0.9.0	
Operating Systems	Added support for the following Operating Systems: RHEL8.6, RHEL9.0, SLES15-SP4.
General	Bug fixes

5.1.2 Bug Fixes History

This table lists the bugs fixed in the last three major GA releases. For a list of old bug fixes, please refer to the release notes of the desired version.

Below are the bugs fixed in this version. For a list of fixes previous version, see Bug Fixes History.

Internal Reference Number	Description
4140272 / 4496401 / 4567694 / 4567695	**Description:** Fixed a rare issue where the kernel API `ib_free_cq()` could crash if a new IRQ or CQ work was submitted immediately after disabling the IRQ or canceling CQ work, potentially running concurrently with `destroy_cq`.
	Keywords: ib_free_cq, destroy_cq
	Discovered in Release: 23.10-5.1.4.0
	Fixed in Release: 23.10-6.1.6.1
4573786 / 4500815	**Description:** Fixed an issue that caused packet loss when enabling or disabling promiscuous mode on a network interface.
	Keywords: Promiscuous mode
	Discovered in Release: 23.10-5.1.4.0
	Fixed in Release: 23.10-6.1.6.1
4639994 / 4634995 / 4635609 / 4635610 / 4797618	**Description:** Fixed an issue where `mlx5dv_devx_umem_reg_ex()` would fail if `ib_umem_dmabuf_get_pinned()` was not defined.
	Keywords: UMEM_DMABUF_GET_PINNED, ib_umem_dmabuf_get_pinned
	Discovered in Release: 23.10-5.1.4.0
	Fixed in Release: 23.10-6.1.6.1
4648901 / 4606693	**Description:** Fixed a rare crash that could occur when a MAD completion queue (CQ) was destroyed while RDMA CM traffic was still active.
	Keywords: MAD CQ
	Discovered in Release: 23.10-5.1.4.0

Internal Reference Number	Description
	Fixed in Release: 23.10-6.1.6.1
4719792 / 4553499	**Description:** Fixed an issue where, on devices that do not support BlueFlame, allocation of a new Transport Domain (TD) could fail when attempting to allocate a dedicated UAR.
	Keywords: BlueFlame, Transport Domain (TD), UAR
	Discovered in Release: 23.10-5.1.4.0
	Fixed in Release: 23.10-6.1.6.1

Internal Reference Number	Description
4410029	**Description:** Fixed an issue where installing mlnx-ofa_kernel drivers on SLES 15 SP5 with kernel version 5.14.21-150500.55.68-default (and newer) failed due to weak-modules falling back to the original inbox modules. The failure was caused by a mismatch: the original build kernel (5.14.21-150500.53-default) did not include the mana_ib driver, so no dummy module was provided, while the newer kernel did include it. This mismatch led to weak-modules sanity check errors due to the presence of the inbox mana_ib driver.
	Keywords: mlnx-ofa_kernel, SLES 15 SP5
	Discovered in Release: 23.10-4.0.9.1
	Fixed in Release: 23.10-5.1.4.0
4471811	**Description:** Resolved NVMe driver compilation issue on Linux kernel version 6.6.87.
	Keywords: NVMe driver
	Discovered in Release: 23.10-4.0.9.1
	Fixed in Release: 23.10-5.1.4.0
4253229	**Description:** Fixed a race condition between the firmware syndrome report and driver initialization during boot.
	Keywords: Race condition
	Discovered in Release: 23.10-4.0.9.1
	Fixed in Release: 23.10-5.1.4.0
4442965	**Description:** Fixed performance degradation on older kernel versions using RX cache, particularly on slower ARM CPUs with larger RX buffers. The issue was caused by the driver attempting to allocate new RX pages too quickly, leading to head-of-line blocking in the RX cache. The fix improves RX cache usage by triggering page allocation for a bulk of at least 2 WQEs, allowing the application more time to process packets and return buffers to the RX cache, thereby reducing blocking and enhancing performance.
	Keywords: Performance, kernel, Rx cache, page allocation
	Discovered in Release: 23.10-4.0.9.1
	Fixed in Release: 23.10-5.1.4.0

Internal Reference Number	Description
4243800	**Description:** Resolved improper page deallocation handling issue present in some kernels.
	Keywords: Page deallocation
	Discovered in Release: 23.10-4.0.9.1
	Fixed in Release: 23.10-5.1.4.0
4466255	**Description:** Fixed an issue where a kernel crash could occur if a device event arrives during the event subscription process.
	Keywords: DevX, event_fd
	Discovered in Release: 23.10-4.0.9.1
	Fixed in Release: 23.10-5.1.4.0
4441119	**Description:** Fixed a crash caused by handling multiple CMA net events occurring in quick succession on the same CMA ID.
	Keywords: CMA
	Discovered in Release: 23.10-4.0.9.1
	Fixed in Release: 23.10-5.1.4.0
4405723	**Description:** Fixed a potential deadlock that could occur during the handling of peer memory registration failures.
	Keywords: Deadlock, peer memory registration
	Discovered in Release: 23.10-4.0.9.1
	Fixed in Release: 23.10-5.1.4.0
4340109	**Description:** Fixed a sysfs issue that occurred when accessing hardware counters from within a namespace.
	Keywords: sysfs
	Discovered in Release: 23.10-4.0.9.1
	Fixed in Release: 23.10-5.1.4.0
4248125	**Description:** Fixed the UMR QP recovery flow to ensure proper functionality and prevent tasks from getting stuck in the kernel. Additionally, resolved a race condition in the ODP MR area that could lead to a CQE error in the UMR QP.
	Keywords: UMR QP recovery flow
	Discovered in Release: 23.10-4.0.9.1
	Fixed in Release: 23.10-5.1.4.0
4235682	**Description:** Resolved corruption of SA MAD Congestion Control FIFO queue when all elements are canceled and a dequeue operation is attempted.
	Keywords: SA legacy congestion control mechanism
	Discovered in Release: 23.10-4.0.9.1
	Fixed in Release: 23.10-5.1.4.0

Internal Reference Number	Description

4409282	**Description:** Increased the size of the slow FDB table to prevent hitting the following error when switching to SwitchDev mode.
	`mlx5_core 0000:03:00.0: mlx5_cmd_out_err:835:(pid 24362): CREATE_FLOW_GROUP(0x933) op_mod(0x0) failed, status bad parameter(0x3), syndrome (0x4065f0), err(-22)`
	`mlx5_core 0000:03:00.0: E-Switch: Failed to create peer miss flow group err(-22)`

	Keywords: Slow FDB table
	Discovered in Release: 23.10-4.0.9.1
	Fixed in Release: 23.10-5.1.4.0

Internal Reference Number	Description
3992667	**Description:** Fixed an issue that caused the MLNX_EN uninstall script to keep some leftover packages from the last installation process.
	Keywords: Installation, MLNX_EN
	Discovered in Release: 23.10-3.2.2.0
	Fixed in Release: 23.10-4.0.9.1

Internal Reference Number	Description
4192798	**Description:** Fixed the issue where the device failed to initialize after a write-combine test failure. The device now loads with Blueflame capabilities disabled instead.
	Keywords: write-combine; Blueflame; device initialization
	Discovered in Release: 23.10-3.2.2.0
	Fixed in Release: 23.10-4.0.9.1
4111625	**Description:** MLNX_OFED can now successfully be built with `add-kernel-support` flag over SLES15-SP5 kernel 5.14.21-150500.55.73.
	Keywords: SLES; Kernel; operating system; OS
	Discovered in Release: 23.10-3.2.2.0
	Fixed in Release: 23.10-4.0.9.1
4128400	**Description:** Removed the udev rule error message "/usr/lib/udev/rules.d/90-ib.rules:4 Only network interfaces can be renamed" from the log files. The udev rule included a line in a syntax that is no longer valid that triggered the mentioned error.
	Keywords: udev rule
	Discovered in Release: 23.10-3.2.2.0
	Fixed in Release: 23.10-4.0.9.1

Internal Reference Number	Description
4001035 / 4001038	**Description:** Fixed an issue that resulted in corrupt SMP MAD requests list when the sent list was accessed while the unregistered flow was running.
	Keywords: SMP MAD requests
	Discovered in Release: 23.10-3.2.2.0
	Fixed in Release: 23.10-4.0.9.1
3991835	**Description:** Fixed a stack overrun warning by reducing the size of the local on-stack array used for optimization by 192 bytes.
	Keywords: Kernel Stack
	Discovered in Release: 23.10-3.2.2.0
	Fixed in Release: 23.10-4.0.9.1
4004300	**Description:** Fixed an issue that prevented netdev queue value from being updated in mqprio param when switchdev mode was enabled and the netdev queue number was reset to 1.
	Keywords: PF TXQ mapping
	Discovered in Release: 23.10-3.2.2.0
	Fixed in Release: 23.10-4.0.9.1
4176804	**Description:** Fixed the receive queue cache size calculation to take into account the host page size.
	Keywords: Memory allocation
	Discovered in Release: 23.10-3.2.2.0
	Fixed in Release: 23.10-4.0.9.1
4027218	**Description:** Fixed the packet inspection parsing to avoid data corruption when GRE offload was turned on by parsing the outer header as UDP and not as TCP.
	Keywords: UDP, TCP, GRE offload
	Discovered in Release: 23.10-3.2.2.0
	Fixed in Release: 23.10-4.0.9.1
4001551	**Description:** Fixed a CT entry update failure that was caused because of a firmware limitation, the old modify header context was not freed and had leaks.
	Keywords: CT entry update
	Discovered in Release: 23.10-3.2.2.0
	Fixed in Release: 23.10-4.0.9.1
4012710	**Description:** Increased the `MLX5E_TC_MAX_INT_PORT_NUM` value to 32 to avoid cases of rules not being offloaded.
	Keywords: Rules offloaded
	Discovered in Release: 23.10-3.2.2.0
	Fixed in Release: 23.10-4.0.9.1
4014362	**Description:** Fixed an issue that prevented the "hash" `lag_port_select_mode` from working properly with ConnectX-7 adapter cards on some old OSs.
	Keywords: LAG

Internal Reference Number	Description
	Discovered in Release: 23.10-3.2.2.0
	Fixed in Release: 23.10-4.0.9.1

Internal Reference Number	Description
3932946	**Description:** Fixed the setting of ATS for DMABUF MRs that caused some MRs to miss the ATS enablement. Lack of ATS enablement on DMABUF MRs results in slower performance when using these MRs.
	Keywords: ATS, DMABUF, MRs
	Discovered in Release: 23.10-2.1.3.1
	Fixed in Release: 23.10-3.2.2.0
3894403	**Description:** On rare occasions, rdmacm applications could not find the device upon creating new RDMA devices, as the CMA driver lost some of the devices due to an overflow issue.
	Keywords: rdmacm
	Discovered in Release: 23.10-2.1.3.1
	Fixed in Release: 23.10-3.2.2.0
3807155	**Description:** Fixed an issue that could have caused memory corruption when running XDP traffic.
	Keywords: tc_wrap tool, VLAN
	Discovered in Release: 23.10-2.1.3.1
	Fixed in Release: 23.10-3.2.2.0
3848999	**Description:** Fixed an issue that prevented the tc_wrap tool from properly working when VLAN is configured as the tool wrongly handled the library function return value.
	Keywords: tc_wrap tool, VLAN
	Discovered in Release: 23.10-2.1.3.1
	Fixed in Release: 23.10-3.2.2.0
3822520	**Description:** Fixed an issue that caused a deadlock in the flow of disabling the LAG when changing eswitch mode from switchdev to legacy when a LAG bond existed on the machine.
	Keywords: MR cache cleanup
	Discovered in Release: 23.10-2.1.3.1
	Fixed in Release: 23.10-3.2.2.0
3822520	**Description:** Fixed an issue related to the comparison process between the SW steering and FW steering modes to avoid kernel crashes incidences.
	Keywords: MR cache cleanup
	Discovered in Release: 23.10-2.1.3.1
	Fixed in Release: 23.10-3.2.2.0
3947195	**Description:** Fixed an issue related to the driver's internal MR cache cleanup that caused high memory consumption on the host.

Internal Reference Number	Description
	Keywords: MR cache cleanup
	Discovered in Release: 23.10-2.1.3.1
	Fixed in Release: 23.10-3.2.2.0

Internal Reference Number	Description
3729466	**Description:** Resolved a discalculation issue where more Q-counters were freed than allocated when moving to switchdev mode.
	Keywords: Q-counters, switchdev
	Discovered in Release: 23.10-1.1.9.0
	Fixed in Release: 23.10-2.1.3.1
3727822	**Description:** Fixed an issue that allowed concurrent creation of encap entries, and could potentially cause double free vulnerabilities.
	Keywords: encap entries, double free
	Discovered in Release: 23.10-1.1.9.0
	Fixed in Release: 23.10-2.1.3.1
3728381	**Description:** Fixed an issue that exposed debugfs entries for non supported RoCE general parameters, such as rtt_resp_dscp.
	Keywords: debugfs, RoCE
	Discovered in Release: 23.10-1.1.9.0
	Fixed in Release: 23.10-2.1.3.1
3710957	**Description:** Fixed an issue that triggered an error message by updating the rule actions STE apply flow. Following the update, the flow checks if the rule domain is different from the ASO CT action domain when applying the ASO CT action.
	Keywords: Software Steering
	Discovered in Release: 23.10-1.1.9.0
	Fixed in Release: 23.10-2.1.3.1

Internal Reference Number	Description
3663363	**Description:** Fixed an issue where an error was triggered in case devlink reload was attempted when there were allocated subfunctions.
	Keywords: devlink reload, allocated subfunctions
	Discovered in Release: 23.10-0.5.5.0
	Fixed in Release: 23.10-1.1.9.0
3660998	**Description:** Resolved an issue on ConnectX-4 Lx, where the VF state was not configured correctly following the activation of SR-IOV.

Internal Reference Number	Description
	Keywords: ConnectX-4 Lx, VF state
	Discovered in Release: 23.10-0.5.5.0
	Fixed in Release: 23.10-1.1.9.0
3653417	**Description:** Fixed an issue where changing the steering mode to firmware steering was unsupported for policy IPsec rules.
	Keywords: Firmware steering
	Discovered in Release: 23.10-0.5.5.0
	Fixed in Release: 23.10-1.1.9.0

Internal Reference Number	Description
3602955	**Description:** Fixed an issue that occurred when a VF was set to get allmulti traffic. The issue caused the steering rules to send the multicast traffic received by the NIC back to the uplink.
	Keywords: VF, allmulti traffic
	Discovered in Release: 23.07-0.5.0.0
	Fixed in Release: 23.10-0.5.5.0
3553766	**Description:** Fixed an issue where the `enable_remote_dev_reset` Devlink parameter was not supported on kernel versions below v5.10.
	Keywords: Devlink parameter
	Discovered in Release: 23.07-0.5.0.0
	Fixed in Release: 23.10-0.5.5.0
3546694	**Description:** Fixed an issue where MAC address configuration for PFs could fail if SR-IOV was enabled at the same time.
	Keywords: PF, MAC address, SR-IOV
	Discovered in Release: 23.07-0.5.0.0
	Fixed in Release: 23.10-0.5.5.0
3538018	**Description:** Fixed an issue where firmware sync reset (with the '`mlxfwreset -d <device> -l 3 r --sync 1`' command) could fail on a system configured for hotplug on the PCIe slot on which the mlx5 card was mounted.
	Keywords: Firmware sync reset, mlx5 card
	Discovered in Release: 23.07-0.5.0.0
	Fixed in Release: 23.10-0.5.5.0
3587834	**Description:** Fixed an issue where the `enable_remote_dev_reset` Devlink parameter was not supported on kernel versions below v5.10.
	Keywords: Devlink parameter

Internal Reference Number	Description
	Discovered in Release: 23.07-0.5.0.0
	Fixed in Release: 23.10-0.5.5.0
3576351	**Description:** Resolved a warning that was triggered when starting the openibd service, which pertained to an unidentified 'ExecRestart' value within the 'Service' section.
	Keywords: openibd, warning
	Discovered in Release: 23.07-0.5.0.0
	Fixed in Release: 23.10-0.5.5.0
3557482	**Description:** Fixed an issue where the 'mlnx_tune -l' list of supported operating systems did not include several operating systems that were actually supported, such as RHEL8.6 and Ubuntu 22.04.
	Keywords: mlnx_tune -l
	Discovered in Release: 23.07-0.5.0.0
	Fixed in Release: 23.10-0.5.5.0
3549684	**Description:** Fixed a signature-related issue that occurred when installing DOCA on SLES15SP4 using the repository.
	Keywords: DOCA, SLES15SP4
	Discovered in Release: 23.07-0.5.0.0
	Fixed in Release: 23.10-0.5.5.0
3380263	**Description:** Fixed an issue where users who attempted to use OFED with Device ID NVD0000000033, had to install the firmware manually.
	Keywords: Device ID NVD0000000033
	Discovered in Release: 23.07-0.5.0.0
	Fixed in Release: 23.10-0.5.5.0
3228788	**Description:** Fixed an issue where running rx-tls-offload over Korg6.0 as its TLS module did not work properly.
	Keywords: NetDev, TLS
	Discovered in Release: 23.07-0.5.0.0
	Fixed in Release: 23.10-0.5.5.0

Internal Reference Number	Description
3546304	**Description:** Resolved the kernel crash resulting from sysfs calls to profiles lacking TC (Traffic Control) support.
	Keywords: sysfs calls, Trafic Control
	Discovered in Release: 23.04-0.5.3.3
	Fixed in Release: 23.07-0.5.0.0
3531986	**Description:** Fixed an issue that prevented OS booting following an installation of the EN and RoCE drivers.

Internal Reference Number	Description
	Keywords: OS booting, EN, RoCE
	Discovered in Release: 23.04-0.5.3.3
	Fixed in Release: 23.07-0.5.0.0
3489233	**Description:** Fixed an issue in SLES 15 SP4 where the openibd service failed to start automatically after system boot.
	Keywords: SLES 15 SP4,openibd, system boot
	Discovered in Release: 23.04-0.5.3.3
	Fixed in Release: 23.07-0.5.0.0
3431430	**Description:** Fixed an issue that prevented the installation of OFED on RHEL systems using a non-default Python version.
	Keywords: Installation, RHEL, Python
	Discovered in Release: 5.9-0.5.6
	Fixed in Release: 23.07-0.5.0.0
3422823	**Description:** Fixed an OFED installation issue on BCLinux 21.10 that occurred when using the "--add-kernel-support" installation flag.
	Keywords: Installation, BCLinux 21.10, "--add-kernel-support"
	Discovered in Release: 5.9-0.5.6
	Fixed in Release: 23.07-0.5.0.0
3264588	**Description:** Resolved a problem where the system boot process would hang when more than two Network Interface Cards were installed.
	Keywords: System boot, Network Interface Cards
	Discovered in Release: 5.7-1.0.2.0
	Fixed in Release: 23.07-0.5.0.0
3499136	**Description:** Fixed an issue where the sysfs PHY counters displayed outdated information.
	Keywords: sysfs PHY counters
	Discovered in Release: 23.04
	Fixed in Release: 23.07-0.5.0.0

Internal Reference Number	Description
2883451	**Description:** Installing mlnx_tune on Python3 did not work properly. mlnx_tune now supports Python3 in addition to Python2.
	Keywords: Installation, mlnx_tune, Python3
	Discovered in Release: 5.5-1.0.3.2
	Fixed in Release: 23.04-0.5.3.3

Internal Reference Number	Description
3219842	**Description:** When creating a bond interface for all ports on a ConnectX-7 4-port HCA, the wrong bond name appeared in ibdev2netdev.
	Keywords: RDMA, Bond Name, ibdev2netdev, ConnectX-7
	Discovered in Release: 5.8-1.0.1.1
	Fixed in Release: 23.04-0.5.3.3
3333919	**Description:** Changing traffic class via the sysfs while modifying QPs in parallel causes a deadlock.
	Keywords: RDMA, TC, Sysfs, QP
	Discovered in Release: 5.0-2.1.8.0
	Fixed in Release: 23.04-0.5.3.3
3406019	**Description:** Due to a bug in the emulation layer, performance degradation might be experienced when running GPUDirect over Virtual Functions.
	Keywords: RDMA, GPUDirect, performance, VF
	Discovered in Release: 5.9-0.5.6.0
	Fixed in Release: 23.04-0.5.3.3
3233799	**Description:** debugfs directories cannot be created for representors and sub-functions, thus the log might show error warning for either of the scenarios.
	Keywords: NetDev, debugfs, SF, logging
	Discovered in Release: 5.8-1.0.1.1
	Fixed in Release: 23.04-0.5.3.3
1892663/1800633/2883451	**Description:** mlnx_tune script does not support Python3 interpreter.
	Keywords: mlnx_tune, Python3
	Discovered in Release: 4.7-1.0.0.1
	Fixed in Release: 23.04-0.5.3.3
3340542	**Description:** The verion number for perftest was not-standard resulting in some distribution packages recieving a higher version number than the OFED version for no good reason. Changed the naming of perftest to MAJOR.MINOR.PATCH.
	Keywords: Installation, perftest
	Fixed in Release: 23.04-0.5.3.3
3428775	**Description:** knem did not fully support RHEL8.7 and newer releases.
	Keywords: Installation, knem, RHEL
	Discovered in Release: 5.8-1.0.1.1
	Fixed in Release: 23.04-0.5.3.3
3431430	**Description:** Installing MLNX_OFED on a RHEL system that uses a non-default version of Python (e.g., Python3.9 on RHEL8.6, where the default is 3.6) may fail with an error that mlnx-tools is missing a dependency on 'python(abi)'. mlnx-tools includes a single script, mlnx_qos, that depends on a specific version of python. In such a case, after the fix, it may fail to run with such a non-standard version of Python.
	Keywords: Installation, Python, RHEL, mlnx-tools

Internal Reference Number	Description
	Discovered in Release: 5.9-0.5.6.0
	Fixed in Release: 23.04-0.5.3.3

5.2 User Manual Revision History

Release	Date	Description
5.7	August 2022	• Added Out of Order (OOO) under RoCE section
5.3	April 15, 2021	• Added PTP Cyc2time Hardware Translation Offload section • Updated Persistent Naming section
5.2	January 12, 2021	• Added Offloaded Traffic Sniffer section. • Added Tx Port Time-Stamping section. • Added VLAN Push/Pop section. • Added sFLOW section. • Added E2E Cache section. • Added Geneve Encapsulation/Decapsulation section. • Added Parallel Offloads section. • Updated SR-IOV VF LAG section. • Removed Installing MLNX_EN on Innova™ IPsec Adapter Cards section. • Removed Updating Firmware and FPGA Image on Innova IPsec Cards section.
5.1	August 16, 2020	Updated the content of the entire document following the removal of support for ConnectX-3, ConnectX-3 Pro and Connect-IB adapter cards, as well as the deprecation of RDMA experimental verbs library (mlnx_lib).
		Added Interrupt Request (IRQ) Naming section.
		Added Kernel Transport Layer Security (kTLS) Offloads section.
5.0	March 15, 2020	• Added IPsec Crypto Offload section. • Updated .Installing MLNX_EN v4.5-1.0.1.0 section.
4.7	December 29, 2019	• Added section .Mediated Devices v4.7-3.2.9.0. • Added "num_of_groups" entry to table mlx5_core Module Parameters. • Added Performance Tuning Based on Traffic Patterns section.
4.5	December 19, 2018	• Reorganized Chapter 2, “Installation”: Consolidated the separate installation procedures under Installing MLNX_EN and Additional Installation Procedures

6 Legal Notices and 3rd Party Licenses

The following are the drivers' software, tools and HCA firmware legal notices and 3rd party licenses.

Product	Version	Legal Notices and 3rd Party Licenses
MLNX_OFED	23.10-6	• License • 3rd Part Notice
Firmware	xx.39.51xx	• HCA Firmware EULA • 3rd Party Unify Notice • License
MFT	4.26.1-35	• License • 3rd Party Notice • 3rd Party Unify Notice
Clusterkit	1.11	• License • 3rd Party Notice
DPCP	1.1.43	• License • 3rd Party Notice
VMA	9.8.40	• 3rd Party Unify Notice • 3rd Party Notice
XLIO	3.20.8	• License • 3rd Party Unify Notice
HCOLL	4.8	• License • 3rd Party Notice
SHARP	3.5.2	• License • 3rd Party Notice
ibutils2	2.15	• License • 3rd Party Notice
OpenSM	5.17.2	• 3rd Party Unify Notice • 3rd Party Notice
mpitests	3.2.21	• License • 3rd Party Notice

Notice

This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. Neither NVIDIA Corporation nor any of its direct or indirect subsidiaries and affiliates (collectively: “NVIDIA”) make any representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assumes no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice. Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer (“Terms of Sale”). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document.

NVIDIA products are not designed, authorized, or warranted to be suitable for use in medical, military, aircraft, space, or life support equipment, nor in applications where failure or malfunction of the NVIDIA product can reasonably be expected to result in personal injury, death, or property or environmental damage. NVIDIA accepts no liability for inclusion and/or use of NVIDIA products in such equipment or applications and therefore such inclusion and/or use is at customer’s own risk.

NVIDIA makes no representation or warranty that products based on this document will be suitable for any specified use. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer’s sole responsibility to evaluate and determine the applicability of any information contained in this document, ensure the product is suitable and fit for the application planned by customer, and perform the necessary testing for the application in order to avoid a default of the application or the product. Weaknesses in customer’s product designs may affect the quality and reliability of the NVIDIA product and may result in additional or different conditions and/or requirements beyond those contained in this document. NVIDIA accepts no liability related to any default, damage, costs, or problem which may be based on or attributable to: (i) the use of the NVIDIA product in any manner that is contrary to this document or (ii) customer product designs.

No license, either expressed or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this document. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA.

Reproduction of information in this document is permissible only if approved in advance by NVIDIA in writing, reproduced without alteration and in full compliance with all applicable export laws and regulations, and accompanied by all associated conditions, limitations, and notices.

THIS DOCUMENT AND ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, “MATERIALS”) ARE BEING PROVIDED “AS IS.” NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL NVIDIA BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA’s aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms of Sale for the product.

Trademarks

NVIDIA