Corigine

Agilio CoreNIC User Manual

V22.04

www.corigine.com

Corigine

Contents

Corigine

Document Revision History

Revision	Date	Description
V22.04	29 Apr 2022	Corigine initial public release.

Corigine

About this Manual

This is the User Manual for Agilio CoreNic Fireware 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 grey background and expected results in grey ink.

Audience

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

Contact Us

2F West, Building 1, No. 1516 Hongfeng Road, Wuxing Dist., Huzhou, Zhejiang, 313000	
+86-572-2361505	
https://www.corigine.com	smartnic-support@corigine.com

Abbreviations and Terms

Abbreviation/Term	Meaning/Definition
BPF	Berkeley Packet Filter
BSP	Board Support Package
COTS	Commercial Off-The-Shelf
DPDK	Data Plane Development Kit
DKMS	Dynamic Kernel Module Support
EOR	End of Row
EM	Element Management
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
NFP	Network Flow Processor
NFV	Network Functions Virtualization
NFVI	Network Functions Virtualization Infrastructure
NFVO	Network Functions Virtualization Orchestrator

NIC	Network Interface Card
NM	Network Management
NSD	Network Service Descriptor
NUMA	Non Uniform Memory Access Architecture
OS	Operating System
OOT	Out of Tree
OVS	Open vSwitch
PCI	Peripheral Component Interconnect
PF	Physical Functions
PNF	Physical Network Functions
PXE	Preboot Execute Environment
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
VDU	Virtualization Deployment Unit
VF	Virtual Functions
VIM	Virtualized Infrastructure Manager
VLAN	Virtual Local Area Network
VNF	Virtualized Network Functions
VNFC	Virtualized Network Functions Component
VNFD	Virtualized Network Functions Descriptor
VNFFG	Virtualized Network Functions Forwarding Graph
VNFM	Virtualized Network Functions Manager
VXLAN	Visual Extensible Local Area Network

1 PRODUCTS OVERVIEW AND LAYOUT

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.cn/smartnic.html).

1.1 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
Agilio CX 2x10	2x10G
Agilio CX 2x25	2x10G 2x 25G
Agilio CX 2x25 (v2)	2x10G 2x25G
Agilio CX 1x40	1x40G 4x10G
Agilio CX 2x40	4x10G 1x40G per port
Agilio GX 2x10	2x10G

1.2 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 lazer.

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.

1.3 The Agilio SmartNIC Architecture

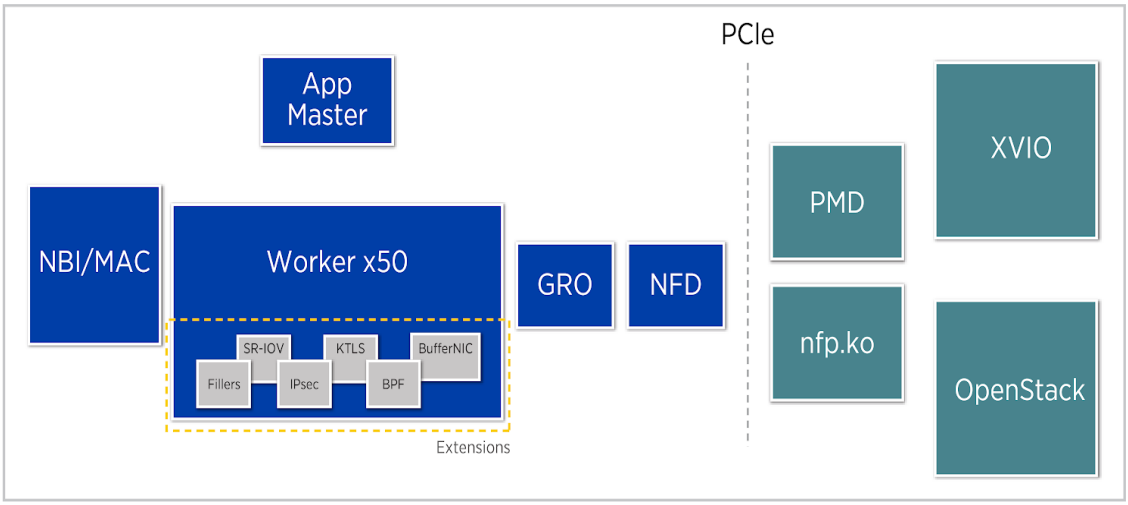

Figure 1: The conceptual architecture of the Agilio SmartNIC

The Agilio CX 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*.

1.4 Standards and Regulations

The Agilio SmartNICs adhere to the following regulations.

1.4.1 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)

1.4.2 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

2 HARDWARE INSTALLATION

This user guide focuses on x86 deployments of Corigine's Agilio hardware. As detailed in *Validating the Driver*, 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 useof the upstreamed drivers. Wherever applicable, separate instructions for RHEL/CentOS and Ubuntu are provided.

2.1 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 methodson 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}
```

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.

2.2 Physical Installation

Physically install the SmartNIC in the host server and ensure proper cooling e.g. airflow over the card. Ensure that the PCI slot is at least Gen3 x8 (the SmartNIC can be placed in a Gen3 x16 slot). Once

installed, power up the server and open a terminal. For additional support, contact smartnic-support@corigine.com.

2.3 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 the following command to validate that the SmartNIC is being correctly detected by the host server and identify its PCI address, 19ee is the specific PCI vendor identifier:

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

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.

3 VALIDATING THE DRIVER

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 respectivekernels 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

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
```

3.1 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 versions to include the upstreamed drivers.

3.2 Confirm that the NFP Driver is Loaded

Use the *lsmod* command to list the loaded driver modules and *grep* to look 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
```

3.3 SmartNIC Netdev Interfaces

The *agilio-naming-policy* package ensures consistent naming of Corigine SmartNIC network interfaces. Please note that this package is **optional** and not required if your distribution has a sufficiently new systemd installation.

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

For Ubuntu:

```
# apt-get install agilio-naming-policy
```

For RHEL or CentOS:

```
# yum install agilio-naming-policy
```

At 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 PCI address obtained in *Hardware Installation* e.g.

```
#!/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
```

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.

3.3.1 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 othernetdevs.
- 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 andanother according to systemd schemes.

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

Refer to the online biosdevname documentation for more details about the naming policy convention thatwill be applied.

3.4 Validating the Firmware

Corigine's 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 varydepending 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 17nic_AMDA0081-0001_4x10.nffw -> nic_AMDA0097.nffw
lrwxrwxrwx. 1 17nic_AMDA0096-0001_2x10.nffw -> nic_AMDA0096.nffw
-rw-r--r--. 1   3987240    nic_AMDA0096.nffw
lrwxrwxrwx. 1 17nic_AMDA0097-0001_2x40.nffw -> nic_AMDA0097.nffw
lrwxrwxrwx. 1 17nic_AMDA0097-0001_4x10_1x40.nffw ->
    nic_AMDA0097.nffw
lrwxrwxrwx. 1 17nic_AMDA0097-0001_8x10.nffw -> nic_AMDA0097.nffw
-rw-r--r--. 1   3988184    nic_AMDA0097.nffw
lrwxrwxrwx. 1 17nic_AMDA0099-0001_2x10.nffw -> nic_AMDA0099.nffw
lrwxrwxrwx. 1 17nic_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
-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 thefollowing 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 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/netdev port> 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
```

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.

3.5 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.cn/Download.html).

3.5.1 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
# dracut -f
```

For RHEL 8 and CentOS 8:

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

3.5.2 Upgrading Firmware from Package Installations

The latest firmware can be obtained at the downloads area of the Corigine Support site (https://www.corigine.com.cn/Download.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
# dracut -f
```

For RHEL 8 and CentOS 8:

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

4 USING THE LINUX DRIVER

4.1 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.

4.1.1 Configuring Interface Link-speed

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

Down the respective interface(s):

```
#  ip link set dev enp4s0np0 down
```

Set interface link-speed to 10G:

```
# ethtool -s enp4s0np0 speed 10000
```

Set interface link-speed to 25G:

```
# ethtool -s enp4s0np0 speed 25000
```

Reload driver for changes to take effect:

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

Note: The settings above only apply to Agilio CX 25G SmartNICs and older drivers/firmware changes may require a system reboot for changes to take effect.

4.2 Configuring Interface Maximum Transmission Unit (MTU)

The MTU of interfaces can temporarily be set using the iproute2 or ifconfig tools. Note that this change will not persist. Setting this via *`Network Manager`*, or an other appropriate OS configuration tool, is recommended.

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.

4.3 Configuring FEC Modes

Agilio CX 2x25GbE SmartNICs support FEC mode configuration, e.g. Auto, Firecode BaseR, Reed Solomonand Off modes. Each physical port's FEC mode can be set independently via the ethtool command. To view the currently supported FEC modes of the interface use the following:

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

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

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:  020025.020025.020072
```

This example lists a version of 020025.020025.020072 which is sufficient to support ethtool FEC mode configuration. To update your SmartNIC NVRAM flash, refer to *Appendix E: Updating NFP Flash* or 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 enp130s0np0
Settings for enp130s0np0:
...
Supported FEC modes:  None
```

To show the currently active FEC mode for either the *<netdev>* or its physical port(s) <netdev port>:

```
# ethtool --show-fec <netdev>/<netdev port>
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 autonegotiation 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 BaseR:

```
# 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.

4.4 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 following section on configuring interfaces: *Configure Media Settings*.

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

Determine the card's PCI address:

```
# lspci -Dkd 19ee:4000
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
```

List the devices:

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

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 Agilio CX 2x40GbE SmartNICs. The first physical 40G port cannot be split without the second physical port also being split, hence 1x40G + 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 theport configuration to take effect.

4.5 Confirming Connectivity

4.5.1 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.

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

4.5.2 Pinging Interfaces

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

```
# 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
```

5 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.

5.1 Install iPerf

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
```

5.2 Run iPerf Test

5.2.1 Server

Run iPerf on the server:

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

5.2.2 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
```

5.3 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
[5] 0.00-10.04    sec   10.2 GBytes 8.7   Gbits/sec   receiver
Total: 37.7 Gbits/sec
95.49% of 40GbE link
```

6 BASIC FIRMWARE FEATURES

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

6.1 Summary of Features

The following table summarizes the features of the Agilio SmartNICs.

Agilio CoreNIC feature list		
Feature	**Description**	**Notes**
FW version	Getting device hardware firmware info	-
UEFI PXE boot	Support UEFI BIOS PXE booting	-
Legacy PXE boot	Suport Legacy BIOS PXE booting	-
Firmware flashing	Ethtool support BSP firmware updating	-
Extended stats	Support extended ethtool statistics reporting	-
TSO (LSO)	TCP (Large) Segment Offload	-
SR-IOV	Single-Root Virtual Functions and virtual ethernet bridge	-
SR-IOV MAC VLAN	VLAN offload from SRIOV MACs	-
RSS: Receive Side Scaling	Core/Interrupt/Queue packet routing	-
TCP/UDP checksum	Rx/Tx	IP/UDP/TCP - Driver offloads checksum calculation When TSO slice, the ip csum will be re-calculated by nic
Jumbo frame support	MTU setting	-
eBPF offload	eBPF rules on NFP	-
Inner L3 checksum	-	-
Inner L4 checksum	-	-
Outer L3 checksum	-	-
Support for TSO	-	-
NVGRE	Microsoft, included tenant network id over GRE	-

Adaptive RX/TX	Change interrupt rates under load (IRQ handling)	-
CX 2x25GbE v2 10G+25G	Allow 10G+25G in any port-combination on 2-port cards	P0 - 10G, P1 - 25G
Support CX 2x25GbE v2 board	Firmware build for the new card type	-

6.2 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*.

6.3 Multiple Queues

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

6.3.1 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
```

6.3.2 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 <intf> rx 0 tx 0 combined 8
```

6.4 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.

6.4.1 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
```

6.4.2 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.

6.4.3 Configuring the Key

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

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

6.5 View Interface Parameters

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

```
# ethtool -k <netdev>
Features for enp4s0np0:
rx-checksumming: off [fixed]
tx-checksumming: off
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
tx-scatter-gather: off [fixed]
tx-scatter-gather-fraglist: off [fixed]
tcp-segmentation-offload: off
tx-tcp-segmentation: off [fixed]
tx-tcp-ecn-segmentation: off [fixed]
tx-tcp6-segmentation: off [fixed]
tx-tcp-mangleid-segmentation: off [fixed]
udp-fragmentation-offload: off [fixed]
generic-segmentation-offload: off [requested on]
generic-receive-offload: on
large-receive-offload: off [fixed]
rx-vlan-offload: off [fixed]
tx-vlan-offload: off [fixed]
ntuple-filters: off [fixed]
receive-hashing: off [fixed] highdma: off [fixed]
rx-vlan-filter: off [fixed]
vlan-challenged: off [fixed]
tx-lockless: off [fixed]
netns-local: off [fixed]
…
```

6.5.1 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
```

6.5.2 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
```

6.5.3 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 outputas 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: on
tx-scatter-gather: on
generic-segmentation-offload: on
```

6.5.4 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
```

6.5.5 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
```

6.5.6 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 <interface>* to get bus-info will not work on representorsas this information is not populated for representor devices.

6.6 Interrupt Coalescing

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

6.6.1 View Current Coalescing Parameters

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

```
# ethtool -c ens1np0
Coalesce parameters for  ens1np0:
Adaptive RX: off TX: off
stats-block-usecs: 0
sample-interval: 0
pkt-rate-low: 0
pkt-rate-high: 0
rx-usecs: 50
rx-frames:  64
rx-usecs-irq: 0
rx-frames-irq: 0
tx-usecs: 50
tx-frames: 64
tx-usecs-irq: 0
tx-frames-irq: 0
rx-usecs-low:  0
rx-frame-low:  0
tx-usecs-low:  0
tx-frame-low:  0
rx-usecs-high:  0
rx-frame-high:  0
tx-usecs-high:  0
tx-frame-high:  0
```

6.6.2 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
```

7 INSTALLING, CONFIGURING AND USING DPDK

7.1 Enabling IOMMU

In order to use the NFP device with DPDK applications, the VFIO/IGB module has to be loaded. Firstly, the machine has to have 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 UIO instead of VFIO, and also mentions the VFIO no-IOMMU mode.
Although DPDK focuses on avoiding interrupts, there is an option of a 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.

7.1.1 Edit Grub Configuration File

This change 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.

7.1.2 Implement Changes

Apply kernel parameters changes and reboot.

For Ubuntu:

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

For RHEL or CentOS:

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

7.2 DPDK Sources with PF PMD Support

7.2.1 PF PMD Multiport Support

The PMD can work with up to 8 ports on the same 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 VF PCI device. However, with the NFP PF multiport 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 multiport support: RX interrupts and device hot-plugging are not supported.

7.3 Installing DPDK

Physical Function PMD support has been upstreamed into DPDK 18.11.

Install prerequisites for Ubuntu:

```
# apt-get -y install libnuma-dev  build-essential
```

Install prerequisites for RHEL 7 or CentOS 7:

```
# yum -y install libnuma-dev  build-essential
```

Install prerequisites for RHEL 8 or CentOS 8:

```
# dnf -y install libnuma-dev  build-essential
```

Obtain DPDK sources:

```
# cd /usr/src/
# wget http://fast.dpdk.org/rel/dpdk-17.11.tar.xz
# tar xf  dpdk-17.11.tar.xz
# export DPDK_DIR=/usr/src/dpdk-17.11
# cd $DPDK_DIR
```

Configure and install DPDK:

```
# export DPDK_TARGET=x86_64-native-linuxapp-gcc
# export DPDK_BUILD=$DPDK_DIR/$DPDK_TARGET
# make install T=$DPDK_TARGET  DESTDIR=install
```

7.4 Binding DPDK PF Driver

Note: This section details the binding of dpdk-enabled drivers to the Physical Functions.

7.4.1 Attaching vfio-pci Driver

Load vfio-pci driver module:

```
# modprobe  vfio-pci
```

Unbind current drivers:

```
# PCIA=0000:$(lspci -d 19ee:4000 | awk '{print $1}')
# echo $PCIA > /sys/bus/pci/devices/$PCIA/driver/unbind
```

Bind vfio-pci driver:

```
# echo 19ee 4000 > /sys/bus/pci/drivers/vfio-pci/new_id
```

7.4.2 Attaching igb-uio Driver

Load igb-uio driver module:

```
# modprobe uio
# DRKO=$(find $DPDK_DIR -iname 'igb_uio.ko' | head1)
# insmod  $DRKO
```

Unbind current drivers:

```
# PCIA=0000:$(lspci -d 19ee:4000 | awk '{print $1}')
# echo $PCIA > /sys/bus/pci/devices/$PCIA/driver/unbind
```

Bind igb_uio driver:

```
# echo 19ee 4000 > /sys/bus/pci/drivers/igb_uio/new_id
```

7.4.3 Confirm Attached Driver

Confirm that the driver has been attached:

```
# lspci -kd 19ee:
01:00.0 Ethernet controller: Netronome Systems, Inc. Device 4000
Subsystem: Netronome Systems, Inc. Device 4001
Kernel driver in use: nfp
Kernel modules: nfp
01:08.0 Ethernet controller: Netronome Systems, Inc. Device 6003
Subsystem: Netronome Systems, Inc. Device  4001
Kernel driver in use: igb_uio
Kernel modules:  nfp
```

7.4.4 Unbind Driver

Determine card address:

```
# PCIA=$(lspci -d 19ee: | awk '{print $1}')
```

Unbind vfio-pci driver:

```
# echo 0000:$PCIA > /sys/bus/pci/drivers/vfio-pci/unbind
```

Unbind igb_uio driver:

```
# echo 0000:$PCIA > /sys/bus/pci/drivers/igb_uio/unbind
```

7.5 Using DPDK PF Driver

7.5.1 Create Default Symlink

Note: This workaround applies to dpdk versions < 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 Agilio 2x40G:

```
# cp -s nic_AMDA0097-0001_2x40.nffw nic_dpdk_default.nffw
```

For Agilio 2x25G:

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

For Agilio 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
Agilio CX 2x10G	AMDA0096
Agilio CX 2x25G	AMDA0099
Agilio CX 2x25G v2	AMDA0161
Agilio CX 1x40G	AMDA0081
Agilio CX 2x40G	AMDA0097
Agilio GX 2x10G	AMDA0145

8 USING SR-IOV

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.

8.1 Installing the SR-IOV Capable Firmware

Before installing the SR-IOV capable firmware, ensure that SR-IOV is enabled in the BIOS of the hostmachine. 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 can be determined by the ethtool command. As an example, Ubuntu 18.04 LTS contains the following upstreamed firmware:

```
# ethtool -i enp2s0np0 | head -3
driver: nfp
version: 4.15.0-20-generic SMP mod_unload
firmware-version: 0.0.3.5 0.22 nic-2.0.4 nic
```

From the above output, the upstreamed firmware is nic-2.0.4. The prefix nic indicates that the firmware implements the basic NIC functionality. The suffix 2.0.4 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.

8.1.1 The linux-firmware Package

The SR-IOV capable firmware has been upstreamed into the linux-firmware package. For rpm packages, this will be available from linux-firmware 20181008-88 onwards. As of Ubuntu 18.10, the linux-firmware Debian package does not yet contain SR-IOV capable firmware.

Ensure that the latest linux-firmware package is installed. For RHEL / Fedora / CentOS:

```
# 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 SmartNICfunctionality. Consider the following tree structure:

```
# tree /lib/firmware/netronome
/lib/firmware/netronome/
    flower
        nic_AMDA0081-0001_1x40.nffw -> nic_AMDA0081.nffw
        nic_AMDA0081-0001_4x10.nffw -> nic_AMDA0081.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, nic-sriov) are supplied by the linux-firmware package. If nic-sriov is missing, follow the *The Support Site* method below. Point the symboliclinks to the specific application required, in this case nic-sriov:

8.1.2 The Support Site

The SR-IOV capable firmware can be obtained from the Corigine support site. Upon downloading the packaged firmware, install the firmware files.

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

For Debian / Ubuntu:

```
# dpkg -i  agilio-sriov-firmware-2.1.x.deb
```

For RHEL / Fedora / CentOS:

```
# yum -y install agilio-sriov-firmware-2.1.x.rpm
```

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. This can be confirmed with:

```
# 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
...
```

8.1.3 Load Firmware to SmartNIC

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

```
# modprobe -r  nfp
# modprobe  nfp
```

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

```
# ethtool -i enp2s0np0 | head -3
driver: nfp
version: 4.15.0-20-generic SMP mod_unload
firmware-version: 0.0.3.5 0.22 sriov-2.1.14 nic
```

Notice that the firmware has successfully changed from nic-2.0.4 to sriov-2.1.14.

Note: Because the */lib/firmware/netronome* directory is managed by the linux-firmware package, an update to this package will cause the symbolic links to point back to the nic firmware files. If a system reboot or a driver reload occurs after the links were changed, the incorrect firmware will be loaded to the SmartNIC. In this event, repeat the *Installing the SR-IOV Capable Firmware* procedure to restore the desired functionality. A workaround is possible, but involves additional configuration of the initramfs file system. Customers interested in this workaround can *Contact Us* for more information.

8.2 Configuring SR-IOV

At this stage, there are still zero VFs, and only one PF (assuming only one Corigine SmartNIC is installed):

```
# 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
```

The number of supported VFs on a netdev is exposed by sriov_totalvfs in sysfs. For example, if enp2s0np0 is the interface associated with the SmartNIC's PF, the following command will return the total supported number of VFs:

```
# cat /sys/class/net/enp2s0np0/device/sriov_totalvfs
56
```

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

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

The new VFs, together with the PF, can be observed with the lspci command:

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

In this example, 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. ForRHEL 7.x systems however, the VFs will use the nfp driver.

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 reality, the VFs are only connected to phyport 0.

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:

```
# echo 0 > /sys/class/net/enp2s0np0/device/sriov_numvfs
```

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

In order to persist the VFs on the system, the system initialisation scripts can be updated to manage them. The following snippet illustrates how to do this for the PF enp2s0np0:

```
#!/bin/sh
cat > /etc/init.d/vf-init << 'EOF'
#!/bin/sh
ip link set mtu 9216 dev enp2s0np0
ip link set up dev  enp2s0np0
cat /sys/class/net/enp2s0np0/device/sriov_totalvfs  >
    /sys/class/net/enp2s0np0/device/sriov_numvfs
EOF
chmod u+x  /etc/init.d/vf-init
```

To enable PCI passthrough, edit the kernel command line at /etc/default/grub. Add the parameters intel_iommu=on iommu=pt to the existing command line:

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

Then:

```
# update-grub
```

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
```

8.3 Using virtio-forwarder

virtio-forwarder is a userspace networking application that forwards bi-directional traffic between SR-IOVVFs and virtio networking devices in 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 *Using SR-IOV* has been completed.

8.3.1 Installing virtio-forwarder

For Debian / Ubuntu:

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

For RHEL / Fedora / 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. Carry out the instructions of *Installing DPDK*.

8.3.2 Configuring Hugepages

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:

```
/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,
```

Also edit the existing line, such that:

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

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 following should be added to the Linux kernel command line parameters:

```
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 tothe 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/  )
```

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

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:

```
# hugeadm --pool-list
      Size  Minimum  Current  Maximum  Default
   2097152     2048     2048     2048        *
1073741824        8        8        8
```

8.3.3 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.

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

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

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

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

Notice that the Kernel driver in use attribute was removed. 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
```

8.3.4 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 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
```

To manually start the service after installation, run:

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

8.3.5 Adding VF Ports to virtio-forwarder

Modify socket permissions:

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

Dynamically map the PCI address of each VF to virtio-forwarder as follows:

```
# /usr/lib/virtio-forwarder/virtioforwarder_port_control.py add  \
--virtio-id 1 --pci-addr 02:08.0 status: OK
# /usr/lib/virtio-forwarder/virtioforwarder_port_control.py add  \
--virtio-id 2 --pci-addr 02:08.1 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 02:08.0 status: OK
# /usr/lib/virtio-forwarder/virtioforwarder_port_control.py remove
    \--virtio-id 2 --pci-addr 02:08.1 status: OK
```

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.

8.3.6 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>
```

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.

9 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.

9.1 Importing GPG-key

Download and Import GPG-key to your local machine:

```
# wget https://download.corigine.com.cn/public/Corigine.pub
```

For RHEL and CentOS, import the public key:

```
# rpm --import Corigine.pub
```

For Ubuntu based systems, import the public key:

```
# apt-key add Corigine.pub
```

9.2 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
# echo "deb [trusted=yes]
https://download.corigine.com.cn/public/apt stable main" > \
/etc/apt/sources.list.d/corigine.list
# apt-get update
```

10 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.

10.1 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.

10.1.1 RHEL 8 and CentOS 8

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:

```
# dnf install epel-release
```

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

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

10.1.2 Ubuntu

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

10.1.3 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 `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 by adding

--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 and 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:

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

10.2 Building from Source

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

10.2.1 RHEL 8 and CentOS 8 Dependencies

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

10.2.2 Ubuntu Dependencies

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

10.2.3 Clone, Build and Install

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

11 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.

11.1 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.

11.1.1 RHEL 7 and CentOS 7

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

11.1.2 RHEL 8 and CentOS 8

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

11.1.3 Ubuntu

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

11.2 Install Software From deb/rpm Package

11.2.1 Obtain Software

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

11.2.2 Install the Prerequisite Dependencies

RHEL and CentOS Dependencies

No dependency installation required.

Ubuntu Dependencies

To install the BSP package dependencies on Ubuntu, run:

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

11.2.3 NFP BSP Package

Install the NFP BSP package provided by Corigine Support.

RHEL 7 and CentOS 7 Install

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

RHEL 8 and CentOS 8 Install

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

Ubuntu Install

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

11.3 Using BSP Tools

11.3.1 Enable CPP Access

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

Follow the steps from *Install Driver via Corigine Repository* to install the oot nfp driver. After the nfp module has been built load the driver with CPP access:

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

To persist this option across reboots, several options are available; the distribution specific documentation will detail the process more thoroughly. Care must be taken that the settings are also applied to any initramfs images generated.

11.3.2 Configure Media Settings

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

Agilio 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 mayresult 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.

Agilio CX 1x40GbE - AMDA0081

Set port 0 to 40G mode:

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

Set port 0 to 4x10G fanout mode:

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

Agilio 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
```

12 APPENDIX D: UPDATING NFP FLASH

The NVRAM flash software on the SmartNIC can be updated via the BSP userspace tools. T he BSP package needs to be installed to gain access to the intendedflash image. After the flash has been updated, the system needs to be rebooted for changes to take effect.

Refer to *Appendix D: Working with Board Support Package* to acquire the BSP tool package.

12.1 Update via BSP Userspace Tools

12.1.1 Obtain Out of Tree NFP Driver

To update the flash using the BSP userspace tools, use the following steps. Refer to *Appendix C: Installingthe Out-of-Tree NFP Driver* on installing the out of tree NFP driver and to load the driver with CPP access.

12.1.2 Flash the Card

The following commands may be executed for each card installed in the system using the PCI address of the particular card. In this section, the card's PCI address is assumed to be 0000:04:00.0. First reload the NFP drivers with CPP access enabled:

```
# rmmod nfp
# modprobe nfp nfp_pf_netdev=0 nfp_dev_cpp=1
```

Then use the included Corigine flashing tools to reflash the card:

```
# /opt/netronome/bin/nfp-fw-update -u
# reboot
```

13 APPENDIX E: UPGRADING THE KERNEL

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

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

13.1 RHEL

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

13.2 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_4
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
```

13.3 Ubuntu

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

13.3.1 Acquire Packages

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

13.3.2 Install Packages

```
dpkg -i \
    linux-headers-4.18.0-041800_4.18.0-041800.201808122131_all.deb
    \
    linux-headers-4.18.0-041800-generic_4.18.0-
    041800.201808122131_amd64.deb   \
    linux-image-unsigned-4.18.0-041800-generic_4.18.0-
    041800.201808122131_amd64.deb   \
    linux-modules-4.18.0-041800-generic_4.18.0-
    041800.201808122131_amd64.deb
```