

Agilio CoreNIC User Manual

V24.07

Contents:

1	Introduction	1
1.1	Revision History	1
1.2	About this Guide.....	2
1.3	Audience.....	2
1.4	Contact Us	2
2	Safety standard	3
2.1	Safety	3
2.2	Standards and Regulations.....	4
2.2.1	Environmental Compliance.....	4
2.2.2	Regulatory Compliance	4
3	Product Selection	5
3.1	Supported Products.....	5
3.2	GX 2x10G	5
3.3	GX 2x25G	6
3.4	GX 2x40G	7
3.5	GX 4x10G	8
4	The Agilio SmartNIC Architecture	10
5	Basic Firmware Features	11
5.1	Summary of Features	11
5.2	VF TX Rate Limit.....	16
5.2.1	Ingress and Egress	16
5.2.2	Configuring VF Rate Limit.....	17
5.2.3	VF TX Rate Limit Validation	18
5.2.4	Expected results.....	18
5.3	Setting Interface Settings.....	21
5.4	Multiple Queues	21
5.4.1	View Current Settings.....	21
5.4.2	Configure Queues	21
5.5	Receive Side Scaling (RSS)	22
5.5.1	View Current Hash Parameters	22
5.5.2	Set Hash Parameters	22
5.5.3	Configuring the Key.....	22
5.6	View Interface Parameters.....	22
5.6.1	Receive Checksumming (rx-checksumming).....	24
5.6.2	Transmit Checksumming (tx-checksumming).....	24
5.6.3	Scatter and Gather (scatter-gather)	24
5.6.4	TCP Segmentation Offload (TSO).....	25
5.6.5	Generic Segmentation Offload (GSO).....	25

5.6.6	Generic Receive Offload (GRO)	25
5.7	Interrupt Coalescing	26
5.7.1	View Current Coalescing Parameters	26
5.7.2	Configure Coalescing	26
5.8	Flow Steering	27
6	Hardware Installation	29
6.1	Physical Installation	29
6.2	Identification	29
6.3	Validation	30
6.4	SmartNIC Netdev Interfaces	30
6.4.1	Support for <code>Biosdevname</code>	32
6.5	Accessory	32
7	Linux Driver	33
7.1	Preparing for Installation	33
7.1.1	Checking the kernel version	33
7.1.2	Checking the NPF Driver	34
7.2	Installing the Linux Driver	34
7.2.1	Install Driver via Corigine Repository	34
7.2.2	Install Driver via Software Package	35
7.2.3	Building from Source	37
7.2.4	(Optional) Kernel Changes	38
7.2.5	Confirm that the NFP Driver is Loaded	39
7.3	Using the Linux Driver	39
7.3.1	Configuring Interface Media Mode	39
7.3.2	Configuring interface Maximum Transmission Unit (MTU)	40
7.3.3	Configuring FEC modes	41
7.3.4	Setting Interface Breakout Mode	43
7.3.5	Configuring interface private-flags	45
7.3.6	Confirming Connectivity	46
8	VMware Driver	47
8.1	Driver Download	47
8.2	Driver Installation	49
8.3	Using the VMware Driver	50
8.3.1	Configuring Link-speed	50
8.3.2	Confirming Connectivity	50
9	Windows Driver	52
9.1	Driver Download	52
9.2	Driver Installation	52
9.3	Driver Uninstallation	55
9.4	Using the Windows Driver	56
9.4.1	Configuring Link-speed	56
9.4.2	Configuring FEC modes	56
9.4.3	Configuring Interface MTU	57
9.4.4	Confirming Connectivity	57

10 Firmware Installation	58
10.1 Validating the Firmware	58
10.2 Upgrading the firmware	60
10.2.1 Upgrading firmware via the Corigine repository	60
10.2.2 Upgrading Firmware from Package Installations	60
11 BSP Installation	62
11.1 Install Software from Corigine Repository	62
11.2 Install Software from DEB/RPM Package	62
11.2.1 Obtain Software	62
11.2.2 Install the Prerequisite Dependencies	63
11.2.3 NFP BSP Package	63
11.3 Using BSP Tools	63
11.3.1 Enable CPP Access	63
11.3.2 Configure Media Settings	64
11.4 Upgrade Flash Firmware	65
12 Basic Performance Test	66
12.1 Install iPerf	66
12.2 Run iPerf Test	66
12.2.1 Server	66
12.2.2 Client	66
12.3 Using iPerf3	67
13 Installing, Configuring and Using DPDK	69
13.1 Introduction to DPDK	69
13.2 Enabling IOMMU	69
13.2.1 Edit Grub Configuration File	70
13.2.2 Implement Changes	70
13.3 DPDK Sources with PF PMD Support	70
13.3.1 Single-PF PMD Multi-port Support	70
13.3.2 Multi-PF PMD Multi-port Support	71
13.4 Installing DPDK	71
13.5 Binding DPDK PF Driver	72
13.5.1 Attaching VFIO-PCI Driver	72
13.5.2 Confirm Attached Driver	73
13.5.3 Unbind Driver	73
13.6 Using DPDK PF Driver	73
13.6.1 Create Default Symlink	73
14 Using SR-IOV	75
14.1 Installing the SR-IOV Capable Firmware	75
14.1.1 The Linux-Firmware Package	75
14.1.2 The Support Site	77
14.2 Load Firmware to SmartNIC	78
14.3 Configuring SR-IOV	78
14.4 Using Virtio-Forwarder	81
14.4.1 Installing Virtio-Forwarder	82

14.4.2	Configuring Hugepages	82
14.4.3	Binding to VFIO-PCI	84
14.4.4	Launching Virtio-Forwarder	86
14.4.5	Adding VF Ports to Virtio-Forwarder	86
14.4.6	Modify Guest VM XML Files	88
15	Using RoCEv2	89
15.1	Introduction to RDMA/RoCEv2	89
15.2	Installing	90
15.2.1	Requirements	90
15.2.2	Install Userspace Lib via Software Package	91
15.2.3	Upgrading Firmware from Package Installations	91
15.2.4	Install Kernel Driver via Software Package	91
15.2.5	Confirm BSP Version	93
15.3	Using and Basic Testing	94
15.3.1	Install Perftest	94
15.3.2	Run Test	94
16	Using IPsec	97
16.1	Introduction to IPsec/IPsec VPN	97
16.2	IPsec Process	97
16.3	IPsec Offloading	98
16.4	IPsec Features	98
16.5	Requirements	99
16.6	IPsec Driver and Firmware Installation	99
16.7	DPDK Installation and Configuration for IPsec	100
16.8	Kernel-based IPsec Offloading	100
16.8.1	Kernel Version	100
16.8.2	Environment Setup	100
16.8.3	Configuration Instructions	100
16.8.4	Basic Testing	100
16.9	DPDK-based IPsec Offloading	101
16.9.1	DPDK Version	101
16.9.2	Environment Deployment	102
16.9.3	Configuration Instructions	103
16.9.4	Basic Testing	103
17	Upgrading the Kernel	105
17.1	RHEL	105
17.2	CentOS	105
17.3	Ubuntu	105
17.3.1	Acquire Packages	106
17.3.2	Install Packages	106
18	UEFI Secure Boot with Out-of-Tree NFP Driver	107
18.1	NFP Driver Module is Signed with a DKMS Module Signing key	107
18.1.1	RHEL and CentOS	108
18.1.2	Ubuntu	108

18.2 NFP Driver Module is Not Signed or Signed with Unknown Keys	108
19 Abbreviations and Terms	110

1 Introduction

1.1 Revision History

Revision	Date	Description
V22.04	29 Apr 2022	Corigine initial public release.
V22.07	30 Jul 2022	Corigine second public release.
V22.10	31 Oct 2022	Corigine third public release.
V23.01	20 Jan 2023	Add 6 features Add 7.5 Configuring interface private-flags Add Appendix F: UEFI Secure Boot with Out-of-Tree NFP Driver
V23.04	7 Apr 2023	Add 5 features
V23.07	15 Jun 2023	Add 6 features
V23.10	30 Oct 2023	Add 8 features Add 9.8 Flow Steering Add 12 Using RoCEv2 Add Appendix F: Installing the Windows Driver
V24.01	22 Mar 2024	Add 4 features Add 13 Using IPsec
V24.04	11 May 2024	Add 3 features Add 3 Product Selection Add 6.5 Accessory Add 8.3 Using the VMware Driver Add 9.4 Using the Windows Driver
V24.07	30 Aug 2024	Add 4 features

1.2 About this Guide

This is the User Guide for Agilio CoreNIC Firmware and support provided by Corigine to its customers. The reader can find more elaborated information about the different topics in the links and references provided throughout the document. Bash scripts are indicated with a light blue background.

1.3 Audience

This document is intended for the installer and user of the SmartNIC.

1.4 Contact Us

Corigine Systems, Inc. 2F West, Building 1 No. 1516 Hongfeng Road Wuxing Dist., Huzhou Zhejiang, 313000	
400-615-0098	
https://www.corigine.com/	smartnic-support@corigine.com

2 Safety standard

2.1 Safety

This section contains **Warnings!** and **Cautions!** Warnings are safety related. Failure to follow warnings may lead to injury or equipment damage. Cautions are requirements for proper function. Failure to follow cautions may result in improper operation. All products are low voltage PCIe cards (12V-, 3.3V-supplied per PCIe standard). All lasers in optional transceiver plug-ins are Class 1 or Class 1M. Avoid long-term viewing of laser.

Warning: No user serviceable parts are present.

Warning: Replacements must be performed by qualified personnel only. All installation instructions and requirements specified for the end-use system must be followed.

Caution: None of the units in this document are hot-swappable. Damage will result. Please disconnect all system power feeds before attempting to install or replace any of these products in a system.

Caution: These products may be vulnerable to static electricity. Electro-Static Discharge (ESD) mitigation controls (e.g. static straps) must be used while handling and installing these products. These products should be stored in antistatic bags or containers when not in use.

2.2 Standards and Regulations

The Agilio SmartNICs adhere to the following regulations.

2.2.1 Environmental Compliance

- European Union RoHS II Directive: 2011/65/EU
- European Union REACH Directive: 2006/121/EC
- Administrative Measure on the Control of Pollution Caused by Electronic Information Products (“China ROHS”)
- Congo Conflict Minerals Act of 2009 (Section 1502 of Dodd-Frank Wall Street Reform and Consumer Protection Act including SEC ruling 17 CFR PARTS 240 and 249b)

2.2.2 Regulatory Compliance

- CFR 47 FCC Part 15 Subpart B Class A emissions requirements (USA)
- European Union EMC Directive: 2004/108/EC
- ICES-0003 Issue 4 Class A Digital Apparatus emissions requirements (Canada)
- EN 55022:2010/AC:2011 Class A ITE emissions requirements (EU / CE Mark)
- EN 55024:2010 ITE - immunity characteristics (EU / CE Mark)
- EN 61000-4-2
- EN 61000-4-3
- EN 61000-4-4
- EN 61000-4-6
- EN 61000-4-8
- Kylin Software NeoCertify Certification

3 Product Selection

3.1 Supported Products

An Agilio SmartNIC product can support different speed types. The following table shows Agilio SmartNIC products that are currently supported and their different supported port speeds.

Supported Agilio product	Supported port speeds
CX 2x25G	2x10G 2x25G 1x10G + 1x25G
CX 2x40G	4x10G + 4x10G 2x40G 4x10G + 1x10G
GX 2x10G	2x1G 2x10G 1x1G + 1x10G
GX 2x25G	2x10G 2x25G 1x10G + 1x25G
GX 4x10G	Each port supports 10G/1G

3.2 GX 2x10G

Specification Parameters	Description
Interfaces	2-port SFP+ (10GbE), backwards compatible to SFP (1GbE)
Port Speeds	1/10G speed auto-negotiation
Dimension	PCIe half-height, half-length, single width (full-height bracket included) 68.9mm (H) x 167.65mm (L) x 18.71mm (W)
Weight	125g (not including optical modules)

continues on next page

continued from previous page

Specification Parameters	Description
PCIe Base	PCIe gen 3.0 x8, compatible with gen 1.1 and 2.0, MSI/MSI-X compatible 2.5, 5.0, 8.0GT/s link rate x8/x4/x2/x1 lanes
LED Indicators	LINK (Green solid), ACTIVITY (Green blinking) LINK SPEED (Green=10GbE, Yellow=1GbE)
Network Flow Processor	Corigine, Inc. Network Flow Processor 3800
IEEE Standards	IEEE 802.3z 1 Gigabit Ethernet IEEE 802.3ae 10 Gigabit Ethernet IEEE 802.3ap based auto-negotiation and KR startup IEEE 802.3ab (LLDP) IEEE 802.3ad, 802.1AX Link aggregation IEEE 802.1q, 802.1p VLAN tags and priority IEEE 802.1au (QCN) IEEE 802.1Qbb (PFC)

Power Consumption	Description
Typical Power	9.5 W
Maximum Power	13.5 W
Optical Module Power (for reference)	1.5 W (two SFP+ SR)

3.3 GX 2x25G

Specification Parameters	Description
Interfaces	2-port SFP28 (25GbE), backwards compatible to SFP+ (10GbE)
Port Speeds	10/25G speed auto-negotiation
Dimension	PCIe half-height, half-length, single width (full-height bracket included) 68.9mm (H) x 167.65mm (L) x 18.71mm (W)
Weight	134g (not including optical modules)

continues on next page

continued from previous page

Specification Parameters	Description
PCIe Base	PCIe gen 3.0 x8, compatible with gen 1.1 and 2.0, MSI/MSI-X compatible 2.5, 5.0, 8.0GT/s link rate x8/x4/x2/x1 lanes
LED Indicators	LINK (Green solid), ACTIVITY (Green blinking) LINK SPEED (Green=25GbE, Yellow=10GbE)
Network Flow Processor	Corigine, Inc. Network Flow Processor 3800
IEEE Standards	IEEE 802.3ae 10 Gigabit Ethernet IEEE 802.3by 25 Gigabit Ethernet IEEE 802.3ap based auto-negotiation and KR startup IEEE 802.3ab (LLDP) IEEE 802.3ad, 802.1AX Link aggregation IEEE 802.1q, 802.1p VLAN tags and priority IEEE 802.1au (QCN) IEEE 802.1Qbb (PFC)

Power Consumption	Description
Typical Power	13.5 W
Maximum Power	17.5 W
Optical Module Power (for reference)	2.5 W (two SFP28 SR)

3.4 GX 2x40G

Specification Parameters	Description
Interfaces	2-port QSFP (40GbE)
Dimension	PCIe half-height, half-length, single width (full-height bracket included) 68.9mm (H) x 167.65mm (L) x 18.71mm (W)
Weight	133g (not including optical modules)
PCIe Base	PCIe gen 3.0 x8, compatible with gen 1.1 and 2.0, MSI/MSI-X compatible 2.5, 5.0, 8.0GT/s link rate x8/x4/x2/x1 lanes

continues on next page

continued from previous page

Specification Parameters	Description
LED Indicators	LINK (Green solid), ACTIVITY (Green blinking)
Network Flow Processor	Corigine, Inc. Network Flow Processor 3800
IEEE Standards	IEEE 802.3ae 10 Gigabit Ethernet IEEE 802.3ba 40 Gigabit Ethernet IEEE 802.3ap based auto-negotiation and KR startup IEEE 802.3ab (LLDP) IEEE 802.3ad, 802.1AX Link aggregation IEEE 802.1q, 802.1p VLAN tags and priority IEEE 802.1au (QCN) IEEE 802.1Qbb (PFC)

Power Consumption	Description
Typical Power	15 W
Maximum Power	20 W

3.5 GX 4x10G

Specification Parameters	Description
Interfaces	4-port SFP+ (10GbE), backwards compatible to SFP (1GbE)
Port Speeds	1/10G speed auto-negotiation
Dimension	PCIe half-height, half-length, single width (full-height bracket included) 68.9mm (H) x 167.65mm (L) x 18.71mm (W)
Weight	140g (not including optical modules)
PCIe Base	PCIe gen 3.0 x8, compatible with gen 1.1 and 2.0, MSI/MSI-X compatible 2.5, 5.0, 8.0GT/s link rate x8/x4/x2/x1 lanes
LED Indicators	LINK (Green solid), ACTIVITY (Green blinking) LINK SPEED (Green=10GbE, Yellow=1GbE)
Network Flow Processor	Corigine, Inc. Network Flow Processor 3800

continues on next page

continued from previous page

Specification Parameters	Description
IEEE Standards	IEEE 802.3z 1 Gigabit Ethernet IEEE 802.3ae 10 Gigabit Ethernet IEEE 802.3ap based auto-negotiation and KR startup IEEE 802.3ab (LLDP) IEEE 802.3ad, 802.1AX Link aggregation IEEE 802.1q, 802.1p VLAN tags and priority IEEE 802.1au (QCN) IEEE 802.1Qbb (PFC)

Power Consumption	Description
Typical Power	11 W
Maximum Power	15 W
Optical Module Power (for reference)	3 W (four SFP+ SR)

4 The Agilio SmartNIC Architecture

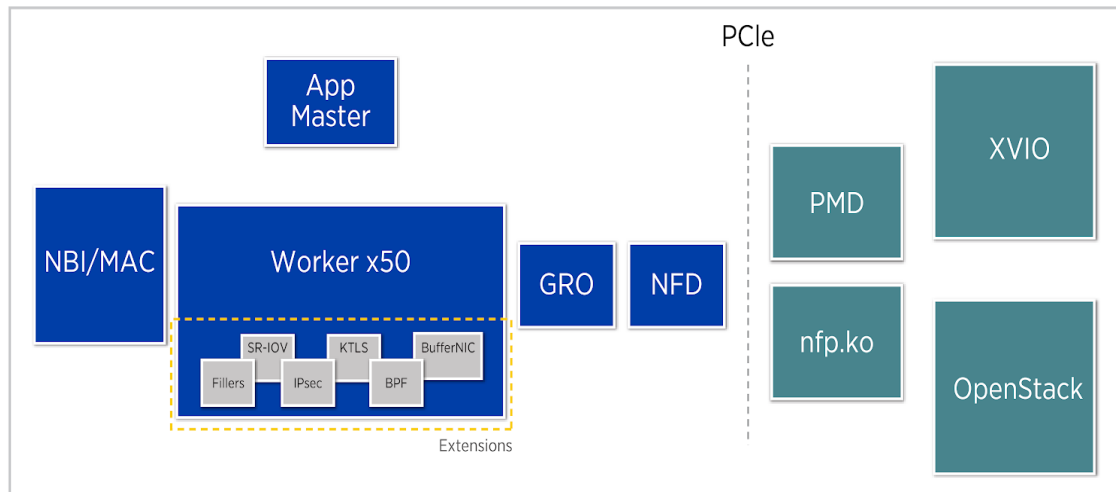


Fig. 1: The conceptual architecture of the Agilio SmartNIC

The Agilio Series SmartNICs are based on the NFP-4000 and NFP-3800 and are available in low profile PCIe and OCM v2 NIC form factors suitable for use in Commercial Off-The-Shelf (COTS) servers. These are 60 and 36 core processors respectively, with eight cooperatively multithreaded threads per core. The flow processing cores have an instruction set that is optimized for networking. This ensures an unrivalled level of flexibility within the data plane while maintaining performance. The Open vSwitch (OVS) datapath can also be enabled without a server reboot.

Further extensions such as Berkeley Packet Filter (BPF) offload, Single Root I/O Virtualization (SR-IOV) or custom offloads can be added without any hardware modifications or a server reboot. These extensions are not covered by this guide, which deals with the basic firmware only.

The basic firmware offers a wide variety of features including Receive Side Scaling (RSS), Checksum Offload (IPv4/IPv6, TCP, UDP, Tx/Rx), Large Segmentation Offload (LSO), IEEE 802.3ad, Link flow control, 802.1AX Link Aggregation, etc. For more details regarding currently supported features refer to the section [Basic Firmware Features](#).

5 Basic Firmware Features

In this section `ethtool` will be used to view and configure SmartNIC interface parameters.

5.1 Summary of Features

The following table summarizes the features of the Agilio SmartNICs.

Feature	Description
FW version	Getting device hardware firmware info Note: <code>nfp-hwinfo</code> and <code>query</code> supported
UEFI PXE boot	Support UEFI BIOS PXE booting
Legacy PXE boot	Support Legacy BIOS PXE booting
Firmware flashing	Support for BSP firmware updating
Extended stats	Support extended <code>ethtool</code> statistics reporting
TSO (LSO)	TCP Large Segment Offload enablement
SR-IOV	Single-Root Virtual Functions and virtual ethernet bridge
SR-IOV MAC VLAN	VLAN offload from SRIOV MACs. Supports transmitting packets according to the Virtual Ethernet Bridge (VEB) lookup result using MAC+VLAN
RSS: Receive Side Scaling	Core/Interrupt/Queue packet routing
TCP/UDP checksum	On both Rx and Tx Note: IP/UDP/TCP - Driver offloads checksum calculation. When TSO slice, the IP checksum will be re-calculated by the SmartNIC.
Jumbo frame support	MTU setting Note: 9532 bytes is largest available MTU
eBPF offload	eBPF rules on NFP
Checksum support	Inner L3 checksum Outer L3 checksum Inner L4 checksum
NVGRE	Microsoft, included tenant network id over GRE

continues on next page

continued from previous page

Feature	Description
Adaptive RX/TX	Change interrupt rates under load (IRQ handling). This is useful for optimizing for latency or throughput
CX 2x25GbE v2 10G+25G	Allow 10G+25G in any port-combination on 2-port cards (only on NFP3800 chips) Note: Only one port-combination is possible on NFP4000 chips (P0 - 10G, P1 - 25G)
VXLAN Support for TSO	Full Encap/Decap of fragments
DPDK driver	DPDK driver support for VXLAN tunnel inner TSO
DPDK VXLAN tunnel RSS	DPDK VXLAN driver support for tunnel RSS
Distinguish SVLAN and CVLAN VIDs (TPID)	Support different VLAN protocols (TPID) using different VLAN IDs. For example with TPID value 0x8100 vs 0x88a8.
BMC support	Out-of-band management using a SMBUS/I2C interface. Currently support gets the information from FRU/CPLD.
VF TX rate limit (QoS)	Support setting the VF rate limit with command <code>ip link set <dev> vf <num> max_tx_rate <rate></code>
VLAN transparent mode	The interface can be set to work in VLAN transparent mode, under which, the VLAN tag in the packages that passes through the interface, is not changed. For packets moving specifically out of the VM, if there is no tag in the header, a default tag can be added.
Support command <code>ethtool -p</code>	Support the identifying of the NIC port with the command <code>ethtool -p: ethtool [FLAGS] -p -identify DEVNAME</code>
Support command <code>ethtool -a</code>	Queries the specified Ethernet device for pause parameter information
Packet type offload for DPDK	Packet filtering based on various metrics e.g. packet size, protocol
Support for <code>Auto</code> FEC mode	FEC mode automatically chosen based on speed and SFP type
Support for AMDA3000 SmartNICs	Support for variant of GX 2x10G SmartNIC
UEFI HII Display NIC Info	SmartNIC information now listed in UEFI HII menu (vendor, MAC address, product information)
IEEE 802.1Q VLAN tags	Hardware tag insertion and deletion
VEPA	SRIOV supports Virtual Port Aggregator Mode (VEPA) mode

continues on next page

continued from previous page

Feature	Description
Support command <code>ethtool -t</code>	Executes adapter self-test on the specified network device
Uniform PXE firmware	One PXE firmware supports all applicable CPUs (X86, ARM, ...)
IPv6 PXE (UEFI mode)	Pre-boot Execution Environment via IPv6 protocol
Auto-negotiation on 25Gb SmartNICs	Support 25Gb/10Gb speed and FEC mode auto-negotiation on 25Gb SmartNICs
Auto-negotiation on 10Gb SmartNICs	Support 10Gb/1Gb speed auto-negotiation on 10Gb SmartNICs
1Gb mode on 10Gb SmartNICs	10Gb product supports 1Gb speed mode
SRIOV VF multi-queues	Support more than 16 queues for one VF, 64 queues in total Note: The feature is only on the GX cards
Support VF split	Single PF cards support assignment of VFs to different physical ports
Support command <code>ethtool -r</code>	Restarts auto-negotiation on the specified Ethernet device, if auto-negotiation is enabled
Support command <code>ethtool -e</code> / <code>ethtool -E</code>	Support dump/update EEPROM content with ethtool command
New product update	Update new product PCIE vendor ID to Corigine 0X1DA8
BSP packages for ARM	nfp-bsp_aarch64.rpm/deb, equivalents of nfp-bsp_x86_64.rpm and nfp-bsp_amd64.deb
LLDP	Send out LLDP packet without OS software on host
Ethtool display "link mode"	Display "Supported link modes" and "Advertised link modes" by command <code>ethtool <netdev></code>
Multicast MAC filter	Support passing through multicast packets with configured mac address (with command <code>ip maddress add <mac> dev <netdev></code>) while dropping packets with other mac addresses, in default mode. Note: Configuration is asynchronous, so a large list of mac addresses might become fully active a few seconds after the last <code>ip maddress add</code> command completes. Support passing through all multicast packets in ALLMULTI mode with command <code>ip link set allmulticast on <netdev></code> .
LLDP Support for CX Card	Add LLDP feature for CX series card

continues on next page

continued from previous page

Feature	Description
Data Center Bridging (DCB)	Converged Ethernet
IEEE 802.1P VLAN priorities	QoS for VLANs
DPDK Crypto Library	Crypto offload support for DPDK
Support New Type of NIC Products	Agilio GX 4x10G SmartNIC: AMDA2002-1113 AMDA2002-1114 AMDA2002-1133 AMDA2002-1134 AMDA2005-1001 Agilio GX 2x25G SmartNIC: AMDA2000-1103 AMDA2000-1113 AMDA2000-1104 AMDA2000-1114 Agilio GX 2x10G SmartNIC: AMDA2001-1103 AMDA2001-1113 AMDA2001-1104 AMDA2001-1114 AMDA2001-1123 AMDA2001-1133 AMDA2001-1124 AMDA2001-1134 AMDA2001-1105 AMDA2001-1106
Extend PF number	Two PFs for two phy ports of GX card
Interface status: config down vs link down	While phy port is configured down, it should have light signal and VF should work.
Fix the ports' order of GX card	The order of the two ports on 2x10G/2x25G need to be changed.
Queue extending	Extend hardware queue from 64 to 128
IRQ affinity script - allow thread pool specification	Optimize the scripition and intergrade into driver package
Trouble shoot tools update	Add more collection information of nfp-bsp tools; intergrade into driver package
Support ethtool -A tx on/off and rx on/off	Set the specified Ethernet device for pause parameter

continues on next page

continued from previous page

Feature	Description
Support Windows Driver	Pass Microsoft certification and publish windows driver package
Support RoCEv2	Support RDMA/RoCEv2 offload, support user space and kernel space RDMA verbs.
Support DCQCN	Support DCQCN flow control
Support Flow Steering	Support using DPDK RTE_FLOW or ethtool command to configure flows to steer input traffic.
MCTP for BMC information	Out-of-band management support standard MCTP to communicate FRU information (basic hardware info, including temp, power, SN, status, negotiation mode, MAC, speed, etc.) with BMC
Add new P/N encoding in VPD	Add new P/N number data in VPD, and use it for checking and validating the firmware type which decides the function set of NIC.
Support VF promiscuous mode	Run the <code>ip link set <netdev> promisc on/off</code> command to set promiscuous mode. When promiscuous mode is enabled, the VF can receive packets that reach the physical port, including: DMAC unmatched unicast/multicast packets, broadcast packets, and DMAC matched unicast packets. Can only enable promiscuous mode for trusted VF.
PFC feature	Priority-based flow control Support <code>dcb pfc set dev <netdev> prio-pfc all:on</code> to enable and <code>dcb pfc show dev <netdev></code> to view info
Gain PN from in-band command	Run <code>devlink dev info</code> to display the required PN
Support IPSec	Support IPSec protocols encapsulation/decapsulation offload and crypto algorithms offload Up to 16K simultaneous SAs
Configure LLDP	Configuring and reading LLDP state per network interface via BMC
NCSI over RMII	Support BMC manage the NIC with NCSI over RMII Support BMC configuration the NCSI feature Support passthrough the legal management traffic
Load firmware from master/slave flash partition	Load firmware from flash, independent from the host OS Master/Slave partition for firmware on FLASH

continues on next page

Feature	Description
Support <code>ethtool -C <netdev> adaptive-rx on/off adaptive-tx on/off</code>	Adaptive RX/TX coalescing is an algorithm implemented by some drivers to improve latency under low packet rates and improve throughput under high packet rates
Support IPv6 for RoCEv2	Support IPv6 protocol
Support SRQ	Allow multiple queue pairs to share receive work requests and the data segments associated with receive work requests
Support loopback message for same RDMA device	Support communication between QPs which belong to same RDMA device

5.2 VF TX Rate Limit

The VF rate limit feature is designed to limit the amount of bandwidth that a specific virtual machine can get from a pipeline.

5.2.1 Ingress and Egress

Quality of Service (QoS) policies are applied to ingress or egress, however, the definition of these terms may be reversed depending on the perspective taken. Ingress and egress can be defined as either of the following descriptions:

1. Host or VM Perspective: This is the perspective taken by OpenStack and is the perspective used in this document.
 - a. Ingress: Packets received by a host or VM.
 - b. Egress: Packets sent from a host or VM.
2. SmartNIC Perspective: This perspective is the exact opposite of the host or VM perspective. This is the perspective taken by OVS and is mentioned here for completeness.
 - a. Ingress: Packets received by the NIC via a physical port, or PCIe PF or VF.
 - b. Egress: Packets transmitted by the NIC via a physical port, or PCIe PF or VF.

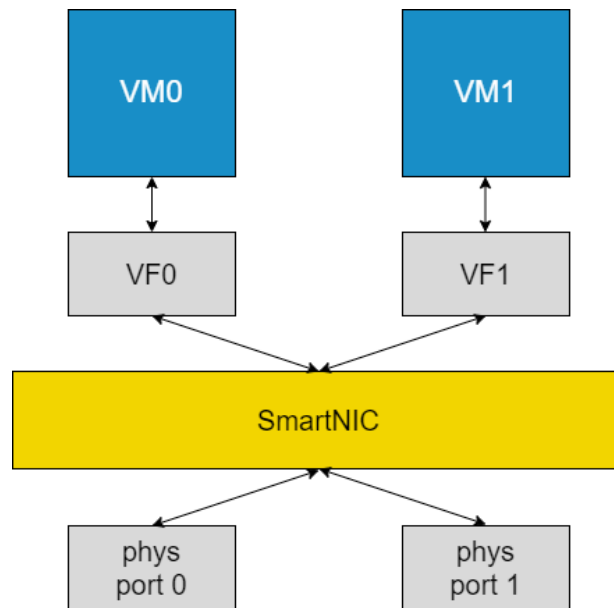


Fig. 1: Packet flow possibilities between VMs and SmartNIC

Currently, VF rate limiting is only supported on egress (based on definition 1 of ingress and egress).

5.2.2 Configuring VF Rate Limit

Allocate Virtual Functions (VFs) to a network interface (<netdev>) using the following commands:

```
# echo 0 > /sys/class/net/<netdev>/device/sriov_numvfs
# echo 2 > /sys/class/net/<netdev>/device/sriov_numvfs
```

After allocating the VFs, set the rate limit using `ip link set`. For example, to set the rate limit to 1000 for VF 1 on the <netdev> `ens4np0`, use the following command:

```
# ip link set ens4np0 vf 1 max_tx_rate 1000 min_tx_rate 0
```

Note: VF numbering starts at 0. In the case of this example, VF 1 refers to the Virtual Function on the client (sender) side.

Note: `max_tx_rate` changes the allowed maximum transmit bandwidth for the specified VF. Setting this parameter to 0 disables rate limiting. The VF parameter must be specified. `min_tx_rate` changes the allowed minimum transmit bandwidth for the specified VF. This value can only be configured at 0, as it is currently not supported.

5.2.3 VF TX Rate Limit Validation

To validate if the VF TX rate limiter is working, two namespaces need to be set up, using the following commands:

```
# ip netns add h1
# ip netns add h2
# ifconfig <netdev> up
# ip link set <vf0> netns h1
# ip link set <vf1> netns h2
# ip netns exec h1 ip link set <vf0> up
# ip netns exec h2 ip link set <vf1> up
# ip netns exec h1 ip addr add 192.168.x.x/24 dev <vf0>
# ip netns exec h2 ip addr add 192.168.x.y/24 dev <vf1>
```

Note: Replace <netdev> with the machine's specific interface associated with the SmartNIC's physical port, which is expected to be something like ens4np0. Replace <vf0> and <vf1> with the Virtual Function interfaces, such as ens4np0v0 and ens4np0v1.

Open two Windows, one as server and one as client.

On the server, run the command:

```
# ip netns exec h1 iperf3 -s
```

On the client, run the command:

```
# ip netns exec h2 iperf3 -c 192.168.x.x -b 10000M -t 10 -P 4
```

Note: `iperf3 -c` is used to transmit the packets and `iperf3 -s` is used to receive the packets.

5.2.4 Expected results

Output with rate limit disabled:

On the client:

```
# ip netns exec h2 iperf3 -c 192.168.1.1 -b 10000M -t 10 -P 4
Connecting to host 192.168.1.1, port 5201
[ 5] local 192.168.1.2 port 51282 connected to 192.168.1.1 port 5201
[ 7] local 192.168.1.2 port 51284 connected to 192.168.1.1 port 5201
[ 9] local 192.168.1.2 port 51286 connected to 192.168.1.1 port 5201
[11] local 192.168.1.2 port 51288 connected to 192.168.1.1 port 5201
[ ID] Interval           Transfer     Bitrate        Retr  Cwnd
[ 5]   0.00-1.00   sec    698 MBytes  5.86 Gbits/sec   39  1.72 MBytes
[ 7]   0.00-1.00   sec    688 MBytes  5.77 Gbits/sec  128  1.70 MBytes
```

(continues on next page)

(continued from previous page)

[9]	0.00-1.00	sec	367 MBytes	3.08 Gbits/sec	47	1012 KBytes
[11]	0.00-1.00	sec	551 MBytes	4.62 Gbits/sec	33	1.39 MBytes
[SUM]	0.00-1.00	sec	2.25 GBytes	19.3 Gbits/sec	247	

[5]	1.00-2.00	sec	736 MBytes	6.18 Gbits/sec	0	2.01 MBytes
[7]	1.00-2.00	sec	642 MBytes	5.38 Gbits/sec	1	1.42 MBytes
[9]	1.00-2.00	sec	443 MBytes	3.72 Gbits/sec	0	1.28 MBytes
[11]	1.00-2.00	sec	472 MBytes	3.96 Gbits/sec	23	1.23 MBytes
[SUM]	1.00-2.00	sec	2.24 GBytes	19.2 Gbits/sec	24	

On the server:

```
# ip netns exec h1 iperf3 -s
-----
Server listening on 5201
-----
Accepted connection from 192.168.1.2, port 51280
[ 5] local 192.168.1.1 port 5201 connected to 192.168.1.2 port 51282
[ 8] local 192.168.1.1 port 5201 connected to 192.168.1.2 port 51284
[ 10] local 192.168.1.1 port 5201 connected to 192.168.1.2 port 51286
[ 12] local 192.168.1.1 port 5201 connected to 192.168.1.2 port 51288
[ ID] Interval          Transfer          Bitrate
[ 5]  0.00-1.00    sec    696 MBytes    5.84 Gbits/sec
[ 8]  0.00-1.00    sec    685 MBytes    5.75 Gbits/sec
[ 10] 0.00-1.00    sec    364 MBytes    3.05 Gbits/sec
[ 12] 0.00-1.00    sec    549 MBytes    4.60 Gbits/sec
[SUM] 0.00-1.00    sec    2.24 GBytes   19.2 Gbits/sec
-----
[ 5]  1.00-2.00    sec    736 MBytes    6.18 Gbits/sec
[ 8]  1.00-2.00    sec    642 MBytes    5.38 Gbits/sec
[ 10] 1.00-2.00    sec    443 MBytes    3.72 Gbits/sec
[ 12] 1.00-2.00    sec    472 MBytes    3.96 Gbits/sec
[SUM] 1.00-2.00    sec    2.24 GBytes   19.2 Gbits/sec
```

The [SUM] lines are important as they show the bitrate. It is seen above that before enabling the rate limit, the total bitrate is 19.2 or 19.3 Gbits/sec.

Output with rate limit enabled and set to 1000 Mbits/sec:

On the client:

```
# ip netns exec h2 iperf3 -c 192.168.1.1 -b 10000M -t 10 -P 4
Connecting to host 192.168.1.1, port 5201
[ 5] local 192.168.1.2 port 51258 connected to 192.168.1.1 port 5201
[ 7] local 192.168.1.2 port 51260 connected to 192.168.1.1 port 5201
[ 9] local 192.168.1.2 port 51262 connected to 192.168.1.1 port 5201
[ 11] local 192.168.1.2 port 51264 connected to 192.168.1.1 port 5201
[ ID] Interval          Transfer          Bitrate          Retr  Cwnd
[ 5]  0.00-1.00    sec    64.5 MBytes    541 Mb/s    4067    9.90 KBytes
[ 7]  0.00-1.00    sec    18.9 MBytes    158 Mb/s    1166    8.48 KBytes
```

(continues on next page)

(continued from previous page)

[9]	0.00-1.00	sec	46.4 MBytes	389 Mbits/sec	3932	11.3 KBytes
[11]	0.00-1.00	sec	21.2 MBytes	178 Mbits/sec	1646	199 KBytes
[SUM]	0.00-1.00	sec	151 MBytes	1.27 Gbits/sec	10811	

[5]	1.00-2.00	sec	29.4 MBytes	246 Mbits/sec	957	7.07 KBytes
[7]	1.00-2.00	sec	42.1 MBytes	353 Mbits/sec	2802	17.0 KBytes
[9]	1.00-2.00	sec	26.6 MBytes	223 Mbits/sec	2035	11.3 KBytes
[11]	1.00-2.00	sec	18.1 MBytes	152 Mbits/sec	1715	2.83 KBytes
[SUM]	1.00-2.00	sec	116 MBytes	975 Mbits/sec	7509	

On the server:

```
# ip netns exec h1 iperf3 -s
-----
Server listening on 5201
-----
Accepted connection from 192.168.1.2, port 51256
[ 5] local 192.168.1.1 port 5201 connected to 192.168.1.2 port 51258
[ 8] local 192.168.1.1 port 5201 connected to 192.168.1.2 port 51260
[ 10] local 192.168.1.1 port 5201 connected to 192.168.1.2 port 51262
[ 12] local 192.168.1.1 port 5201 connected to 192.168.1.2 port 51264
[ ID] Interval          Transfer          Bitrate
[ 5]  0.00-1.00    sec   61.8 MBytes    518 Mbits/sec
[ 8]  0.00-1.00    sec   16.4 MBytes    138 Mbits/sec
[ 10] 0.00-1.00    sec   44.4 MBytes    373 Mbits/sec
[ 12] 0.00-1.00    sec   19.8 MBytes    166 Mbits/sec
[SUM] 0.00-1.00    sec   142 MBytes    1.19 Gbits/sec
-----
[ 5]  1.00-2.00    sec   29.4 MBytes    247 Mbits/sec
[ 8]  1.00-2.00    sec   42.0 MBytes    352 Mbits/sec
[ 10] 1.00-2.00    sec   25.9 MBytes    217 Mbits/sec
[ 12] 1.00-2.00    sec   18.0 MBytes    151 Mbits/sec
[SUM] 1.00-2.00    sec   115 MBytes    967 Mbits/sec
```

After enabling the rate limit, the bitrate is reduced to be close to the `max_tx_rate`. In this case, both the server and client bitrates are close to 1000 Mbits/sec. The client has 1270 and 975 Mbits/sec and the server has 1190 and 967 Mbits/sec bitrates.

5.3 Setting Interface Settings

Unless otherwise stated, changing the interface settings detailed below will not require reloading of the NFP drivers for changes to take effect, unlike the interface breakouts described in *Configuring Interface Media Mode*.

5.4 Multiple Queues

The Physical Functions on a SmartNIC support multiple transmit and receive queues.

5.4.1 View Current Settings

The `-l` flag can be used to view current queue/channel configuration e.g:

```
# ethtool -l <netdev>
Channel parameters for ens1np0:
Pre-set maximums:
RX:                20
TX:                20
Other:             2
Combined:          20
Current hardware settings:
RX:                0
TX:                12
Other:             2
Combined:          8
```

5.4.2 Configure Queues

The `-L` flag can be used to change interface queue/channel configuration. The following parameters can be configured:

rx Receive ring interrupts

tx Transmit ring interrupts

combined Interrupts that service both rx & tx rings

Note: Having RXR-only and TXR-only interrupts are not allowed.

In practice use this formula to calculate parameters for the `ethtool` command: $\text{combined} = \min(\text{RXR}, \text{TXR})$; $\text{rx} = \text{RXR} - \text{combined}$; $\text{tx} = \text{TXR} - \text{combined}$.

To configure 8 combined interrupt servicing:

```
# ethtool -L <netdev> rx 0 tx 0 combined 8
```

5.5 Receive Side Scaling (RSS)

RSS is a technology that focuses on effectively distributing received traffic to the spectrum of RX queues available on a given network interface based on a hash function.

5.5.1 View Current Hash Parameters

The `-n` flag can be used to view current RSS configuration, for example by default:

```
# ethtool -n <netdev> rx-flow-hash tcp4
TCP over IPV4 flows use these fields for computing Hash flow key:
IP SA
IP DA
L4 bytes 0 & 1 [TCP/UDP src port]
L4 bytes 2 & 3 [TCP/UDP dst port]

# ethtool -n <netdev> rx-flow-hash udp4
UDP over IPV4 flows use these fields for computing Hash flow key:
IP SA
IP DA
```

5.5.2 Set Hash Parameters

The `-N` flag can be used to change interface RSS configuration e.g:

```
# ethtool -N <netdev> rx-flow-hash tcp4 sdfn
# ethtool -N <netdev> rx-flow-hash udp4 sdfn
```

The `ethtool` man pages can be consulted for full details of what RSS flags may be set.

5.5.3 Configuring the Key

The `-x` flag can be used to view current interface key configuration, for example:

```
# ethtool -x <netdev>
# ethtool -X <netdev> <hkey>
```

5.6 View Interface Parameters

The `-k` flag can be used to view current interface configurations, for example using a CX 1x40GbE SmartNIC which has an interface id (netdev) `enp4s0np0`:

```
# ethtool -k <netdev>
Features for enp4s0np0:
rx-checksumming: off [fixed]
```

(continues on next page)

```
tx-checksumming: off [fixed]
    tx-checksum-ipv4: off [fixed]
    tx-checksum-ip-generic: off [fixed]
    tx-checksum-ipv6: off [fixed]
    tx-checksum-fcoe-crc: off [fixed]
    tx-checksum-sctp: off [fixed]
scatter-gather: off [fixed]
    tx-scatter-gather: off [fixed]
    tx-scatter-gather-fraglist: off [fixed]
tcp-segmentation-offload: off [fixed]
    tx-tcp-segmentation: off [fixed]
    tx-tcp-ecn-segmentation: off [fixed]
    tx-tcp6-segmentation: off [fixed]
    tx-tcp-mangleid-segmentation: off [fixed]
udp-fragmentation-offload: off [fixed]
generic-segmentation-offload: off [requested on]
generic-receive-offload: on
large-receive-offload: off [fixed]
rx-vlan-offload: off [fixed]
tx-vlan-offload: off [fixed]
ntuple-filters: off [fixed]
receive-hashing: off [fixed]
highdma: off [fixed]
rx-vlan-filter: off [fixed]
vlan-challenged: off [fixed]
tx-lockless: off [fixed]
netns-local: off [fixed]
tx-gso-robust: off [fixed]
tx-fcoe-segmentation: off [fixed]
tx-gre-segmentation: off [fixed]
tx-ipip-segmentation: off [fixed]
tx-sit-segmentation: off [fixed]
tx-udp_tnl-segmentation: off [fixed]
fcoe-mtu: off [fixed]
tx-nocache-copy: off [fixed]
loopback: off [fixed]
rx-fcs: off [fixed]
rx-all: off [fixed]
tx-vlan-stag-hw-insert: off [fixed]
rx-vlan-stag-hw-parse: off [fixed]
rx-vlan-stag-filter: off [fixed]
busy-poll: off [fixed]
tx-gre-csum-segmentation: off [fixed]
tx-udp_tnl-csum-segmentation: off [fixed]
tx-gso-partial: off [fixed]
tx-sctp-segmentation: off [fixed]
l2-fwd-offload: off [fixed]
hw-tc-offload: on
rx-udp_tunnel-port-offload: off [fixed]
```

5.6.1 Receive Checksumming (rx-checksumming)

When enabled, checksum calculation and error checking comparison for received packets is offloaded to the NFP SmartNIC's flow processor rather than the host CPU.

To enable rx-checksumming:

```
# ethtool -K <netdev> rx on
```

To disable rx-checksumming:

```
# ethtool -K <netdev> rx off
```

5.6.2 Transmit Checksumming (tx-checksumming)

When enabled, checksum calculation for outgoing packets is offloaded to the NFP SmartNIC's flow processor rather than the host's CPU.

To enable tx-checksumming:

```
# ethtool -K <netdev> tx on
```

To disable tx-checksumming:

```
# ethtool -K <netdev> tx off
```

5.6.3 Scatter and Gather (scatter-gather)

When enabled the NFP will use scatter and gather I/O, also known as Vectored I/O, which allows a single procedure call to sequentially read data from multiple buffers and write it to a single data stream. Only changes to the scatter-gather interface settings (from `on` to `off` or `off` to `on`) will produce a terminal output as shown below:

To enable scatter-gather:

```
# ethtool -K <netdev> sg on
Actual changes:
scatter-gather: on
tx-scatter-gather: on
generic-segmentation-offload: on
```

To disable scatter-gather:

```
# ethtool -K <netdev> sg off
Actual changes:
scatter-gather: off
tx-scatter-gather: off
generic-segmentation-offload: off
```

5.6.4 TCP Segmentation Offload (TSO)

When enabled, this parameter causes all functions related to the segmentation of TCP packets at egress to be offloaded to the NFP.

To enable tcp-segmentation-offload:

```
# ethtool -K <netdev> tso on
```

To disable tcp-segmentation-offload:

```
# ethtool -K <netdev> tso off
```

5.6.5 Generic Segmentation Offload (GSO)

This parameter offloads segmentation for transport layer protocol data units other than segments and datagrams for TCP/UDP respectively to the NFP. GSO operates at packet egress.

To enable generic-segmentation-offload:

```
# ethtool -K <netdev> gso on
```

To disable generic-segmentation-offload:

```
# ethtool -K <netdev> gso off
```

5.6.6 Generic Receive Offload (GRO)

This parameter enables software implementation of Large Receive Offload (LRO), which aggregates multiple packets at ingress into a large buffer before they are passed higher up the networking stack.

To enable generic-receive-offload:

```
# ethtool -K <netdev> gro on
```

To disable generic-receive-offload:

```
# ethtool -K <netdev> gro off
```

Note: Do take note that scripts that use `ethtool -i <netdev>` to get bus-info will not work on representors as this information is not populated for representor devices.

5.7 Interrupt Coalescing

Interrupt coalescing is used to generate a single interrupt for multiple packets. This is a trade-off between latency and throughput.

5.7.1 View Current Coalescing Parameters

The `-c` flag can be used to view current coalescing configuration, e.g:

```
# ethtool -c <netdev>
Coalesce parameters for <netdev>:
Adaptive RX: off TX: off
stats-block-usecs: n/a
sample-interval: n/a
pkt-rate-low: n/a
pkt-rate-high: n/a

rx-usecs: 50
rx-frames: 64
rx-usecs-irq: n/a
rx-frames-irq: n/a

tx-usecs: 50
tx-frames: 64
tx-usecs-irq: n/a
tx-frames-irq: n/a

rx-usecs-low: n/a
rx-frame-low: n/a
tx-usecs-low: n/a
tx-frame-low: n/a

rx-usecs-high: n/a
rx-frame-high: n/a
tx-usecs-high: n/a
tx-frame-high: n/a
```

5.7.2 Configure Coalescing

The `-C` flag can be used to change coalescing configuration. The following parameters can be configured:

rx-usecs/tx-usecs How many microseconds to delay a rx/tx interrupt after a packet is received/sent.

rx-frames/tx-frames Maximum number of packets to receive/send before a rx/tx interrupt.

adaptive-rx/adaptive-tx Enable or disable adaptive rx/tx coalescing, only supported with kernel 5.15 or newer.

For example, to enable adaptive rx and tx coalescing:


```
# ethtool -C <netdev> adaptive-rx on adaptive-tx on
```

5.8 Flow Steering

It allows user to configure flows into NIC to steer input traffic by `ethtool` command when using kernel driver or `rte_flow` interface when using DPDK application. Supported actions:

- Drop
- Direct to specified rx queue
- Passthrough, the default behaviour
- Mark, only can be configured when using DPDK

When running with kernel driver, `ethtool -n|-u|--show-nfc|--show-ntuple` is used to show steering rules:

```
# ethtool -n <netdev>
8 RX rings available
Total 1 rules

Filter: 1023
  Rule Type: Raw IPv6
  Src IP addr: :: mask: ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
  Dest IP addr: :: mask: ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
  Traffic Class: 0x0 mask: 0xff
  Protocol: 0 mask: 0xff
  L4 bytes: 0x0 mask: 0xffffffff
  Action: Drop
```

`ethtool -N|-U|--config-nfc|--config-ntuple` is used to add or delete steering rules. Currently we only support following flow-types: ether, ip4, ip6, tcp4, udp4, sctp4, tcp6, udp6, sctp6. And for each flow-type, there're some limitations as well:

ether only 'proto' is supported, and proto of ipv4(0x0800) or ipv6(0x86dd) is not allowed, please use ip4/ip6 instead.

ip4/ip6 only 'dst-ip', 'src-ip', 'l4proto' are supported.

tcp4/udp4/sctp4/tcp6/udp6/sctp6 only 'dst-ip', 'src-ip', 'dst-port', 'src-port' are supported.

Examples:

Direct LLDP traffic to queue 0:

```
# ethtool -N <netdev> flow-type ether proto 0x88cc queue 0
```

Drop the ipv4 packets with dst-ip 192.168.1.1/24:

```
# ethtool -N <netdev> flow-type ip4 dst-ip 192.168.1.1 m 0.0.0.255 action -1
```

The same restrictions of supported fields exist when running DPDK application as well.

In DPDK, the rules are configured by using `rte_flow` commands of the `dpdk-testpmd` app, refer http://doc.dpdk.org/guides/testpmd_app_ug/testpmd_funcs.html#flow-rules-management.

Examples:

List existing flow rules sorted by priority:

```
flow list {port_id} [...]
```

Check whether a flow rule can be created:

```
flow validate {port_id}
  [priority {level}]
  pattern {item} [/ {item} [...]] / end
  actions {action} [/ {action} [...]] / end
```

Create a flow rule:

```
flow create {port_id}
  [priority {level}]
  pattern {item} [/ {item} [...]] / end
  actions {action} [/ {action} [...]] / end
```

Destroy specific flow rules:

```
flow destroy {port_id} rule {rule_id} [...]
```

Destroy all flow rules:

```
flow flush {port_id}
```

6 Hardware Installation

This user guide focuses on x86 deployments of Corigine's Agilio hardware.

6.1 Physical Installation

Physically install the SmartNIC in the host server and ensure proper cooling e.g. airflow over card. Ensure the PCI slot is at least Gen3 x8 (The SmartNIC can be placed in Gen3 x16 slot). Once installed, power up the server and open a terminal. For additional support, contact smartnic-support@corigine.com.

6.2 Identification

In a running system the assembly ID and serial number of a PCI device may be determined using the `ethtool` debug interface. This requires knowledge of the physical function network device identifier, or `<netdev>`, assigned to the SmartNIC under consideration. Consult the section [SmartNIC Netdev Interfaces](#) for methods on determining this identifier. The interface name `<netdev>` can be otherwise identified using the `ip link` command. The following shell snippet illustrates this method for some particular `<netdev>` whose name is cast as the argument `$1`:

```
#!/bin/bash
DEVICE=$1
ethtool -W ${DEVICE} 0
DEBUG=$(ethtool -w ${DEVICE} data /dev/stdout | strings)
SERIAL=$(echo "${DEBUG}" | grep "^SN:")
ASSY=$(echo ${SERIAL} | grep -oE AMDA[0-9]{4})
echo ${SERIAL}
echo Assembly: ${ASSY}
```

To run the script execute:

```
# ./<script name> <netdev>
```

Example output of the script:

```
SN: SMAAMDA0099-000117070631 (carbon)
Assembly: AMDA0099
```

Note: The `strings` command is commonly provided by the `binutils` package. This can be installed with the command `yum install binutils` or `apt-get install binutils`, depending on your distribution.

6.3 Validation

The Agilio SmartNIC is a plug and play device. This means that after hardware installation, everything should be working perfectly. To ensure that everything is working as it should, the following validations can be run.

Use one of the following `lspci` commands to validate that the SmartNIC is being correctly detected by the host server and identify its PCI address. The PCI vendor identifier for SmartNICs with Board Support Package (BSP) versions before 22.09 is 19ee and the specific PCI vendor identifier for SmartNICs with AMDA2XXX product codes, with a BSP version of at least 22.09, is 1da8. The device tuples are 3800, 4000 and 6000 respectively.

For SmartNICs with a vendor ID of 19ee:

```
# lspci -bDnnd 19ee:
0000:02:00.0 Ethernet controller [0200]: Netronome Systems, Inc. Device [19ee:4000]
```

Or for SmartNICs with a vendor ID of 1da8:

```
# lspci -bDnnd 1da8:
0000:17:00.0 Ethernet controller [0200]: Corigine, Inc. Device [1da8:3800]
```

Note: The `lspci` command is commonly provided by the `pciutils` package. This can be installed with the command `yum install pciutils` or `apt-get install pciutils`, depending on your distribution.

6.4 SmartNIC Netdev Interfaces

After NFP driver initialization new `netdev` interfaces will be created:

```
# ip link
4: enp6s0np0s0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT_
->group default qlen 1000
    link/ether 00:15:4d:13:01:db brd ff:ff:ff:ff:ff:ff
5: enp6s0np0s1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT_
->group default qlen 1000
    link/ether 00:15:4d:13:01:dd brd ff:ff:ff:ff:ff:ff
6: enp6s0np0s2: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT_
```

(continues on next page)

(continued from previous page)

```
↪group default qlen 1000
    link/ether 00:15:4d:13:01:de brd ff:ff:ff:ff:ff:ff
7: enp6s0np0s3: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT
↪group default qlen 1000
    link/ether 00:15:4d:13:01:df brd ff:ff:ff:ff:ff:ff
8: enp6s0np1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT
↪group default qlen 1000
    link/ether 00:15:4d:13:01:dc brd ff:ff:ff:ff:ff:ff
```

Note: Netdev naming may vary depending on your linux distribution and configuration e.g. enpAsXn-pYsZ, pXpY.

To confirm the names of the interfaces, view the contents of `/sys/bus/pci/devices/<pci addr>/net`, using the specific vendor ID and PCI address obtained in above.

For SmartNICs with a vendor ID of 19ee:

```
#!/bin/bash
PCIA=$(lspci -d 19ee:4000 | awk '{print $1}' | xargs -Iz echo 0000:z)
echo $PCIA | tr ' ' '\n' | xargs -Iz echo "ls /sys/bus/pci/devices/z/net" | bash
```

Or for SmartNICs with a vendor ID of 1da8:

```
#!/bin/bash
PCIA=$(lspci -d 1da8:3800 | awk '{print $1}' | xargs -Iz echo 0000:z)
echo $PCIA | tr ' ' '\n' | xargs -Iz echo "ls /sys/bus/pci/devices/z/net" | bash
```

The output of such a script would be similar to:

```
enp6s0np0s0 enp6s0np0s1 enp6s0np0s2 enp6s0np0s3 enp6s0np1
```

In the worst case scenario, netdev types can also be discovered by reading the kernel logs.

6.4.1 Support for Biosdevname

Corigine NICs support `biosdevname` netdev naming with recent versions of the utility, circa December 2018, e.g. RHEL 8.0 onwards. There are some notable points to be aware of:

- Whenever an unsupported netdev is considered for naming, the `biosdevname` naming will be skipped and the next inline naming scheme will take preference, e.g. the `systemd` naming policies.
- Netdevs in breakout mode are not supported for naming.
- VF netdevs will still be subject to `biosdevname` naming irrespective of the breakout mode of other netdevs.
- When using an older version of the `biosdevname` utility, users will observe inconsistent naming of netdevs on multiport NICs, i.e. one netdev may be named according to the `biosdevname` scheme and another according to `systemd` schemes.

To disable `biosdevname` users can add `biosdevname=0` to the kernel command line.

Refer to the [online biosdevname documentation](#), created by Dell for more details about the naming policy convention that will be applied.

6.5 Accessory

For details about optical module cables, see <https://www.corigine.com/testedCablesAndOptics.html>

7 Linux Driver

7.1 Preparing for Installation

7.1.1 Checking the kernel version

The Corigine SmartNIC physical function driver is included in Linux 4.13 and later kernels. The list of minimum required operating system distributions and their respective kernels, which include the NFP driver, are as follows:

Operating System	Kernel package version
RHEL/CentOS 7.5	3.10.0-862.el7
RHEL/CentOS 7.6	3.10.0-957.el7
RHEL/CentOS 7.7	3.10.0-1062.el7
RHEL 8.0	4.18.0-80.el8
Ubuntu 18.04 LTS	4.15.0-20.21
Kylin V10 (Sword)	4.19.90-25.14.v2101.ky10
UOS V20 (1050d)	4.19.0
OpenEuler 22.03	5.10.0-5.10.1.25.oe1
BC Linux 7.6	3.10.0-957HG.el7.x86_64

In order to upgrade Ubuntu 16.04.0 - 16.04.3 to a supported version, the following commands must be run:

```
# apt-get update
# apt-get upgrade
# apt-get dist-upgrade
```

7.1.2 Checking the NFP Driver

Use the `modinfo` command to confirm that your current Operating System contains the upstreamed `nfp` module:

```
# modinfo nfp | head -3
filename:
/lib/modules/<kernel package version>/kernel/drivers/net/ethernet/netronome/nfp/
↪nfp.ko.xz
description:    The Netronome Flow Processor (NFP) driver.
license:       GPL
```

Note: If the module is not found in your current kernel, refer to [Installing the Linux Driver](#).

7.2 Installing the Linux Driver

7.2.1 Install Driver via Corigine Repository

Importing GPG-Key

For RHEL and CentOS, add the Corigine GPG-key:

```
# rpm --import https://download.corigine.com.cn/public/Corigine.pub
```

For Ubuntu based systems, add the Corigine GPG-key:

```
# curl -fsSLo /usr/share/keyrings/corigine-archive-keyring.gpg \
https://download.corigine.com.cn/public/Corigine.gpg
```

Configuring Repositories

For RHEL 7 and CentOS 7, the RPM repository can be added:

```
# yum-config-manager --add-repo \
https://download.corigine.com.cn/public/corigine.repo
```

For RHEL 8+ and CentOS 8+, the RPM repository can be added:

```
# dnf config-manager --add-repo \
https://download.corigine.com.cn/public/corigine.repo
```

For Ubuntu based systems:

```
# mkdir -p /etc/apt/sources.list.d
# KEY=/usr/share/keyrings/corigine-archive-keyring.gpg
```

(continues on next page)

(continued from previous page)

```
# REPOLINK=https://download.corigine.com.cn/public/apt
# OUPUTPATH=/etc/apt/sources.list.d/corigine.list
# echo "deb [arch=all signed-by=${KEY}] ${REPOLINK} stable main" > ${OUPUTPATH}
# apt-get update
```

RHEL and CentOS

Installing the NFP Dynamic Kernel Module Support (DKMS) driver package depends on DKMS to be installed. On RHEL based systems, DKMS is provided in the EPEL repository. If this is not installed, it must first be done before installing the NFP driver package. The EPEL repository can be installed using:

```
# yum install epel-release
```

Ensure that the correct kernel-development package is installed that matches the current kernel version. The following command will check the kernel-devel version and, if needed, install the correct kernel-devel package:

```
# yum install kernel-devel-$(uname -r)
```

Installing the driver from the Corigine repository, should automatically install all dependencies:

```
# yum search agilio-nfp-driver-dkms
# yum install agilio-nfp-driver-dkms
```

Ubuntu

```
# apt-cache search agilio-nfp-driver-dkms
# apt-get install agilio-nfp-driver-dkms
```

7.2.2 Install Driver via Software Package

The latest Driver packages can be obtained at the downloads area of the Corigine Support site (<https://www.corigine.com/DPUDownload.html>) . For example, at the time of writing, v22.07 is the newest version.

RHEL and CentOS

CentOS 7:

```
# wget https://download.corigine.com.cn/public/packages/agilio-nfp-driver-dkms-22.07-1.noarch.rpm
# yum -y install agilio-nfp-driver-dkms-22.07-1.noarch.rpm
# rmmod nfp; modprobe nfp
# dracut -f
```

CentOS 8 and later:

```
# wget https://download.corigine.com.cn/public/packages/agilio-nfp-driver-dkms-22.07-1.noarch.rpm
# dnf -y install agilio-nfp-driver-dkms-22.07-1.noarch.rpm
# rmmod nfp; modprobe nfp
# dracut -f
```

Ubuntu

```
# wget https://download.corigine.com.cn/public/apt/pool/main/a/agilio-nfp-driver-dkms-all/agilio-nfp-driver-dkms_22.07-1_all.deb
# apt-get -y install agilio-nfp-driver-dkms_22.07-1_all.deb
# rmmod nfp; modprobe nfp
# update-initramfs -u
```

KylinOS

Installing the NFP DKMS driver package depends on DKMS to be installed. If the DKMS version of OS is an earlier version, please upgrade it first (it is highly recommended to use DKMS provided in the EPEL repository):

```
# wget https://download.corigine.com.cn/public/packages/agilio-nfp-driver-dkms-22.07-0.noarch.rpm
# yum -y install agilio-nfp-driver-dkms-22.07-0.noarch.rpm
# rmmod nfp; modprobe nfp
# dracut -f
```

UOS

Take UOS V20 1050d as an example, in order to install the driver, the following commands can be run:

```
# wget https://download.corigine.com.cn/public/apt/pool/main/a/agilio-nfp-driver-dkms-all/agilio-nfp-driver-dkms_22.07-1_all.deb
# dpkg -i agilio-nfp-driver-dkms_22.07-1_all.deb
```

OpenEuler

Take OpenEuler 22.03 LTS as an example, in order to install the driver, the following commands can be run:

```
# wget https://download.corigine.com.cn/public/packages/agilio-nfp-driver-dkms-22.08-0.noarch.rpm
# yum install agilio-nfp-driver-dkms-22.08-0.noarch.rpm
```

Note: The V22.08.0 and later driver version is necessary.

BC Linux

Take BC Linux 7.6 as an example, in order to install the driver, the following commands can be run:

```
# wget https://download.corigine.com.cn/public/packages/agilio-nfp-driver-dkms-22.07-0.noarch.rpm
# yum install agilio-nfp-driver-dkms-22.07-0.noarch.rpm
```

7.2.3 Building from Source

Driver sources for Corigine Network Flow Processor devices, including the NFP-3800, NFP-4000 and NFP-6000 models can be found at: <https://github.com/Corigine/nfp-driv-kmods>

RHEL 7 and CentOS 7 Dependencies

```
# yum install -y kernel-devel-$(uname -r)
# yum groupinstall -y "Development Tools"
```

RHEL 8 and CentOS 8 Dependencies

```
# dnf install -y kernel-devel-$(uname -r)
# dnf groupinstall -y "Development Tools"
```

Ubuntu Dependencies

```
# apt-get update
# apt-get install -y linux-headers-$(uname -r) build-essential libelf-dev
```

Clone, Build and Install

```
# git clone https://github.com/Corigine/nfp-driv-kmods.git
# cd nfp-driv-kmods
# make
# make install
# depmod -a
```

7.2.4 (Optional) Kernel Changes

Take note that installing the DKMS driver will only install it for the currently running kernel. When you upgrade the installed kernel it may not automatically update the the `nfp` module to use the version in the DKMS package. In kernel versions older than v4.16 the `MODULE_VERSION` parameter of the in-tree module was not set, which causes DKMS to pick the module with the highest `srcversion` hash (<https://github.com/dell/dkms/issues/14>). This is worked around by the package install step adding a `--force` to the DKMS install, but this will not trigger on a kernel upgrade. To work around this issue, boot into the new kernel and then re-install the `agilio-nfp-driver-dkms` package.

This should not be a problem when upgrading from kernels v4.16 and newer as the `MODULE_VERSION` has been added, so the DKMS version check should work properly. It's not possible to determine which `nfp.ko` file was loaded by only relying on information provided by the kernel. However, it's possible to confirm that the binary signature of a file on disk and the module loaded in memory is the same.

To confirm that the module in memory is the same as the file on disk, compare the `srcversion` tag. The in-memory module's tag is at `/sys/module/nfp/srcversion`. The default on-disk version can be queried with `modinfo`:

```
# cat /sys/module/nfp/srcversion # In-memory module
# modinfo nfp | grep "^srcversion:" # On-disk module
```

If these tags are in sync, the filename of the module provided by a `modinfo` query will identify the origin of the module:

```
# modinfo nfp | grep "^filename:"
```

If these tags are not in sync, there are likely conflicting copies of the module on the system: the `initramfs` may be out of sync or the module dependencies may be inconsistent.

The in-tree kernel module is usually located at the following path (please note, this module may be compressed with a `.xz` extension):

```
/lib/modules/$(uname -r)/kernel/drivers/net/ethernet/netronome/nfp/nfp.ko
```

The DKMS module is usually located at the following path:

```
/lib/modules/$(uname -r)/updates/dkms/nfp.ko
```

To ensure that the out-of-tree driver is correctly loaded instead of the in-tree module, the following commands can be run:

Ubuntu:

```
# mkdir -p /etc/depmod.d
# echo "search nfp * extra updates" > /etc/depmod.d/netronome.conf
# depmod -a
# rmmod nfp; modprobe nfp
# update-initramfs -u
```

CentOS:

```
# mkdir -p /etc/depmod.d
# echo "search nfp * extra updates" > /etc/depmod.d/netronome.conf
# depmod -a
# modprobe -r nfp; modprobe nfp
# dracut -f
```

7.2.5 Confirm that the NFP Driver is Loaded

Use `lsmod` to list the loaded driver modules and use `grep` to search for the `nfp` string:

```
# lsmod | grep nfp
nfp                161364  0
```

If the NFP driver is not loaded, the following command loads it manually:

```
# modprobe nfp
```

Note: If the driver cannot be loaded after the OOT driver is installed in the case of UEFI secure boot enable, please refer to [UEFI Secure Boot with Out-of-Tree NFP Driver](#).

7.3 Using the Linux Driver

The Linux driver supports Corigine's line of Flow Processor devices, including the NFP3800, NFP4000, NFP5000, and NFP6000 models, which are also incorporated in the company's family of Agilio SmartNICs.

The driver is used to expose networking devices (`netdevs`) and/or user space access to the card via a character device created by the driver.

7.3.1 Configuring Interface Media Mode

The following sections detail the configuration of the SmartNIC `netdev` interfaces.

Note: For older kernels that do not support the configuration methods outlined below, please refer to [BSP Installation](#) on how to make use of the BSP toolset to configure interfaces.

Configuring interface link-speed

The following steps explain how to change between 10G mode and 25G mode on CX 2x25GbE cards. The changing of port speed must be done in order, port 0 (p0) must be set to 10G before port 1 (p1) may be set to 10G.

Down the respective interface(s):

```
# ip link set dev <netdev port 0> down
# ip link set dev <netdev port 1> down
```

Set interface link-speed to 10G:

```
# ethtool -s <netdev port 0> speed 10000
# ethtool -s <netdev port 1> speed 10000
```

Set interface link-speed to 25G:

```
# ethtool -s <netdev port 0> speed 25000
# ethtool -s <netdev port 1> speed 25000
```

Set interface auto-negotiation:

```
# ethtool -s <netdev port 0> autoneg on/off
# ethtool -s <netdev port 1> autoneg on/off
```

For NFP3800 cards with driver versions later than V22.10.0 in use:

```
# ip link set dev <netdev port 0> up
# ip link set dev <netdev port 1> up
```

For earlier driver versions, reload the driver for changes to take effect:

```
# rmmod nfp; modprobe nfp
```

Note: IP addresses of the server and the host machine are on the same range.

7.3.2 Configuring interface Maximum Transmission Unit (MTU)

The MTU of interfaces can temporarily be set using the `iproute2`, `ip link` or `ifconfig` tools. Note that this change will not persist. Setting this via *Network Manager*, or another appropriate OS configuration tool, is recommended as changes to the MTU using *Network Manager* can be made to persist.

Set interface MTU to 9000 bytes:

```
# ip link set dev <netdev port> mtu 9000
```

It is the responsibility of the user or the orchestration layer to set appropriate MTU values when handling jumbo frames or utilizing tunnels. For example, if packets sent from a VM are to be encapsulated on the card and egress a physical port, then the MTU of the VF should be set to lower than that of the physical port to account for the extra bytes added by the additional header.

If a setup is expected to see fallback traffic between the SmartNIC and the kernel then the user should also ensure that the PF MTU is appropriately set to avoid unexpected drops on this path.

7.3.3 Configuring FEC modes

CX 2x25GbE SmartNICs support Forward Error Correction (FEC) mode configuration, e.g. Auto, Firecode Base-R, Reed-Solomon and Off modes. Each physical port's FEC mode can be set independently via the `ethtool` command. To view the currently supported FEC modes of the interface use the following:

```
# ethtool <netdev>
Settings for <netdev>:
  Supported ports: [ FIBRE ]
  Supported link modes:   Not reported
  Supported pause frame use: No
  Supports auto-negotiation: No
  Supported FEC modes: None BaseR RS
  Advertised link modes:  Not reported
  Advertised pause frame use: No
  Advertised auto-negotiation: No
  Advertised FEC modes: BaseR RS
  Speed: 25000Mb/s
  Duplex: Full
  Port: Direct Attach Copper
  PHYAD: 0
  Transceiver: internal
  Auto-negotiation: on
  Link detected: yes
```

One can see above which FEC modes are supported for this interface. Note that the CX 2x25GbE SmartNIC used for the example above only supports Firecode BaseR FEC mode on ports that are forced to 10G speed.

Note: Replace `<netdev>` with the machine's specific interface number associated with the SmartNIC's PF, which is expected to be something like `ens3np0` or `enp130s0np0`.

Note: Ethtool FEC support is only available in kernel 4.14 and newer or RHEL/Centos 7.5 and equivalent distributions. The Corigine upstream kernel driver provides ethtool FEC support from kernel 4.15. Furthermore, the SmartNIC NVRAM version must be at least 020025.020025.02006e to support ethtool FEC get/set operations.

To determine your version of the current SmartNIC NVRAM, look at the following system log:

```
# dmesg | grep 'nfp.*BSP'
[2387.682046] nfp 0000:82:00.0: BSP: 22.10-0
```

This example lists a version of 22.10-0 which is sufficient to support `ethtool` FEC mode configuration. To update your SmartNIC NVRAM flash, please contact [Corigine support](#).

If the SmartNIC NVRAM or the kernel does not support `ethtool` modification of FEC modes, no supported FEC modes will be listed in the `ethtool` output for the port. This could be because of an outdated kernel version or an unsupported distribution (e.g. Ubuntu 16.04 irrespective of the kernel version):

```
# ethtool <netdev>
Settings for <netdev>:
...
Supported FEC modes: None
```

To show the currently active FEC mode for `<netdev>` (eg. `enp130s0np0`):

```
# ethtool --show-fec <netdev>
FEC parameters for <netdev>:
Configured FEC encodings: Auto Off BaseR RS
Active FEC encoding: Auto
```

To force the FEC mode for a particular port, auto-negotiation must be disabled with the following:

```
# ip link set enp130s0np0 down
# ethtool -s enp130s0np0 autoneg off
# ip link set enp130s0np0 up
```

Note: In order to change the auto-negotiation configuration the port must be down.

Note: Changing the auto-negotiation configuration will not affect the SmartNIC port speed. Please see [Configuring interface link-speed](#) to adjust this setting.

To modify the FEC mode to Firecode Base-R:

```
# ethtool --set-fec <netdev port> encoding baser
```

Verify the newly selected mode:

```
# ethtool --show-fec enp130s0np0
FEC parameters for enp130s0np0:
Configured FEC encodings: Auto Off BaseR RS
Active FEC encoding: BaseR
```

To modify the FEC mode to Reed-Solomon:


```
# ethtool --set-fec enp130s0np0 encoding rs
```

Verify the newly selected mode:

```
# ethtool --show-fec enp130s0np0
FEC parameters for enp130s0np0:
Configured FEC encodings: Auto Off BaseR RS
Active FEC encoding: RS
```

Revert back to the default Auto setting:

```
# ethtool --set-fec enp130s0np0 encoding auto
```

Finally verify the setting again:

```
# ethtool --show-fec enp130s0np0
FEC parameters for enp130s0np0:
Configured FEC encodings: Auto Off BaseR RS
Active FEC encoding: Auto
```

FEC and auto-negotiation settings are persisted on the SmartNIC across reboots.

Note: In this context setting the interface mode to `auto` specifies that the encoding scheme should be automatically determined if possible. It does **not** enable auto-negotiation of link speed between 10Gbps and 25Gbps.

7.3.4 Setting Interface Breakout Mode

The following commands only work on kernel versions 4.13 and later. If your kernel is older than 4.13 or you do not have devlink support enabled refer to the section on configuring interfaces: [Configure Media Settings](#).

Note: Breakout mode settings are only applicable to CX 40GbE and CX 2x40GbE SmartNICs.

Determine the card's PCI address with the correct vendor ID. The PCI vendor identifier for SmartNICs with Board Support Package (BSP) versions before 22.09 is 19ee and the specific PCI vendor identifier for SmartNICs with AMDA2XXX product codes, with a BSP version of at least 22.09 is 1da8.

For SmartNICs with a vendor ID of 19ee:

```
# lspci -Dkd 19ee:
0000:04:00.0 Ethernet controller: Netronome Systems, Inc. Device 4000
    Subsystem: Netronome Systems, Inc. Device 4001
    Kernel driver in use: nfp
    Kernel modules: nfp
```

Or for SmartNICs with a vendor ID of 1da8:

```
# lspci -Dkd 1da8:
0000:17:00.0 Ethernet controller: Corigine, Inc. Device 3800
    Subsystem: Corigine, Inc. Device 7ff5
    Kernel driver in use: nfp
    Kernel modules: nfp
```

List the devices:

```
# devlink dev show
pci/0000:04:00.0
```

The output of `devlink dev show` should then be used for the following commands.

Split the first physical 40G port from 1x40G to 4x10G ports:

```
# devlink port split pci/0000:04:00.0/0 count 4
```

Split the second physical 40G port from 1x40G to 4x10G ports:

```
# devlink port split pci/0000:04:00.0/4 count 4
```

If the SmartNIC's port is already configured in breakout mode (it has already been split) then `devlink` will respond with an argument error. Whenever changes to the port configuration are made, the original `netdev(s)` associated with the port will be removed from the system:

```
# dmesg | tail
[ 5696.432306] nfp 0000:04:00.0: nfp: Port #0 config changed, unregistering.
↳Driver reload required before port will be operational again.
[ 6270.553902] nfp 0000:04:00.0: nfp: Port #4 config changed, unregistering.
↳Driver reload required before port will be operational again.
```

The driver needs to be reloaded for the changes to take effect. Older driver/SmartNIC NVRAM versions may require a system reboot for changes to take effect. The driver communicates events related to port split/unsplit in the system logs. The driver may be reloaded with the following command:

```
# rmmod nfp; modprobe nfp
```

After reloading the driver, the `netdevs` associated with the split ports will be available for use:

```
# ip link show
...
68: enp4s0np0s0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT
↳group default qlen 1000
69: enp4s0np0s1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT
↳group default qlen 1000
70: enp4s0np0s2: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT
↳group default qlen 1000
71: enp4s0np0s3: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT
↳group default qlen 1000
72: enp4s0np1s0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT
```

(continues on next page)

(continued from previous page)

```
↔group default qlen 1000
73: enp4s0np1s1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT_
↔group default qlen 1000
74: enp4s0np1s2: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT_
↔group default qlen 1000
75: enp4s0np1s3: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT_
↔group default qlen 1000
```

Note: There is an ordering constraint to splitting and unsplitting the ports on CX 2x40GbE SmartNICs. The one physical 40G port cannot be split without the other physical port also being split, hence a setup of one port at 1x40G and another port at 4x10G is always invalid even if it's only intended to be a transitional mode. The driver will reject such configurations.

Breakout mode persists on the SmartNIC across reboots. To revert back to the original 2x40G ports use the `unsplit` subcommand.

Unsplit Port 1:

```
# devlink port unsplit pci/0000:04:00.0/4
```

Unsplit Port 0:

```
# devlink port unsplit pci/0000:04:00.0/0
```

The NFP drivers will again have to be reloaded (`rmmod nfp` then `modprobe nfp`) for unsplit changes in the port configuration to take effect.

7.3.5 Configuring interface private-flags

Setting interface `disable-fw-lldp`

The `fw-lldp` feature is that sending packet of lldp by the firmware and it is enabled by default for the nic. The following steps explain how to turn `fw-lldp` on or off.

Turn off the `fw-lldp`:

```
# ethtool --set-priv-flags <netdev> disable-fw-lldp on
```

Turn on the `fw-lldp`:

```
# ethtool --set-priv-flags <netdev> disable-fw-lldp off
```

To show the current private-flags:

```
# ethtool --show-priv-flags <netdev>
Private flags for <netdev>:
disable-fw-lldp: off
```

7.3.6 Confirming Connectivity

Allocating IP Addresses

Under RHEL and CentOS, the network configuration is managed by default using *NetworkManager*.

The following commands can be used to set the IPv4 address statically:

```
# ip address add 10.0.0.2/24 dev <netdev port>
# ip link set <netdev port> up
```

Pinging interfaces

After you have successfully assigned IP addresses to the NFP interfaces, perform a standard ping test to confirm connectivity between localhost and the NFP device:

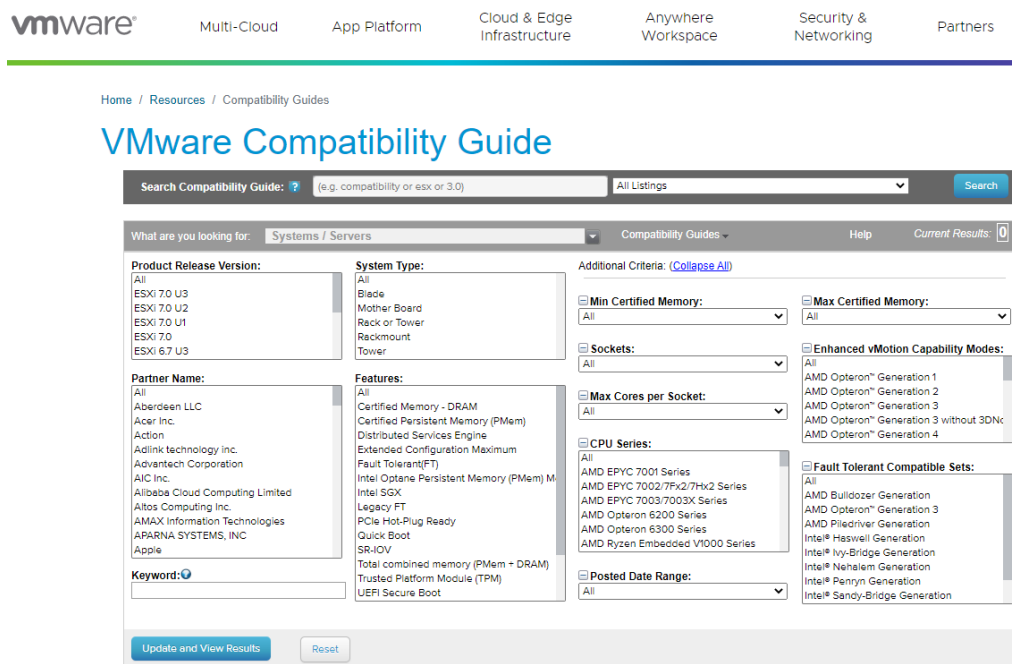
```
# ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.067 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.062 ms
```

8 VMware Driver

The minimum OS version required for the Agilio series NIC driver is ESXi 7.0 U3.

8.1 Driver Download

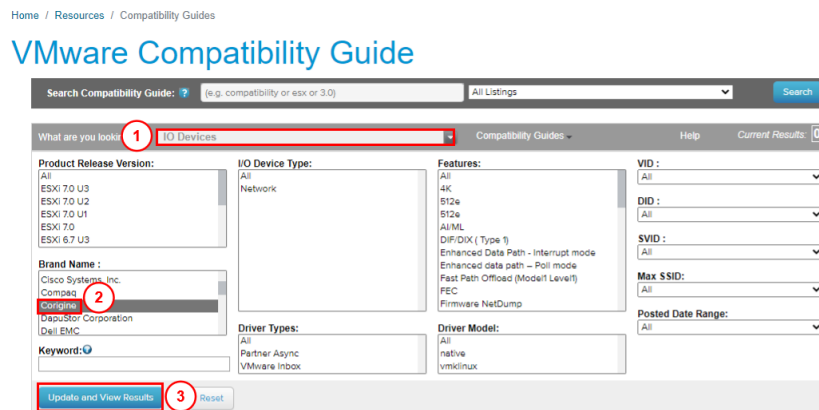
1. Visit the VMware Website.



The screenshot shows the VMware Compatibility Guide search interface. The search bar contains the text "Search Compatibility Guide: ? (e.g. compatibility or esx or 3.0)". The "What are you looking for:" dropdown is set to "Systems / Servers". The "Brand Name" filter is set to "All". The "Product Release Version" filter is set to "All". The "System Type" filter is set to "All". The "Additional Criteria" section includes filters for "Min Certified Memory", "Max Certified Memory", "Sockets", "Max Cores per Socket", "CPU Series", and "Posted Date Range". The "Features" section includes filters for "Certified Memory - DRAM", "Certified Persistent Memory (PMem)", "Distributed Services Engine", "Extended Configuration Maximum", "Fault Tolerant(FT)", "Intel Optane Persistent Memory (PMem) M", "Intel SGX", "Legacy FT", "PCIe Hot-Plug Ready", "Quick Boot", "SR-IOV", "Total combined memory (PMem + DRAM)", "Trusted Platform Module (TPM)", and "UEFI Secure Boot". The "Partner Name" filter is set to "All". The "Keyword" field is empty. The "Update and View Results" button is highlighted.

2. Select the corresponding driver

- a. Select IO Devices in What are you looking for, and Corigine in Brand Name. Click Update and View Results.



The screenshot shows the VMware Compatibility Guide search interface with the following filters: "What are you looking for:" is set to "IO Devices" (circled in red with a '1'); "Brand Name" is set to "Corigine" (circled in red with a '2'); "Product Release Version" is set to "All"; "I/O Device Type" is set to "Network"; "Features" is set to "All"; "VID" is set to "All"; "DID" is set to "All"; "SVID" is set to "All"; "Max SSID" is set to "All"; "Posted Date Range" is set to "All"; "Driver Types" is set to "All"; "Driver Model" is set to "All". The "Update and View Results" button is highlighted with a red box and a '3' (circled in red).

The results are as follows:

I/O Device and Model Information

The detailed lists show actual vendor devices that are either physically tested or are similar to the devices tested by VMware or VMware partners. VMware provides support only for the devices that are listed in this document.

Click on the **'Model'** to view more details and to subscribe to RSS feeds.

[Bookmark](#) | [Print](#) | [Export to CSV](#)

Search Results: Your search for "IO Devices" returned 3 results. [Back to Top](#) [Turn Off Auto Scroll](#) Display: 10 ▼

Brand Name	Model	Device Type	Supported Releases
Corigine	Agilio CX	Network	ESXi 7.0 U3
Corigine	Agilio GX 2x10GbE	Network	ESXi 7.0 U3
Corigine	Agilio GX 2x25GbE	Network	ESXi 7.0 U3

- b. Click [Agilio GX 2x10GbE](#) in Model column, and view Model Details. Take the [Agilio GX 2x10GbE](#) as an example.

[Home](#) / [Resources](#) / [Compatibility Guides](#) / [Detail](#)


VMware Compatibility Guide

[Back to Search Results](#) Print

Model Details

Model : [Agilio GX 2x10GbE](#)
 Device Type : Network DID : 3800
 Brand Name : Corigine SVID : 0b64
 Number of Ports : 2 SSID : 19ee
 VID : 19ee

Notes: Firmware versions listed are the minimum supported versions. Refer to <http://kb.vmware.com/kb/2030818> for additional information on other supported driver and firmware combinations

[Click here to be notified when this page is updated: rss feed](#) 
[Click here to export this page: Export to CSV](#)

Model Release Details [Expand All](#) | [Collapse All](#)

VMware Product Name : [ESXi 7.0 U3](#) ▼

Release	Device Driver(s)	Firmware Version	Additional Firmware Version	Type	Features
<input type="checkbox"/> ESXi 7.0 U3	nfp version 1.0.0.3	22.07	N/A	Partner Async, native	View

[Back to Search Results](#) Print

- c. Click +, view Model Release Details.

Model Release Details [Expand All](#) | [Collapse All](#)

VMware Product Name : [ESXi 7.0 U3](#) ▼

Release	Device Driver(s)	Firmware Version	Additional Firmware Version	Type	Features
<input checked="" type="checkbox"/> ESXi 7.0 U3	nfp version 1.0.0.3	22.07	N/A	Partner Async, native	View

Feature Category: IO Device
 Features: GENEVE-Offload, IPv6, RSS, SR-IOV, VXLAN-Offload

Footnotes : Download driver from <https://customerconnect.vmware.com/en/downloads/details?downloadGroup=DT-ESXi70-CORIGINE-NFP-1003&productId=974>
 Please refer to <http://kb.vmware.com/kb/2045704> for GOS supported on SR-IOV in VMware vSphere 5.1 or later

- d. Click Footnotes download link, the download page is displayed. Select the corresponding driver file and download.

Product Downloads ©

File	Information
VMware ESXi 7.0 nfp 1.0.0.3 NIC Driver for Ethernet specific functions and plugs	DOWNLOAD NOW
File size: 196 MB File type: zip Read More	
Release_Notes_nfp-1.0.0.3	DOWNLOAD NOW
File size: 194.19 KB File type: pdf Read More	

8.2 Driver Installation

Connect to ESXi HOST using SSH and place the above download driver file to the following path (as an example):

```
/vmfs/volumes/datastore1/VMW-esx-7.0.3-Corigine-nfp-1.0.0.3-1OEM.703.0.0.18644231.  
↪zip
```

Run the following command to unzip the driver file:

```
# unzip Corigine-nfp_1.0.0.3-1OEM.703.0.0.18644231_20272745-package.zip
```

Run the following command to install the driver file:

```
# esxcli software component apply -d /vmfs/volumes/datastore1/Corigine-nfp_1.0.0.3-  
↪1OEM.703.0.0.18644231_20272745.zip
```

Note:

- The path in above command is an absolute path.
- Some users fail to run the above command because the certificate is not installed. In this case, add `--no-sig-check` behind this command. For example, the above command should be changed to `esxcli software component apply -d /vmfs/volumes/datastore1/VMW-esx-7.0.3-Corigine-nfp-1.0.0.3-1OEM.703.0.0.18644231.zip --no-sig-check`.

After the installation is complete, a message similar to the following is displayed:

```
Installation Result  
  Components Installed: Corigine-nfp_1.0.0.3-1OEM.703.0.0.18644231  
  Components Removed:  
  Components Skipped:  
  Message: Operation finished successfully.  
  Reboot Required: false
```

Reboot the machine. When finished, SSH reconnect. Run the `esxcli software component list |grep nfp` command to view the corresponding nfp driver information that means the installation is successful:

```
# esxcli software component list |grep nfp  
Corigine-nfp          Corigine NFP Ethernet Driver          1.0.0.4-10          EM.  
↪703.0.0.18644231    1.0.0.4-0          Corigine 08-29-2022  ↪  
↪VMwareCertified
```

At this time, run the `esxcli network nic list` command to view the corresponding vmnic port of nfp:

```
# esxcli network nic list  
Name   PCI Device   Driver Admin Status Link Status Speed Duplex
```

(continues on next page)

(continued from previous page)

MAC Address	MTU	Description
-----	-----	-----
vmnic0 0000:b1:00.0 nfp	Up	Up
a8c:1f:64:30:6f:27 1500	Corigine Inc. nfp	Kestrel Ethernet Controller

8.3 Using the VMware Driver

8.3.1 Configuring Link-speed

Run the `esxcli network nic list` command and view the current port link-speed:

```
# esxcli network nic list
Name      PCI Device      Driver      Admin Status  Link Status  Speed
-----
vmnic5    0000:06:00.0    nfp         Up             Up           10000
```

Run the `esxcli network nic set -S <speed(Mbps)> -D full -n <port_name>` command and set the port link-speed:

```
# esxcli network nic set -S 1000 -D full -n vmnic5
```

8.3.2 Confirming Connectivity

The following commands need to be performed in the esxi shell.

View the NIC list:

```
# esxcli network nic list
```

Add a vSwitch and bind the NIC port:

```
# esxcli network vswitch standard uplink add -u vmnic5 -v nfp-switch
```

Add a portgroup and bind the vSwitch:

```
# esxcli network vswitch standard portgroup add -p nfp-port0 -v nfp-switch
```

Add a vmkernel NIC and bind the portgroup:

```
# esxcli network ip interface add -i vmk2 -p nfp-port0
```

Set the static IP address of vmkernel:

```
# esxcli network ip interface ipv4 set -i vmk2 -I 5.5.5.6 -N 255.255.255.0 -g 5.5.
↪5.254 -t static
```

Configure interface MTU:


```
# esxcli network ip interface set --mtu 9000 --interface-name vmk2
```

Ping the peer IP address:

```
# ping 5.5.5.1
Pinging 5.5.5.1 with 32 bytes of data:
Reply from 5.5.5.1: bytes=32 time<1ms TTL=64
Reply from 5.5.5.1: bytes=32 time<1ms TTL=64
```

9 Windows Driver

The Microsoft Windows version supported by the Agilio series NIC are Windows Server 2012/2012R2/2016/2019/2022.

9.1 Driver Download

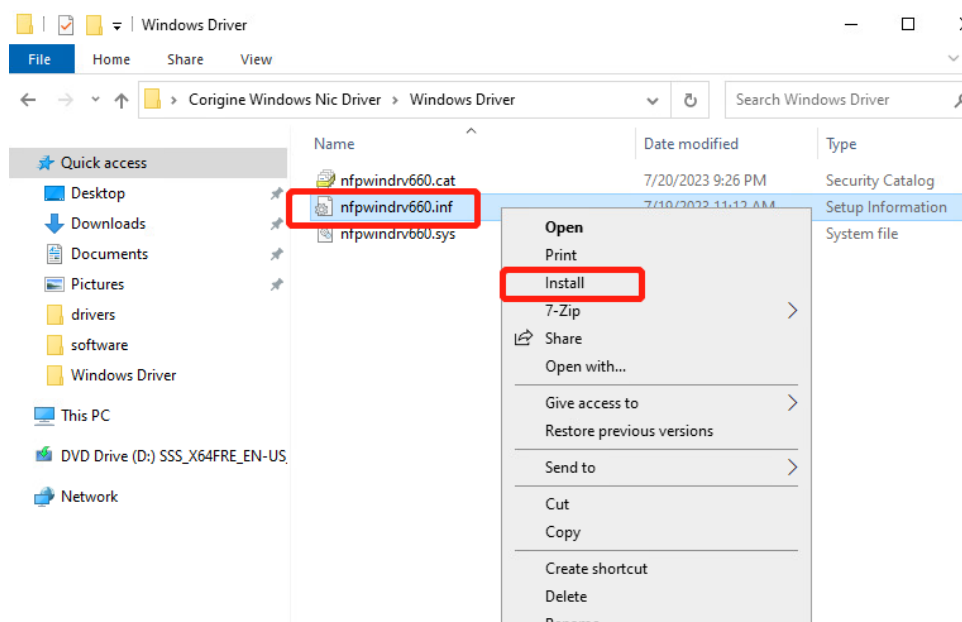
1. Visit [Corigine Support Site](#). Download the corresponding driver based on the OS version.
2. Decompress the driver file.

9.2 Driver Installation

Install the driver with .inf file, CMD line or Device Manager, choose one suitable for your environment.

1. Install the driver with the `nfpwindrv660.inf` file.

Right click `nfpwindrv660.inf` file in the decompressing driver directory, then click Install, wait for several seconds.



2. Install the driver at the CMD line

Enter the `pnputil.exe /add-driver .\nfpwindrv660.inf /install` into the CMD line selecting the Administrator privilege in the decompressing driver directory.

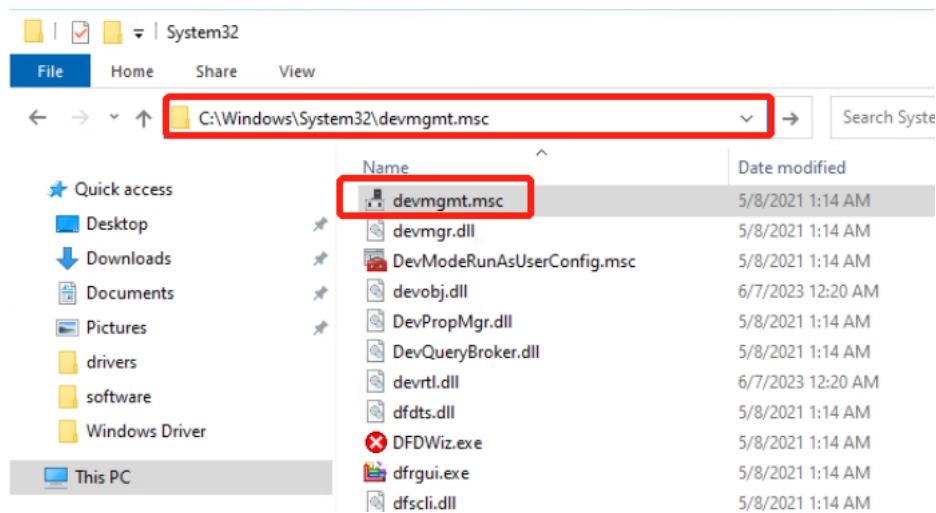
After the installation is complete, a message similar to the following is displayed in the CMD line:

```

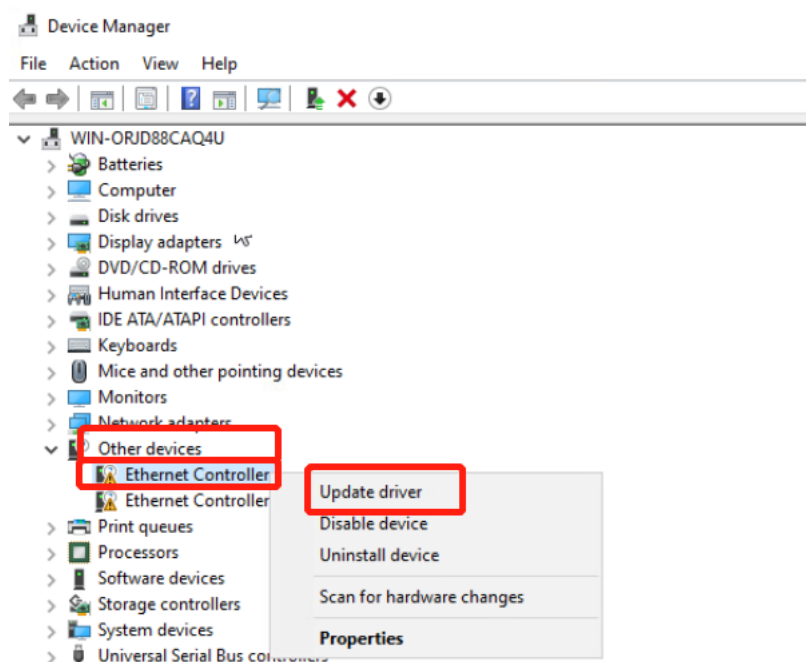
C:\Corigine-Windows-Driver>pnputil.exe /add-driver .\nfpwindrv660.inf /
↵install
Microsoft PnP Utility
Adding driver package: nfpwindrv660.inf
Driver package added successfully.
Published Name: oem3.inf
Driver package installed on device: PCI\VEN_1DA8&DEV_3800&SUBSYS_3800...
Driver package installed on device: PCI\VEN_1DA8&DEV_3800&SUBSYS_BFD5...
Total driver packages: 1
Added driver packages: 1
  
```

3. Install the driver with the Device Manager

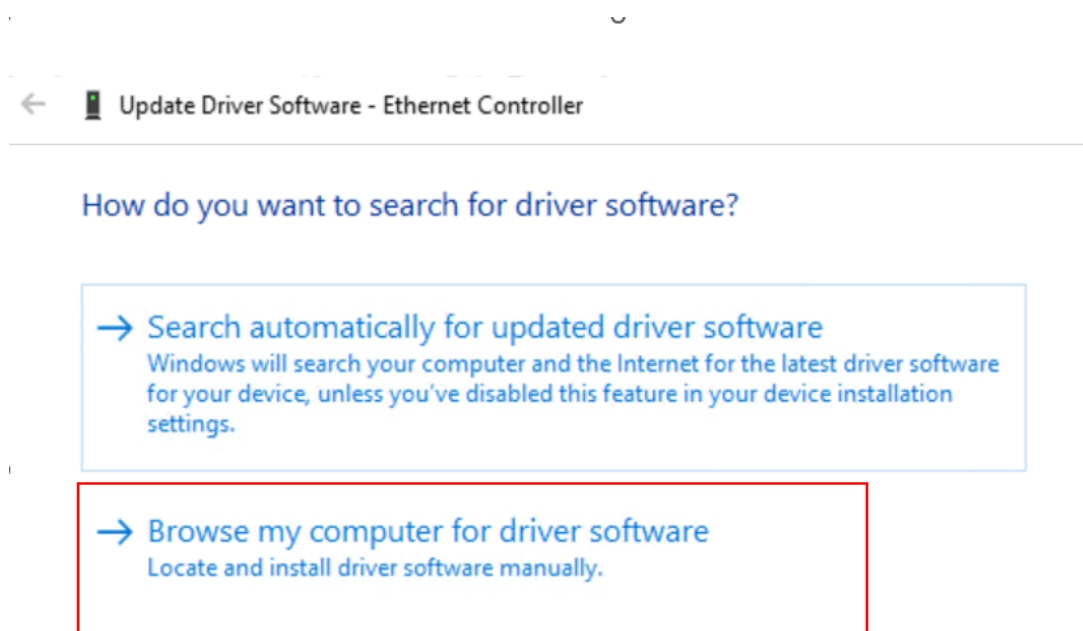
- a. Type `C:\Windows\System32\devmgmt.msc` in the File Explore address field to open Device Manager.



- b. Unfold Other devices, then right click Ethernet Controller. Choose Update Driver.



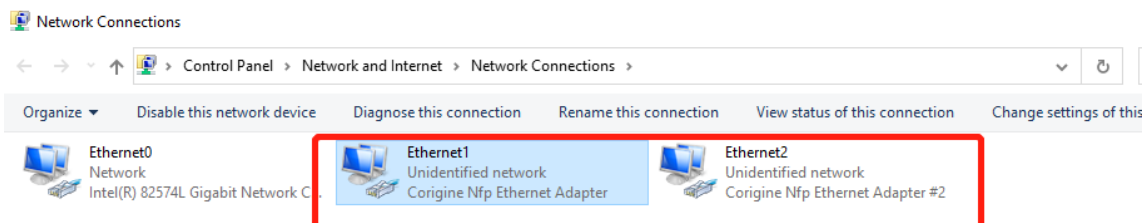
- c. Browse My computer for driver software and choose Windows-Driver-SRV directory after decompressing.



- d. The same installation should be acted on another Ethernet controller.

Check the Ethernet Adapter status

1. In the Network Connections, you will see two Corigine Nfp Ethernet Adapters if two physical ports connect suitably with the optical fiber.

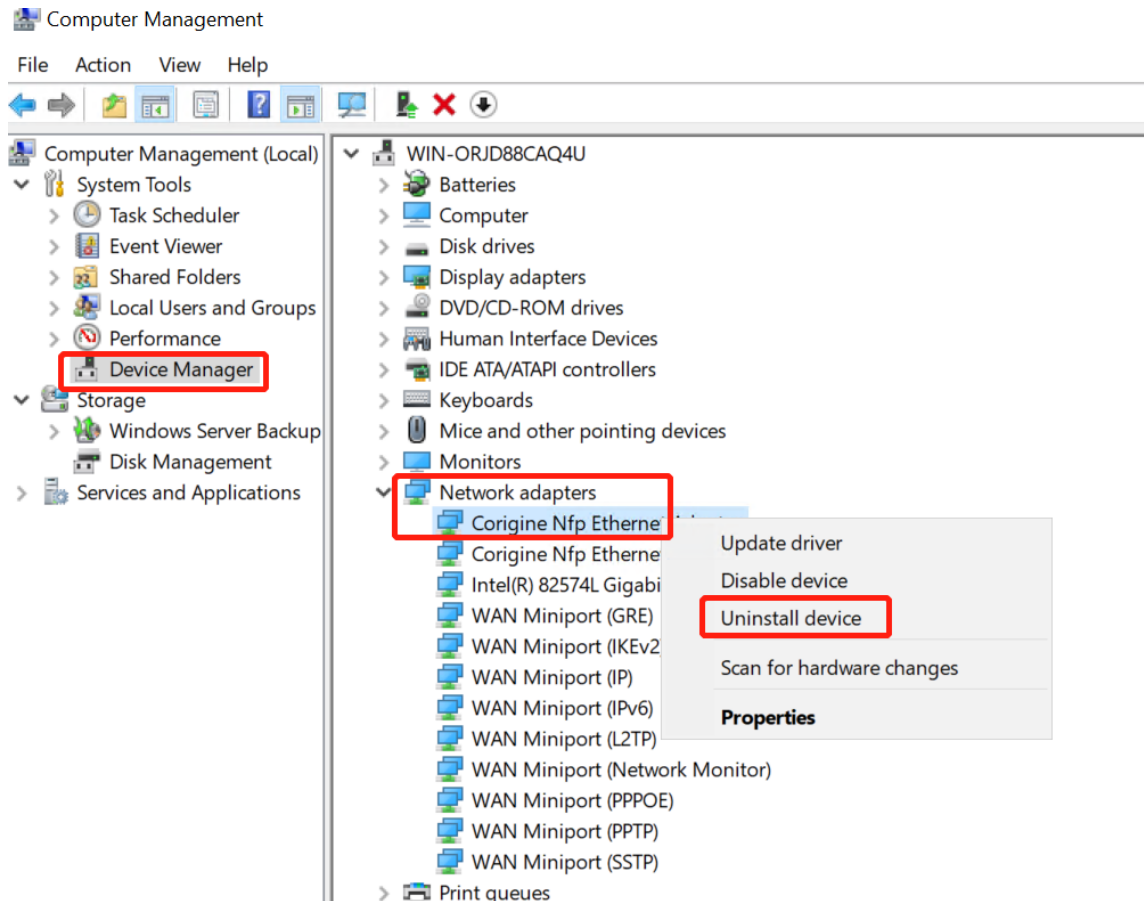


2. Or you can enter the `Get-NetAdapter` in the Powershell to check ethernet status Normal adapter status is following:

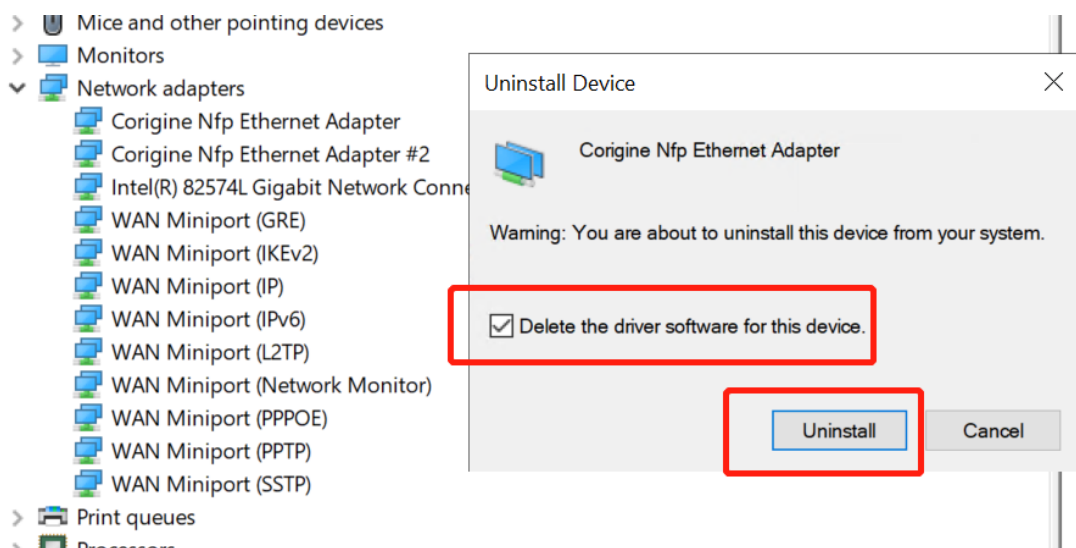
```
PS C:\Users\Administrator> Get-NetAdapter
Name      InterfaceDescription  ifIndex Status MacAddress      LinkSpeed
-----
Ethernet1 Corigine Ethernet Adapter  25 Up      88-3C-C5-A0-61-FB 10 Gbps
Ethernet2 Corigine Ethernet Adapter#2 29 Up      88-3C-C5-A0-61-FC 10 Gbps
```

9.3 Driver Uninstallation

1. Click the Computer Management. In the Device Manager, right click the Corigine Nfp Ethernet Adapter and choose the Uninstall device.



2. In the new window, please tick the Delete the driver software for this device and Uninstall finally.

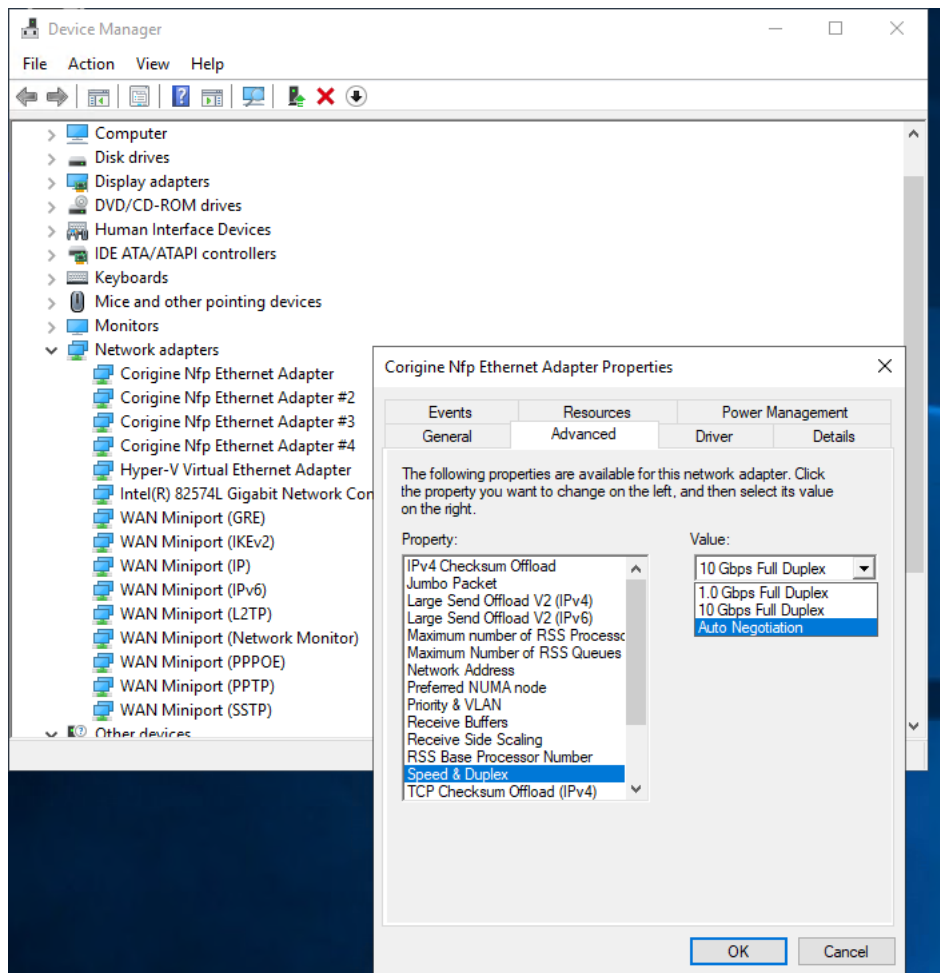


3. Uninstall another adapter with the same process.

9.4 Using the Windows Driver

9.4.1 Configuring Link-speed

Open Device Manager, select Network adapters and double-click to view its properties. Click the “Advanced” tab, select “Speed & Duplex” and the corresponding speed value.



9.4.2 Configuring FEC modes

Open Device Manager, select Network adapters and double-click to view its properties. Click the “Advanced” tab, select “FEC Mode” and the corresponding mode.

9.4.3 Configuring Interface MTU

Open Device Manager, select Network adapters and double-click to view its properties. Click the “Advanced” tab, select “Jumbo Packet” and the corresponding MTU value.

9.4.4 Confirming Connectivity

Open PowerShell and enter “Get-NetAdapter” to get the ifIndex of the port::

```
PS C:\Users\Administrator> Get-NetAdapter
```

Name	InterfaceDescription	ifIndex
Ethernet0	Intel(R) 82574L Gigabit Network Conn	22 Up
pciPassthru0	Corigine Nfp Ethernet Adapter #5	3 Up

Configure IP address:

```
New-NetIPAddress -InterfaceIndex 3 -IPAddress 5.5.5.5 -PrefixLength 24 -  
↔DefaultGateway 5.5.5.5.254
```

Ping the peer IP address:

```
PS C:\Users\Administrator> ping 5.5.5.1  
Pinging 5.5.5.1 with 32 bytes of data:  
Reply from 5.5.5.1: bytes=32 time<1ms TTL=64
```

10 Firmware Installation

10.1 Validating the Firmware

Corigine SmartNICs are fully programmable devices and thus depend on the driver to load firmware onto the device at runtime. It is important to note that the functionality of the SmartNIC significantly depends on the firmware loaded. The firmware files should be present in the following directory (contents may vary depending on the installed firmware):

```
# ls -ogR --time-style="+" /lib/firmware/netronome/
/lib/firmware/netronome/:
total 8
drwxr-xr-x. 2 4096 flower
drwxr-xr-x. 2 4096 nic
lrwxrwxrwx. 1 31 nic_AMDAA0081-0001_1x40.nffw -> nic/nic_AMDAA0081-0001_1x40.nffw
lrwxrwxrwx. 1 31 nic_AMDAA0081-0001_4x10.nffw -> nic/nic_AMDAA0081-0001_4x10.nffw
lrwxrwxrwx. 1 31 nic_AMDAA0096-0001_2x10.nffw -> nic/nic_AMDAA0096-0001_2x10.nffw
lrwxrwxrwx. 1 31 nic_AMDAA0097-0001_2x40.nffw -> nic/nic_AMDAA0097-0001_2x40.nffw
lrwxrwxrwx. 1 36 nic_AMDAA0097-0001_4x10_1x40.nffw -> nic/nic_AMDAA0097-0001_4x10_
↪1x40.nffw
lrwxrwxrwx. 1 31 nic_AMDAA0097-0001_8x10.nffw -> nic/nic_AMDAA0097-0001_8x10.nffw
lrwxrwxrwx. 1 36 nic_AMDAA0099-0001_1x10_1x25.nffw -> nic/nic_AMDAA0099-0001_1x10_
↪1x25.nffw
lrwxrwxrwx. 1 31 nic_AMDAA0099-0001_2x10.nffw -> nic/nic_AMDAA0099-0001_2x10.nffw
lrwxrwxrwx. 1 31 nic_AMDAA0099-0001_2x25.nffw -> nic/nic_AMDAA0099-0001_2x25.nffw
lrwxrwxrwx. 1 34 pci-0000:04:00.0.nffw -> flower/nic_AMDAA0097-0001_2x40.nffw
lrwxrwxrwx. 1 34 pci-0000:06:00.0.nffw -> flower/nic_AMDAA0096-0001_2x10.nffw

/lib/firmware/netronome/flower:
total 11692
lrwxrwxrwx. 1 17 nic_AMDAA0081-0001_1x40.nffw -> nic_AMDAA0097.nffw
lrwxrwxrwx. 1 17 nic_AMDAA0081-0001_4x10.nffw -> nic_AMDAA0097.nffw
lrwxrwxrwx. 1 17 nic_AMDAA0096-0001_2x10.nffw -> nic_AMDAA0096.nffw
-rw-r--r--. 1 3987240 nic_AMDAA0096.nffw
lrwxrwxrwx. 1 17 nic_AMDAA0097-0001_2x40.nffw -> nic_AMDAA0097.nffw
lrwxrwxrwx. 1 17 nic_AMDAA0097-0001_4x10_1x40.nffw -> nic_AMDAA0097.nffw
lrwxrwxrwx. 1 17 nic_AMDAA0097-0001_8x10.nffw -> nic_AMDAA0097.nffw
-rw-r--r--. 1 3988184 nic_AMDAA0097.nffw
lrwxrwxrwx. 1 17 nic_AMDAA0099-0001_2x10.nffw -> nic_AMDAA0099.nffw
lrwxrwxrwx. 1 17 nic_AMDAA0099-0001_2x25.nffw -> nic_AMDAA0099.nffw
-rw-r--r--. 1 3990552 nic_AMDAA0099.nffw

/lib/firmware/netronome/nic:
total 12220
```

(continues on next page)

(continued from previous page)

```
-rw-r--r--. 1 1380496 nic_AMDAA0081-0001_1x40.nffw
-rw-r--r--. 1 1389760 nic_AMDAA0081-0001_4x10.nffw
-rw-r--r--. 1 1385608 nic_AMDAA0096-0001_2x10.nffw
-rw-r--r--. 1 1385664 nic_AMDAA0097-0001_2x40.nffw
-rw-r--r--. 1 1391944 nic_AMDAA0097-0001_4x10_1x40.nffw
-rw-r--r--. 1 1397880 nic_AMDAA0097-0001_8x10.nffw
-rw-r--r--. 1 1386616 nic_AMDAA0099-0001_1x10_1x25.nffw
-rw-r--r--. 1 1385608 nic_AMDAA0099-0001_2x10.nffw
-rw-r--r--. 1 1386368 nic_AMDAA0099-0001_2x25.nffw
```

The NFP driver will search for firmware in `/lib/firmware/netronome`. Firmware is searched for in the following order and the first firmware to be successfully found and loaded is used by the driver:

```
1: serial-_SERIAL_.nffw
2: pci-_PCI_ADDRESS_.nffw
3: nic-_ASSEMBLY-TYPE___BREAKOUTxMODE_.nffw
```

This search is logged by the kernel when the driver is loaded. For example:

```
# dmesg | grep -A 4 nfp.*firmware
[ 3.260788] nfp 0000:04:00.0: nfp: Looking for firmware file in order of priority:
[ 3.260810] nfp 0000:04:00.0: nfp:   netronome/serial-00-15-4d-13-51-0c-10-ff.
↪nffw: not found
[ 3.260820] nfp 0000:04:00.0: nfp:   netronome/pci-0000:04:00.0.nffw: not found
[ 3.262138] nfp 0000:04:00.0: nfp:   netronome/nic_AMDAA0097-0001_2x40.nffw: found,
↪ loading...
```

The firmware found and used by the driver is indicated by the `found, loading...` tag. In the example above, the `nfp: netronome/nic_AMDAA0097-0001_2x40.nffw` firmware is used by the driver.

The version of the loaded firmware for a particular `<netdev>` interface, as found in [SmartNIC Netdev Interfaces](#) (for example `enp4s0`), or an interface's port `<netdev port>` (e.g. `enp4s0np0`) can be displayed with the `ethtool` command:

```
# ethtool -i <netdev>
driver: nfp
version: 3.10.0-862.el7.x86_64 SMP mod_u
firmware-version: 0.0.3.5 0.22 nic-2.0.4 nic
expansion-rom-version:
bus-info: 0000:04:00.0
```

Note: Replace `<netdev>` with the machine's specific interface associated with the SmartNIC's PF, which is expected to be something like `ens3np0` or `enp2s0np0`.

Firmware versions are displayed in order; NFD version, NSP version, APP FW version, driver APP. The specific output above shows that basic NIC firmware is running on the card, as indicated by "nic" in the `firmware-version` field.

10.2 Upgrading the firmware

The preferred method to upgrading Agilio firmware is via the Corigine repositories, however, if this is not possible, the corresponding installation packages can be obtained from Corigine Support (<https://www.corigine.com/DPUDownload.html>).

10.2.1 Upgrading firmware via the Corigine repository

Please refer to *Importing GPG-Key* and *Configuring Repositories* on how to configure the Corigine repository applicable to your distribution. When the repository has been successfully added, install the `agilio-nic-firmware` package using the commands below.

For Ubuntu:

```
# apt-get install agilio-nic-firmware
# rmmod nfp; modprobe nfp
# update-initramfs -u
```

For RHEL 7 and CentOS 7:

```
# yum install agilio-nic-firmware
# rmmod nfp; modprobe nfp
```

For RHEL 8 and CentOS 8:

```
# dnf install agilio-nic-firmware
# rmmod nfp; modprobe nfp
```

10.2.2 Upgrading Firmware from Package Installations

The latest firmware can be obtained at the downloads area of the Corigine Support site (<https://www.corigine.com/DPUDownload.html>).

Install the packages provided by Corigine Support using the commands below.

For Ubuntu:

```
# dpkg -i agilio-nic-firmware-*.deb
# rmmod nfp; modprobe nfp
# update-initramfs -u
```

For RHEL 7 and CentOS 7:

```
# yum install -y agilio-nic-firmware-*.rpm
# rmmod nfp; modprobe nfp
```

For RHEL 8 and CentOS 8:

```
# dnf install -y agilio-nic-firmware-*.rpm  
# rmmod nfp; modprobe nfp
```

11 BSP Installation

The Corigine Board Support Package (BSP) provides infrastructure software and a development environment for managing NFP based platforms.

11.1 Install Software from Corigine Repository

Please refer to *Importing GPG-Key* and *Configuring Repositories* on how to configure the Corigine repository applicable to your distribution. When the repository has been successfully added install the BSP package using the commands below.

RHEL 7 and CentOS 7:

```
# yum list available | grep nfp-bsp
# yum install nfp-bsp
# reboot
```

RHEL 8 and CentOS 8:

```
# dnf list available | grep nfp-bsp
# dnf install nfp-bsp
# reboot
```

Ubuntu:

```
# apt-cache search nfp-bsp
# apt-get install nfp-bsp
```

11.2 Install Software from DEB/RPM Package

11.2.1 Obtain Software

The latest BSP packages can be obtained at the downloads area of the Corigine Support site (<https://www.corigine.com/DPUDownload.html>).

11.2.2 Install the Prerequisite Dependencies

RHEL and CentOS Dependencies

The libftdi package is required to install BSP software, it can be installed from the EPEL repository. Install the EPEL repository by running:

```
# yum install -y epel-release
```

Then install the libftdi package by running:

```
# yum install -y libftdi
```

Ubuntu Dependencies

To install the BSP package dependencies on Ubuntu, run:

```
# apt-get install -y libjansson4 libftdi
```

11.2.3 NFP BSP Package

Install the NFP BSP package provided by Corigine Support.

RHEL 7 and CentOS 7 Install:

```
# yum install -y nfp-bsp*.rpm
```

RHEL 8 and CentOS 8 Install:

```
# dnf install -y nfp-bsp*.rpm
```

Ubuntu Install:

```
# dpkg -i nfp-bsp*.deb
```

11.3 Using BSP Tools

11.3.1 Enable CPP Access

The NFP has an internal Command Push/Pull (CPP) bus that allows debug access to the SmartNIC internals. CPP access allows user space tools raw access to chip internals and is required to enable the use of most BSP tools. Only the *out-of-tree* (OOT) driver allows CPP access.

Follow the steps from [Install Driver via Corigine Repository](#) to install the OOT NFP driver. After the `nfp` module has been built, load the driver with CPP access:

```
# depmod -a
# rmmod nfp
# modprobe nfp nfp_dev_cpp=1
```

To persist this option across reboots, several options are available; the distribution specific documentation, which can be found at [RHEL](#), [CentOS](#) and [Ubuntu](#), will detail that process more thoroughly. Care must be taken that the settings are also applied to any initramfs images generated.

11.3.2 Configure Media Settings

Alternatively to the process described in [Configuring Interface Media Mode](#), BSP tools can be used to configure the port speed of the SmartNIC using the following commands. Note, a reboot is still required for changes to take effect.

CX 2x25GbE - AMDA0099

To set the port speed of the CX 2x25GbE, the following commands can be used:

Set port 0 and port 1 to 10G mode:

```
# nfp-media phy1=10G phy0=10G
```

Set port 1 to 25G mode:

```
# nfp-media phy1=25G+
```

To change the FEC settings of the 2x25GbE, the following commands can be used:

```
nfp-media --set-aneg=phy0=[S|A|I|C|F] --set-fec=phy0=[A|F|R|N]
```

Where the parameters for each argument are:

--set-aneg=:

S

search - Search through supported modes until link is found. Only one side should be doing this. It may result in a mode that can have physical layer errors depending on SFP type and what the other end wants. Long DAC cables with no FEC will have physical layer errors.

A

auto - Automatically choose mode based on speed and SFP type.

C

consortium - Consortium 25G auto-negotiation with link training.

I

IEEE - IEEE 10G or 25G auto-negotiation with link training.

F

forced - Mode is forced with no auto-negotiation or link training.

--set-fec=:

A

auto - Automatically choose FEC based on speed and SFP type.

F

Firecode - BASE-R Firecode FEC compatible with 10G.

R

Reed-Solomon - Reed-Solomon FEC new for 25G.

N

none - No FEC is used.

CX 1x40GbE - AMDA0081

Set port 0 to 40G mode:

```
# nfp-media phy0=40G
```

Set port 0 to 4x10G fanout mode:

```
# nfp-media phy0=4x10G
```

CX 2x40GbE - AMDA0097

Set port 0 and port 1 to 40G mode:

```
# nfp-media phy0=40G phy1=40G
```

Set port 0 to 4x10G fanout mode:

```
# nfp-media phy0=4x10G
```

For mixed configuration the highest port must be in 40G mode e.g:

```
# nfp-media phy0=4x10G phy1=40G
```

11.4 Upgrade Flash Firmware

Agilio GX 2x10GbE card (AMDA2006) supports loading firmware from flash, and by default the firmware needs to be upgraded to the flash. First need to copy the firmware to the `/lib/firmware` directory, then run the following command to upgrade:

```
# ethtool -f <netdev port> <firmware>
```

The netdev port can be any port on NIC, reload NFP driver will load the new upgraded firmware.

12 Basic Performance Test

iPerf is a basic traffic generator and network performance measuring tool that can be used to quickly determine the throughput achievable by a device.

This basic performance test requires two machines, a server and a client, which are connected to each other, in order to run this test.

12.1 Install iPerf

iPerf needs to be installed on both the server and host machines.

Ubuntu:

```
# apt-get install -y iperf
```

For RHEL 7 or CentOS 7:

```
# yum install -y iperf
```

For RHEL 8 or CentOS 8:

```
# dnf install -y iperf
```

12.2 Run iPerf Test

12.2.1 Server

Run iPerf on the server:

```
# ip address add 10.0.0.1/24 dev <netdev>
# iperf -s
```

12.2.2 Client

Allocate an ip address on the same range as used by the server, then execute the following on the client to connect to the server and start running the test:

```
# iperf -c 10.0.0.1 -P 4
```

Example output of 1x40G link:


```
# iperf -c 10.0.0.1 -P 4
-----
Client connecting to 10.1, TCP port 5001 TCP window size: 85.0 KByte
(default)
-----
[5] local 10.0.0.2 port 56938 connected with 10.0.0.1 port 5001
[3] local 10.0.0.2 port 56932 connected with 10.0.0.1 port 5001
[4] local 10.0.0.2 port 56934 connected with 10.0.0.1 port 5001
[6] local 10.0.0.2 port 56936 connected with 10.0.0.1 port 5001
[ID] Interval Transfer Bandwidth
[6] 0.0-10.0 sec 11.9 GBytes 10.3 Gbits/sec
[3] 0.0-10.0 sec 9.85 GBytes 8.46 Gbits/sec
[4] 0.0-10.0 sec 11.9 GBytes 10.2 Gbits/sec
[5] 0.0-10.0 sec 10.2 GBytes 8.75 Gbits/sec
[SUM] 0.0-10.0 sec 43.8 GBytes 37.7 Gbits/sec
```

12.3 Using iPerf3

iPerf3 can also be used to measure performance, however multiple instances have to be chained to properly create multiple threads:

On the server:

```
# iperf3 -s -p 5001 & iperf3 -s -p 5002 & iperf3 -s -p 5003 &
iperf3 -s -p 5004 &
```

On the client:

```
# iperf3 -c 102.0.0.6 -i 30 -p 5001 & iperf3 -c 102.0.0.6 -i 30 -p 5002 &
iperf3 -c 102.0.0.6 -i 30 -p 5003 & iperf3 -c 102.0.0.6 -i 30 -p 5004 &
```

Example output:

```
[ID] Interval Transfer Bandwidth
[5] 0.00-10.04 sec 0.00 Bytes 0.00 bits/sec sender
[5] 0.00-10.04 sec 9.39 GBytes 8.03 Gbits/sec receiver
[5] 10.00-10.04 sec 33.1 MBytes 7.77 Gbits/sec
-----
[ID] Interval Transfer Bandwidth
[5] 0.00-10.04 sec 0.00 Bytes 0.00 bits/sec sender
[5] 0.00-10.04 sec 9.86 GBytes 8.44 Gbits/sec receiver
[5] 10.00-10.04 sec 53.6 MBytes 11.8 Gbits/sec
-----
[ID] Interval Transfer Bandwidth
[5] 0.00-10.04 sec 0.00 Bytes 0.00 bits/sec sender
[5] 0.00-10.04 sec 11.9 GBytes 10.2 Gbits/sec receiver
[5] 10.00-10.04 sec 42.1 MBytes 9.43 Gbits/sec
-----
[ID] Interval Transfer Bandwidth
[5] 0.00-10.04 sec 0.00 Bytes 0.00 bits/sec sender
```

(continues on next page)

(continued from previous page)

```
[5] 0.00-10.04 sec 10.2 GBytes 8.70 Gbits/sec receiver
Total: 37.7 Gbits/sec
95.49% of 40GbE link
```

Note: If the kernel version is earlier than 5.1 and iperf3 is used to test the NIC performance, the test result is not satisfactory. Please run the `sysctl -w net.ipv4.tcp_limit_output_bytes=1048576` command to modify system configurations.

13 Installing, Configuring and Using DPDK

13.1 Introduction to DPDK

The Data Plane Development Kit (DPDK) allows a user to bypass the Linux network stack within its Kernel space and allows DPDK libraries to communicate directly with a Network Flow Processor (NFP) from a DPDK application in the Linux Userspace. The Linux Userspace is typically reserved for applications software which will make use of various libraries to interact with the kernel itself. Kernel space consists of the operating system kernel, which handles interactions between hardware and software components, kernel extensions and device drivers. The Kernel space makes use of interrupts to notify the system that a packet has been received and then specially allocates a buffer for that packet. It then only frees that buffer once the packet is passed to Userspace. As more packets come in, more buffer memory is consumed. Resources are also consumed when switching between Kernel space and Userspace.

With DPDK, the direct communication between Userspace applications and a NFP is done using the Poll Mode Drivers (PMD) that continuously poll the NFP to check for new packets. It allows a user to map memory in Userspace applications to memory in the NFP, allowing for the PMD to poll for new packets coming into the NFP and immediately process them. Thus, DPDK provides a solution that minimizes consumption of resources as well as provides direct access to the NFP to process packets much faster.

13.2 Enabling IOMMU

In order to use the NFP device with DPDK applications, the Virtual Functions Input/Output (VFIO) module has to be loaded.

Firstly, the machine has to have the Input/Output Memory Management Unit (IOMMU) enabled. The following link: <http://dpdk-guide.gitlab.io/dpdk-guide/setup/binding.html> contains some generic information about binding devices including the possibility of using Userspace Input/Output (UIO) instead of VFIO, and also mentions the VFIO no-IOMMU mode.

Although DPDK focuses on avoiding interrupts, there is an option of a New Application Programming Interface (NAPI)-like approach using RX interrupts. This is supported by PMD NFP and with VFIO it is possible to have an RX interrupt per queue (with UIO just one interrupt per device). Because of this VFIO is the preferred option.

13.2.1 Edit Grub Configuration File

This is required for working with VFIO, however, when using kernels 4.5+, it is possible to work with VFIO and no-IOMMU mode. If your system comes with a kernel > 4.5, you can work with VFIO and no-IOMMU if desired by enabling this mode:

```
# echo 1 > /sys/module/vfio/parameters/enable_unsafe_noiommu_mode
```

For kernels older than 4.5, working with VFIO requires the enabling of IOMMU in the kernel at boot time. Add the following kernel parameters to `/etc/default/grub` to enable IOMMU:

```
GRUB_CMDLINE_LINUX="intel_iommu=on iommu=pt intremap=on"
```

It is worth noting that `iommu=pt` is not required for DPDK if VFIO is used, but it does avoid a performance impact in host drivers, such as the NFP netdev driver, when `intel_iommu=on` is enabled.

13.2.2 Implement Changes

Apply kernel parameters changes and reboot.

Ubuntu:

```
# update-grub2
# reboot
```

CentOS/RHEL:

```
# grub2-mkconfig -o /boot/grub2/grub.cfg
# reboot
```

13.3 DPDK Sources with PF PMD Support

13.3.1 Single-PF PMD Multi-port Support

While the NIC does not support multiple PF, it can run multiple ports with only single PF.

As mentioned in the introduction, the PMD polls a memory address within the NFP. The PF PMD multi-port support is necessary because the Userspace application needs access to the PF to poll for new packets coming in to the NFP.

The PMD can work with up to 8 ports on the same Physical Function (PF) device. The number of available ports is firmware and hardware dependent, and the driver looks for a firmware symbol during initialization to know how many can be used.

DPDK apps work with ports, and a port is usually a PF or a Virtual Function (VF) PCI device. However, with the NFP PF multi-port there is just one PF PCI device. Supporting this particular configuration requires the PMD to create ports in a special way, although once they are created, DPDK apps should be able to use them as normal PCI ports.

NFP ports belonging to same PF can be seen inside PMD initialization with a suffix added to the PCI ID: **www:xx:yy.z_portn**. For example, a PF with PCI ID 0000:03:00.0 and four ports is seen by the PMD code as:

```
0000:03:00.0_port0
0000:03:00.0_port1
0000:03:00.0_port2
0000:03:00.0_port3
```

Some DPDK applications can choose to use the MAC address to identify ports, OVS-DPDK is one such example, please refer to: <https://docs.openvswitch.org/en/latest/howto/dpdk/>

Note: There are some limitations with multi-port support: RX interrupts and device hot-plugging are not supported.

13.3.2 Multi-PF PMD Multi-port Support

With Multi-PF feature supported, the configuration is more simple. As the number of ports and PFs are 1:1 mapping, one NFP port belongs to one PF, the PCI ID can be directly used for a corresponding port in PMD. For example, port0 and port1 with their PCI ID:

```
0000:03:00.0
0000:03:00.1
```

13.4 Installing DPDK

For Agilio CX products, Physical Function (PF) PMD support has been upstreamed into *DPDK 18.11*. For Agilio GX products, PF PMD support has been upstreamed into *DPDK 22.11*. For Multi-PF feature, PF PMD support hasn't been upstreamed, we can provide a patch based on *DPDK 22.11*, and will upstream in a future version.

Install prerequisites for Ubuntu:

```
# apt-get -y install gcc libnuma-dev build-essential
# pip3 install meson ninja pyelftools
```

Install prerequisites for RHEL 7 or CentOS 7:

```
# yum -y group install "Development Tools"
# yum -y install numactl-devel
# pip3 install meson ninja pyelftools
```

Install prerequisites for RHEL 8 or CentOS 8:

```
# dnf -y group install "Development Tools"
# dnf -y install numactl-devel
# pip3 install meson ninja pyelftools
```

Obtain DPDK sources:

```
# cd /usr/src/
# wget http://fast.dpdk.org/rel/dpdk-22.11.tar.xz
# tar xf dpdk-22.11.tar.xz
# export DPDK_DIR=/usr/src/dpdk-22.11
# cd $DPDK_DIR
```

Configure and install DPDK:

```
# meson build
# ninja -C build
# ninja -C build install
```

13.5 Binding DPDK PF Driver

This section details the binding of DPDK-enabled drivers to the Physical Functions (PF) on your SmartNIC's NFP. The following commands detach the SmartNIC device from the Kernel space drivers and attach your SmartNIC's PFs to the VFIO-PCI driver in Userspace.

13.5.1 Attaching VFIO-PCI Driver

Load VFIO-PCI driver module:

- With SR-IOV support:

```
# sudo modprobe vfio-pci enable_sriov=1
```

- Without SR-IOV support:

```
# sudo modprobe vfio-pci
```

DPDK includes a user tool for binding devices to drivers in Userspace. To see the status of all your network ports, execute:

```
# sudo ./usertools/dpdk-devbind.py --status
```

The terminal should print an output that is similar to the output below:

```
Network devices using kernel driver
=====
0000:02:00.0 'Device 4000' if=ens6np1,ens6np0 drv=nfp unused=vfio-pci
0000:03:00.0 'Ethernet Controller XL710 for 40GbE QSFP+ 1584' if=ens4 drv=i40e_
```

(continues on next page)

(continued from previous page)

```
↔unused=vfio-pci  
...
```

The output shows which PCI ID the NFP belongs to (in this case, 0000:02:00.0).

To bind the NFP's PF to the VFIO-PCI driver in Userspace, use the command with it's corresponding PCI ID by replacing XXXX:XX:XX.X below:

```
# sudo ./usertools/dpdk-devbind.py --bind=vfio-pci XXXX:XX:XX.X
```

13.5.2 Confirm Attached Driver

Confirm that the driver has been attached, type the command:

```
# sudo ./usertools/dpdk-devbind.py --status
```

This command returns the output of the Kernel driver in use as shown:

```
Network devices using DPDK-compatible driver  
=====  
0000:02:00.0 'Device 4000' drv=vfio-pci unused=nfp  
  
Network devices using kernel driver  
=====  
0000:03:00.0 'Ethernet Controller XL710 for 40GbE QSFP+ 1584' if=ens4 drv=i40e_  
↔unused=vfio-pci  
...
```

13.5.3 Unbind Driver

To unbind VFIO-PCI driver:

```
# sudo ./usertools/dpdk-devbind.py --bind=nfp XXXX:XX:XX.X
```

13.6 Using DPDK PF Driver

13.6.1 Create Default Symlink

Note: This workaround applies to DPDK versions lower than version 18.05.

In order to use the PF in DPDK applications, a symlink named `nic_dpdk_default.nffw` pointing to the applicable firmware needs to be created e.g.

Navigate to firmware directory:

```
# cd /lib/firmware/netronome
```

For CX 2x40G:

```
# cp -s nic_AMD0097-0001_2x40.nffw nic_dpdk_default.nffw
```

For CX 2x25G:

```
# cp -s nic_AMD0099-0001_2x25.nffw nic_dpdk_default.nffw
```

For CX 2x40G w/ first port in breakout mode:

```
# cp -s nic_AMD0097-0001_4x10_1x40.nffw nic_dpdk_default.nffw
```

The following table can be used to map product names to their codes:

SmartNIC	Code
CX 2x25G	AMDA0099
CX 2x40G	AMDA0097

14 Using SR-IOV

Single Root I/O Virtualization (SR-IOV) is a PCI feature that allows Virtual Functions (VFs) to be created from a Physical Function (PF). The VFs thus share the resources of a PF, while VFs remain isolated from each other. The isolated VFs are typically assigned to Virtual Machines (VMs) on the host. In this way, the VFs allow the VMs to directly access the PCI device, thereby bypassing the host kernel.

14.1 Installing the SR-IOV Capable Firmware

Before installing the SR-IOV capable firmware, ensure that SR-IOV is enabled in the BIOS of the host machine. If SR-IOV is disabled or unsupported by the motherboard/chipset being used, the kernel message log will contain a `PCI SR-IOV: -12` error when trying to create a VF at a later stage. This can be queried using the `dmesg` tool.

The firmware currently running on the SmartNIC card can be determined by the `ethtool` command. As an example, CentOS Stream 8 contains the following upstreamed firmware:

```
# ethtool -i <netdev> | head -3
driver: nfp
version: 5.15.2
firmware-version: 0.0.3.5 0.31 nic-2.1.16.1 nic
```

Note: Replace `<netdev>` with the machine's specific interface associated with the SmartNIC's PF, which is expected to be something like `ens3np0` or `enp2s0np0`.

From the above output, the upstreamed firmware is `nic-2.1.16.1`. The prefix `nic` indicates that the firmware implements the Agilio CoreNIC functionality. The suffix `2.1.16.1` indicates the firmware version.

Firmware `sriov-2.1.x` or greater provides SR-IOV capability. There are two methods in which the firmware can be obtained, either from the `linux-firmware` package or from the support site.

14.1.1 The Linux-Firmware Package

The SR-IOV capable firmware has been upstreamed into the `linux-firmware` package. For `rpm` packages, this is available from `linux-firmware 20181008-88` version and onwards. For Ubuntu, the `linux-firmware` package contains SR-IOV capable firmware from version 20.04.

Ensure that the latest `linux-firmware` package is installed.

For Ubuntu run:

```
# apt update linux-firmware
```

For RHEL or Fedora or CentOS run:

```
# yum update linux-firmware
```

The linux-firmware package will store the Corigine firmware files in the `/lib/firmware/netronome` directory. This directory contains symbolic links which point to the actual firmware files. The actual firmware files will be located in subdirectories, with each subdirectory related to a different SmartNIC functionality. Consider the following tree structure:

```
# tree /lib/firmware/netronome
/lib/firmware/netronome/
├── flower
│   ├── nic_AMDA0058-0011_2x40.nffw -> nic_AMDA0058.nffw
│   ├── nic_AMDA0058-0001_4x10.nffw -> nic_AMDA0058.nffw
│   └── ...
├── nic
│   ├── nic_AMDA0058-0011_2x40.nffw
│   ├── nic_AMDA0058-0012_2x40.nffw
│   └── ...
├── nic-sriov
│   ├── nic_AMDA0058-0011_2x40.nffw
│   ├── nic_AMDA0058-0012_2x40.nffw
│   └── ...
├── nic_AMDA0058-0011_2x40.nffw -> nic/nic_AMDA0058-0011_2x40.nffw
├── nic_AMDA0058-0012_2x40.nffw -> nic/nic_AMDA0058-0012_2x40.nffw
└── ...
```

As can be seen from the tree structure, three functionalities (`flower`, `nic` and `nic-sriov`) are supplied by the linux-firmware package. If `nic-sriov` is missing, follow [The Support Site](#) method below. If `nic-sriov` is present, point the symbolic links to the specific application required, in this case `nic-sriov`, with the following command and thereafter continue to [Load Firmware to SmartNIC](#):

```
# ln -sf /lib/firmware/netronome/nic-sriov/* /lib/firmware/netronome/
```

Symbolic links can be confirmed with the following command where `sriov` is included on the right hand side:

```
# ls -og --time-style="+" /lib/firmware/netronome
...
lrwxrwxrwx 1 64 nic_AMDA0058-0011_2x40.nffw
-> /opt/netronome/agilio-sriov-firmware/nic_AMDA0058-0011_2x40.nffw
...
```

14.1.2 The Support Site

The SR-IOV capable firmware can be obtained from the Corigine website in the [DPU Software Download](#) page, found under the Support and Services tab. This chapter takes the V22.04 release of the package as an example.

If `wget` is installed, the firmware can be downloaded as follows:

For Ubuntu:

```
# BASE=https://download.corigine.com.cn/public
# VERSION=apt/pool/main/a/agilio-sriov-firmware-all
# wget -nc ${BASE}/${VERSION}/agilio-sriov-firmware-22.04-4_all.deb
```

For RHEL or Fedora or CentOS:

```
# BASE=https://download.corigine.com.cn/public
# wget -nc ${BASE}/packages/agilio-sriov-firmware-22.04-4.noarch.rpm
```

For other:

```
# BASE=https://download.corigine.com.cn/public
# wget -nc ${BASE}/tgz/agilio-sriov-firmware-22.04-4.tgz
```

After downloading the packaged firmware, install the firmware files.

For Ubuntu:

```
# dpkg -i agilio-sriov-firmware-22.04-4_all.deb
```

For RHEL or Fedora or CentOS:

```
# yum -y install agilio-sriov-firmware-22.04-4.noarch.rpm
```

For other:

```
# tar -zxvf agilio-sriov-firmware-22.04-4.tgz
```

The `/lib/firmware/netronome` directory contains symbolic links which point to the actual firmware files. When installing the above firmware package, the symbolic links are automatically updated to point to the new SR-IOV capable firmware files. An exception if the *other* option was used for installation, then the links need to be manually created with the following command:

```
# ln -sf /lib/firmware/netronome/nic-sriov/* /lib/firmware/netronome/
```

Symbolic links can be confirmed with the following command where `sriov` is included on the right hand side:

```
# ls -og --time-style="+ " /lib/firmware/netronome
...
lrwxrwxrwx 1 64 nic_AMDAA0058-0011_2x40.nffw
```

(continues on next page)

(continued from previous page)

```
-> /opt/netronome/agilio-sriov-firmware/nic_AMD0058-0011_2x40.nffw
...
```

14.2 Load Firmware to SmartNIC

Remove and reload the driver. The driver will subsequently install the new firmware to the SmartNIC card:

```
# rmmmod nfp
# modprobe nfp
```

Note: For multi-PF, remove the driver and wait for 3 seconds before reloading the driver to verify that the correct firmware has been loaded.

The `ethtool` command can be used to verify that the correct firmware has been loaded onto the SmartNIC card:

```
# ethtool -i <netdev> | head -3
driver: nfp
version: 5.15.2
firmware-version: 0.0.3.5 0.31 sriov-22.04.3 nic
```

Notice that the firmware has successfully changed from `nic-2.1.16.1` to `sriov-22.04.3`.

14.3 Configuring SR-IOV

At this stage, there is still zero VFs (assuming only one Corigine SmartNIC card is installed). This is seen with the `lspci -kd <vendor ID>` command where each entry starts with an address in the format of `Bus:Dev.Fn`. For `<vendor ID>` of `19ee` use:

```
# lspci -kd 19ee:
02:00.0 Ethernet controller: Netronome Systems, Inc. Device 4000
    Subsystem: Netronome Systems, Inc. Device 4001
    Kernel driver in use: nfp
    Kernel modules: nfp
```

And for `<vendor ID>` of `1da8` use:

```
# lspci -kd 1da8:
17:00.0 Ethernet controller: Corigine, Inc. Device 3800
    Subsystem: Corigine, Inc. Device 7ff5
    Kernel driver in use: nfp
    Kernel modules: nfp
```

Note: If multiple Corigine SmartNIC cards are installed, the information of the additional cards, each with a different bus number, will appear below one another when `lspci -kd <vendor ID>` is run.

The number of supported VFs on an interface (`netdev`), associated with the SmartNIC's PF, is exposed by `sriov_totalvfs` in `sysfs`. The following command will return the total supported number of VFs:

```
# cat /sys/class/net/<netdev>/device/sriov_totalvfs
48
```

Note: Replace `<netdev>` with the machine's specific interface number associated with the SmartNIC's PF, which is expected to be something like `ens3np0` or `enp2s0np0`. In multi-PF setup, all PFs share the total VF number supported by firmware, thus the number of supported VFs of each PF is configurable, please contact us if it's required to change.

VFs can be allocated to a network interface by writing an integer to the `sysfs` file. For example, to allocate two VFs to `enp2s0np0`, run:

```
# echo 2 > /sys/class/net/enp2s0np0/device/sriov_numvfs
```

The newly created VFs, together with the PF, can be observed with the `lspci` command:

```
# lspci -kd <vendor ID>:
02:00.0 Ethernet controller: Netronome Systems, Inc. Device 4000
    Subsystem: Netronome Systems, Inc. Device 4001
    Kernel driver in use: nfp
    Kernel modules: nfp
02:08.0 Ethernet controller: Netronome Systems, Inc. Device 6003
    Subsystem: Netronome Systems, Inc. Device 4001
    Kernel driver in use: nfp_netvf
    Kernel modules: nfp
02:08.1 Ethernet controller: Netronome Systems, Inc. Device 6003
    Subsystem: Netronome Systems, Inc. Device 4001
    Kernel driver in use: nfp_netvf
    Kernel modules: nfp
```

Note: Replace `<vendor ID>` with the PCI vendor identifier. For SmartNICs with Board Support Package (BSP) version before 22.09, use `19ee` as the `<vendor ID>` and for SmartNICs with AMDA2XXX product codes, with a BSP version of at least 22.09, use `1da8` as the `<vendor ID>`.

In the example above, the PF is located at PCI address `02:00.0`. The two VFs are located at `02:08.0` and `02:08.1`. Notice that the VFs are identified by `Device 6003`, and that they use the `nfp_netvf` kernel driver. However, for RHEL 7.x systems, the VFs will use the NFP driver.

If one of the following errors occur during the allocation of VFs to a network interface, it means that the virtualization is not enabled in the BIOS setup. Errors are either one of the following:

```
Error: bind failed for 0000:81:00.4 - Cannot bind to driver vfio-pci
```

```
Error: write error: Cannot allocate memory
```

Then it is necessary to reboot the machine and enter its BIOS setup. Each BIOS may have a different name for the SR-IOV setting that needs to be enabled. Here are some examples that may require enabling: - Advanced -> Integrated IO Configuration -> Intel(R) VT for Directed IO - Advanced -> PCIe/PCI/PnP Configuration -> SR-IOV Support - Integrated Devices -> SR-IOV Global Enable

Note: In single-PF setup, if the SmartNIC has more than one physical port (phyport), the VFs will appear to be connected to all the phyports (as reported by the `ip link` command). This happens due to the PF being shared among all VFs. In practice the VFs are only connected to phyport 0.

In order to persist the VFs on the system, which means to auto recreate the VFs on reboot, it is suggested that the system's networking scripts are updated to manage the VFs. The following script can be run to persist the VFs with the help of *NetworkManager* for a specific PF by using the PF's <netdev> number:

```
#!/bin/sh
cat > /etc/init.d/vf-init << 'EOF'
#!/bin/sh
ip link set mtu 9532 dev <netdev>
ip link set up dev <netdev>
cat /sys/class/net/<netdev>/device/sriov_totalvfs > \
/sys/class/net/<netdev>/device/sriov_numvfs
EOF
chmod u+x /etc/init.d/vf-init
```

Note: Replace <netdev> with the machine's specific interface number associated with the SmartNIC's PF, which is expected to be something like `ens3np0` or `enp2s0np0`.

SR-IOV VFs cannot be reallocated dynamically. In order to change the number of allocated VFs, existing functions must first be deallocated by writing a 0 to the `sysfs` file. Otherwise, the system will return a `device or resource busy error`.

Note: Ensure that any VMs are shut down and that applications which may be using the VFs are stopped before deallocation.

Deallocating VFs:

```
# echo 0 > /sys/class/net/<netdev>/device/sriov_numvfs
```

PCI passthrough passes the VF to the VM which makes the VM think that the card is directly attached

to the PCI port. To enable the PCI passthrough, use `nano` or `vi` to edit the kernel command line in the `/etc/default/grub` file. Edit the command line in the file by adding the parameters `intel_iommu=on` `iommu=pt` to the existing command line with:

```
GRUB_CMDLINE_LINUX_DEFAULT="console=tty1 console=ttyS0,115200 intel_iommu=on  
↵ iommu=pt "
```

Apply kernel parameters:

For Ubuntu:

```
# update-grub2
```

For CentOS or RHEL:

```
# grub2-mkconfig -o /boot/grub2/grub.cfg
```

Ensure that the `/boot/grub/grub.cfg` file is updated with the aforementioned parameters:

```
# reboot
```

After reboot, confirm that the kernel has been started with the parameters:

```
# cat /proc/cmdline  
BOOT_IMAGE=/boot/vmlinuz-4.15.0-20-generic  
root=UUID=179b45a3-def2-48b0-8f2f-7a5b6b3f913b  
ro  
console=tty1  
console=ttyS0,115200  
intel_iommu=on  
iommu=pt
```

14.4 Using Virtio-Forwarder

Virtio-forwarder is a userspace networking application that forwards bi-directional traffic between SR-IOV VFs and virtio networking devices in QuickcEMUlator (QEMU) virtual machines. Virtio-forwarder implements a virtio backend driver using the DPDK's vhost-user library and services designated VFs by means of the DPDK poll mode driver (PMD) mechanism.

The steps shown here closely correlate with the comprehensive [virtio-forwarder docs](#). Ensure that the [Requirements](#) are met and that the setup of the above has been completed.

14.4.1 Installing Virtio-Forwarder

For Ubuntu:

```
# add-apt-repository ppa:netronome/virtio-forwarder
# apt-get update
# apt-get install virtio-forwarder
```

For RHEL or Fedora or CentOS:

```
# yum install yum-plugin-copr
# yum copr enable netronome/virtio-forwarder
# yum install virtio-forwarder
```

Virtio-forwarder makes use of the DPDK library, therefore DPDK has to be installed. Ensure DPDK is installed when the following command returns something similar:

```
# /usr/src/dpdk-22.03/usertools/dpdk-devbind.py -s
Network devices using DPDK-compatible driver
=====
0000:02:00.0 'Device 4000' drv=vfio-pci unused=nfp

Network devices using kernel driver
=====
0000:01:00.0 'MT27700 Family [ConnectX-4] 1013' if=ens2f0np0 drv=mlx5_core_
↳unused=vfio-pci
0000:01:00.1 'MT27700 Family [ConnectX-4] 1013' if=ens2f1np1 drv=mlx5_core_
↳unused=vfio-pci
...
```

Otherwise, carry out the instructions of [Installing DPDK](#).

14.4.2 Configuring Hugepages

Hugepages are a contiguous space reserved in memory to make it possible for the operating system (OS) to support memory pages greater than the default. This can help with performance when reading or writing to memory.

For Ubuntu, modify libvirt's apparmor permissions to allow read/write access to the hugepages directory and library files for QEMU. Add the following lines to the end of `/etc/apparmor.d/abstractions/libvirt-qemu` using `vi` or `nano`:

```
/tmp/virtio-forwarder/** rwmix,
# for latest QEMU
/usr/lib/x86_64-linux-gnu/qemu/* rmix,
# for access to hugepages
owner "/dev/hugepages/libvirt/qemu/**" rw,
owner "/dev/hugepages-1G/libvirt/qemu/**" rw,
```


For Ubuntu, also edit the existing line, such that:

```
/tmp/{,**} r,
```

For Ubuntu, restart the apparmor service:

```
# systemctl restart apparmor.service
```

For virtio-forwarder, 2M hugepages are required whereas QEMU/KVM performs better with 1G hugepages. It is recommended that at least 1375 pages of 2M be reserved for virtio-forwarder. The hugepages can be configured during boot time, for which the Linux kernel command line parameters should be edited. Use `nano` or `vi` to add the following parameters to the existing kernel command line parameters in the `/etc/default/grub` file at the end of the `GRUB_CMDLINE_LINUX=` line:

```
hugepagesz=2M hugepages=1375 default_hugepagesz=1G hugepagesz=1G hugepages=8
```

Alternatively, hugepages can be configured manually after each boot. Reserve at least $1375 * 2M$ for virtio-forwarder:

```
# echo 2048 > /sys/kernel/mm/hugepages/hugepages-2048kB/nr_hugepages
```

Reserve 8G for application hugepages (modify this as needed):

```
# echo 8 > /sys/kernel/mm/hugepages/hugepages-1048576kB/nr_hugepages
```

Since non-fragmented memory is required for hugepages, it is recommended that hugepages be configured during boot time.

`hugetlbfs` needs to be mounted on the file system to allow applications to create and allocate handles to the mapped memory. The following lines mount the two types of hugepages on `/dev/hugepages` (2M) and `/dev/hugepages-1G` (1G):

```
# grep hugetlbfs /proc/mounts | grep -q "pagesize=2M" || \
(mkdir -p /dev/hugepages && \
mount nodev -t hugetlbfs -o rw,pagesize=2M /dev/hugepages/)
# grep hugetlbfs /proc/mounts | grep -q "pagesize=1G" || \
(mkdir -p /dev/hugepages-1G && \
mount nodev -t hugetlbfs -o rw,pagesize=1G /dev/hugepages-1G/)
```

Verify that Hugepages are set up correctly with:

```
# cat /proc/meminfo | grep Huge
AnonHugePages:      18432 kB
ShmemHugePages:     0 kB
FileHugePages:      0 kB
HugePages_Total:    8
HugePages_Free:     8
HugePages_Rsvd:     0
HugePages_Surp:     0
```

(continues on next page)

(continued from previous page)

```
Hugepagesize:    1048576 kB
Hugetlb:        14020608 kB
```

Finally, `libvirt` requires a special directory inside the hugepages mounts with the correct permissions in order to create the necessary per-VM handles:

```
# mkdir /dev/hugepages-1G/libvirt
# mkdir /dev/hugepages/libvirt
# chown qemu:kvm -R /dev/hugepages-1G/libvirt
# chown qemu:kvm -R /dev/hugepages/libvirt
```

Note: If the above `chown qemu:kvm -R ...` commands do not work, try to use `chown libvirt-qemu:kvm -R /dev/hugepages-1G/libvirt` and `chown libvirt-qemu:kvm -R /dev/hugepages/libvirt` or check what users exist on the system with `awk -F: '{ print $1}' /etc/passwd` and use the applicable user in the `qemu` place.

Note: Substitute `/dev/hugepages[-1G]` with your actual hugepage mount directory. A 2M hugepage mount location is created by default by some distributions.

Restart the `libvirt` daemon:

```
# systemctl restart libvirtd
```

To check that hugepages are correctly reserved for each page size, the `hugeadm` utility can be used if `libhugetlbfs-utils` is installed:

```
# hugeadm --pool-list
```

	Size	Minimum	Current	Maximum	Default
	2097152	2048	2048	2048	*
	1073741824	8	8	8	

14.4.3 Binding to VFIO-PCI

Since the VFs need to communicate directly with `virtio-forwarder`, a pass-through style driver, such as `vfio-pci` is required. The `vfio-pci` module is the preferred driver, compared to `uio_pci_generic` and `igb_uio`, of which the former lacks SR-IOV compatibility whereas the latter is considered outdated.

Note: The following commands use the `<vendor ID>:<device tuple>` from the previous example of `19ee:6003`. Please use the vendor ID and device tuple of your VF as determined by the section [Configuring SR-IOV](#).

First, unbind the VF PCI devices from their current drivers:

```
# lspci -Dd 19ee:6003 | awk '{print $1}' | xargs -I{} echo \  
"echo {} > /sys/bus/pci/devices/{}/driver/unbind;" | bash
```

The VFs which now have their drivers unbound, can be observed with the `lspci` command:

```
# lspci -kd 19ee:  
02:00.0 Ethernet controller: Netronome Systems, Inc. Device 4000  
    Subsystem: Netronome Systems, Inc. Device 4001  
    Kernel driver in use: nfp  
    Kernel modules: nfp  
02:08.0 Ethernet controller: Netronome Systems, Inc. Device 6003  
    Subsystem: Netronome Systems, Inc. Device 4001  
    Kernel modules: nfp  
02:08.1 Ethernet controller: Netronome Systems, Inc. Device 6003  
    Subsystem: Netronome Systems, Inc. Device 4001  
    Kernel modules: nfp
```

Notice that the `Kernel driver in use` attribute was removed for the VFs. To bind the `vfio-pci` driver to the VFs, first load the `vfio-pci` driver to the Linux kernel:

```
# modprobe vfio-pci
```

Then bind the driver to the VFs:

```
# echo 19ee 6003 > /sys/bus/pci/drivers/vfio-pci/new_id
```

The VFs are now bound to the `vfio-pci` driver:

```
# lspci -kd 19ee:  
02:00.0 Ethernet controller: Netronome Systems, Inc. Device 4000  
    Subsystem: Netronome Systems, Inc. Device 4001  
    Kernel driver in use: nfp  
    Kernel modules: nfp  
02:08.0 Ethernet controller: Netronome Systems, Inc. Device 6003  
    Subsystem: Netronome Systems, Inc. Device 4001  
    Kernel driver in use: vfio-pci  
    Kernel modules: nfp  
02:08.1 Ethernet controller: Netronome Systems, Inc. Device 6003  
    Subsystem: Netronome Systems, Inc. Device 4001  
    Kernel driver in use: vfio-pci  
    Kernel modules: nfp
```

14.4.4 Launching Virtio-Forwarder

In this guide, the use case will be virtio-forwarder acting as a server. This means virtio-forwarder will create and host the sockets to which VMs can connect at a later stage. To configure virtio-forwarder as the server, edit `/etc/default/virtioforwarder` with `vi` or `nano` so that `VIRTIOFWD_VHOST_CLIENT` is assigned a blank value:

```
# Non-blank enables vhostuser client mode (default: server mode)
VIRTIOFWD_VHOST_CLIENT=
```

The virtio-forwarder service can be configured to start during boot time:

```
# systemctl enable virtio-forwarder
Created symlink ...
```

To manually start the service after installation, run:

```
# systemctl start virtio-forwarder
```

To check the status of virtio-forwarder, run:

```
# systemctl status virtio-forwarder
```

Note: A DPDK version of at least 21.11.1 is needed for the virtio-forwarder to work optimally.

14.4.5 Adding VF Ports to Virtio-Forwarder

Modify socket permissions:

```
# chown -R qemu:kvm /tmp/virtio-forwarder/
```

Note: If the above `chown -R qemu:kvm ...` command do not work, try to use `chown chown -R libvirt-qemu:kvm /tmp/virtio-forwarder/` or check what users exist on the system with `awk -F: '{ print $1}' /etc/passwd` and use the applicable user in the `qemu` place.

Dynamically map the PCI address of each VF to virtio-forwarder. The PCI address is the B:D.F number which can be seen as the first part of the `lspci` command:

```
# lspci -nd 19ee:
02:00.0 0200: 19ee:4000
02:08.0 0200: 19ee:6003
02:08.1 0200: 19ee:6003
```

Note: Alternatively, use the vendor ID of `1da8` in the place of `19ee` when using SmartNICs with

AMDA2XXX product codes that have a Board Support Package (BSP) version of at least 22.09.

The second and third PCI addresses above are the PCI addresses of the VFs, which can be referred to as <B:D.F of VF1> and <B:D.F of VF2>. Dynamically map the PCI address of each VF to virtio-forwarder then as follows:

```
# /usr/lib/virtio-forwarder/virtioforwarder_port_control.py add \  
--virtio-id 1 --pci-addr <B:D.F of VF1>  
status: OK  
# /usr/lib/virtio-forwarder/virtioforwarder_port_control.py add \  
--virtio-id 2 --pci-addr <B:D.F of VF2>  
status: OK
```

The `virtio-id` parameter is compulsory and denotes the id of the relay through which traffic is routed. A relay can accept only a single PCI device and a single VM.

The VF ports added to virtio-forwarder can be confirmed with:

```
# /usr/lib/virtio-forwarder/virtioforwarder_stats.py \  
--include-inactive | grep DPDK_ADDED  
relay_1.vf_to_vm.internal_state=DPDK_ADDED  
relay_2.vf_to_vm.internal_state=DPDK_ADDED
```

The VF ports can be removed in a similar fashion:

```
# /usr/lib/virtio-forwarder/virtioforwarder_port_control.py remove \  
--virtio-id 1 --pci-addr <B:D.F of VF1>  
status: OK  
# /usr/lib/virtio-forwarder/virtioforwarder_port_control.py remove \  
--virtio-id 2 --pci-addr <B:D.F of VF2>  
status: OK
```

Note: Replace <B:D.F of VF1> and <B:D.F of VF2> with the respective VF's Bus:Dev.Fn (B:D.F) address previously seen with the `lspci` command.

It is useful to watch the virtio-forwarder journal while adding or removing ports:

```
# journalctl -fu virtio-forwarder
```

The VF entries can also be modified statically within the `/etc/default/virtioforwarder` file. Consult the [virtio-forwarder docs](#) for more information.

14.4.6 Modify Guest VM XML Files

The snippets in this section should be inserted in each VM's XML file.

The following snippet configures the connection between the VM and the virtio-forwarder service. Note that `virtio-forwarder1.sock` refers to `virtio-id 1` and `relay_1`. The MAC address should be assigned the value of the specific VF to be paired with the VM. If left unassigned, libvirt will assign a random MAC address which will cause the VM's traffic to be rejected by the SmartNIC. The PCI address is internal to the VM and can be chosen arbitrarily, but should be unique within the VM itself.

```
<devices>
<interface type='vhostuser'>
  <mac address='1e:a3:32:f8:3e:83'/>
  <source type='unix' path='/tmp/virtio-forwarder/virtio-forwarder1.sock' mode=
  ↪ 'client'/>
  <model type='virtio'/>
  <alias name='net1'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x06' function='0x0'/>
</interface>
</devices>
```

The VM also has to be configured to make use of the 1G hugepages that was reserved for this purpose:

```
<memoryBacking>
<hugepages>
  <page size='1048576' unit='KiB' nodeset='0'/>
</hugepages>
</memoryBacking>
```

Allocate CPUs and memory to the VM. It is especially important to specify `memAccess='shared'`, as this allows the host and guest VM to share RAM. This is required by virtio-forwarder to write the packets to the VM.

```
<cpu mode='custom' match='exact'>
<model fallback='allow'>SandyBridge</model>
<feature policy='require' name='ssse3'/>
<numa>
  <cell id='0' cpus='0-1' memory='3670016' unit='KiB' memAccess='shared'/>
</numa>
</cpu>
```

The VMs can now be booted. Observing the host's CPU usage (e.g. `htop`) will show that some of the cores will be utilized to the maximum (polling mechanism). The default number of cores dedicated for virtio-forwarder is 2, and can be adjusted in `/etc/default/virtioforwarder` by modifying the `VIRTIOFWD_CPU_MASK` value.

15 Using RoCEv2

15.1 Introduction to RDMA/RoCEv2

RDMA is a technology that facilitates direct memory access remotely, enabling user mode requests to access peer memory directly through hardware. This access bypasses the peer CPU and is directly handled by the NIC. Compared with traditional TCP/IP network RDMA has the following performance advantages:

- Zero copy - Data is directly transferred from user mode to hardware without copying through kernel mode
- Kernel bypass - The data path is completed between user mode and hardware, and no longer experiences context switching to kernel mode
- CPU is not disturbed - Read/write peer memory does not require the peer CPU to participate, all packet encapsulation and parsing, memory handling are completed by hardware

These features allow RDMA's end-to-end latency to be reduced to the microsecond level, while also greatly reducing the CPU load. RDMA itself can be overlaid on 4 types of transport layers: InfiniBand, Ethernet (RoCEv1), UDP/IP (RoCEv2) and TCP/IP (iWARP). The InfiniBand transport layer requires proprietary hardware, while the others can use Ethernet.

RoCE is a network protocol defined in the InfiniBand Trade Association (IBTA) standard, which allows the use of RDMA over an Ethernet network. That is, RoCE allows high-performance, low-latency, and high-throughput data transmission over Ethernet. It does not require the use of special network hardware equipment and can be achieved with standardized Ethernet equipment, greatly reducing costs (High-performance, low-latency, and high-throughput data transmission can be achieved through Ethernet adapters and switches that support RDMA without frequent CPU involvement during the transmission process). RoCE technology follows the same transport mechanism as InfiniBand, allowing network adapters to directly access host memory, significantly reducing latency and increasing throughput.

For comparison, RoCEv1, which is based on the network link layer, can only communicate within the L2 subnet. Thus, it has very limited applications. RoCEv2 is based on UDP and can be deployed on L3 networks. It has good scalability, and can achieve relatively good throughput and latency, so it is a solution adopted on a large scale.

15.2 Installing

15.2.1 Requirements

The following table shows the Requirements of RoCEv2 feature for the Agilio SmartNICs.

Requirements	Description
Agilio GX 2x10G SmartNIC	AMDA0145-0002 AMDA0145-0012 AMDA0145-1001 AMDA0145-1012 AMDA2001-1001 AMDA2001-1002 AMDA2001-1103 AMDA2001-1104 AMDA2001-1113 AMDA2001-1114 AMDA2001-1123 AMDA2001-1124 AMDA2001-1133 AMDA2001-1134
Agilio GX 2x25G SmartNIC	AMDA2000-1001 AMDA2000-1002 AMDA2000-1103 AMDA2000-1113 AMDA2000-1104 AMDA2000-1114
Platform	x86_64
Operating System	CentOS 8.5 (4.18.0-348.el8)
BSP	23.07.0 and later
Kernel Driver	23.10.0 and later
Userspace Lib	23.10.0 and later
Firmware	23.10.0 and later

15.2.2 Install Userspace Lib via Software Package

The *Userspace Lib* packages can be provided separately through technical support.

```
# tar -xvf rdma-core-35.0-23.10.0.tgz
# cd x86_64/
# yum localinstall -y ./libibverbs-35.0-1.el8.x86_64.rpm
# yum localinstall -y ./libibverbs-utils-35.0-1.el8.x86_64.rpm
# yum localinstall -y ./libibumad-35.0-1.el8.x86_64.rpm
# yum localinstall -y ./librdmacm-35.0-1.el8.x86_64.rpm
# yum localinstall -y ./librdmacm-utils-35.0-1.el8.x86_64.rpm
# yum localinstall -y ./infiniband-diags-35.0-1.el8.x86_64.rpm
# yum localinstall -y ./infiniband-diags-compatible-35.0-1.el8.x86_64.rpm
# yum localinstall -y ./rdma-core-35.0-1.el8.x86_64.rpm
# yum localinstall -y ./rdma-core-devel-35.0-1.el8.x86_64.rpm
```

If the following output message is encountered during the installation of any of the packages above, it indicates that a different version is already installed.

```
# yum localinstall -y ./libibverbs-35.0-1.el8.x86_64.rpm
Last metadata expiration check: 0:45:13 ago on Mon 18 Sep 2023 03:03:06 PM CST.
Package libibverbs-35.0-1.el8.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
```

Then use the following command to install the specified version, substituting the name of the package that the message above was received for.

```
# yum reinstall -y ./libibverbs-35.0-1.el8.x86_64.rpm
```

15.2.3 Upgrading Firmware from Package Installations

The *firmware* for RoCEv2 can be provided separately through technical support.

```
# yum -y install agilio-roce-firmware-23.10.0-rc0.noarch.rpm
```

Please refer to [Validating the Firmware](#) to make sure that correct firmware is used after drivers have been loaded.

15.2.4 Install Kernel Driver via Software Package

The *Kernel Driver* packages can be provided separately through technical support.

```
# yum -y install agilio-rdma-driver-dkms-23.10.0-rc0.noarch.rpm
```

The package installs two kernel driver modules, `nfp` and `crdma`. Use `modinfo` command to confirm:

```

# modinfo nfp
filename:          /lib/modules/4.18.0/extra/nfp.ko
nfp_build_path:   /var/lib/dkms/agilio-rdma-driver/23.10.0-rc0/build/deps/nfp_drv.
↳git/src
nfp_build_host:   Server15
nfp_build_user:   root
nfp_build_user_id: root
nfp_src_path:     /var/lib/dkms/agilio-rdma-driver/23.10.0-rc0/build/deps/nfp_drv.
↳git/src/
nfp_src_version: 29fd1741 (o-o-t)
version:          6.5.0
description:      The Network Flow Processor (NFP) driver.
license:          GPL
author:           Corigine, Inc. <oss-drivers@corigine.com>
...
rhelversion:     8.5
srcversion:      4C6103B2E1A0741B9FD5F2D
...
parm:            nfp_roce_ints_num:Number of RoCE interrupt vectors (default 4)↳
↳(uint)
parm:            nfp_roce_enabled:Enable RoCE interface registration (default =↳
↳False) (bool)
...

# modinfo crdma
filename:          /lib/modules/4.18.0/extra/crdma.ko
version:          0.5
license:          Dual BSD/GPL
description:      Corigine NFP RoCEv2 HCA provider driver
author:           Corigine Inc.
rhelversion:     8.5
srcversion:      32C67171263E65A5D8647F8
depends:          ib_core,nfp,ib_uverbs
name:            crdma
vermagic:        4.18.0 SMP mod_unload modversions
...
parm:            mad_cq_event_wa:Enables a temporary work around to support QP1↳
↳interface testing while ucode CQ event notification is not implemented (default:↳
↳false (bool)
parm:            send_loopback:A development test only parameter to set the↳
↳internal loop-back flag on kernel QP post sends (default: false) (bool)
parm:            have_interrupts:During bring-up, allows selective use of event↳
↳driven command mode (default: false) (bool)
parm:            dcqcn_enable:During bring-up, allows selective use of setting↳
↳dcqcn enable (default: false) (bool)

```

Next load the installed kernel driver modules. `crdma` module depends on `nfp` module.

```

// if crdma or nfp has been already loaded, unload it first
# rmmod crdma

```

(continues on next page)

(continued from previous page)

```
# rmmmod nfp
// need add `nfp_roce_enabled=1` parameter to nfp driver
# modprobe nfp nfp_roce_enabled=1
# modprobe crdma
```

Once the drivers are loaded, use the `rdma` command to check if the RDMA links have been created. This also verifies that the RoCEv2 feature of the Agilio SmartNIC is operational. The following shell output indicates that it's operational. We're assuming that port 0 is active and linked up.

```
# rdma link
link crdma0/1 state ACTIVE physical_state LINK_UP netdev ens6f0np0
link crdma1/1 state DOWN physical_state DISABLED netdev ens6f1np1
```

15.2.5 Confirm BSP Version

To ensure that RoCEv2 works as expected, ensure that the BSP version meets the required version. After drivers have been loaded, use the `dmesg` command to check it:

```
# lspci -bDnnd 19ee:
# lspci -bDnnd 1da8:
0000:02:00.0 Ethernet controller [0200]: Corigine, Inc. Network Flow Processor_
↔3800 [1da8:3800]
0000:02:00.1 Ethernet controller [0200]: Corigine, Inc. Network Flow Processor_
↔3800 [1da8:3800]
# dmesg | grep 0000:02:00 | grep BSP
[ 119.709993] nfp 0000:02:00.0: BSP: 23.07-0
[ 131.818201] nfp 0000:02:00.1: BSP: 23.07-0
```

Note: If the bsp version doesn't meet the requirements, please contact [Corigine support](#).

15.3 Using and Basic Testing

Perftest is a set of test programs written based on uverbs, which is a benchmark related to RDMA performance, and can be used for software and hardware tuning and functional testing.

15.3.1 Install Perftest

```
# yum install -y perftest
```

15.3.2 Run Test

Note: Perftest contains several test commands. For RoCEv2 on Agilio SmartNICs, only the commands below are currently supported.

`ib_send_lat`: latency test with send transactions

`ib_send_bw`: bandwidth test with send transactions

`ib_write_lat`: latency test with RDMA write transactions

`ib_write_bw`: bandwidth test with RDMA write transactions

`ib_read_lat`: latency test with RDMA read transactions

`ib_read_bw`: bandwidth test with RDMA read transactions

Here is an example using command `ib_send_bw`.

Run test on the server:

```
# ip address add 10.1.1.200/24 dev <netdev>
# ip link set dev <netdev> up
# ip link set dev <netdev> mtu 9000
# ib_send_bw -a -d crdma0 -F --report_gbits -q 2
```

Run test on the client. Allocate an IP address in the same range as used by the server, then execute the following on the client to connect to the server and start running the test:

```
# ip address add 10.1.1.100/24 dev <netdev>
# ip link set dev <netdev> up
# ip link set dev <netdev> mtu 9000
# ib_send_bw -a -d crdma0 10.1.1.200 -F --report_gbits -q 2
```

The test result outputs on server as follows:

```
# ib_send_bw -a -d crdma0 -F --report_gbits -q 2

*****
* Waiting for client to connect... *
```

(continues on next page)

```

*****
-----
                        Send BW Test
Dual-port      : OFF          Device      : crdma0
Number of qps  : 2           Transport type : IB
Connection type : RC         Using SRQ    : OFF
PCIe relax order: Unsupported
ibv_wr* API    : OFF
TX depth       : 128
RX depth       : 512
CQ Moderation  : 100
Mtu            : 4096[B]
Link type      : Ethernet
GID index      : 1
Max inline data : 0[B]
rdma_cm QPs    : OFF
Data ex.method : Ethernet
-----

local address: LID 0000 QPN 0x0002 PSN 0x843d1d
GID: 00:00:00:00:00:00:00:00:00:00:255:255:10:01:01:200
local address: LID 0000 QPN 0x0003 PSN 0x2ff07a
GID: 00:00:00:00:00:00:00:00:00:00:255:255:10:01:01:200
remote address: LID 0000 QPN 0x0002 PSN 0x29ae4a
GID: 00:00:00:00:00:00:00:00:00:00:255:255:10:01:01:100
remote address: LID 0000 QPN 0x0003 PSN 0x551b6c
GID: 00:00:00:00:00:00:00:00:00:00:255:255:10:01:01:100
-----

#bytes      #iterations      BW peak[Gb/sec]      BW average[Gb/sec]      MsgRate[Mpps]
2           2000                0.000000             0.004733                 0.295819
4           2000                0.000000             0.009445                 0.295143
8           2000                0.000000             0.018922                 0.295649
16          2000                0.000000             0.037734                 0.294800
32          2000                0.000000             0.075721                 0.295784
64          2000                0.00                0.15                     0.294702
128         2000                0.00                0.30                     0.293482
256         2000                0.00                0.60                     0.293498
512         2000                0.00                1.20                     0.294117
1024        2000                0.00                2.43                     0.296196
2048        2000                0.00                4.81                     0.293786
4096        2000                0.00                8.93                     0.272470
8192        2000                0.00                9.79                     0.149412
16384       2000                0.00                9.80                     0.074799
32768       2000                0.00                9.81                     0.037411
65536       2000                0.00                9.81                     0.018712
131072      2000                0.00                9.81                     0.009357
262144      2000                0.00                9.81                     0.004679
524288      2000                0.00                9.81                     0.002340
1048576     2000                0.00                9.81                     0.001170
2097152     2000                0.00                9.81                     0.000585

```

(continued from previous page)

4194304	2000	0.00	9.81	0.000292
8388608	2000	0.00	9.81	0.000146

16 Using IPSec

16.1 Introduction to IPSec/IPSec VPN

IPSec (Internet Protocol Security) is a set of protocols and services designed to provide security for IP networks. It is a widely adopted VPN (Virtual Private Network) technology. When IP packets are transmitted on a public network (such as the Internet), there is a lack of effective security mechanisms and risks such as forgery, theft, and tampering. To address this issue, communication parties establish an IPSec tunnel, which encrypts IP packets during transmission, thereby ensuring their security.

VPN is a technology that establishes a private network on the public network. It is a logical network based on a public network, enabling user data to be transmitted through logical links, differing from traditional dedicated networks where user data travels through end-to-end physical links.

IPSec VPN establishes secure tunnels between hosts, between hosts and network security gateways, or between network security gateways (such as routers and firewalls), protecting point-to-point communication. Operating at the IP layer, it encrypts and authenticates data packets. Due to the encryption of data in IPSec tunnels, IPSec VPN offers higher security compared to other VPN technologies, although its configuration and deployment can be more complex.

16.2 IPSec Process

IPSec performs functions through the following 4 stages:

- **Traffic Identification:** After receiving a packet, network devices match the packet's 5-tuple against configured IPSec policies to determine if the packet needs to be transmitted through an IPSec tunnel. The traffic that requires IPSec tunneling is referred to as interesting traffic.
- **SA (Security Association) Negotiation:** SA defines the elements required for secure data transmission between the two communicating parties, including security protocols, encapsulation modes, encryption and authentication algorithms, and keys for data transmission. Once local network devices identify the interesting traffic, they initiate SA negotiation with the remote network device. During this stage, both parties use the IKE (Internet Key Exchange) protocol to establish an IKE Security Association for identity authentication and key exchange, followed by setting up an IPSec Security Association to ensure data security.
- **Data Transfer:** Upon establishing the IPSec Security Association, the two parties can transmit data through the IPSec tunnel. To secure the data transmission, AH (Authentication Header) or ESP (Encapsulating Security Payload) can be used to encrypt and authenticate the data. Encryption ensures data confidentiality, preventing interception during transmission, while authentication guarantees data integrity and reliability, protecting against forgery or tampering. Modern ESP encapsulations typically include an ICV for authentication data.

- Tunnel Teardown: In most cases, the aging of the session (session teardown) between the two communicating parties indicates that their data exchange has completed. To conserve system resources, the IPSec tunnel automatically disconnects after the tunnel idle timeout period.

16.3 IPSec Offloading

The purpose of IPSec offloading is to shift part of the IPSec protocol processing from the host CPU to SmartNIC.

Two aspects of IPSec need to be offloaded onto the NIC:

- Encryption/Decryption

SmartNICs contain hardware acceleration for encryption and decryption. Leveraging this hardware to perform the IPSec-required encryption and decryption functions improves overall performance, reduces total power consumption, and frees up host processor resources.

- ESP Protocol

The SmartNIC's stream processing engine handles encapsulation and decapsulation of original IPSec packets (ESP), further enhancing performance and freeing up host processor resources.

The remaining parts of IPSec processing, particularly key generation and exchange (IKE), establishment and teardown of secure connections, and maintenance of the security policy database (SPD), continue to be executed by the host processor.

16.4 IPSec Features

The following IPSec feature offloads are supported:

- ESP Protocol Operating Modes
 - Tunnel, Transport
- Cipher Algorithms
 - NULL, AES-128, AES-192, AES-256, 3DES (Agilio GX unsupported)
- Cipher Modes
 - ECB, CBC, CFB, OFB, CTR
- Authentication Hashes
 - SHA1-96, SHA256-96, SHA384-96, SHA512-96, SHA1-80, SHA256-128, SHA384-192, SHA512-256, MD5 (Agilio GX unsupported)
- AEAD Algorithms
 - AES-GCM, CHACHA20-POLY1305 (Agilio CX unsupported)
- ESN Generation
- Anti-replay
- NAT-T

- Up to 16K simultaneous SAs
- Dual-stack IPv4/IPv6

16.5 Requirements

The following table shows the requirements of IPsec offloading feature for the Agilio SmartNICs.

Requirements	Description
Agilio GX 2x10G SmartNIC	AMDA0145-0002 AMDA0145-0012 AMDA0145-1001 AMDA0145-1012 AMDA2001-1001 AMDA2001-1002 AMDA2001-1104 AMDA2001-1114 AMDA2001-1124 AMDA2001-1134
Agilio GX 2x25G SmartNIC	AMDA2000-1001 AMDA2000-1002 AMDA2000-1103 AMDA2000-1104 AMDA2000-1113 AMDA2000-1114
BSP	23.07.0 and later
Kernel Driver	23.10.0 and later
Firmware	23.10.0 and later

16.6 IPsec Driver and Firmware Installation

Please refer to *Installing the Linux Driver* and *Firmware Installation*. Wherein the firmware name is agilio-ipsec-firmware.

16.7 DPDK Installation and Configuration for IPsec

Please refer to *Installing, Configuring and Using DPDK*.

16.8 Kernel-based IPsec Offloading

16.8.1 Kernel Version

Due to the fact that Linux kernels starting from v4.18 and above have XFRM kernel modules that support IPsec offloading, it is necessary to use a Linux system with a kernel version no lower than 4.18 for IPsec offloading testing.

16.8.2 Environment Setup

Typically, a SmartNIC comes with two physical optical ports. For simplicity in testing, these two network ports can be isolated into separate network namespaces using Namespace technology, and then directly connected to each other for testing purposes.

SA can be configured in two ways: IKE negotiation or via static configuration tools. IKE negotiations are commonly performed using open-source applications such as StrongSwan or Libreswan. Static configuration tools utilize the iproute2 suite to configure XFRM policy. In this test, the method of statically configuring SA policy will be adopted.

16.8.3 Configuration Instructions

Refer to the usage of the ip-xfrm command at: <https://www.man7.org/linux/man-pages/man8/ip-xfrm.8.html>

In the following test steps, parameters used include:

```
mode = transport
enc = aes
auth = hmac(sha256)
```

16.8.4 Basic Testing

Load the driver:

```
# modprobe nfp nfp_dev_cpp=1 nfp_reset=1
```

Configure Namespace and IP address:

```
# ifconfig enp1s0np0 192.168.1.1 netmask 255.255.255.0
# ip netns add ns1
# ip link set netns ns1 enp1s0np1
# ip netns exec ns1 ifconfig enp1s0np1 192.168.1.2 netmask 255.255.255.0
```

Configure enp1s0np0:

```
# ip xfrm policy add src 192.168.1.1/32 dst 192.168.1.2/32 dir out tmpl src 192.
↪168.1.1 dst 192.168.1.2 proto esp mode transport reqid 07222000 level required
# ip xfrm policy add src 192.168.1.2/32 dst 192.168.1.1/32 dir in tmpl src 192.168.
↪1.2 dst 192.168.1.1 proto esp mode transport reqid 01234567 level required
# ip xfrm state add src 192.168.1.1 dst 192.168.1.2 proto esp spi 0x3172002 reqid↪
↪07222000 mode transport enc aes 0x0123456789abcdef0123456789abcdef auth
↪"hmac (sha256) " 0x0123456789abcdef0123456789abcdef offload dev enp1s0np0 dir out
# ip xfrm state add src 192.168.1.2 dst 192.168.1.1 proto esp spi 0x5170522 reqid↪
↪01234567 mode transport enc aes 0x0123456789abcdef0123456789abcdef auth
↪"hmac (sha256) " 0x0123456789abcdef0123456789abcdef offload dev enp1s0np0 dir in
```

Configure enp1s0np1:

```
# ip xfrm policy add src 192.168.1.2/32 dst 192.168.1.1/32 dir out tmpl src 192.
↪168.1.2 dst 192.168.1.1 proto esp mode transport reqid 07222000 level required
# ip xfrm policy add src 192.168.1.1/32 dst 192.168.1.2/32 dir in tmpl src 192.168.
↪1.1 dst 192.168.1.2 proto esp mode transport reqid 01234567 level required
# ip xfrm state add src 192.168.1.2 dst 192.168.1.1 proto esp spi 0x5170522 reqid↪
↪07222000 mode transport enc aes 0x0123456789abcdef0123456789abcdef auth
↪"hmac (sha256) " 0x0123456789abcdef0123456789abcdef offload dev enp1s0np1dir out
# ip xfrm state add src 192.168.1.1 dst 192.168.1.2 proto esp spi 0x3172002 reqid↪
↪01234567 mode transport enc aes 0x0123456789abcdef0123456789abcdef auth
↪"hmac (sha256) " 0x0123456789abcdef0123456789abcdef offload dev enp1s0np1dir in
```

View configured policy and status:

```
# ip xfrm policy
# ip xfrm state
```

Use iPerf3 to test the performance:

Server:

```
# iperf3 -s -i 1
```

Client:

```
# iperf3 -c 192.168.1.2 -P 4 -t 100
```

16.9 DPDK-based IPsec Offloading

16.9.1 DPDK Version

NFP's PMD incorporates IPsec offloading in DPDK 23.11 and later, so it is necessary to use a DPDK environment of no lower than v23.11 for IPsec offloading testing.

16.9.2 Environment Deployment

In a DPDK environment, IKE negotiation and IPSec-related processing are both handled by its application, ipsec-secgw. The ipsec-secgw application serves as an example of applications within the DPDK cryptODEV framework. This application supports inline IPSec processing during transmission over Ethernet devices and allows configuring SA and SP (Security Policy) through configuration files.

In order to utilize the entire encryption and decryption process, this test network uses two SmartNICs. As shown in the figure below, Port 0 of SmartNIC 1 is connected to the tester or packet sending device, Port 1 of SmartNIC 1 is connected to Port 0 of SmartNIC 2, and Port 1 of SmartNIC 2 is connected to the tester or packet sending device. Among them, Port 1 of SmartNIC 1 and Port 0 of SmartNIC 2 are respectively configured with IPSec policies for encryption and decryption.

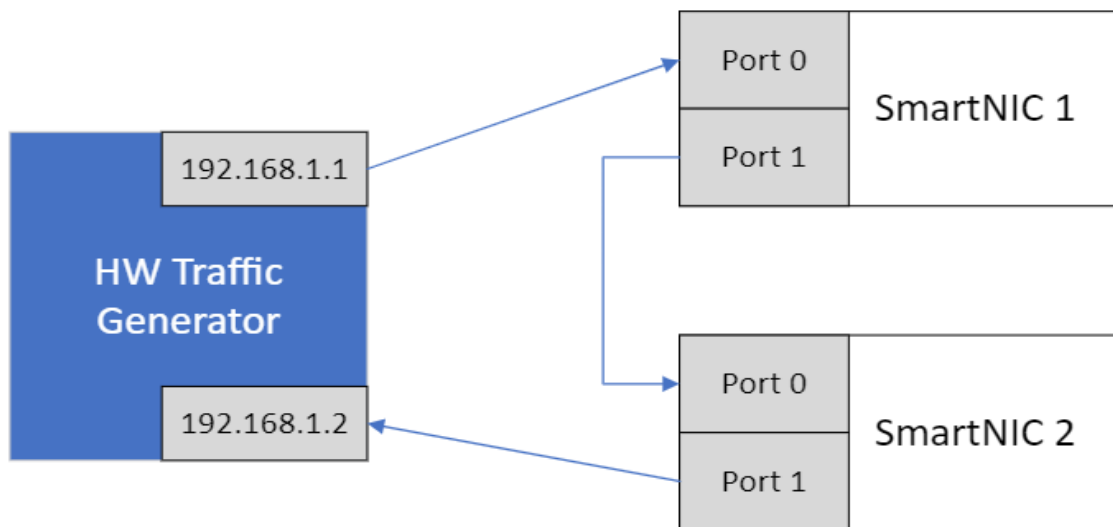


Fig. 1: IPSec DPDK test network

When the tester sends a message with a source address of 192.168.1.1 and a destination address of 192.168.1.2, Port 0 of SmartNIC 1 receives the message and forwards it to Port 1 according to the configured policy and completes the IPSec encryption. Port 0 of SmartNIC 2 receives the IPSec message and decrypts it according to the configured policy and forwards it to Port 1, and finally returns to the tester to complete the whole IPSec encryption/decryption test process. Note that the application ipsec-secgw may not be able to complete ARP learning, you need to manually configure a static ARP table to send IP messages to the other end.

16.9.3 Configuration Instructions

Refer to the usage of ipsec-secgw at: https://doc.dpdk.org/guides/sample_app_ug/ipsec_secgw.html

The configuration is as follows:

```
# sp ipv4 out esp protect 2 pri 2 src 198.168.1.0/24 dst 198.168.1.0/24 sport_
↳0:65535 dport 0:65535
# sa out 1 cipher_algo aes-128-cbc cipher_key_
↳de:ad:be:ef:de:ad:be:ef:de:ad:be:ef:de:ad:be:ef auth_algo sha1-hmac auth_key_
↳de:ad:be:ef:de:ad:be:ef:de:ad:be:ef:de:ad:be:ef:de:ad:be:ef mode transport port_
↳id 1 type inline-protocol-offload
# sa in 2 cipher_algo aes-128-cbc cipher_key_
↳de:ad:be:ef:de:ad:be:ef:de:ad:be:ef:de:ad:be:ef:de:ad:be:ef auth_algo sha1-hmac auth_key_
↳de:ad:be:ef:de:ad:be:ef:de:ad:be:ef:de:ad:be:ef:de:ad:be:ef mode transport port_
↳id 1 type inline-protocol-offload
# rt ipv4 dst 198.18.1.2/32 port 1
# rt ipv4 dst 198.18.1.1/24 port 0
# neigh port 1 0a:00:00:00:00:01
# neigh port 0 01:00:00:00:00:01
```

```
# sp ipv4 out esp protect 1 pri 2 src 198.168.1.0/24 dst 198.168.1.0/24 sport_
↳0:65535 dport 0:65535
# sa out 1 cipher_algo aes-128-cbc cipher_key_
↳de:ad:be:ef:de:ad:be:ef:de:ad:be:ef:de:ad:be:ef:de:ad:be:ef auth_algo sha1-hmac auth_key_
↳de:ad:be:ef:de:ad:be:ef:de:ad:be:ef:de:ad:be:ef:de:ad:be:ef mode transport port_
↳id 1 type inline-protocol-offload
# sa in 2 cipher_algo aes-128-cbc cipher_key_
↳de:ad:be:ef:de:ad:be:ef:de:ad:be:ef:de:ad:be:ef:de:ad:be:ef auth_algo sha1-hmac auth_key_
↳de:ad:be:ef:de:ad:be:ef:de:ad:be:ef:de:ad:be:ef:de:ad:be:ef mode transport port_
↳id 1 type inline-protocol-offload
# rt ipv4 dst 198.18.1.2/32 port 1
# rt ipv4 dst 198.18.1.1/24 port 0
# neigh port 1 0a:00:00:00:00:02
# neigh port 0 01:00:00:00:00:02
```

16.9.4 Basic Testing

Binding vfio-pci driver:

```
# dpdk-devbind -b vfio-pci xxxx.xxxx.xxxx
```

Start ipsec-secgw:

```
# dpdk-ipsec-secgw -l 24,60,61,62,63 -n 4 -a 0000:61:00:1 -a 0000:61:00:0 --socket-
↳mem 0,0,0,1024 --vdev crypto_null -- -P -p 0x3 -u 0x2 --config='(0,0,60),(1,0,
↳60),(0,1,61),(1,1,61),(0,2,62),(1,2,62),(0,3,63),(1,3,63)' -f CONFIG_FILE_PATH
```

```
# dpdk-ipsec-secgw -l 16,84,85,86,87 -n 4 -a 0000:41:00.1 -a 0000:41:00.0 --socket-  
↪mem 0,0,1024 --vdev crypto_null -- -P -p 0x3 -u 0x2 --config='(0,0,84), (1,0,84),  
↪(0,1,85), (1,1,85), (0,2,86), (1,2,86), (0,3,87), (1,3,87)' -f CONFIG_FILE_PATH
```

Use the tester or packet sending device to test performance.

17 Upgrading the Kernel

The minimum recommended Linux distribution versions are those provided in supported releases of distributions. As a guide they are as follows:

Operating System	Kernel Version
CentOS 7.6	3.10.0-957
CentOS 8.0	4.18
Ubuntu 18.04 LTS	4.15

17.1 RHEL

Only kernel packages released by Red Hat which are installable as part of the distribution installation and upgrade procedure are supported.

17.2 CentOS

The CentOS package installer yum will manage an update to the supported kernel version. The command `yum install kernel-${VERSION}` updates the kernel for CentOS. First search for available kernel packages and then install the desired release:

```
# yum list --showduplicates kernel

kernel.x86_64      3.10.0-862.e17      base
kernel.x86_64      3.10.0-862.2.3.e17  updates
kernel.x86_64      3.10.0-862.3.2.e17  updates

# yum install kernel-3.10.0-862.e17
```

17.3 Ubuntu

If desired, alternative kernels may be installed. For example, at the time of writing, v4.18 is the newest stable kernel.

17.3.1 Acquire Packages

```
# BASE=http://kernel.ubuntu.com/~kernel-ppa/mainline/v4.18/
# HEADERS=linux-headers-4.18.0-041800
# IMAGE=linux-image-unsigned-4.18.0-041800
# MODULES=linux-modules-4.18.0-041800-generic
# wget \
  $BASE/${HEADERS}_4.18.0-041800.201808122131_all.deb \
  $BASE/${HEADERS}-generic_4.18.0-041800.201808122131_amd64.deb \
  $BASE/${IMAGE}-generic_4.18.0-041800.201808122131_amd64.deb \
  $BASE/${MODULES}_4.18.0-041800.201808122131_amd64.deb
```

17.3.2 Install Packages

```
# HEADERS=linux-headers-4.18.0-041800
# IMAGE=linux-image-unsigned-4.18.0-041800-generic
# MODULES=linux-modules-4.18.0-041800-generic
# dpkg -i \
  ${HEADERS}_4.18.0-041800.201808122131_all.deb \
  ${HEADERS}-generic_4.18.0-041800.201808122131_amd64.deb \
  ${IMAGE}_4.18.0-041800.201808122131_amd64.deb \
  ${MODULES}_4.18.0-041800.201808122131_amd64.deb
```


18 UEFI Secure Boot with Out-of-Tree NFP Driver

UEFI secure boot ensures that only kernel modules signed with trusted keys can be loaded.

When the NFP driver module is loaded without a signature or with an invalid signature, the errors below may appear:

```
# dmesg
Lockdown: modprobe: unsigned module loading is restricted; see man kernel_lockdown.
↪7
# modprobe nfp
modprobe: ERROR: could not insert 'nfp': Required key not available
```

If the errors above occur when loading the NFP driver, please check the signature of the NFP driver module:

```
# modinfo nfp | grep sig
```

Conditions may as follow:

- If the NFP driver is signed with a DKMS module signing key, as below, please refer to *NFP Driver Module is Signed with a DKMS Module Signing key*:

```
# modinfo nfp | grep sig
signer:          DKMS module signing key
```

- If nothing is printed with command above, then the NFP driver module is not signed, please refer to *NFP Driver Module is Not Signed or Signed with Unknown Keys*.
- If the NFP driver is signed with an unknown key, please refer to *NFP Driver Module is Not Signed or Signed with Unknown Keys*.

18.1 NFP Driver Module is Signed with a DKMS Module Signing key

DKMS supports module signing from version 2.8.1 for Ubuntu/Debian and version 3.0.4 for other OS. More details can be found in the DKMS repository (<https://github.com/dell/dkms/blob/master/README.md#module-signing>).

18.1.1 RHEL and CentOS

The public key is placed in `/var/lib/dkms` by default. Enroll the public key to the MOK list. You may need to set a passphrase for the enrollment:

```
# mokutil --import /var/lib/dkms/mok.pub
```

Reboot the system and enter the enrollment and confirm the passphrase. The system will then boot normally and the NFP driver can be loaded.

18.1.2 Ubuntu

The public key is placed in `/var/lib/shim-signed/mok`. The enrollment is executed by automatically during driver installation via software package. Following the guidance to set the passphrase for the enrollment and reboot is required to finish enrollment.

18.2 NFP Driver Module is Not Signed or Signed with Unknown Keys

When the NFP module is not signed or signed with keys which can't be found. A pair of new keys for module signing may be generated.

1. Key generation

The Machine Owner Key (MOK) can be pre-generated and distributed or generated on the target machine using OpenSSL:

```
# openssl req -x509 -nodes -days 36500 -subj "/CN=Secure Boot DKMS Signing  
↪"  
-newkey rsa:2048 -keyout /root/MOK.priv -outform DER -out /root/MOK.der
```

Note:

- The key generated here is not encrypted with parameter `-nodes`.
- If additional security is required, please refer to the related document in Ubuntu (<https://ubuntu.com/blog/how-to-sign-things-for-secure-boot>).

2. Key enrollment

MOK enrollment process for the generated keys:

```
# mokutil --import /root/MOK.der
```

Reboot the system and finish the enrollment.

3. Manual DKMS module signing

The NFP module can be signed with the enrolled keys.

Check the name of module:

```
# modinfo nfp | grep filename
```

If the module name ends with `.xz`, f.e. `nfp.ko.xz`, then you may need to decompress it:

```
# NFP_DRV_MODULE=$(modinfo nfp | grep filename | awk -F ' ' '{print $2}')
# xz -d ${NFP_DRV_MODULE}
# depmod -a
```

For RHEL and CentOS:

```
# NFP_DRV_MODULE=$(modinfo nfp | grep filename | awk -F ' ' '{print $2}')
# /lib/modules/${uname -r}/build/scripts/sign-file sha256 /root/MOK.priv
/root/MOK.der ${NFP_DRV_MODULE}
```

For Ubuntu:

```
# NFP_DRV_MODULE=$(modinfo nfp | grep filename | awk -F ' ' '{print $2}')
# kmodsign sha256 /root/MOK.priv /root/MOK.der ${NFP_DRV_MODULE}
```

Or:

```
# NFP_DRV_MODULE=$(modinfo nfp | grep filename | awk -F ' ' '{print $2}')
# /usr/src/linux-headers-${uname -r}/scripts/sign-file sha256 /root/MOK.
→priv
/root/MOK.der ${NFP_DRV_MODULE}
```

19 Abbreviations and Terms

Abbreviation/Term	Meaning/Description
AH	Authentication Header
BPF	Berkeley Packet Filter
BSP	Board Support Package
COTS	Commercial Off-The-Shelf
CPP	Command Push/Pull
DAC	Digital to Analog Converter
DPDK	Data Plane Development Kit
DKMS	Dynamic Kernel Module Support
EM	Element Management
ESD	Electro-Static Discharge
ESP	Encapsulating Security Payload
FEC	Forward Error Correction
HPET	High Precision Event Timer
IEEE	Institute of Electrical and Electronics Engineers
IOMMU	Input/Output Memory Management Unit
IPSec	Internet Protocol Security
IKE	Internet Key Exchange
GRE	Generic Routing Encapsulation
KVM	Kernel-based Virtual Machine
LSO	Large Segmentation Offload
MANO	Management and Orchestration
MTU	Maximum Transmission Unit
<netdev>	Network device interface name
<netdev port>	Network device physical port

continues on next page

continued from previous page

Abbreviation/Term	Meaning/Description
NAPI	New Application Programming Interface (API)
NFP	Network Flow Processor
NFV	Network Functions Virtualization
NFVI	Network Functions Virtualization Infrastructure
NFVO	Network Functions Virtualization Orchestrator
NIC	Network Interface Card
NM	Network Management
NSD	Network Service Descriptor
NUMA	Non Uniform Memory Access Architecture
OS	Operating System
OOT	Out of Tree
OVS	Open vSwitch
PCI	Peripheral Component Interconnect
PF	Physical Functions
PMD	Poll Mode Driver
PNF	Physical Network Functions
PXE	Preboot Execute Environment
QoS	Quality of Service
QP	Queue Pair
RAID	Redundant Arrays of Independent Disks
RDMA	Remote Direct Memory Access
RoCE	RDMA over Converged Ethernet
RSC	Receive Side Coalescing
RSS	Receive Side Scaling
SA	Security Association
SP	Security Policy
SPOF	Single Points of Failure

continues on next page

continued from previous page

Abbreviation/Term	Meaning/Description
SR-IOV	Single Root I/O Virtualization
SRQ	Shared Receive Queue
TCP	Transmission Control Protocol
TSO	TCP Segmentation Offload
UDP	User Datagram Protocol
UEFI	Unified Extensible Firmware Interface
UIO	Userspace Input/Output
VDU	Virtualization Deployment Unit
VEB	Virtual Ethernet Bridge
VF	Virtual Functions
VFIO	Virtual Function Input/Output
VIM	Virtualized Infrastructure Manager
VLAN	Virtual Local Area Network
VNF	Virtualized Network Functions
VNFC	Virtualized Network Functions Component
VNFD	Virtualized Network Functions Descriptor
VNFFG	Virtualized Network Functions Forwarding Graph
VNFM	Virtualized Network Functions Manager
VXLAN	Virtual eXtensible Local Area Network