Corigine

Agilio CoreNIC User Manual

V23.01

Contents:

1 Introduction **1**
- 1.1 Revision History 1
- 1.2 About this Guide 1
- 1.3 Audience 1
- 1.4 Contact Us 1

2 Abbreviations and Terms **2**

3 Product Overview **5**
- 3.1 Supported Products 5
- 3.2 Safety 5
- 3.3 Standards and Regulations 6

4 The Agilio SmartNIC Architecture **7**

5 Hardware Installation **8**
- 5.1 Physical Installation 8
- 5.2 Identification 8
- 5.3 Validation 9

6 Driver and Firmware **10**
- 6.1 Confirm Upstreamed NFP Driver 10
- 6.2 Confirm that the NFP Driver is Loaded 11
- 6.3 SmartNIC Netdev Interfaces 11
- 6.4 Validating the Firmware 13
- 6.5 Upgrading the firmware 15

7 Using the Linux Driver **17**
- 7.1 Configuring Interface Media Mode 17
- 7.2 Configuring interface Maximum Transmission Unit (MTU) 18
- 7.3 Configuring FEC modes 18
- 7.4 Setting Interface Breakout Mode 21
- 7.5 Configuring interface private-flags 23
- 7.6 Confirming Connectivity 23

8 Basic Performance Test **24**
- 8.1 Install iPerf 24
- 8.2 Run iPerf Test 24
- 8.3 Using iPerf3 25

9 Basic Firmware Features **27**
- 9.1 Summary of Features 27
- 9.2 VF TX Rate Limit 30

9.3 Setting Interface Settings 34
9.4 Multiple Queues 34
9.5 Receive Side Scaling (RSS) 35
9.6 View Interface Parameters 36
9.7 Interrupt Coalescing 39
9.8 VF Split 40

10 Installing, Configuring and Using DPDK 44
10.1 Introduction to DPDK 44
10.2 Enabling IOMMU 44
10.3 DPDK Sources with PF PMD Support 45
10.4 Installing DPDK 46
10.5 Binding DPDK PF Driver 47
10.6 Using DPDK PF Driver 48

11 Using SR-IOV 50
11.1 Installing the SR-IOV Capable Firmware 50
11.2 Load Firmware to SmartNIC 53
11.3 Configuring SR-IOV 53
11.4 Using Virtio-Forwarder 56

12 Appendix A: Corigine Repositories 64
12.1 Importing GPG-Key 64
12.2 Configuring Repositories 64

13 Appendix B: Installing the Out-of-Tree NFP Driver 65
13.1 Install Driver via Corigine Repository 65
13.2 Install Driver via Software Package 66
13.3 Building from Source 67
13.4 (Optional) Kernel Changes 68

14 Appendix C: Working with Board Support Package 70
14.1 Install Software from Corigine Repository 70
14.2 Install Software from DEB/RPM Package 70
14.3 Using BSP Tools 71

15 Appendix D: Upgrading the Kernel 74
15.1 RHEL 74
15.2 CentOS 74
15.3 Ubuntu 74

16 Appendix E: Installing the VMware Driver 76
16.1 Driver Download 76
16.2 Driver Installation 78

17 Appendix F: UEFI Secure Boot with Out-of-Tree NFP Driver 80
17.1 NFP Driver Module is Signed with a DKMS Module Signing key 80
17.2 NFP Driver Module is Not Signed or Signed with Unknown Keys 81

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

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 Abbreviations and Terms

Abbreviation/Term	Meaning/Description
BPF	Berkeley Packet Filter
BSP	Board Support Package
COTS	Commercial Off-The-Shelf
DPDK	Data Plane Development Kit
DKMS	Dynamic Kernel Module Support
EM	Element Management
FEC	Forward Error Correction
HPET	High Precision Event Timer
IOMMU	Input/Output Memory Management Unit
GRE	Generic Routing Encapsulation
KVM	Kernel-based Virtual Machine
MANO	Management and Orchestration
MTU	Maximum Transmission Unit
<netdev>	Network device interface name
<netdev port>	Network device physical port
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

continues on next page

continued from previous page

Abbreviation/Term	Meaning/Description
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
RAID	Redundant Arrays of Independent Disks
RSC	Receive Side Coalescing
RSS	Receive Side Scaling
SPOF	Single Points of Failure
SR-IOV	Single Root I/O Virtualization
TSO	TCP Segmentation Offload
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

continues on next page

continued from previous page

Abbreviation/Term	Meaning/Description
VNFM	Virtualized Network Functions Manager
VXLAN	Virtual eXtensible Local Area Network

3 Product Overview

The Agilio family of SmartNICs provide the performance, functionality and programmability required by Cloud operators and service providers struggling to meet performance expectations, without consuming massive CPU cores. The Agilio SmartNICs are available in four options: Agilio CX, Agilio FX, Agilio GX and Agilio LX (https://www.corigine.com/smartnic.html).

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	2x10G 2x10G 1x1G + 1x10G
GX 2x25G	2x10G 2x25G 1x10G + 1x25G

3.2 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 (ESD). 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.

3.3 Standards and Regulations

The Agilio SmartNICs adhere to the following regulations.

3.3.1 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)

3.3.2 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

4 The Agilio SmartNIC Architecture

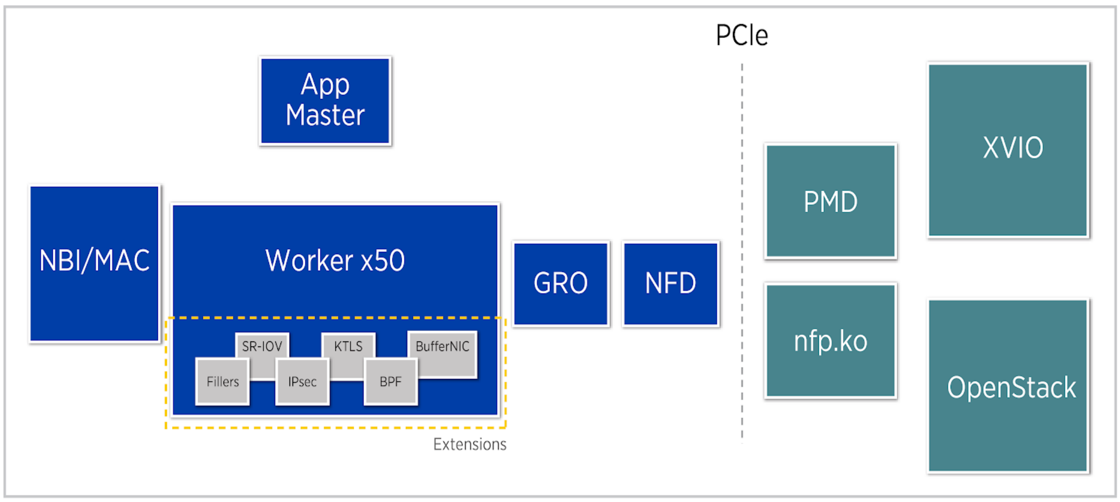


Fig. 1: The conceptual architecture of the Agilio SmartNIC

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 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 OVS datapath can also be enabled without a server reboot.

Further extensions such as BPF offload, 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 RSS (Receive Side Scaling), Checksum Offload (IPv4/IPv6, TCP, UDP, Tx/Rx), LSO (Large Segmentation Offload), 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 Hardware Installation

This user guide focuses on x86 deployments of Corigine's Agilio hardware. As detailed in *Driver and Firmware*, Corigine's Agilio SmartNIC firmware is now upstreamed with certain kernel versions of Ubuntu and RHEL/CentOS. Whilst out-of-tree driver source files are available and build/installation instructions are included in *Appendix A: Corigine Repositories*, it is highly recommended where possible to make use of the upstreamed drivers. Wherever applicable, separate instructions for RHEL/CentOS and Ubuntu are provided.

5.1 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.

5.2 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.

5.3 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:

```
# 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 Driver and Firmware

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
```

6.1 Confirm Upstreamed 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 *Appendix B: Installing the Out-of-Tree NFP Driver* for instructions on installing the out-of-tree NFP driver, or simply upgrade your distribution and kernel version to include the upstreamed drivers.

6.2 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 *Appendix F: UEFI Secure Boot with Out-of-Tree NFP Driver*.

6.3 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␣
↪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. enpAsXnpYsZ, 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 *Hardware Installation* e.g.

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.

6.3.1 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.

6.4 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_AMDA0081-0001_1x40.nffw -> nic/nic_AMDA0081-0001_1x40.nffw
lrwxrwxrwx  1   31  nic_AMDA0081-0001_4x10.nffw -> nic/nic_AMDA0081-0001_4x10.nffw
lrwxrwxrwx  1   31  nic_AMDA0096-0001_2x10.nffw -> nic/nic_AMDA0096-0001_2x10.nffw
lrwxrwxrwx  1   31  nic_AMDA0097-0001_2x40.nffw -> nic/nic_AMDA0097-0001_2x40.nffw
lrwxrwxrwx  1   36  nic_AMDA0097-0001_4x10_1x40.nffw -> nic/nic_AMDA0097-0001_4x10_
↪1x40.nffw
lrwxrwxrwx  1   31  nic_AMDA0097-0001_8x10.nffw -> nic/nic_AMDA0097-0001_8x10.nffw
lrwxrwxrwx  1   36  nic_AMDA0099-0001_1x10_1x25.nffw -> nic/nic_AMDA0099-0001_1x10_
↪1x25.nffw
lrwxrwxrwx  1   31  nic_AMDA0099-0001_2x10.nffw -> nic/nic_AMDA0099-0001_2x10.nffw
lrwxrwxrwx  1   31  nic_AMDA0099-0001_2x25.nffw -> nic/nic_AMDA0099-0001_2x25.nffw
lrwxrwxrwx  1   34  pci-0000:04:00.0.nffw -> flower/nic_AMDA0097-0001_2x40.nffw
lrwxrwxrwx  1   34  pci-0000:06:00.0.nffw -> flower/nic_AMDA0096-0001_2x10.nffw

/lib/firmware/netronome/flower:
total 11692
lrwxrwxrwx. 1      17  nic_AMDA0081-0001_1x40.nffw -> nic_AMDA0097.nffw
lrwxrwxrwx. 1      17  nic_AMDA0081-0001_4x10.nffw -> nic_AMDA0097.nffw
lrwxrwxrwx. 1      17  nic_AMDA0096-0001_2x10.nffw -> nic_AMDA0096.nffw
-rw-r--r--. 1 3987240  nic_AMDA0096.nffw
lrwxrwxrwx. 1      17  nic_AMDA0097-0001_2x40.nffw -> nic_AMDA0097.nffw
lrwxrwxrwx. 1      17  nic_AMDA0097-0001_4x10_1x40.nffw -> nic_AMDA0097.nffw
lrwxrwxrwx. 1      17  nic_AMDA0097-0001_8x10.nffw -> nic_AMDA0097.nffw
-rw-r--r--. 1 3988184  nic_AMDA0097.nffw
lrwxrwxrwx. 1      17  nic_AMDA0099-0001_2x10.nffw -> nic_AMDA0099.nffw
lrwxrwxrwx. 1      17  nic_AMDA0099-0001_2x25.nffw -> nic_AMDA0099.nffw
-rw-r--r--. 1 3990552  nic_AMDA0099.nffw

/lib/firmware/netronome/nic:
total 12220
-rw-r--r--. 1 1380496  nic_AMDA0081-0001_1x40.nffw
-rw-r--r--. 1 1389760  nic_AMDA0081-0001_4x10.nffw
-rw-r--r--. 1 1385608  nic_AMDA0096-0001_2x10.nffw
-rw-r--r--. 1 1385664  nic_AMDA0097-0001_2x40.nffw
-rw-r--r--. 1 1391944  nic_AMDA0097-0001_4x10_1x40.nffw
-rw-r--r--. 1 1397880  nic_AMDA0097-0001_8x10.nffw
-rw-r--r--. 1 1386616  nic_AMDA0099-0001_1x10_1x25.nffw
```

(continues on next page)

(continued from previous page)

```
-rw-r--r--. 1 1385608  nic_AMDA0099-0001_2x10.nffw
-rw-r--r--. 1 1386368  nic_AMDA0099-0001_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_AMDA0097-0001_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_AMDA0097-0001_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.

6.5 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).

6.5.1 Upgrading firmware via the Corigine repository

Please refer to *Appendix A: Corigine 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
```

6.5.2 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
```

7 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 Smart-NICs.

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.

7.1 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 *Appendix C: Working with Board Support Package* on how to make use of the BSP toolset to configure interfaces.

7.1.1 Configuring interface link-speed

The following steps explains 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.

7.2 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 Configuring FEC modes

CX 2x25GbE SmartNICs support 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
```

(continues on next page)

(continued from previous page)

```
    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.

7.4 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␣
↪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.

7.5 Configuring interface private-flags

7.5.1 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
```

7.6 Confirming Connectivity

7.6.1 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
```

7.6.2 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 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.

8.1 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
```

8.2 Run iPerf Test

8.2.1 Server

Run `iPerf` on the server:

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

8.2.2 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
```

8.3 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.

9 Basic Firmware Features

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

9.1 Summary of Features

The following table summarizes the features of the Agilio SmartNICs.

Feature	Description
FW version	Getting device hardware firmware info **Note:** `nfp-hwinfo` and query 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 `ethtool` 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 `ip link set <dev> vf <num> max_tx_rate <rate>`
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 `ethtool -p`	Support the identifying of the NIC port with the command `ethtool -p`: ethtool [FLAGS] -p|--identify DEVNAME
Support command `ethtool -a`	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 `Auto` 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 `ethtool -t`	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 Smart-NICs	Support 25Gb/10Gb speed and FEC mode auto-negotiation on 25Gb SmartNICs
Auto-negotiation on 10Gb Smart-NICs	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
Support VF split	Single PF cards support assignment of VFs to different physical ports
Support command `ethtool -r`	Restarts auto-negotiation on the specified Ethernet device, if auto-negotiation is enabled
Support command `ethtool -e` / `ethtool -E`	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
Inline crypto	Support IPSec crypto offload, it requires specific firmware to work
LLDP	Send out LLDP packet without OS software on host
Ethtool display "link mode"	Display "Supported link modes" and "Advertised link modes" by command `ethtool <netdev>`
Multicast MAC filter	Support passing through multicast packets with configured mac address while drop others in default mode and passing through all multicast packets in ALLMULTI mode

9.2 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.

9.2.1 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.

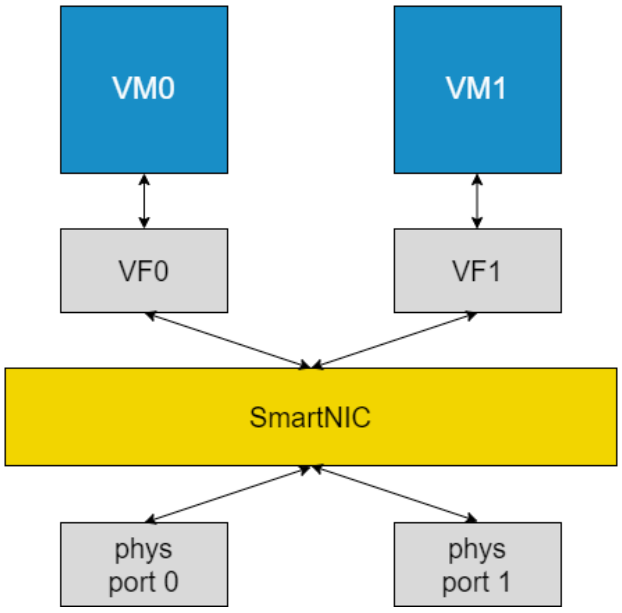


Fig. 1: Packet flow possibilities between VMs and SmartNIC

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

Corigine

9.2.2 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.

9.2.3 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.

9.2.4 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
[  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
```

(continues on next page)

(continued from previous page)

```
[ 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 Mbits/sec  4067   9.90 KBytes
[  7]   0.00-1.00   sec  18.9 MBytes    158 Mbits/sec  1166   8.48 KBytes
[  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
```

(continues on next page)

(continued from previous page)

```
[ 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.

9.3 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*.

9.4 Multiple Queues

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

9.4.1 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
```

9.4.2 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: combined = min(RXR, TXR) ; rx = RXR - combined ; tx = TXR - combined.

To configure 8 combined interrupt servicing:

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

9.5 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.

9.5.1 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
```

9.5.2 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.

9.5.3 Configuring the Key

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

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

9.6 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]
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]
```

(continues on next page)

(continued from previous page)

```
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]
```

9.6.1 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
```

9.6.2 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
```

9.6.3 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
```

9.6.4 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
```

9.6.5 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
```

9.6.6 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.

9.7 Interrupt Coalescing

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

9.7.1 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
```

(continues on next page)

(continued from previous page)

```
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
```

9.7.2 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:

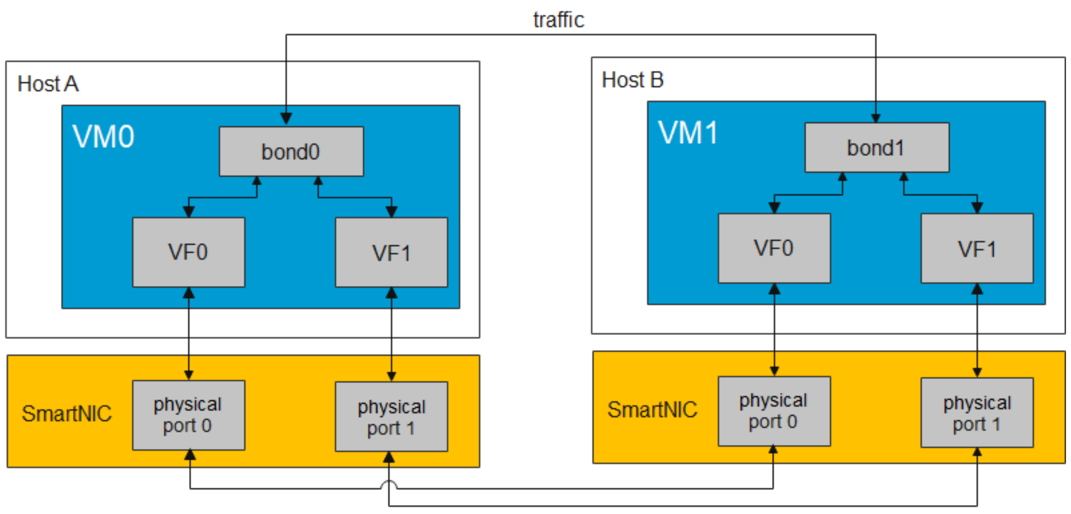
```
# ethtool -C <netdev> adaptive-rx on adaptive-tx on
```

9.8 VF Split

VF split is designed to allow assignment of VFs to different physical ports. This feature may be used to distribute packets between different physical ports in virtualised use-cases by using a LAG of VFs which are assigned to different physical ports.

The default behaviour is for all VFs to be assigned to physical port 0.

This feature is available on NICs with single a PF.



9.8.1 Check Current Status

Show the configuration:

```
# devlink resource show pci/0000:82:00.0
  pci/0000:82:00.0:
  name vf_split size 0 unit entry size_min 0 size_max 48 size_gran 1 dpipe_tables␣
↪none
  resources:
    name port0 size 0 unit entry size_min 0 size_max 48 size_gran 1 dpipe_tables␣
↪none
    name port1 size 0 unit entry size_min 0 size_max 48 size_gran 1 dpipe_tables␣
↪none
    ...
    name portn size 0 unit entry size_min 0 size_max 48 size_gran 1 dpipe_tables␣
↪none
```

9.8.2 Configuring VF Split

Assign VFs to each physical port:

```
# devlink resource set pci/0000:82:00.0 path
/vf_split size <total_vf_cfg_num>
# devlink resource set pci/0000:82:00.0 path /vf_split/port0 size <port0_vf_cfg_
↪num>
# devlink resource set pci/0000:82:00.0 path /vf_split/port1 size <port1_vf_cfg_
↪num>
...
# devlink resource set pci/0000:82:00.0 path /vf_split/portn size <portn_vf_cfg_
↪num>
```

Configuration will not take effect until VFs are created:

```
# devlink resource show pci/0000:82:00.0
  pci/0000:82:00.0:
  name vf_split size 0 size_new <total_vf_cfg_num> unit entry size_min 0 size_max␣
↪48 size_gran 1 dpipe_tables none size_valid true
  resources:
    name port0 size 0 size_new <port0_vf_cfg_num> unit entry size_min 0 size_max␣
↪48 size_gran 1 dpipe_tables none
    name port1 size 0 size_new <port1_vf_cfg_num> unit entry size_min 0 size_max␣
↪48 size_gran 1 dpipe_tables none
    ...
    name portn size 0 size_new <portn_vf_cfg_num> unit entry size_min 0 size_max␣
↪48 size_gran 1 dpipe_tables none

# echo <vf_num> > /sys/class/net/enp130s0np0/device/sriov_numvfs

# devlink resource show pci/0000:82:00.0
  pci/0000:82:00.0:
  name vf_split size <total_vf_cfg_num> unit entry size_min 0 size_max 48 size_
↪gran 1 dpipe_tables none
  resources:
    name port0 size <port0_vf_cfg_num> unit entry size_min 0 size_max 48 size_gran␣
↪1 dpipe_tables none
    name port1 size <port1_vf_cfg_num> unit entry size_min 0 size_max 48 size_gran␣
↪1 dpipe_tables none
    ...
    name portn size <portn_vf_cfg_num> unit entry size_min 0 size_max 48 size_gran␣
↪1 dpipe_tables none
```

Note:

- If `<total_vf_cfg_num>` is not equal to `<port0_vf_cfg_num>` + ... + `<portn_vf_cfg_num>`, an error will be reported:

```
# dmesg
[3258.740101] nfp 0000:82:00.0: nfp: resources size validation failed
```

- If `<vf_num>` is not equal to `<total_vf_cfg_num>`, an error will be reported:

```
# dmesg
[2947.986229] nfp 0000:82:00.0: Trying to create 2 vfs not satisfy the␣
↪configuration of vf_isolation
```

Show the relationship between VFs and physical ports:

```
# ip link show
  309: enp130s0np0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP␣
↪mode DEFAULT group default qlen 1000
```

(continues on next page)

(continued from previous page)

```
  link/ether 00:15:4d:13:51:0d brd ff:ff:ff:ff:ff:ff
  vf 0     link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff, spoof checking off,␣
↪link-state auto, trust off, query_rss off
  vf 1     link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff, spoof checking off,␣
↪link-state auto, trust off, query_rss off
  vf 2     link/ether ff:ff:ff:ff:ff:ff brd ff:ff:ff:ff:ff:ff, spoof checking off,␣
↪link-state auto, trust off, query_rss off
  vf 3     link/ether ff:ff:ff:ff:ff:ff brd ff:ff:ff:ff:ff:ff, spoof checking off,␣
↪link-state auto, trust off, query_rss off
  310: enp130s0np1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP␣
↪mode DEFAULT group default qlen 1000
  link/ether 00:15:4d:13:51:11 brd ff:ff:ff:ff:ff:ff
  vf 0     link/ether ff:ff:ff:ff:ff:ff brd ff:ff:ff:ff:ff:ff, spoof checking off,␣
↪link-state auto, trust off, query_rss off
  vf 1     link/ether ff:ff:ff:ff:ff:ff brd ff:ff:ff:ff:ff:ff, spoof checking off,␣
↪link-state auto, trust off, query_rss off
  vf 2     link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff, spoof checking off,␣
↪link-state auto, trust off, query_rss off
  vf 3     link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff, spoof checking off,␣
↪link-state auto, trust off, query_rss off
```

Note:

- Traffic between VFs assigned to different physical ports is blocked.
- When the VF is not assigned to the physical port, hardcode its mac to ff:ff:ff:ff:ff:ff to distinguish(e.g. vf 2 is not assigned to enp130s0np0).
- The `ip link set <DEVICE> vf <NUM> mac/rate/…/trust <VAL>` command can set VF parameters using any physical port of the same NIC. However, parameters can be only shown using the physical port that the VF is assigned to.

9.8.3 Disabling VF Split

Assign `0` to all resources of `/vf_split` to clear split assignment of all VFs:

```
# devlink resource set pci/0000:82:00.0 path /vf_split size 0
# devlink resource set pci/0000:82:00.0 path /vf_split/port0 size 0
# devlink resource set pci/0000:82:00.0 path /vf_split/port1 size 0
...
# devlink resource set pci/0000:82:00.0 path /vf_split/portn size 0
# echo 0 > /sys/class/net/enp130s0np0/device/sriov_numvfs
```

This assigns the VFs to physical port 0.

10 Installing, Configuring and Using DPDK

10.1 Introduction to DPDK

The Data Plane Development Kit (DPDK) allows a user to bypass the Linux network stack within it's 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.

10.2 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.

10.2.1 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.

10.2.2 Implement Changes

Apply kernel parameters changes and reboot.

Ubuntu:

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

CentOS/RHEL:

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

10.3 DPDK Sources with PF PMD Support

10.3.1 PF PMD Multi-port Support

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: **wwww: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.

10.4 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 22.03*.

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-22.03.tar.xz
# tar xf dpdk-22.03.tar.xz
# export DPDK_DIR=/usr/src/dpdk-22.03
# cd $DPDK_DIR
```

Configure and install DPDK:

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

10.5 Binding DPDK PF Driver

This section details the binding of DPDK-enabled drivers to the Physical Functions (PF) on your Smart-NIC'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.

10.5.1 Attaching VFIO-PCI Driver

Load VFIO-PCI driver module:

```
# sudo modprobe vfio-pci disable_idle_d3=1
```

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␣
↪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
```

10.5.2 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
...
```

10.5.3 Unbind Driver

To unbind VFIO-PCI driver:

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

10.6 Using DPDK PF Driver

10.6.1 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_AMDA0097-0001_2x40.nffw nic_dpdk_default.nffw
```

For CX 2x25G:

```
# cp -s nic_AMDA0099-0001_2x25.nffw nic_dpdk_default.nffw
```

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

```
# cp -s nic_AMDA0097-0001_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
GX 2x10G	AMDA0145
GX 2x25G	AMDA0144

11 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.

11.1 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.

11.1.1 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
...
```

11.1.2 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_AMDA0058-0011_2x40.nffw
```

(continues on next page)

(continued from previous page)

```
-> /opt/netronome/agilio-sriov-firmware/nic_AMDA0058-0011_2x40.nffw
...
```

11.2 Load Firmware to SmartNIC

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

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

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`.

11.3 Configuring SR-IOV

At this stage, there are still zero VFs, and only one PF (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.

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: 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 unless VF split is in effect. please refer to *VF Split*.

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
```

11.4 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 *Using SR-IOV* has been completed.

11.4.1 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*.

11.4.2 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 canchelp 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
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 `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
```

11.4.3 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
```

(continues on next page)

(continued from previous page)

```
        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
```

11.4.4 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.

11.4.5 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.

11.4.6 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.

12 Appendix A: Corigine Repositories

All the software mentioned in this document can be obtained via the official Corigine repositories. Please find instructions below on how to enable access to the aforementioned repositories from your respective Linux distributions.

12.1 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
```

12.2 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
# 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
```

13 Appendix B: Installing the Out-of-Tree NFP Driver

The NFP driver can be installed via the Corigine repository or built from source, depending on your requirement.

13.1 Install Driver via Corigine Repository

Please refer to *Appendix A: Corigine Repositories* on how to configure the Corigine repository applicable to your distribution. When the repository has been successfully added, install the *NFP driver* package using the commands below.

13.1.1 RHEL and CentOS

Installing the NFP 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
```

13.1.2 Ubuntu

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

13.2 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.

13.2.1 RHEL and CentOS

```
# mkdir 22.07.1
# cd 22.07.1/
# 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
```

13.2.2 Ubuntu

```
# mkdir 22.07.1
# cd 22.07.1/
# 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
# dnf -y install agilio-nfp-driver-dkms_22.07-1_all.deb
```

13.2.3 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
# apt -y install agilio-nfp-driver-dkms-22.07-0.noarch.rpm
```

13.2.4 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
```

13.2.5 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.

13.2.6 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
```

13.3 Building from Source

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

13.3.1 RHEL 8 and CentOS 8 Dependencies

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

13.3.2 Ubuntu Dependencies

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

13.3.3 Clone, Build and Install

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

13.4 (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 "override nfp * extra" > /etc/depmod.d/netronome.conf
# depmod -a
# rmmod nfp; modprobe nfp
# update-initramfs -u
```

CentOS:

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

14 Appendix C: Working with Board Support Package

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

14.1 Install Software from Corigine Repository

Please refer to *Appendix A: Corigine 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
```

14.2 Install Software from DEB/RPM Package

14.2.1 Obtain Software

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

14.2.2 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
```

14.2.3 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
```

14.3 Using BSP Tools

14.3.1 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.

14.3.2 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
```

15 Appendix D: 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

15.1 RHEL

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

15.2 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.el7          base
kernel.x86_64        3.10.0-862.2.3.el7      updates
kernel.x86_64        3.10.0-862.3.2.el7      updates

# yum install kernel-3.10.0-862.el7
```

15.3 Ubuntu

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

15.3.1 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
```

15.3.2 Install Packages

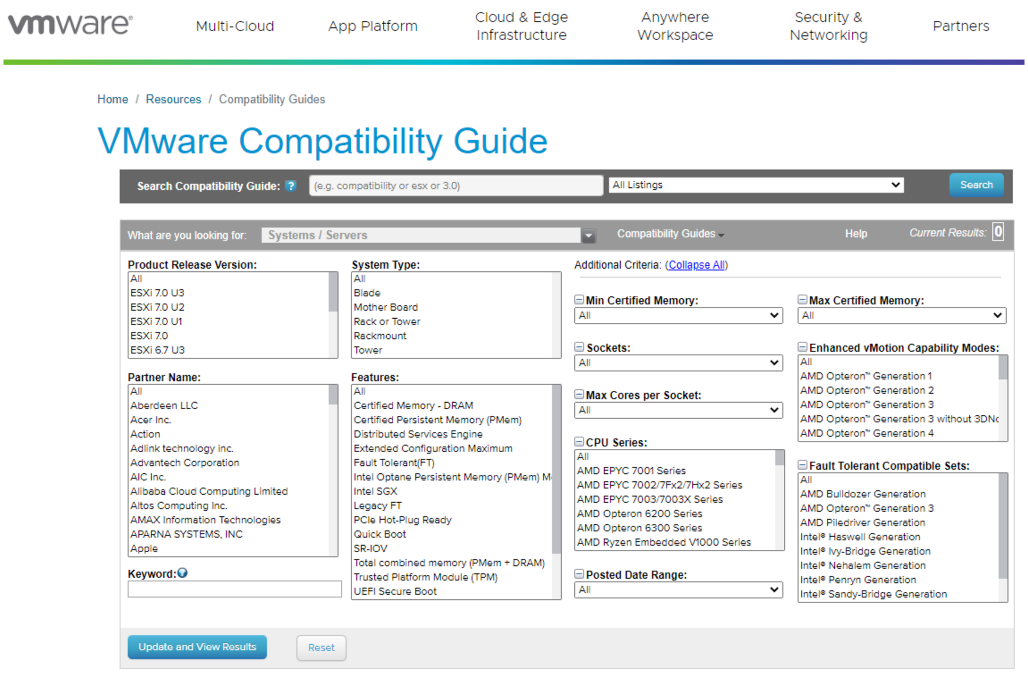
```
# HEADERS=linux-headers-4.18.0-041800
# IMAGE=linux-image-unsigned-4.18.0-041800-generic
# MODULES=linux-modules-4.18.0-041800-generic
# dpkg -i \
    ${HEADERS}_4.18.0-041800.201808122131_all.deb \
    ${HEADERS}-generic_4.18.0-041800.201808122131_amd64.deb \
    ${IMAGE}_4.18.0-041800.201808122131_amd64.deb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

16 Appendix E: Installing the VMware Driver

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

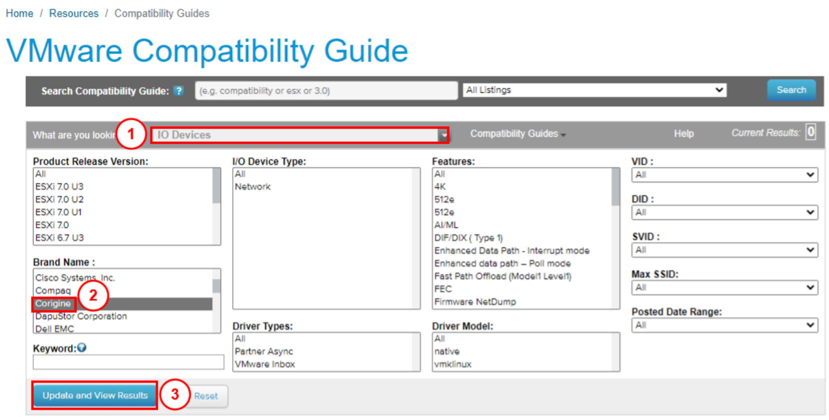
16.1 Driver Download

1. Visit the VMware Website.

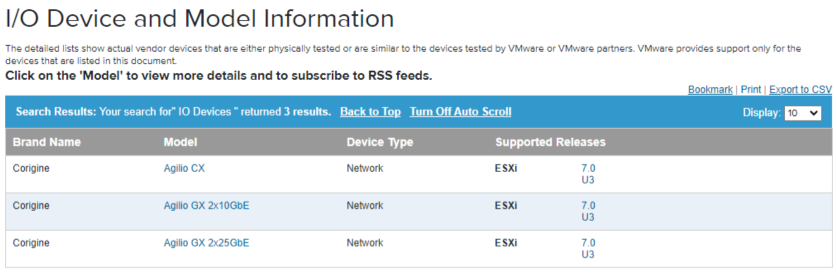

vmware
Multi-Cloud
App Platform
Cloud & Edge Infrastructure
Anywhere Workspace
Security & Networking
Partners
Home / Resources / Compatibility Guides
VMware Compatibility Guide
Search Compatibility Guide:
(e.g. compatibility or esx or 3.0)
All Listings
Search
What are you looking for: Systems / Servers
Compatibility Guides
Help
Current Results: 0
Product Release Version:
All
ESXi 7.0 U3
ESXi 7.0 U2
ESXi 7.0 U1
ESXi 7.0
ESXi 6.7 U3
System Type:
All
Blade
Mother Board
Rack or Tower
Rackmount
Tower
Additional Criteria: (Collapse All)
Min Certified Memory: All
Max Certified Memory: All
Sockets: All
Enhanced vMotion Capability Modes:
All
AMD Opteron™ Generation 1
AMD Opteron™ Generation 2
AMD Opteron™ Generation 3
AMD Opteron™ Generation 3 without 3DNow
AMD Opteron™ Generation 4
Max Cores per Socket: All
CPU Series:
All
AMD EPYC 7001 Series
AMD EPYC 7002/7Fx2/7Hx2 Series
AMD EPYC 7003/7003X Series
AMD Opteron 6200 Series
AMD Opteron 6300 Series
AMD Ryzen Embedded V1000 Series
Fault Tolerant Compatible Sets:
All
AMD Bulldozer Generation
AMD Opteron™ Generation 3
AMD Piledriver Generation
Intel® Haswell Generation
Intel® Ivy-Bridge Generation
Intel® Nehalem Generation
Intel® Penryn Generation
Intel® Sandy-Bridge Generation
Posted Date Range: All
Partner Name:
All
Aberdeen LLC
Acer Inc.
Action
Adlink technology inc.
Advantech Corporation
AIC Inc.
Alibaba Cloud Computing Limited
Altos Computing Inc.
AMAX Information Technologies
APARNA SYSTEMS, INC
Apple
Features:
All
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)
UEFI Secure Boot
Keyword:
Update and View Results
Reset

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 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.

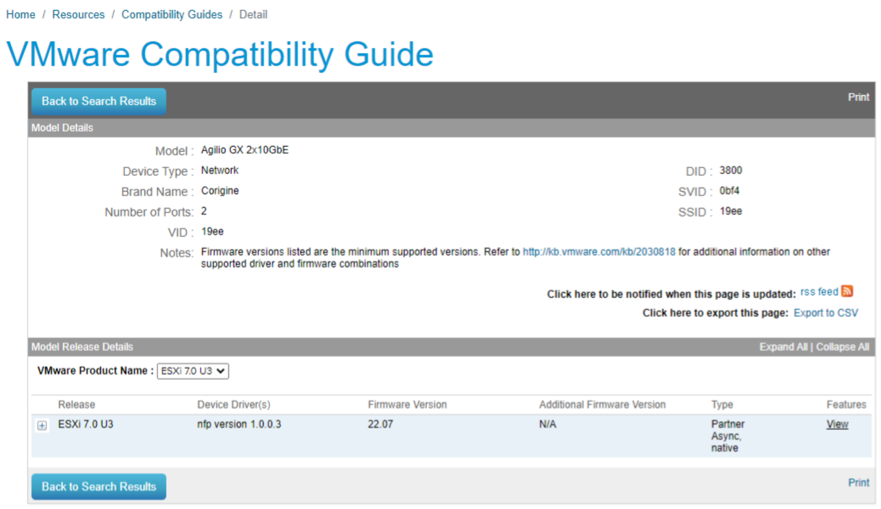
Click on the 'Model' to view more details and to subscribe to RSS feeds.

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

1. 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

Model Details

Model :	Agilio GX 2x10GbE		
Device Type :	Network	DID :	3800
Brand Name :	Corigine	SVID :	0bf4
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

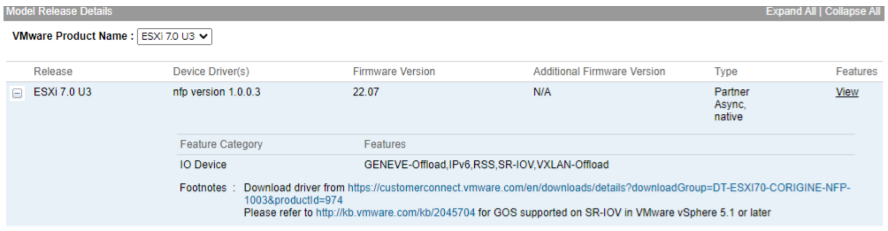
Click here to export this page: Export to CSV

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
ESXi 7.0 U3	nfp version 1.0.0.3	22.07	N/A	Partner Async, native	View

2. Click `+`, view `Model Releass Details`.



Model Release Details — Expand All | Collapse All

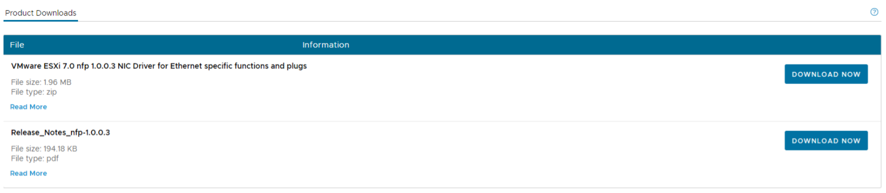
VMware Product Name : ESXi 7.0 U3

Release	Device Driver(s)	Firmware Version	Additional Firmware Version	Type	Features
ESXi 7.0 U3	nfp version 1.0.0.3	22.07	N/A	Partner Async, native	View

Feature Category	Features
IO Device	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

3. 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 File size: 1.96 MB File type: zip Read More		DOWNLOAD NOW
Release_Notes_nfp-1.0.0.3 File size: 194.18 KB File type: pdf Read More		DOWNLOAD NOW

16.2 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
```

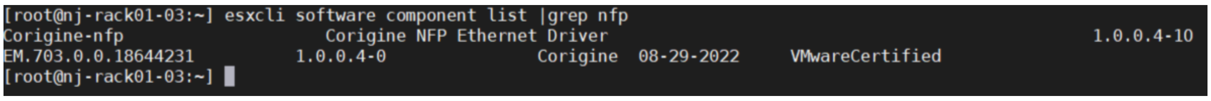
Note:

- 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:

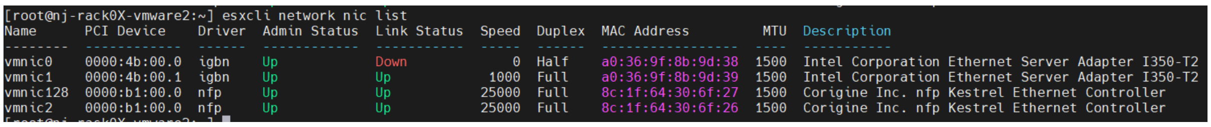
```
Installation Result
   Components Installed: Corigine-nfp_1.0.0.3-1OEM.703.0.0.18644231
   Components Removed:
   Components Skipped:
   Message: Operation finished successfully.
   Reboot Required: false
```

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.



```
[root@nj-rack01-03:~] 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
[root@nj-rack01-03:~]
```

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



```
[root@nj-rack0X-vmware2:~] esxcli network nic list
Name       PCI Device    Driver  Admin Status  Link Status  Speed  Duplex  MAC Address         MTU  Description
--------   ------------  ------  ------------  -----------  -----  ------  -----------------   ---- -----------
vmnic0     0000:4b:00.0  igbn    Up            Down             0  Half    a0:36:9f:8b:9d:38  1500  Intel Corporation Ethernet Server Adapter I350-T2
vmnic1     0000:4b:00.1  igbn    Up            Up            1000  Full    a0:36:9f:8b:9d:39  1500  Intel Corporation Ethernet Server Adapter I350-T2
vmnic128   0000:b1:00.0  nfp     Up            Up           25000  Full    8c:1f:64:30:6f:27  1500  Corigine Inc. nfp Kestrel Ethernet Controller
vmnic2     0000:b1:00.0  nfp     Up            Up           25000  Full    8c:1f:64:30:6f:26  1500  Corigine Inc. nfp Kestrel Ethernet Controller
```

17 Appendix F: 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*.

17.1 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).

17.1.1 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.

17.1.2 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.

17.2 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}
```