Agenda

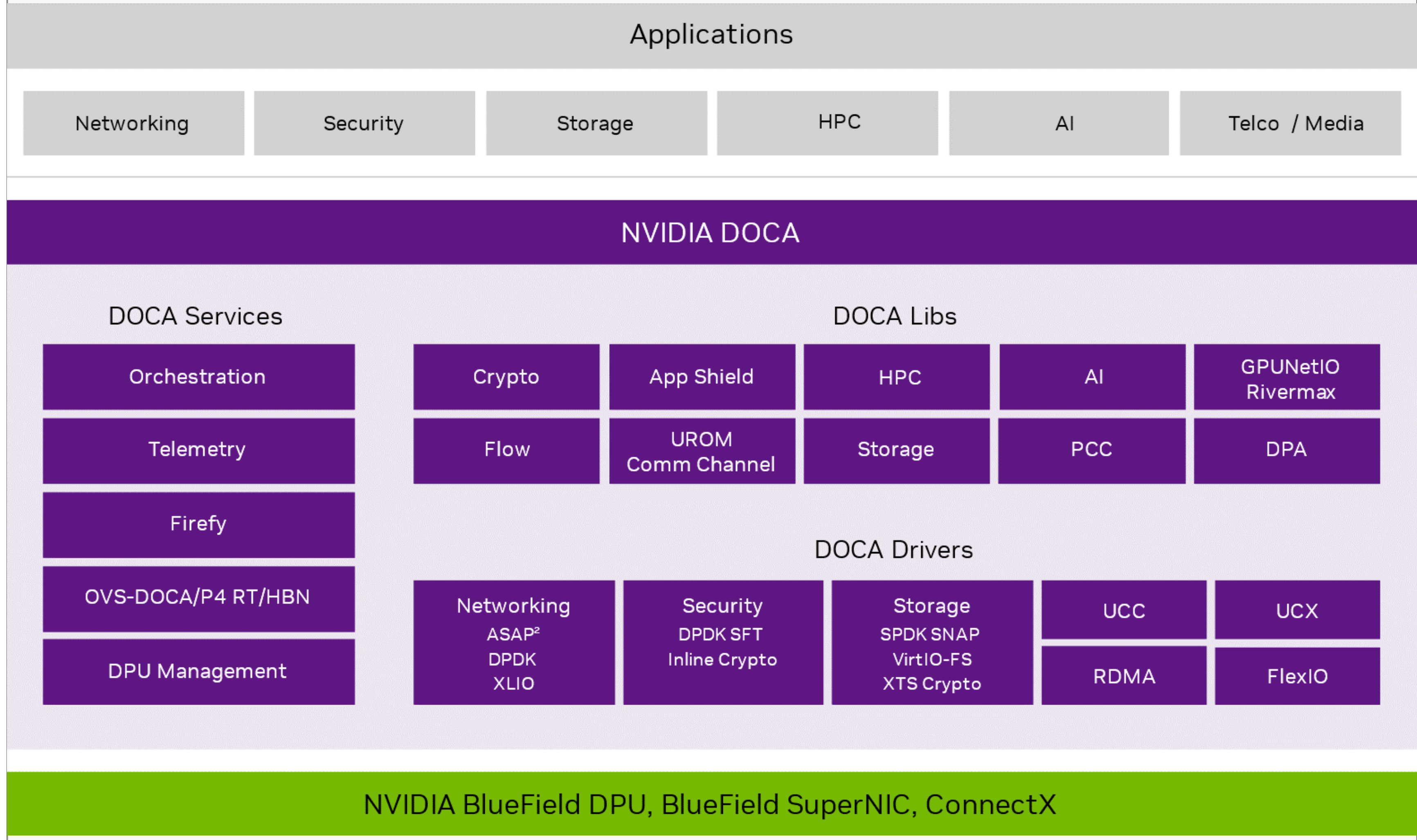
- DOCA 2.7
- DOCA installation
- DPU ARM OS and OVS
- DPU Flow Steering and Bond/LAG
- Virtio-net Hotplug for Bare-metal
- Virtio-net Static for Virtualization
- Q&A

DOCA 2.7

DOCA 2.7 Major Release Overview

Accelerating AI Cloud East-West and Spectrum-X

- Spectrum-X RA 1.0.1 with BlueField-3 SuperNIC
- DOCA Flow and OVS-DOCA Enhancements
- AI Cloud Traffic Encryption – IPsec GA, PSP support
- DOCA Library Introduction
  - UROM, Device Emulation
- DOCA Services Enhancement and Introduction
  - HBN, Firefly, UROM, DOCA Mgmt. Service (DMS)
- BlueField Platform Software
  - BlueField-3 Upgrade in NIC-Mode and NIC FW Upgrade
  - New BF Bundle Type – BF-FW Bundle
- DOCA Unified Package (MOFED compatible)

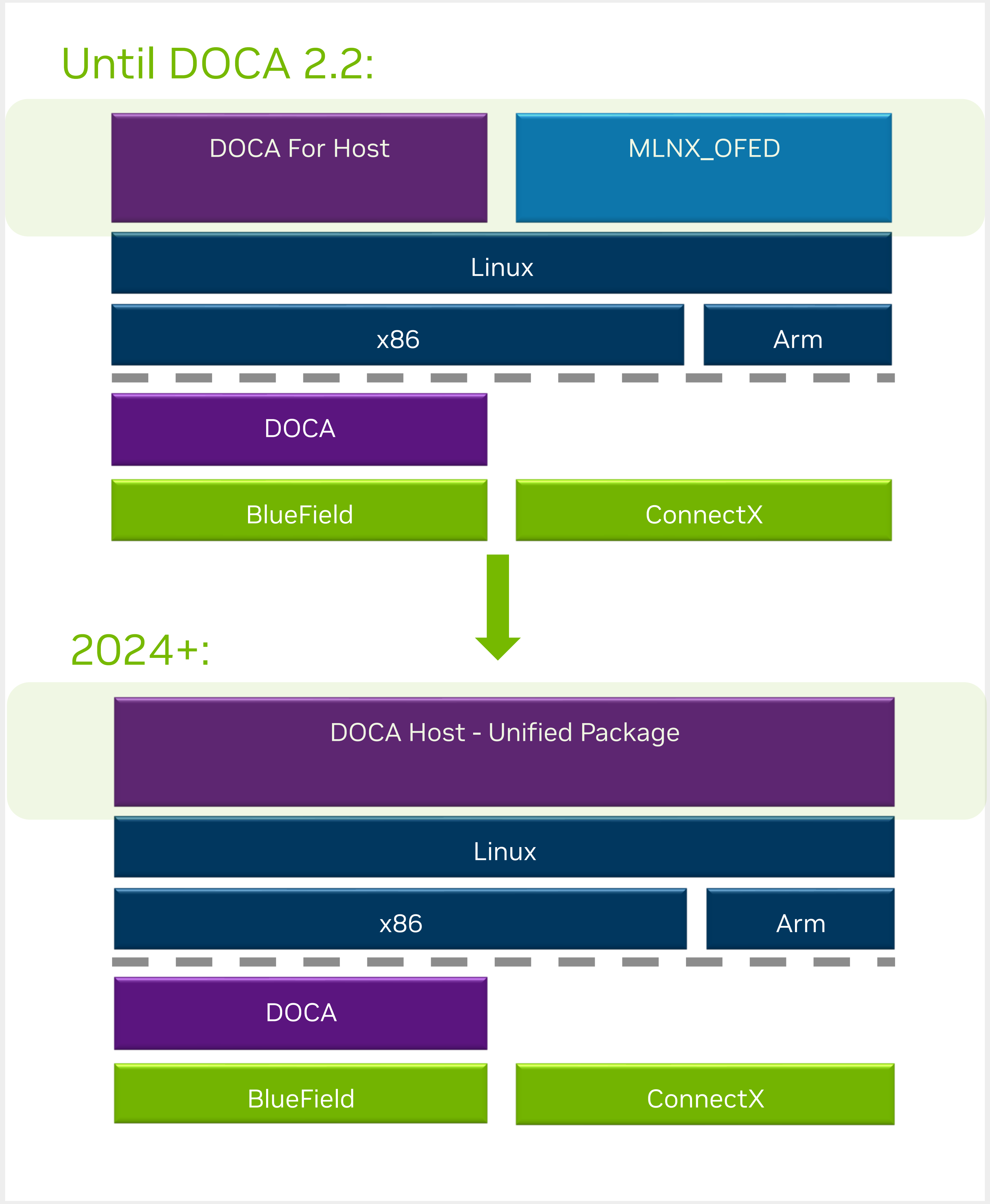


DOCA Profiles and Supported Devices

DOCA Unified Package

One-Stop-Shop for BlueField and ConnectX Devices

- DOCA-Host
  - Single SW package for the host
  - Replacing MLNX_OFED (last standalone release: Oct 2024)
- Since 2024+
  - DOCA-Host is a one-stop-shop supporting both BlueField and ConnectX devices
  - BlueField-3 and ConnectX-8 supported only in DOCA
  - Standard Linux package for host server
  - Host installation profiles: doca-all, doca-networking, doca-ofed
  - **Call for Action:** Promote transition to DOCA Package today
- Use Case and Customers
  - Use DOCA-OFED profile to get the same experience as MLNX-OFED
  - Use the same unified package on a server hosting both BlueField and ConnectX devices



New DOCA Download Pages

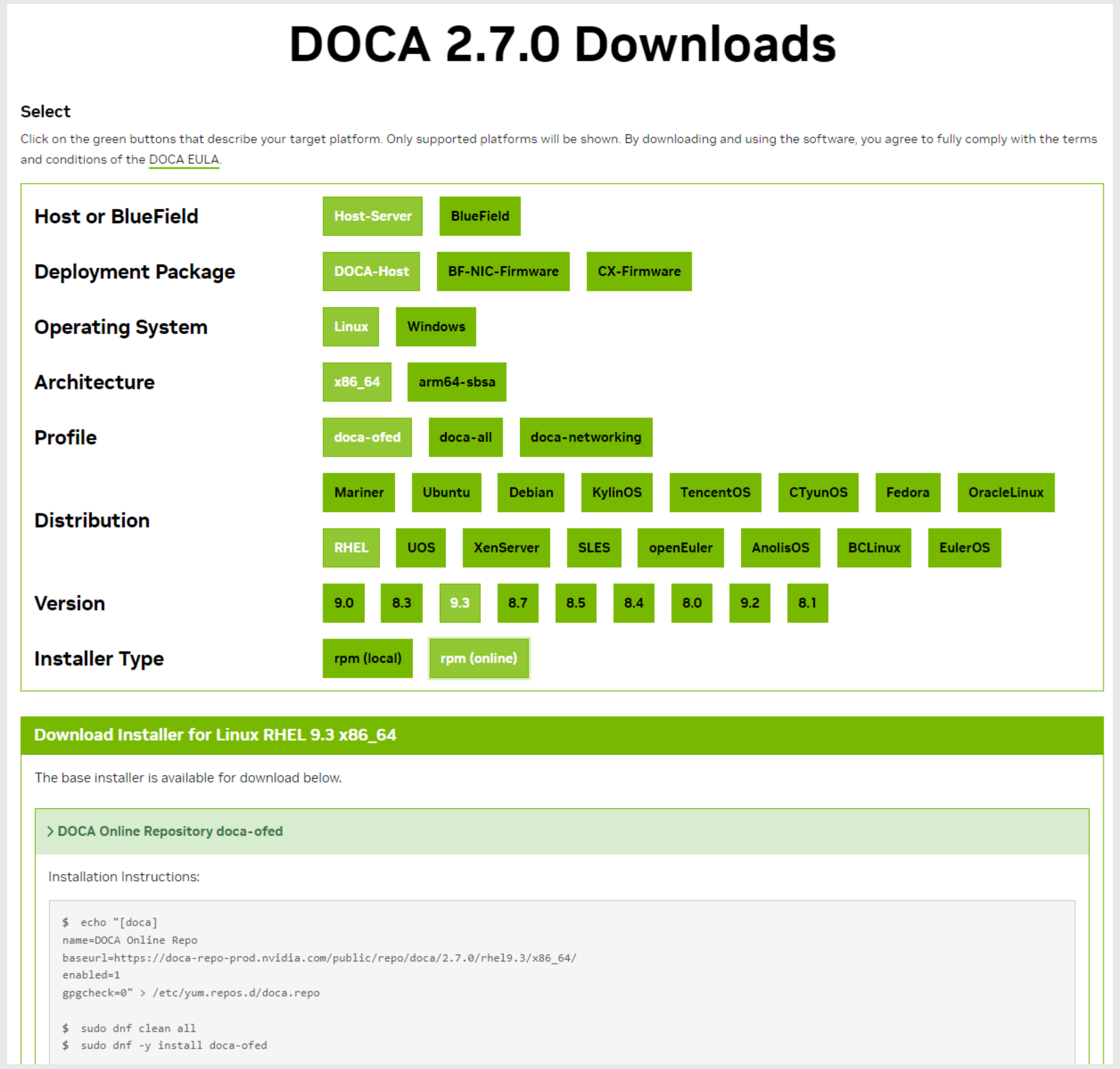
CUDA-like Download Experience

- DOCA for DPU is what CUDA is for GPU, now also similar download experience
- Remove table from web page, replace with dedicated page

Download DOCA

- Latest version always in developer.nvidia.com/doca-downloads
- LTS and Previous versions in developer.nvidia.com/doca-archive

DOCA Download Page



DOCA 2.7.0 Downloads

Select

Click on the green buttons that describe your target platform. Only supported platforms will be shown. By downloading and using the software, you agree to fully comply with the terms and conditions of the DOCA EULA.

Host or BlueField	[✓ Host-Server] BlueField
Deployment Package	[✓ DOCA-Host] BF-NIC-Firmware CX-Firmware
Operating System	[✓ Linux] Windows
Architecture	[✓ x86_64] arm64-sbsa
Profile	[✓ doca-ofed] doca-all doca-networking
Distribution	Mariner Ubuntu Debian KylinOS TencentOS CTyunOS Fedora OracleLinux [✓ RHEL] UOS XenServer SLES openEuler AnolisOS BCLinux EulerOS
Version	9.0 8.3 [✓ 9.3] 8.7 8.5 8.4 8.0 9.2 8.1
Installer Type	rpm (local) [✓ rpm (online)]

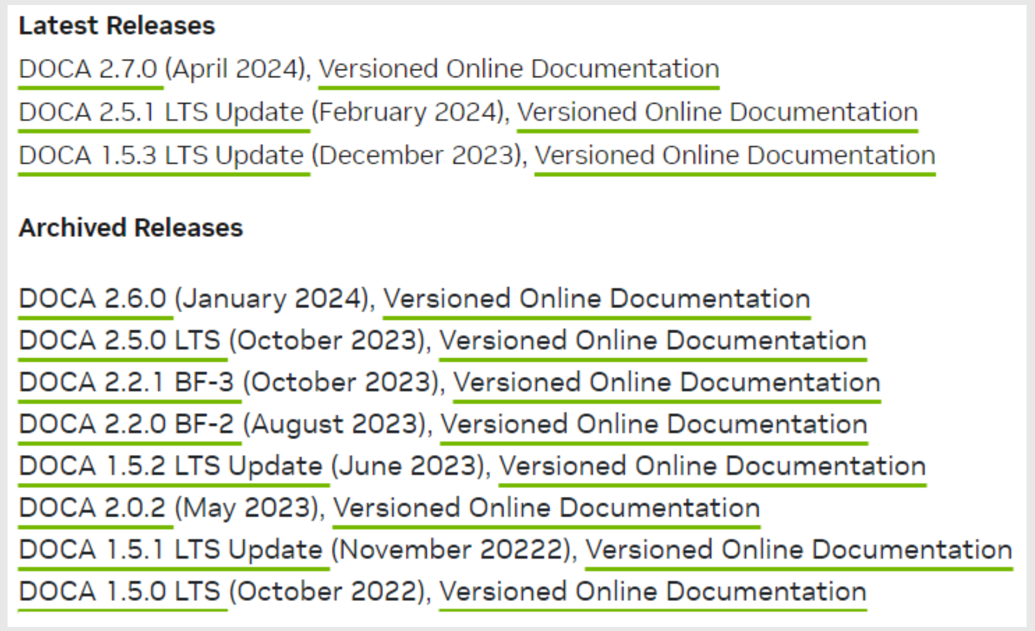
Download Installer for Linux RHEL 9.3 x86_64

The base installer is available for download below.

> DOCA Online Repository doca-ofed

Installation Instructions:

```
$  echo "[doca]
name=DOCA Online Repo
baseurl=https://doca-repo-prod.nvidia.com/public/repo/doca/2.7.0/rhel9.3/x86_64/
enabled=1
gpgcheck=0" > /etc/yum.repos.d/doca.repo

$  sudo dnf clean all
$  sudo dnf -y install doca-ofed
```



Latest Releases

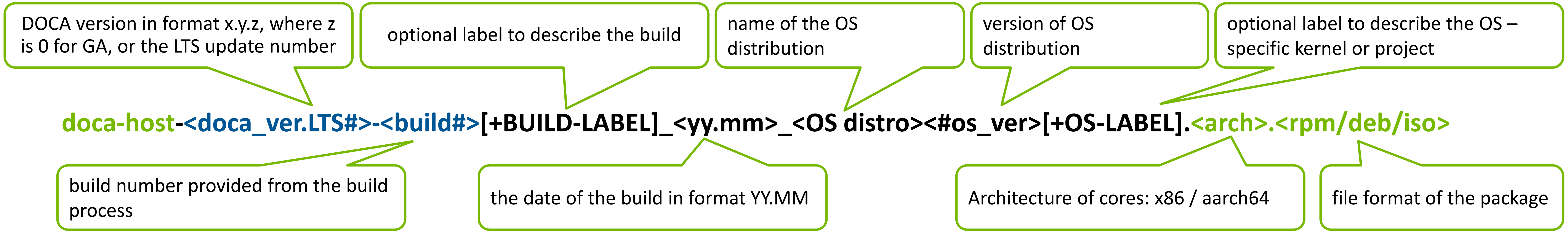
DOCA 2.7.0 (April 2024), Versioned Online Documentation
DOCA 2.5.1 LTS Update (February 2024), Versioned Online Documentation
DOCA 1.5.3 LTS Update (December 2023), Versioned Online Documentation

Archived Releases

DOCA 2.6.0 (January 2024), Versioned Online Documentation
DOCA 2.5.0 LTS (October 2023), Versioned Online Documentation
DOCA 2.2.1 BF-3 (October 2023), Versioned Online Documentation
DOCA 2.2.0 BF-2 (August 2023), Versioned Online Documentation
DOCA 1.5.2 LTS Update (June 2023), Versioned Online Documentation
DOCA 2.0.2 (May 2023), Versioned Online Documentation
DOCA 1.5.1 LTS Update (November 20222), Versioned Online Documentation
DOCA 1.5.0 LTS (October 2022), Versioned Online Documentation

DOCA-Host Image Name Format

doca-host


DOCA version in format x.y.z, where z is 0 for GA, or the LTS update number
optional label to describe the build
name of the OS distribution
version of OS distribution
optional label to describe the OS – specific kernel or project
build number provided from the build process
the date of the build in format YY.MM
Architecture of cores: x86 / aarch64
file format of the package

doca-host-<doca_ver.LTS#>-<build#>[+BUILD-LABEL]_<yy.mm>_<OS distro><#os_ver>[+OS-LABEL].<arch>.<rpm/deb/iso>

The above is for .rpm and .iso files.

For deb: doca-host_<doca_ver.LTS#>-<build#>[+BUILD-LABEL]-<yy.mm>-<OS distro><#os_ver no decimal point>[+OS-LABEL]_<arch>.deb

(to comply with rpm/deb parsing format)

Examples:
doca-host-2.6.0-1234+L0123_24.01_ubuntu2204+61.x86.rpm
LTS: doca-host-2.5.1-1250+L3_24.02_ubuntu2204+61.x86.rpm
FUR: doca-host-2.6.0-1344+MG54_24.03_ubuntu2204+61.x86.rpm

New BF Bundle Types

What's new in DOCA 2.7

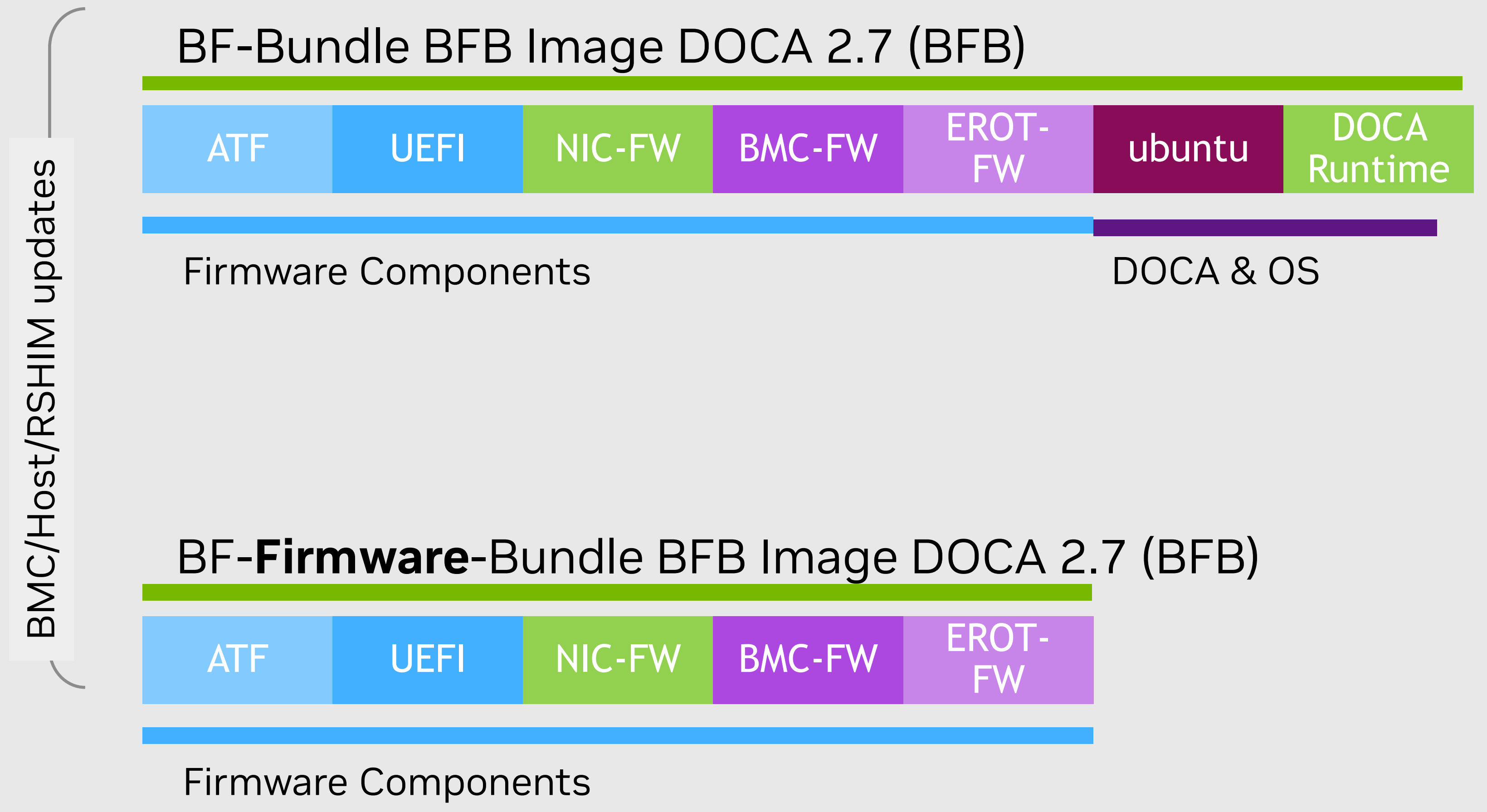
- BF-Bundle – default image
  - The full image with BlueField components, including – ATF, UEFI, nic-fw, bmc-fw, eROT, Ubuntu OS and DOCA runtime

NEW

- BF-Firmware Bundle (BF-FWBundle) - 'firmware only'
  - Smaller image for day2 upgrades – skipping the OS and DOCA upgrade
  - Includes – ATF, UEFI, nic-fw, bmc-fw, eROT
  - DOCA and OS can be separately upgraded using standard Linux tools – apt/yum

- Supported Host and DPU OS Distributions
  - https://docs.nvidia.com/networking/display/bluefielddpuosv470/supported+platforms+and+interoperability

DOCA Installation Guide



BlueField Bundle Image Name Format

bf-bundle

DOCA version in format x.y.z, where z is 0 for GA, or the LTS update number

optional label to describe the build

name of the OS distribution

version of OS distribution

optional label to describe the OS – specific kernel or project

bf-bundle-<doca_ver.LTS#>-<build#>[+BUILD-LABEL]_<yy.mm>_<OS distro><#os_ver>[+OS-LABEL]_<unsigned/dev/prod>.<bfb/iso>

build number provided from the build process

the date of the build in format YY.MM

signing of the image – unsigned / development / production

file format of the package

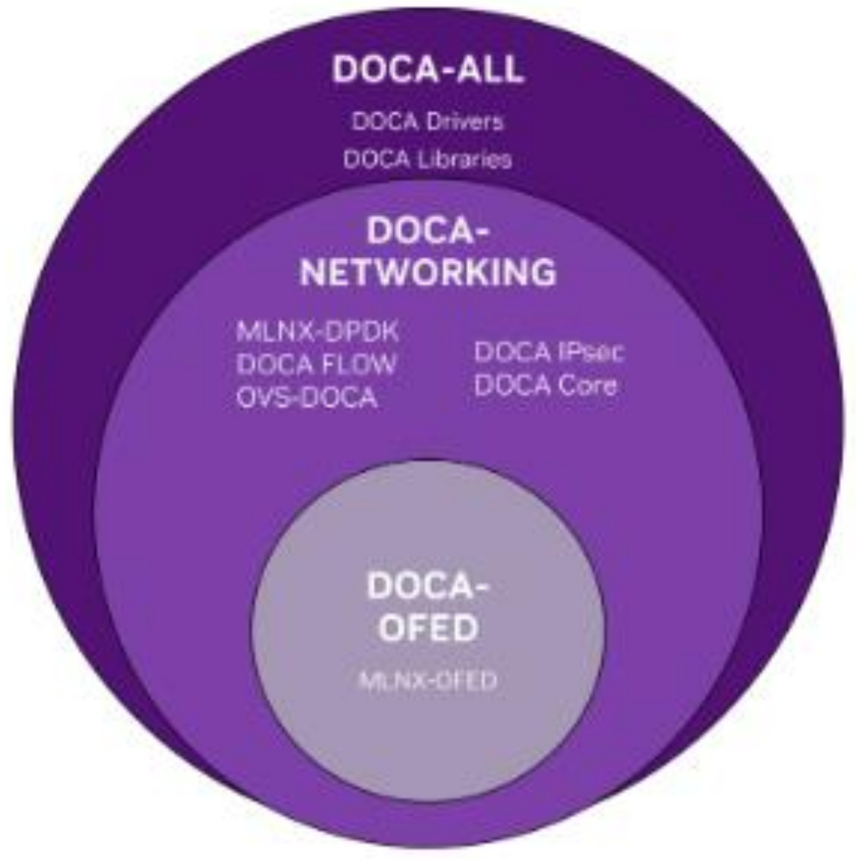
Examples:
bf-bundle-2.6.0-1737_0567-24.01-rhel-8.6_4.19_dev.iso
LTS: bf-bundle-2.5.**1**-1421_24.01_rhel86_prod.bfb
FUR on LTS: bf-bundle-2.5.1+**1422_**24.02_rhel86_prod.bfb
FUR: bf-bundle-2.5.0-1428+**X123**_24.03_rhel86_dev.iso

DOCA Installation

Installing DOCA on the Host

Host or BlueField	[✓ Host-Server]	BlueField				
Deployment Package	[✓ DOCA-Host]	BF-NIC-Firmware	CX-Firmware			
Operating System	[✓ Linux]	Windows				
Architecture	[✓ x86_64]	arm64-sbsa				
Profile	[✓ doca-all]	doca-networking	doca-ofed			
Distribution	[✓ Ubuntu]	RHEL	OracleLinux	Debian	ALinux	CTyunOS
Version	[✓ 22.04]	20.04				
Installer Type	[✓ deb (local)]	deb (online)				

DOCA functionality is limited by the specific device capabilities.



DOCA-HOST Profiles

DOCA Local Repository doca-all

```
wget https://www.mellanox.com/downloads/DOCA/DOCA_v2.7.0/host/doca-host_2.7.0-204000-24.04-ubuntu2204_amd64.deb
sudo dpkg -i doca-host_2.7.0-204000-24.04-ubuntu2204_amd64.deb
sudo apt-get update

sudo apt-get -y install doca-all mlnx-fw-updater
```

DOCA-OFED profile

- This profile is intended for users who wish to have the exact same user experience and content as MLNX_OFED but with DOCA Package. doca-ofed installs the MLNX_OFED drivers and tools but not other DOCA components.
- Note: The rshim package is included in OFED/this profile as well

- The content of the doca-ofed package is as follows:
  - MLNX_OFED drivers
  - MLNX_OFED tools

- Currently supported operating systems and kernel version per DOCA version and profiles are outlined on our DOCA SDK page (note the column for the doca-ofed profile)

https://docs.nvidia.com/doca/sdk/profiles/index.html#supported-host-os-per-doca-profile

- The doca-extra package, located under /opt/mellanox/doca/tools/, contains:
  - doca-info – displays details of all installed dependencies in DOCA
  - doca-kernel-support – if using a kernel not listed in the Supported Kernel Versions table, this tool downloads the appropriate missing packages for your kernel to support DOCA, unless the packages do not support the existing kernel.

Installing DOCA on Host

Verifying successful installation

- Verify the DOCA packages installed on the host.

```
host# dpkg --list| grep doca
ii   doca-apps                  1.5.1007-1                          amd64   DOCA-Based reference applications
ii   doca-apps-dev              1.5.1007-1                          amd64   Development files for DOCA Apps
ii   doca-grpc                  1.5.1007-1                          amd64   gRPC runtime capabilities for DOCA
ii   doca-grpc-dev              1.5.1007-1                          amd64   Development files for DOCA grpc
ii   doca-host-repo-ubuntu1804 1.5.1-0.1.8.1.5.1007.1.5.8.1.1.2.1 amd64   Doca repo bundle package
ii   doca-libs                  1.5.1007-1                          amd64   Data Center on a Chip Architecture (DOCA)
ii   doca-prime-runtime         1.5.1007-1                          amd64   DOCA prime runtime metapackage
ii   doca-prime-sdk             1.5.1007-1                          amd64   DOCA prime sdk metapackage
ii   doca-prime-tools           1.5.1007-1                          amd64   Runtime DOCA Tools
ii   doca-remote-memory-app     22.07.0                             amd64   Nvidia Bluefield regex benchmarking tool
companion remote memory app.
ii   doca-runtime               1.5.1-0.1.8                         amd64   doca-runtime meta-package
ii   doca-samples               1.5.1007-1                          amd64   DOCA Samples
ii   doca-sdk                   1.5.1-0.1.8                         amd64   doca-sdk meta-package
ii   doca-tools                 1.5.1-0.1.8                         amd64   doca-tools meta-package
ii   libdoca-libs-dev           1.5.1007-1                          amd64   Development files for DOCA Libs
```

Install DOCA on DPU

prepare

- Host side
  - Make sure to install host drivers
  - Check rshim status
- DPU side
  - Connect to your BlueField DPU
  - Check BSP version
  - Check NIC-FW/BMC-FW
- Host side
  - Download BFB Image

```
root@localhost:~# flint -d /dev/mst/mt41692_pciconf0 q
Image type:            FS4
FW Version:            32.40.1000
FW Release Date:       4.2.2024
Product Version:       32.40.1000
Rom Info:              type=UEFI Virtio net version=21.4.13 cpu=AMD64,AARCH64
                       type=UEFI Virtio blk version=22.4.12 cpu=AMD64,AARCH64
                       type=UEFI version=14.33.10 cpu=AMD64,AARCH64
                       type=PXE version=3.7.300 cpu=AMD64
Description:           UID                GuidsNumber
Base GUID:             a088c2030075c110        38
Base MAC:              a088c275c110            38
Image VSD:             N/A
Device VSD:            N/A
PSID:                  MT_0000000884
Security Attributes:   secure-fw


root@localhost:~# cat /etc/mlnx-release
DOCA_2.6.0_BSP_4.6.0_Ubuntu_22.04-5.24-01.prod

root@localhost:~# ipmitool mc info
Device ID                 : 1
Device Revision           : 1
Firmware Revision         : 23.04
IPMI Version              : 2.0
```

Install DOCA on DPU

Install BFB

- Host side
  - Run the bfb-install script while indicating the BFB image to install and the DPU device.

```
bfb-install --bfb bf-bundle-2.7.0-33_24.04_ubuntu-22.04_prod.bfb
--rshim /dev/rshim0
```

```
Pushing bfb
Collecting BlueField booting status. Press Ctrl+C to stop…
INFO[PSC]: PSC BL1 START
INFO[BL2]: UEFI loaded
INFO[UEFI]: eMMC init
INFO[UEFI]: eMMC probed
INFO[MISC]: Ubuntu installation started
INFO[MISC]: Installing OS image
INFO[MISC]: Ubuntu installation completed
WARN[MISC]: Skipping BMC components upgrade.
INFO[MISC]: Updating NIC firmware...
INFO[MISC]: NIC firmware upd
INFO[UEFI]: exit Boot Service
INFO[MISC]: Linux up
INFO[MISC]: DPU is ready
```

Install DOCA on DPU

Verify the new BFB version

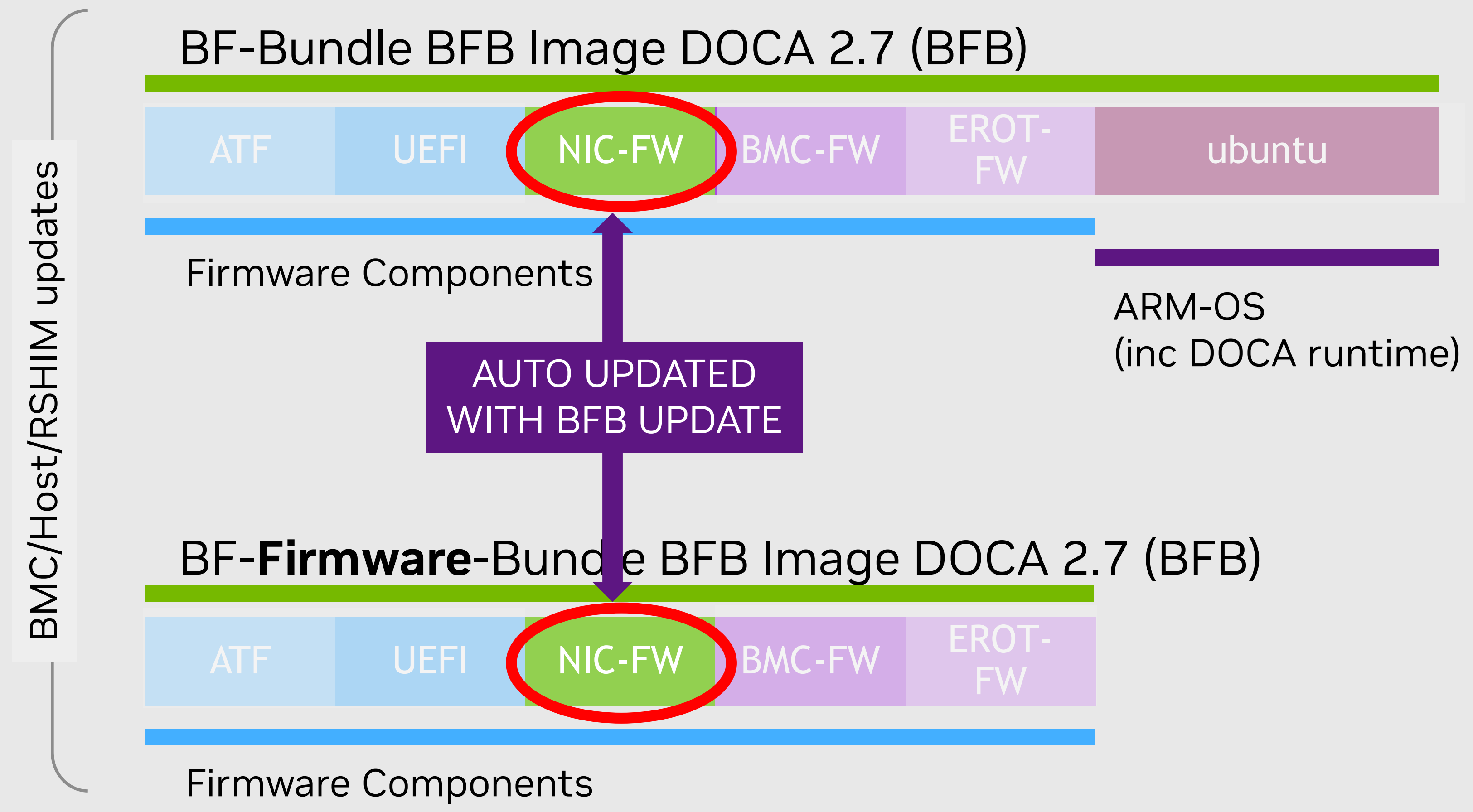
- DPU side
  - After BFB upgrade, the credentials are reset to their default.
  - Default BFB Ubuntu credentials:
    - User: ubuntu
    - Password: ubuntu
  - Check BSP/NIC FW/BMC version

```
ubuntu@localhost:~$ sudo flint -d /dev/mst/mt41692_pciconf0 q
Image type:            FS4
FW Version:            32.41.1000
FW Version(Running):   32.40.1000
FW Release Date:       28.4.2024
Product Version:       32.40.1000
Rom Info:              type=UEFI Virtio net version=21.4.13 cpu=AMD64,AARCH64
                       type=UEFI Virtio blk version=22.4.12 cpu=AMD64,AARCH64
                       type=UEFI version=14.33.10 cpu=AMD64,AARCH64
                       type=PXE version=3.7.300 cpu=AMD64
Description:           UID                GuidsNumber
Base GUID:             a088c2030075c110        38
Base MAC:              a088c275c110            38
Image VSD:             N/A
Device VSD:            N/A
PSID:                  MT_0000000884
Security Attributes:   secure-fw

ubuntu@localhost:~$ cat /etc/mlnx-release
bf-bundle-2.7.0-33_24.04_ubuntu-22.04_prod

ubuntu@localhost:~$ sudo ipmitool mc info
Device ID                 : 1
Device Revision           : 1
Firmware Revision         : 23.04
```

BFB NIC FW Update by Default

Keeping BlueField Components In Sync

- Keeping all BlueField components in sync to an official NVIDIA BlueField release is crucial for the stability of the product
- Starting April/24 DOCA 2.7 – any upgrade done with BFB image (full or 'small') will automatically upgrade the NIC-FW image
- Up to Apr/24 release – customers were required to take manual steps to burn BlueField NIC-FW – this is no longer needed



BMC/Host/RSHIM updates
BF-Bundle BFB Image DOCA 2.7 (BFB)
ATF
UEFI
NIC-FW
BMC-FW
EROT-FW
ubuntu
Firmware Components
ARM-OS (inc DOCA runtime)
AUTO UPDATED WITH BFB UPDATE
BF-**Firmware**-Bundle BFB Image DOCA 2.7 (BFB)
ATF
UEFI
NIC-FW
BMC-FW
EROT-FW
Firmware Components

* For information contact Product Management – Erez Scop

Customizations During BFB Installation

- The BlueField's UEFI system, boot options and more can be customized through the use of configuration parameters in the bf.cfg file.

Command: bfb-install --bfb DOCA_2.6.0_BSP_4.6.0_Ubuntu_22.04-5.24-01.prod.bfb --config bf.cfg --rshim rshim0

Tasks:

- Install target OS if UPDATE_DPU_OS=‘yes’(default)
- Update ATF and UEFI if UPDATE_ATF_UEFI=“yes”(default)
- Update BMC components: UPDATE_BMC_FW=“yes”(default)+UPDATE_CEC_FW=“yes”(default)
- Update NIC firmware if WITH_NIC_FW_UPDATE=“yes”(default)

- , Ubuntu users can provide a unique password that will be applied at the end of the BFB installation.
- Step1: Create password hash

```
# openssl passwd -1
Password:
Verifying - Password:
$1$3B0RIrfX$TlHry93NFUJzg3Nya00rE1
```

- Step2: Add the password hash in quotes to the bf.cfg file

```
# vim bf.cfg
ubuntu_PASSWORD='$1$3B0RIrfX$TlHry93NFUJzg3Nya00rE1'
```

Custom BFB

- How to build your own customized bfb
- GitHub - Mellanox/bfb-build: BFB (BlueField boot stream and OS installer) build environment

BFB/DOCA upgrade process - Today

Upgrade repo, Run:

- ❑ $ wget “doca-dpu-repo”
- ❑ $ dpkg -i “doca-dpu-repo”
- ❑ $ sudo apt-get update
- ❑ $ sudo apt-get upgrade

NOTICE: Today, needed download “doca-dpu-repo” from the artifactory location.

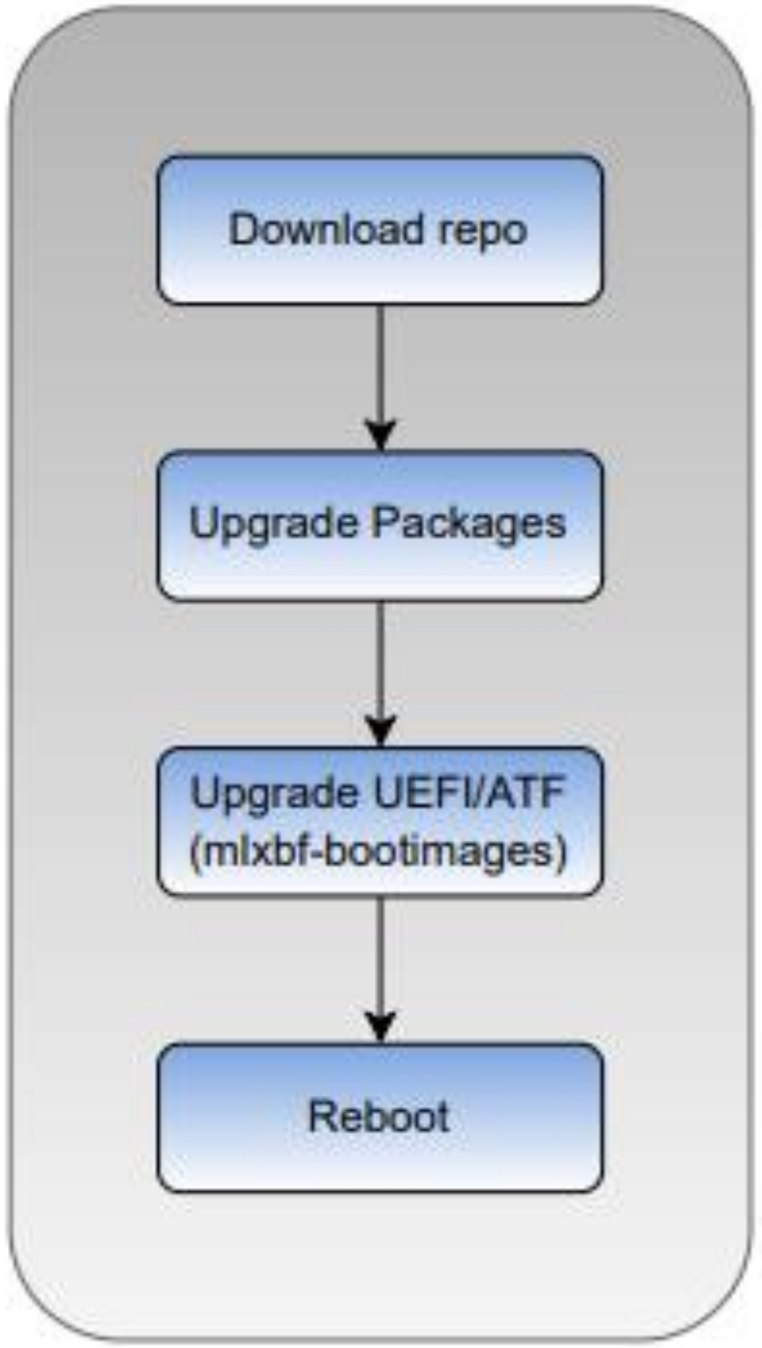
NVIDIA DOCA Installation Guide for Linux - NVIDIA Docs

Upgrade UEFI/ATF (included in mlxbf-bootimages DEB package) on the boot partition, Run:

- ❑ $ bfrec --bootctl --policy dual
- ❑ $ bfrec --capsule /lib/firmware/mellanox/boot/capsule/boot_update2.cap --policy dual
- ❑ $ reboot

Upgrade firmware, Run

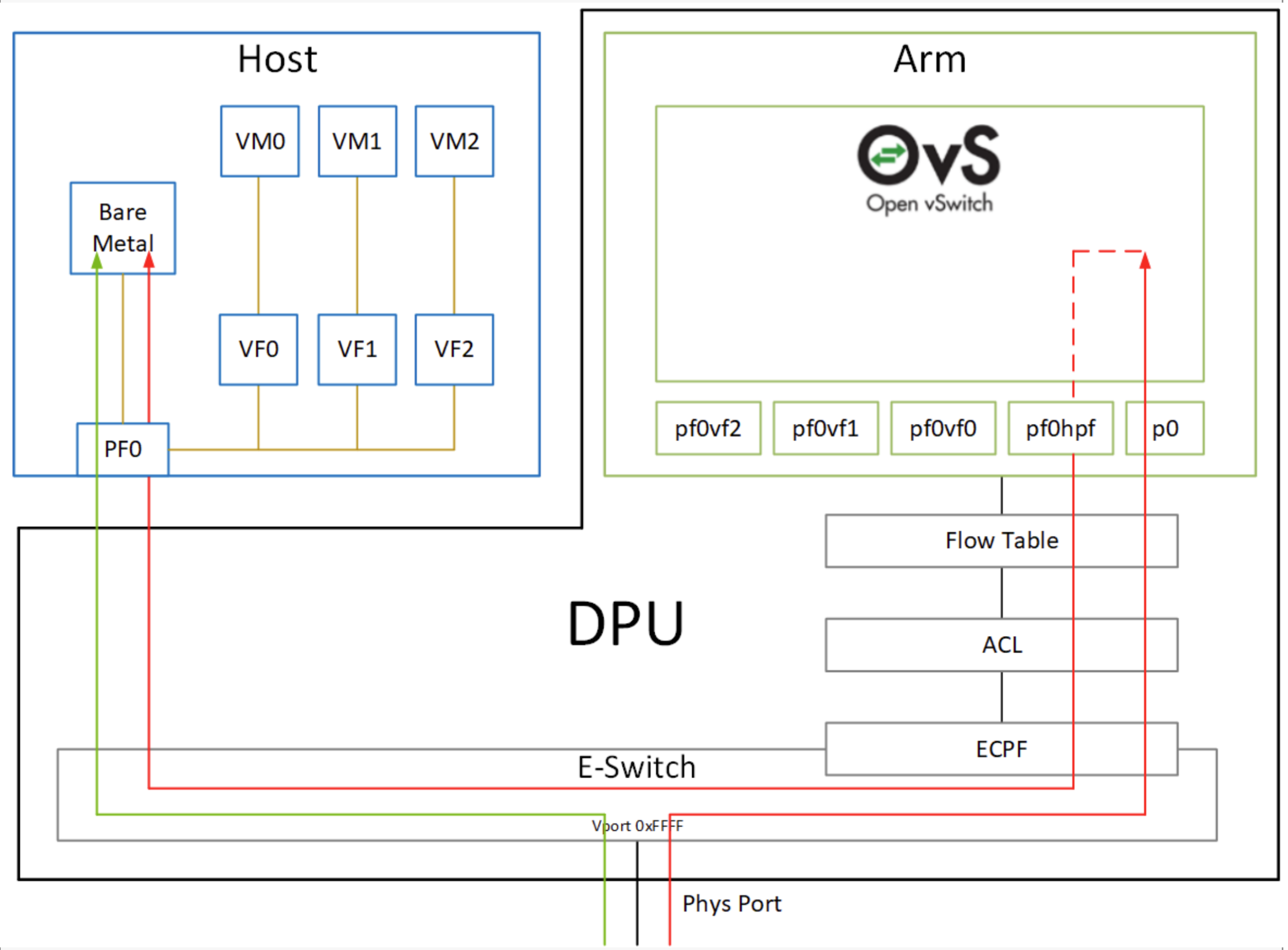
- ❑ dpu# sudo /opt/mellanox/mlnx-fw-updater/mlnx_fw_updater.pl --force-fw-update
- ❑ Check if the driver is supported sync 1
- ❑ dpu# sudo mlxfwreset -d /dev/mst/mt*_pciconf0 --sync 1 -y reset



DPU ARM OS and OVS

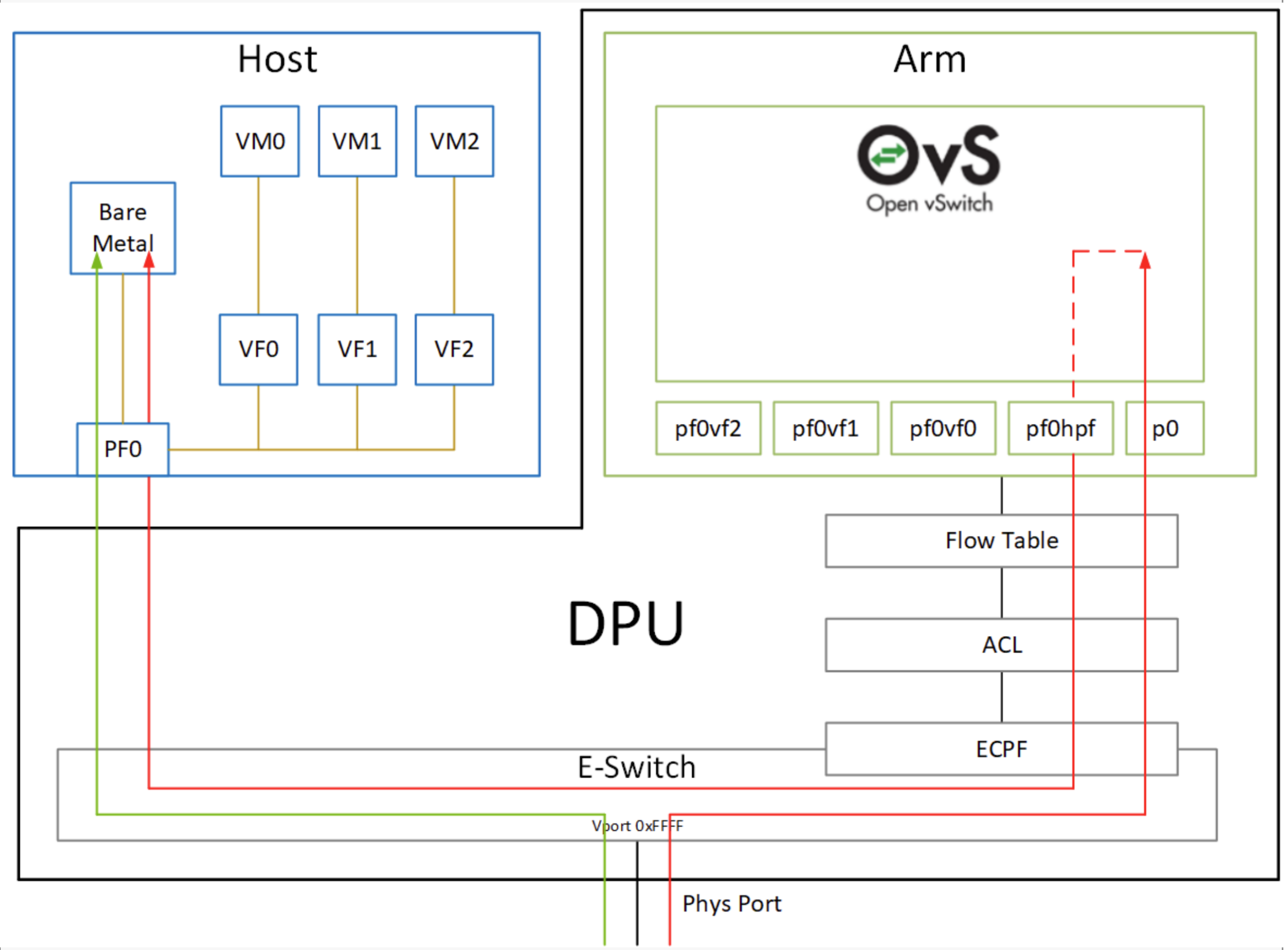
BlueField DPU Interfaces

- The following interfaces can be seen on the DPU Arm OS :
  - OOB interface: **oob_net0**.
    By default, IP address is assigned by DHCP.
  - RShim interface: **tmfifo_net0.**
    Pre-configured with a fixed IP address: 192.168.100.2.
  - DPU's PFs: **p0, p1**
  - Host PF representors: **pf0hpf, pf1hpf**
  - Host VF representors (if VFs are created on the host, one per VF): **pf0vf0, pf1vf0**…
  - Default SFs (one per PF): **enp3s0f0s0, enp3s0f1s0**
    - Default SF representors: **en3f0pf0sf0, en3f1pf1sf0**
  - Default OVS bridge ports: **ovsbr1, ovsbr2**
- Applicable only for DPU mode.



Kernel Representors Model

- BlueField DPU uses netdev representors to map each one of the host-side physical and virtual functions.
- These representors serve as:
  - The tunnel to pass traffic for the virtual switch or application running on the Arm cores to the relevant PF or VF on the host side.
  - The channel to configure the embedded switch with rules to the corresponding represented function.
- Those representors are used as the virtual ports being connected to OVS or any other virtual switch running on the Arm cores.



DPU PFs

Representors for DPU network ports

- Two representors are created for each one of the DPU's network ports:
  - For the uplink
  - For the host-side PF
- Naming convention for PF representors:
  - Uplink representor: p<port_number>
  - Host side PF representor: pf<port_number>hpf

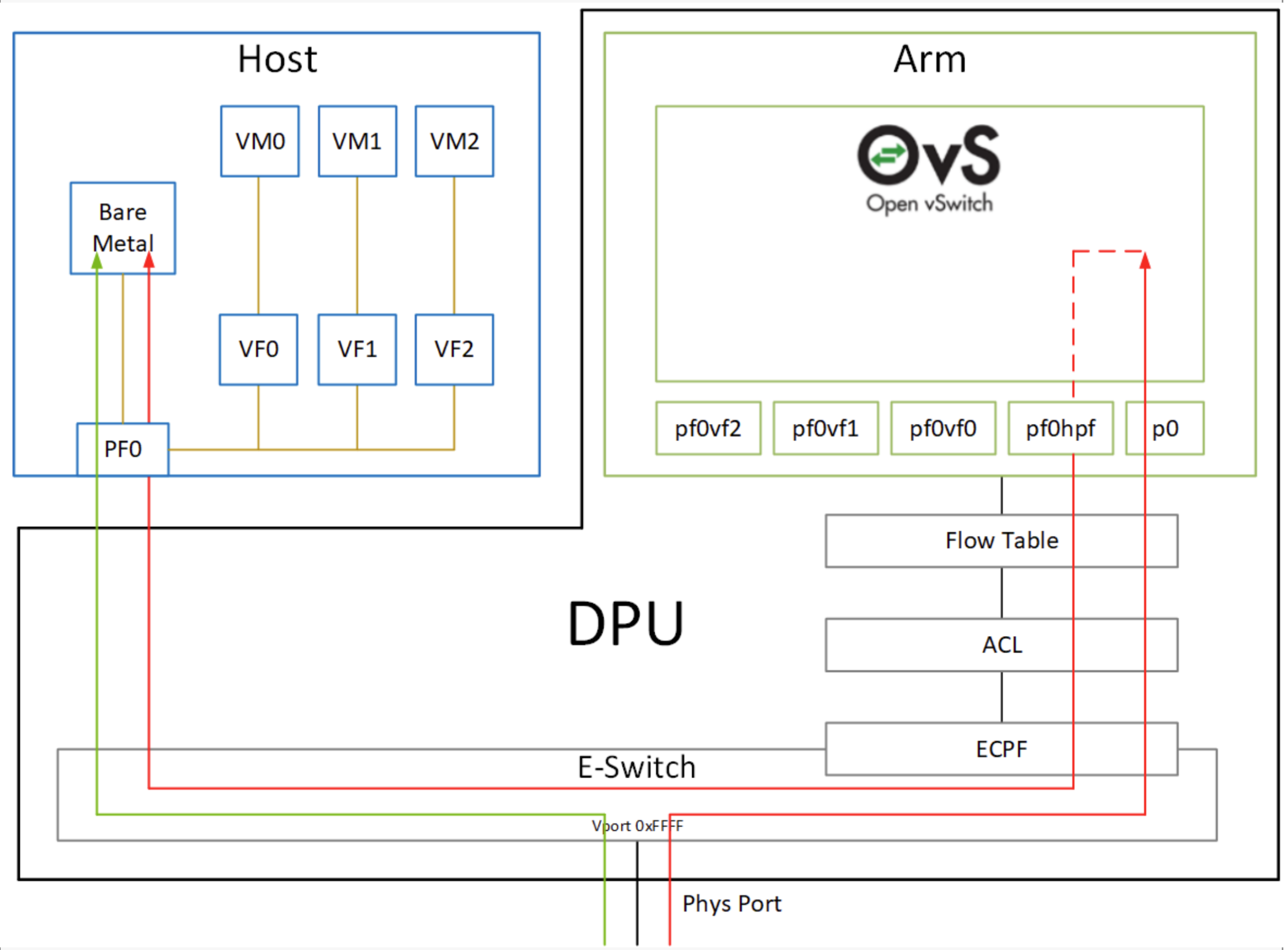
DPU PFs Representor Example

Representors for DPU network ports example

- Run the command “ip a” on the DPU to display a list of network representors:

```
dpu # ip a
4: p0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq master ovs-system state DOWN group default qlen 1000
    link/ether e8:eb:d3:ce:cc:ec brd ff:ff:ff:ff:ff:ff
5: p1: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq master ovs-system state DOWN group default qlen 1000
    link/ether e8:eb:d3:ce:cc:ed brd ff:ff:ff:ff:ff:ff
6: pf0hpf: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq master ovs-system state UP group default qlen 1000
    link/ether 7e:df:8b:a7:ef:e2 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::7cdf:8bff:fea7:efe2/64 scope link
       valid_lft forever preferred_lft forever
7: pf1hpf: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq master ovs-system state UP group default qlen 1000
    link/ether ba:b9:67:c1:e7:e1 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::b8b9:67ff:fec1:e7e1/64 scope link
       valid_lft forever preferred_lft forever
```

- The DPU’s first network port is represented by:
  - Uplink representor: `p0`
  - Host side PF representor: `pf0hpf`
- The DPU’s second network port is represented by:
  - Uplink representor: `p1`
  - Host side PF representor: `pf1hpf`



Host VFs Representors

- For each of the VFs created on the host-side, a corresponding representor is created on the DPU Arm-side.
- Naming convention for VF representors:
  - pf<port_number>vf<function_number>

Host VFs Representor Example

- Create VFs on the host:

Number of VFs to create

```
host # echo 2 > /sys/class/net/p1p1/device/sriov_numvfs
```

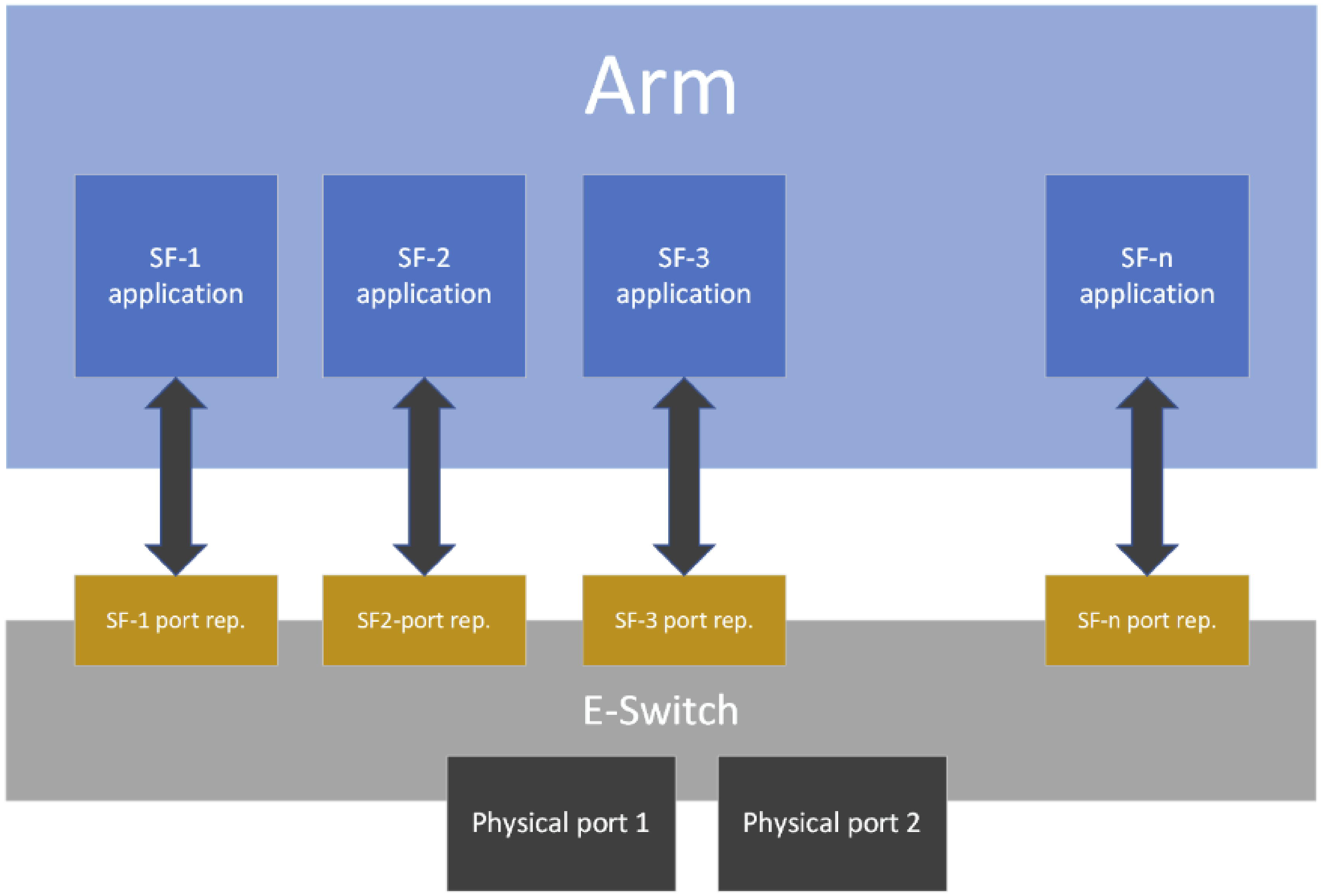
- Display the list of network representors on the DPU:

Representor for VF0

Representor for VF1

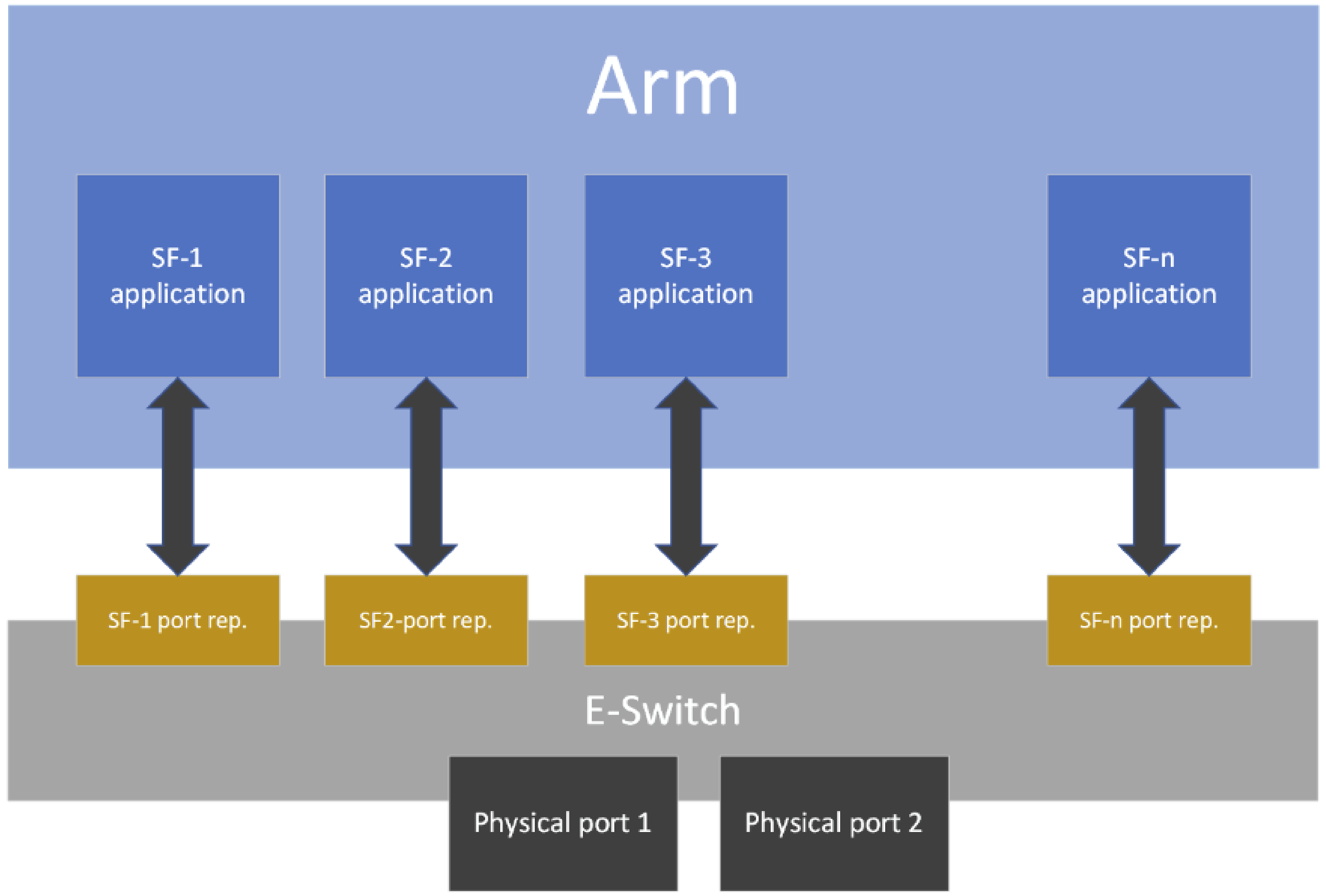
```
Dpu # ip a

16: pf0vf0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 22:f0:f2:cc:d4:a3 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::20f0:f2ff:fecc:d4a3/64 scope link
       valid_lft forever preferred_lft forever
17: pf0vf1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 66:48:3d:87:59:c4 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::6448:3dff:fe87:59c4/64 scope link
       valid_lft forever preferred_lft forever
```

- For each VF created on the host, a corresponding representor is created on the DPU.



Scalable Functions (SFs)

- SFs are very similar to VFs which are part of the SR-IOV solution.
- An SF is a lightweight function which has a parent PCIe function on which it is deployed.
- The SF has access to the capabilities and resources of its parent PCIe function, its own function capabilities, and its own resources.
- SFs co-exist with PCIe SR-IOV virtual functions (on the host) but do not require enabling PCIe SR-IOV.
- SFs can be used as:
  - Internal interface in Arm to use between Arm and the external world.
  - RoCE transport between host and Arm and Arm and the external host (for SNAP etc.).



SF Representors

- One SF per DPU PF is created
  - Default SFs: **enp3s0f0s0, enp3s0f1s0**
  - Defaults SF representors: **en3f0pf0sf0, en3f1pf1sf0**
- Additional SFs can be created on the DPU by the user.

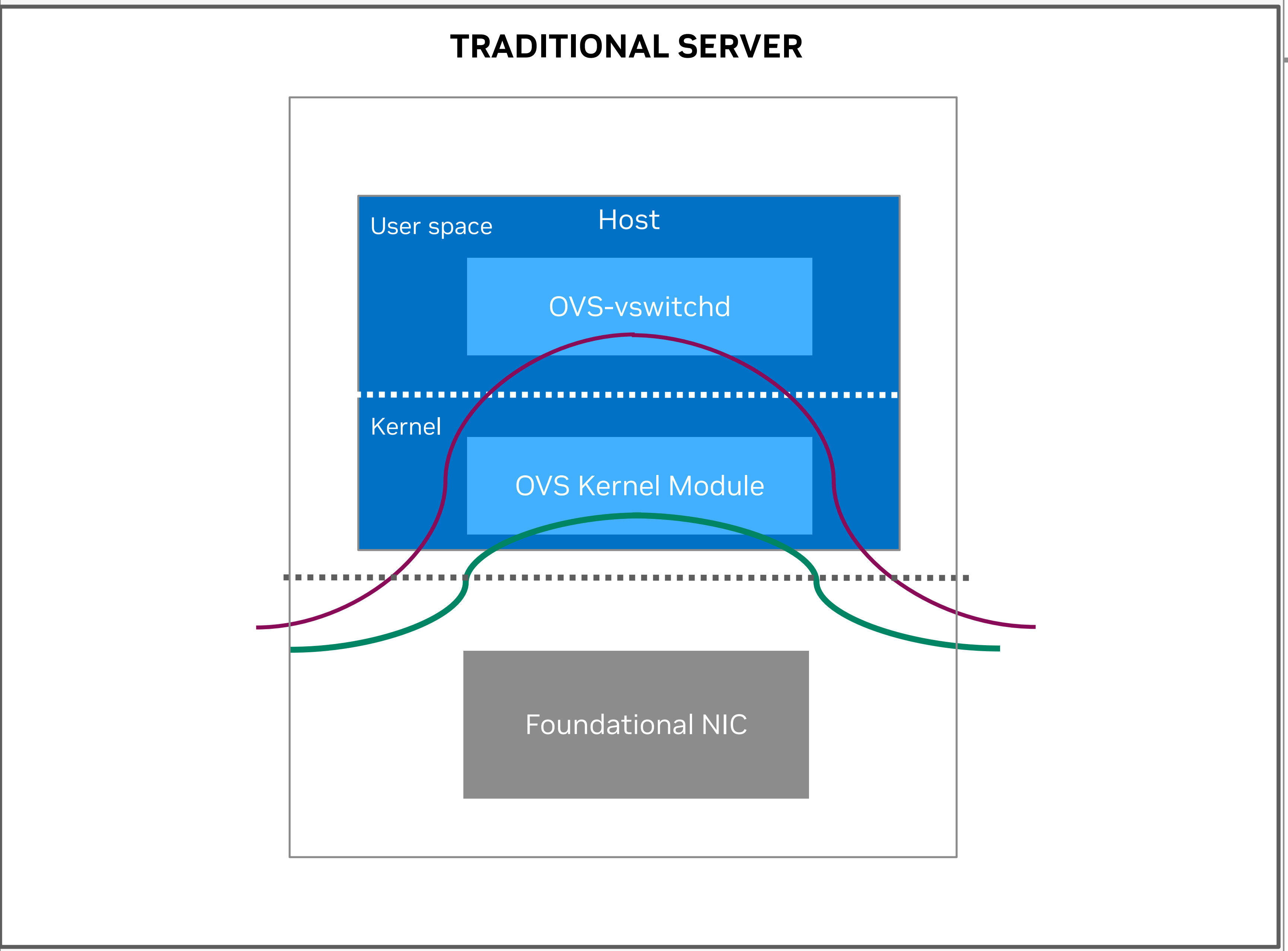
SFs Representors Example

- Run the command “ip a” on the DPU to display a list of network representors.

The first DPU’s PF:
- SF: enp3s0f0s0
- SF representor: en3f0pf0sf0

The second DPU’s PF:
- SF: enp3s0f1s0
- SF representor: en3f1pf1sf0

```
dpu# ip a
8: en3f0pf0sf0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq master ovs-system state UP
group default qlen 1000
    link/ether 8a:45:b3:56:6d:75 brd ff:ff:ff:ff:ff:ff
9: enp3s0f0s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 02:04:5f:68:ef:1e brd ff:ff:ff:ff:ff:ff
    inet6 fe80::4:5fff:fe68:ef1e/64 scope link
       valid_lft forever preferred_lft forever
10: en3f1pf1sf0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq master ovs-system state UP
group default qlen 1000
    link/ether 7a:b5:4c:0c:37:55 brd ff:ff:ff:ff:ff:ff
11: enp3s0f1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen
1000
    link/ether 02:7e:f1:e2:c4:a2 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::7e:f1ff:fee2:c4a2/64 scope link
       valid_lft forever preferred_lft forever
```



Software-only OVS Implementation

High Latency, Low Bandwidth, CPU-Intensive

🟪	First packet in a flow
🟩	Fallback FRWD path

Open vSwitch (OVS)

- Open vSwitch (OVS) is an open-source software switch that allows VMs and containers to communicate with each other and with the outside world.
- OVS software-only solutions are CPU-intensive, affecting system performance and providing poor network bandwidth and higher latency.

DPU OVS Bridges

- Two OVS bridges are created on the DPU by default after BFB installation.
- Default configuration allows default out-of-the box connection from the host to the DPU and from the host to the outside world.
- Default OVS bridges:
  - **ovsbr1** – allows traffic from the host to the DPU and from the host to the outside world via the first network port.
  - **ovsbr2** – allows traffic from the host to the DPU and from the host to the outside world via the second network port
- Default OVS bridge configuration is customizable.
- Additional bridges can be created.
- Different bridge methods can be used instead of OVS (e.g., DPDK).

DPU OVS Default Configuration

- Run the following command on the DPU to display OVS configuration:

- Bridge **ovsbr1** bridges the representors of the first network port, PF0:
  - **p0** – uplink representor
  - **pf0**hpf – host-side representor
  - en3f0**pf0**sf0 – SF representor

- Bridge **ovsbr2** bridges the representors of the second network port, PF1:
  - **p1** – uplink representor
  - **pf1**hpf – host side representor
  - en3f1**pf1**sf1 – SF representor

```
dpu # ovs-vsctl show
ea42c6e8-83e9-4449-87f6-
8a1969e1f0ea
    Bridge ovsbr1
        Port p0
            Interface p0
        Port pf0hpf
            Interface pf0hpf
        Port ovsbr1
            Interface ovsbr1
                type: internal
        Port en3f0pf0sf0
            Interface en3f0pf0sf0
    Bridge ovsbr2
        Port p1
            Interface p1
        Port ovsbr2
            Interface ovsbr2
                type: internal
        Port pf1hpf
            Interface pf1hpf
        Port en3f1pf1sf0
            Interface en3f1pf1sf0
    ovs_version: "2.17.2-9e69ff4"
```

Representor Interfaces

- As noted in previous slides, the DPU OS contains representor interfaces when in ethernet mode for the host OS physical interfaces and virtual function interfaces
- These representor interfaces serve as a tunnel to pass traffic to represented host physical or virtual interface
- It is important to verify OVS (open virtual switch) on the DPU OS side is set correctly to act as the switch/bridge to pass traffic to the host interfaces as needed
- "ovs-vsctl show" can be ran from the DPU OS side to verify this

- If an interface is inside an OVS bridge, the OVS bridge owns the interface
- Assigning an IP to an interface in an OVS bridge will not work. You must either remove that interface from the OVS bridge if possible, or assign an IP to the OVS bridge itself

https://docs.openvswitch.org/en/latest/faq/issues/

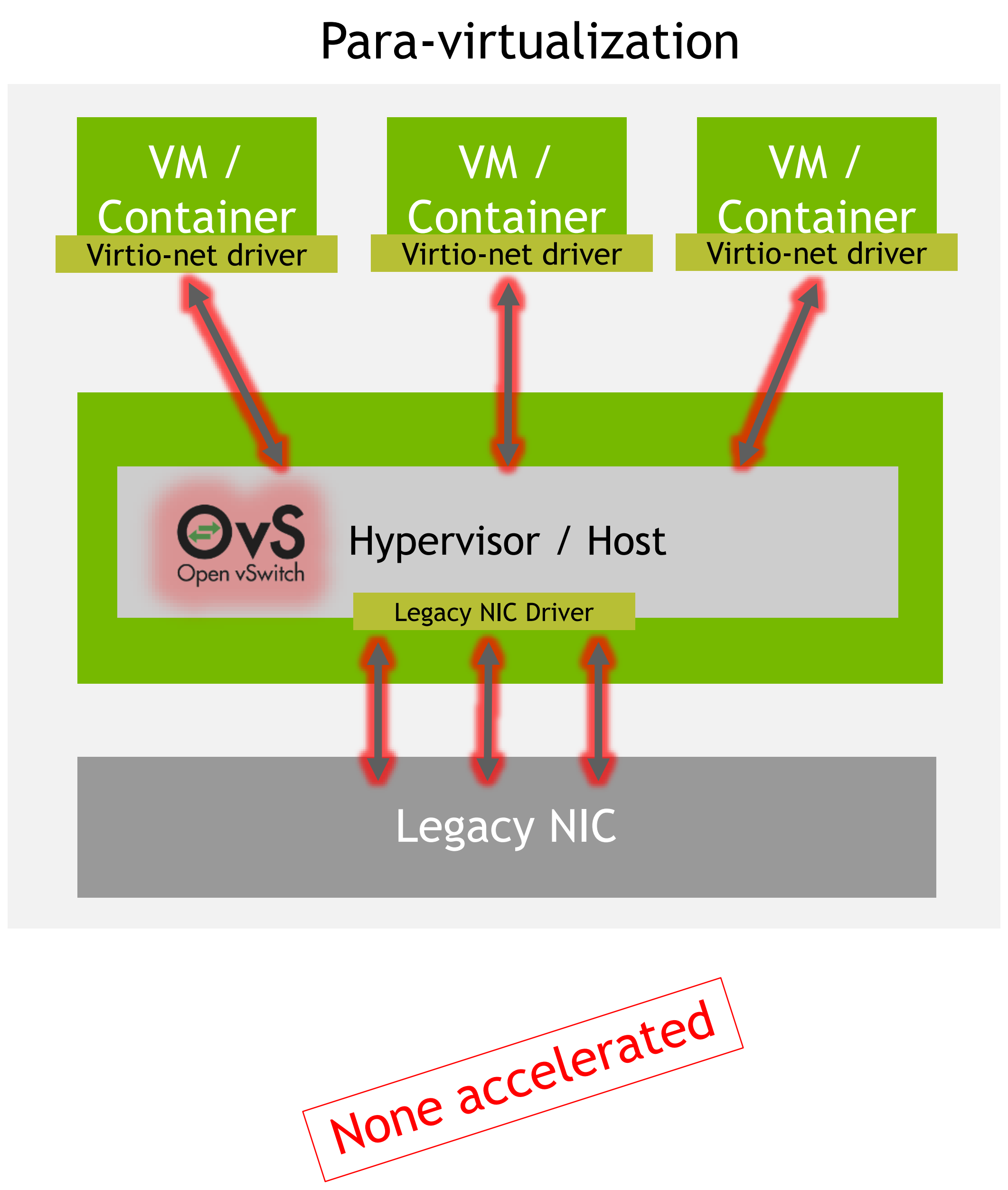
DPUP Flow Steering and Bond/LAG

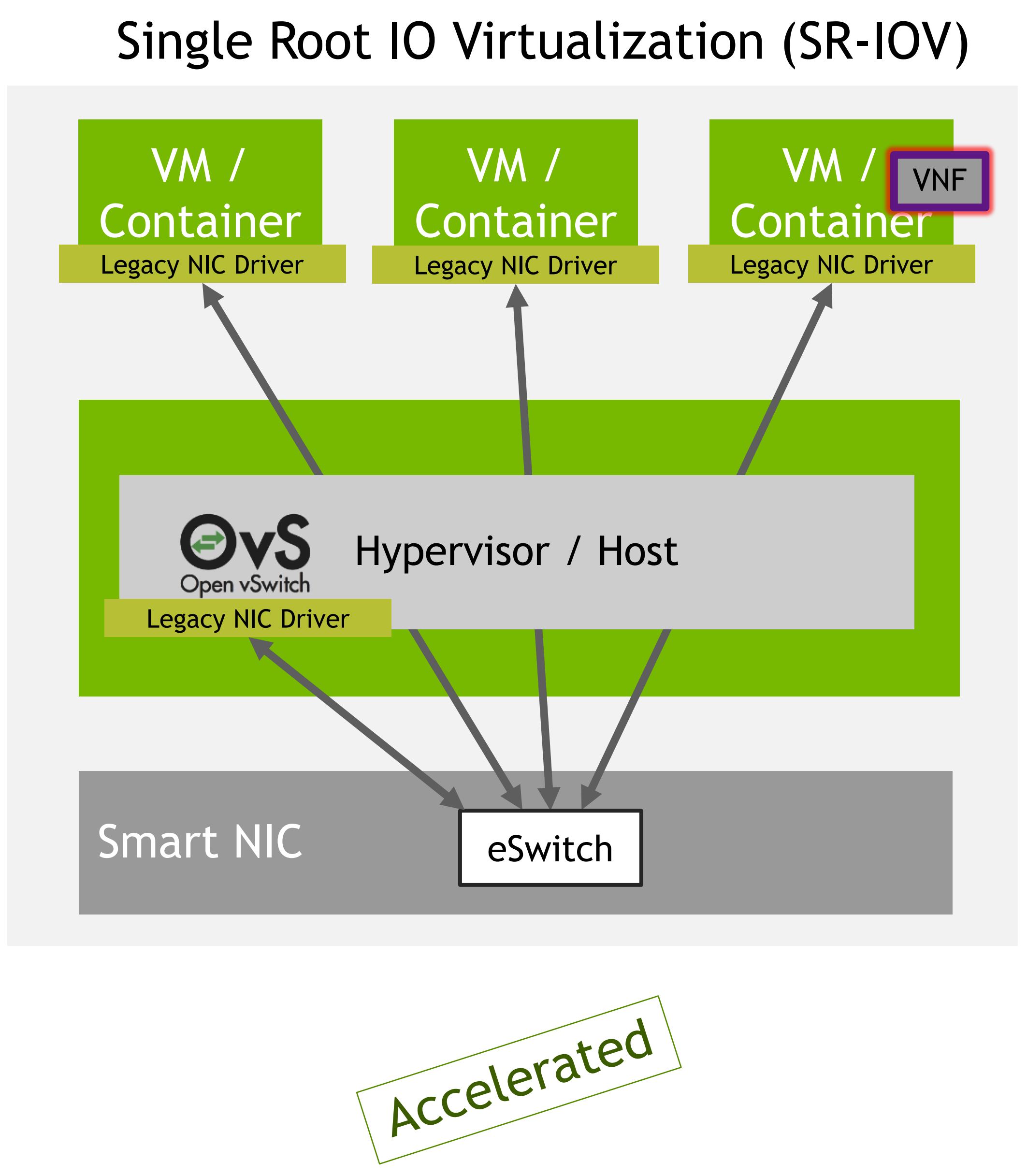
SWS Usage Scenarios

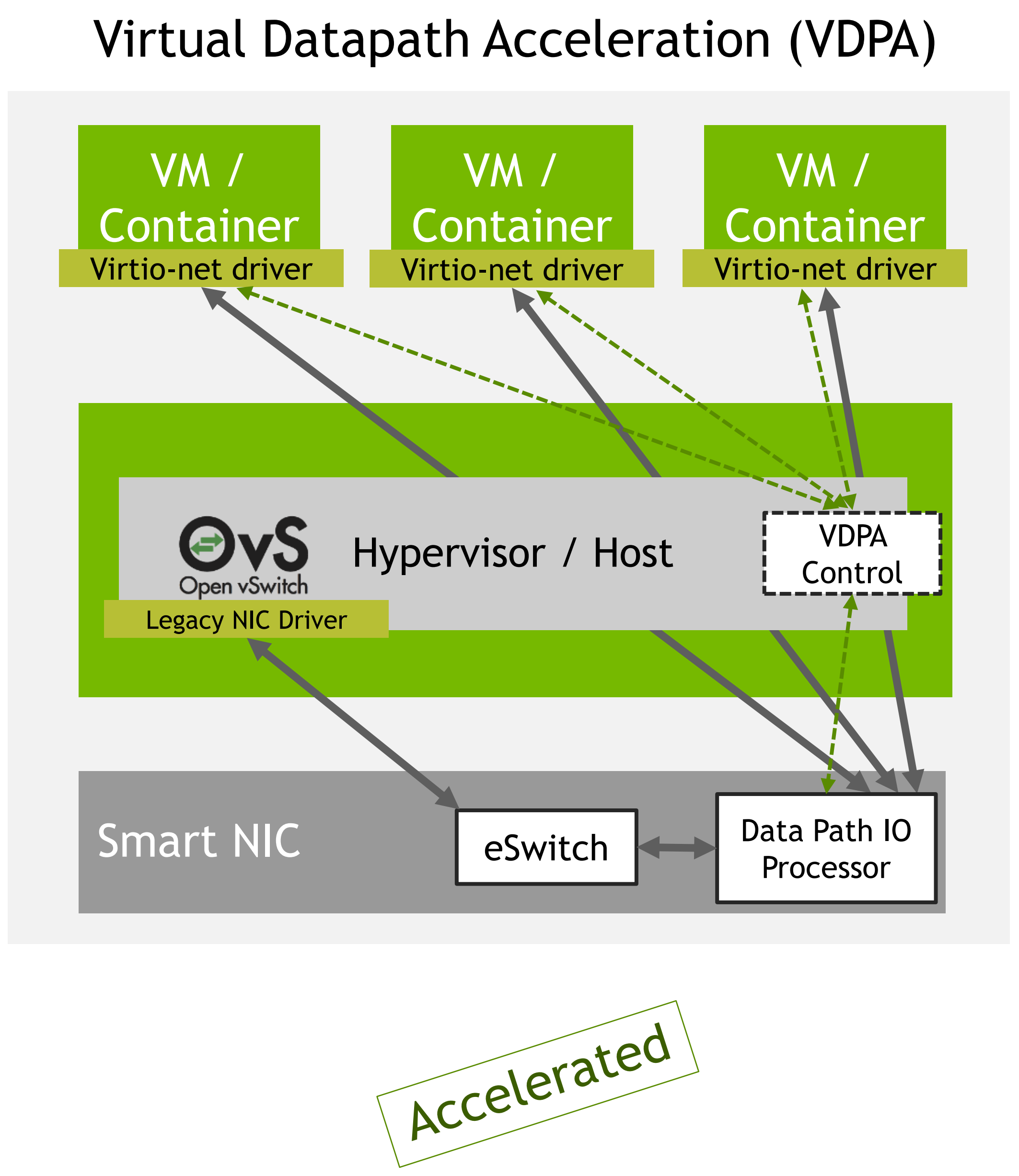
Cloud and GTW

应用场景：

- 云虚拟化和管理， DPDK, ovs-kernel, ovs-dpdk，加速流卸载
- 云网关







Flexible steering

Supported Actions (sample)

Forwarding decision
- Drop / allow
- Mirroring & monitor queue

Counter set
- About 8M counters

VLAN push / pop
- Packets coming from the host / network (respectively)
- Up to 2 VLANs

MPLS push / pop
- Packets coming from the host / network (respectively)
- Up to 5 labels

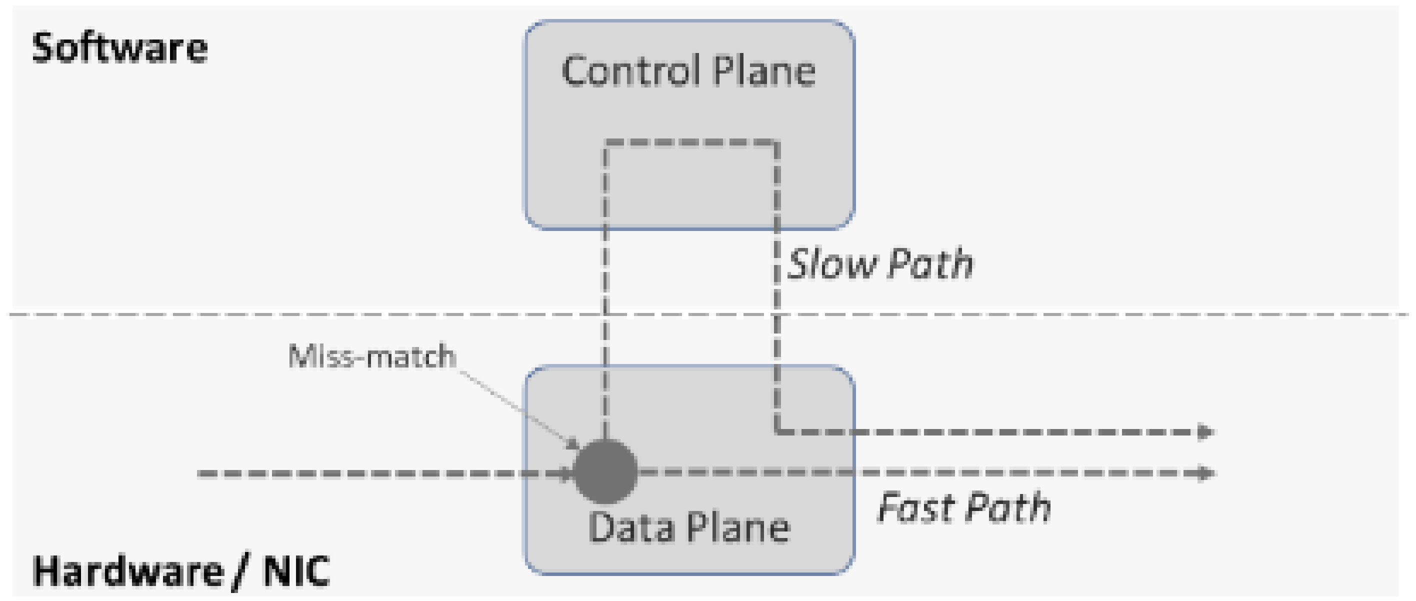
Hairpin
- Forward the packet back to the network

L2/L3 tunnel encapsulation / de-capsulation
- Packets coming from the host / network (respectively)

Header rewrite
- Update a specific header field
- Copy one header field to another
- Add a value to a specific header field

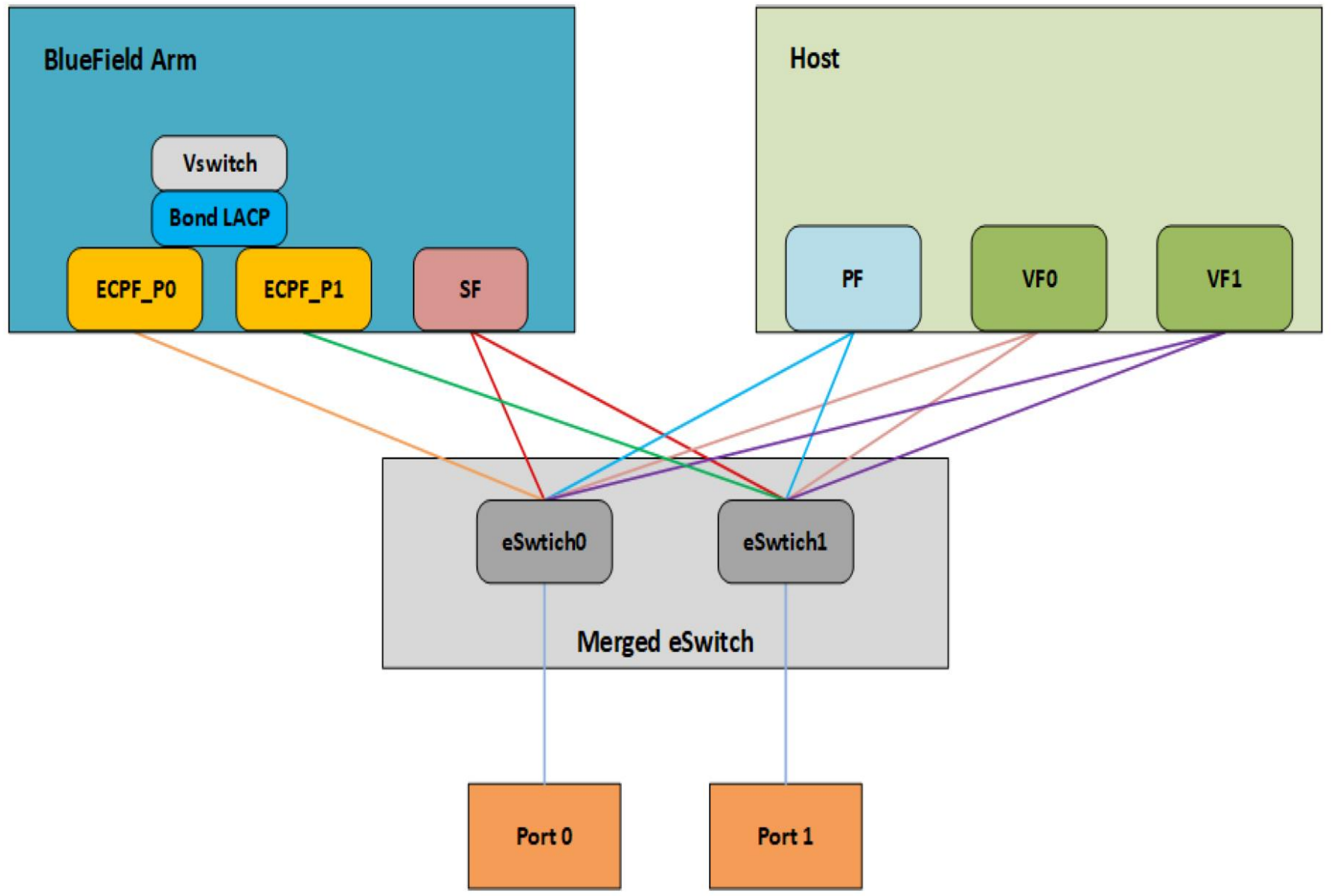
Metadata
- SW ←→ NIC

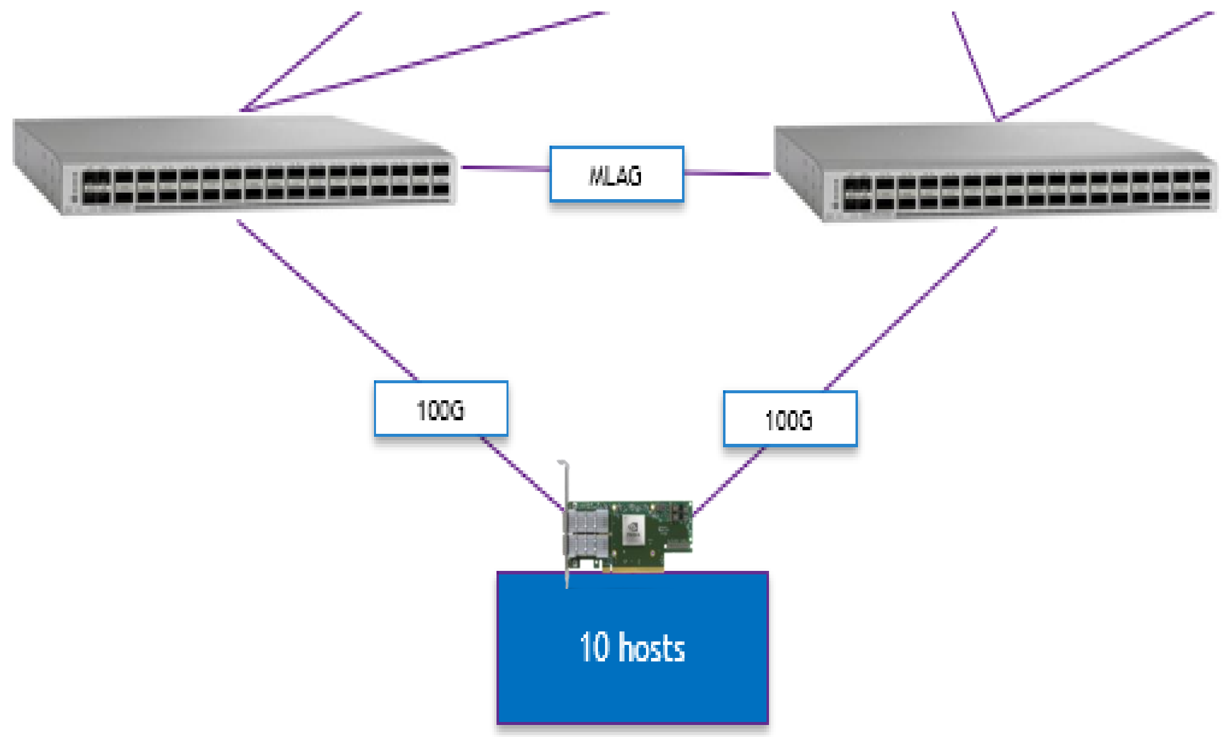
Connection Tracking
Encryption - IPsec & TLS
Generic header Push/Pop
Data policing/ Meters at scale (millions)



Link Aggregation Configure

What's LAG





- Combine two ports e-switch into one, no need to do bond on host
- Host devices (PF/VF/SF, virtio-xxx) have LAG properties for bandwidth and fail-over on uplink ports
- Only supported in DPU mode

Link Aggregation Configure

LAG Mode Configure

1. Firmware set

```
mlxconfig -d /dev/mst/mt41686_pciconf0 s
LAG_RESOURCE_ALLOCATION=1

mlxconfig -d /dev/mst/mt41686_pciconf0 s
HIDE_PORT2_PF=True NUM_OF_PF=1
```

2. Hash mode

```
Add in /etc/mellanox/mlnx-bf.conf for LAG
configuration
    LAG_HASH_MODE="yes"
```

3. Delete the existing OVS

```
ovs-vsctl del-port  ovsbr1  en3f0pf0sf0
ovs-vsctl del-port  ovsbr1  p0
ovs-vsctl del-port  ovsbr1  pf0hpf
ovs-vsctl del-port  ovsbr2  p1
ovs-vsctl del-port  ovsbr2  en3f1pf1sf0
ovs-vsctl del-port  ovsbr2  pf1hpf
```

4. LACP bonding configure

```
root@demo1:~# cat /etc/netplan/70-mlnx.yaml
network:
  renderer: networkd
  ethernets:
    p0:
      dhcp4: no
    p1:
      dhcp4: no
  bonds:
    bond0:
      interfaces: [p0,p1]
      dhcp4: no
      parameters:
        mode: 802.3ad
        transmit-hash-policy: layer3+4
        mii-monitor-interval: 100
  version: 2
```

5. Power cycle

```
ipmitool power cycle
```

Link Aggregation Configure

verify and debug

5. Check bonding device is OK

```
[dpu]
root@localhost:~# ibdev2netdev
mlx5_0 port 1 ==> enp3s0f0s0 (Up)
mlx5_1 port 1 ==> enp3s0f1s0 (Up)
mlx5_bond_0 port 1 ==> en3f0pf0sf0 (Up)
```

6. Add into OVS

```
[dpu]
ovs-vsctl add-br bf-lag
ovs-vsctl add-port bf-lag bond0
ovs-vsctl add-port bf-lag pf0hpf
ovs-vsctl add-port bf-lag en3f1pf1sf0
...
```

** should not add p0,p1

7. Ovs show

```
root@localhost:~# ovs-vsctl show
8a6decf9-5f0f-4096-aa50-dcf0f822ee5f
    Bridge bf-lag
        Port pf0hpf
            Interface pf0hpf
        Port en3f0pf0sf0
            Interface en3f0pf0sf0
        Port en3f1pf1sf0
            Interface en3f1pf1sf0
        Port bond0
            Interface bond0
        Port bf-lag
            Interface bf-lag
                type: internal
    ovs_version: "2.17.8-3feee121f"
```

8. Run rdma perftest on Host PF

```
[Host]
ib_write_bw -d mlx5_0 -b -p 50001 -q 4

[Peer]
ib_write_bw -d mlx5_0 1.1.1.1 -p 50001 -q 4 -b --report_gbits --
  run_infinitely

[Dpu]
mlnx_perf -i p0 -c 1 | grep x_bytes_phy && mlnx_perf -i p1 -c 1 | grep
  x_bytes_phy
      rx_bytes_phy: 25,829,049,110 Bps    = 206,632.39 Mbps
      rx_bytes_phy: 25,913,204,164 Bps    = 207,305.63 Mbps
```

9. Run iperf on Host PF

```
[Host]
numactl -N 0 -m 0 iperf -s -p 50001

[Peer]
numactl -N 0 -m 0 iperf -c 1.1.1.1 -t 1000 -i 2 -p 50001 -P 16

[Dpu]
mlnx_perf -i ens2f0np0 -c 1 | grep rx_bytes_phy && mlnx_perf -i ens5f1np1
  -c 1| grep rx_bytes_phy
      rx_bytes_phy: 22,170,418,304 Bps    = 177,363.34 Mbps
      rx_bytes_phy: 22,623,055,644 Bps    = 180,984.44 Mbps
```

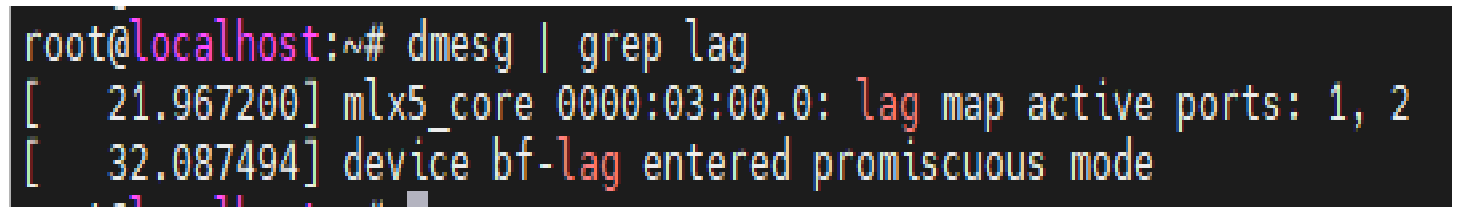
Other ways to config Bond/LAG

Bond config for CentOS

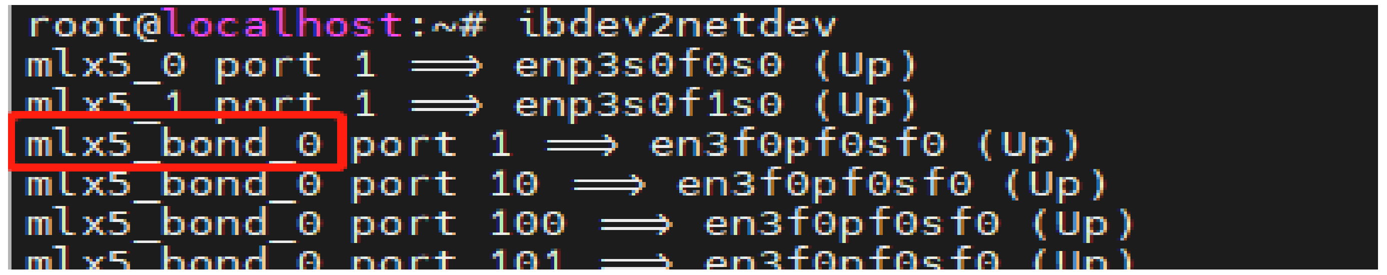
```
/etc/sysconfig/network-scripts/ifcfg-bond0
DEVICE=bond0
NAME=bond0
TYPE=bond
BONDING_MASTER=yes
IPADDR=192.168.99.2
PREFIX=24
ONBOOT=yes
BOOTPROTO=none
BONDING_OPTS="miimon=100 mode=802.3ad lacp_rate=1
xmit_hash_policy=layer3+4"


/etc/sysconfig/network-scripts/ifcfg-ens1f0
DEVICE=p0
NAME=p0
TYPE=Ethernet
BOOTPROTO=none
ONBOOT=yes
MASTER=bond0
SLAVE=yes

/etc/sysconfig/network-scripts/ifcfg-ens1f1
DEVICE=p1
NAME= p1
TYPE=Ethernet
BOOTPROTO=none
ONBOOT=yes
MASTER=bond0
SLAVE=yes
```

Manual scripts

```
Modprobe -r bonding

modprobe bonding miimon=100 mode=4
echo +bond0 >/sys/class/net/bonding_masters
echo "layer3+4" >
/sys/class/net/bond0/bonding/xmit_hash_policy
ifconfig p0 up
ifconfig p1 up
ifconfig bond0 up
ifconfig p0 mtu 9508
ifconfig p1 mtu 9508
ifconfig bond0 mtu 9508
ifenslave bond0 p0 p1

cat /proc/net/bonding/bond0
dmesg | grep "lag map"
```



```
root@localhost:~# dmesg | grep lag
[   21.967200] mlx5_core 0000:03:00.0: lag map active ports: 1, 2
[   32.087494] device bf-lag entered promiscuous mode
```



```
root@localhost:~# ibdev2netdev
mlx5_0 port 1 ==> enp3s0f0s0 (Up)
mlx5_1 port 1 ==> enp3s0f1s0 (Up)
mlx5_bond_0 port 1 ==> en3f0pf0sf0 (Up)
mlx5_bond_0 port 10 ==> en3f0pf0sf0 (Up)
mlx5_bond_0 port 100 ==> en3f0pf0sf0 (Up)
mlx5_bond_0 port 101 ==> en3f0pf0sf0 (Up)
```

Bond/LAG OVS config example

- ovs-kernel

```
ovs-vsctl set Open_vSwitch . Other_config:hw-offload=true
ovs-vsctl add-br ovsbr
ovs-vsctl add-port ovsbr bond1
ovs-vsctl add-port ovsbr pf0hpf
ovs-vsctl add-port ovsbr en3f0pf0sf2000
ovs-vsctl add-port ovsbr en3f0pf0sf2001
ovs-vsctl add-port ovsbr en3f0pf0sf2002
```

- ovs-dpdk

```
ovs-vsctl clear o . Other_config
ovs-vsctl add-br ovsbr -- set bridge ovsbr datapath_type=netdev
ovs-vsctl --no-wait set o . other_config:hw-offload=true other_config:max-idle=30000
ovs-vsctl --no-wait set o . other_config:dpdk-extra="-w 0000:03:00.0,representor=pf0sf[2000-
2030,65535],dv_xmeta_en=1,sys_mem_en=1 mtu_request=9000"
ovs-vsctl --no-wait set o . other_config:dpdk-init=true

ovs-vsctl add-port ovsbr bond0 -- set Interface bond0 type=dpdk options:dpdk-
devargs="0000:03:00.0,dv_xmeta_en=1,sys_mem_en=1" mtu_request=9000

ovs-vsctl add-port ovsbr pf0hpf -- set Interface pf0hpf type=dpdk options:dpdk-
devargs="0000:03:00.0,representor=sf65535,dv_xmeta_en=1,sys_mem_en=1" mtu_request=9000

ovs-vsctl add-port ovsbr pf0sf2000 -- set Interface pf0sf2000 type=dpdk options:dpdk-
devargs="0000:03:00.0,representor=pf0sf2000,dv_xmeta_en=1,sys_mem_en=1" mtu_request=9000
```

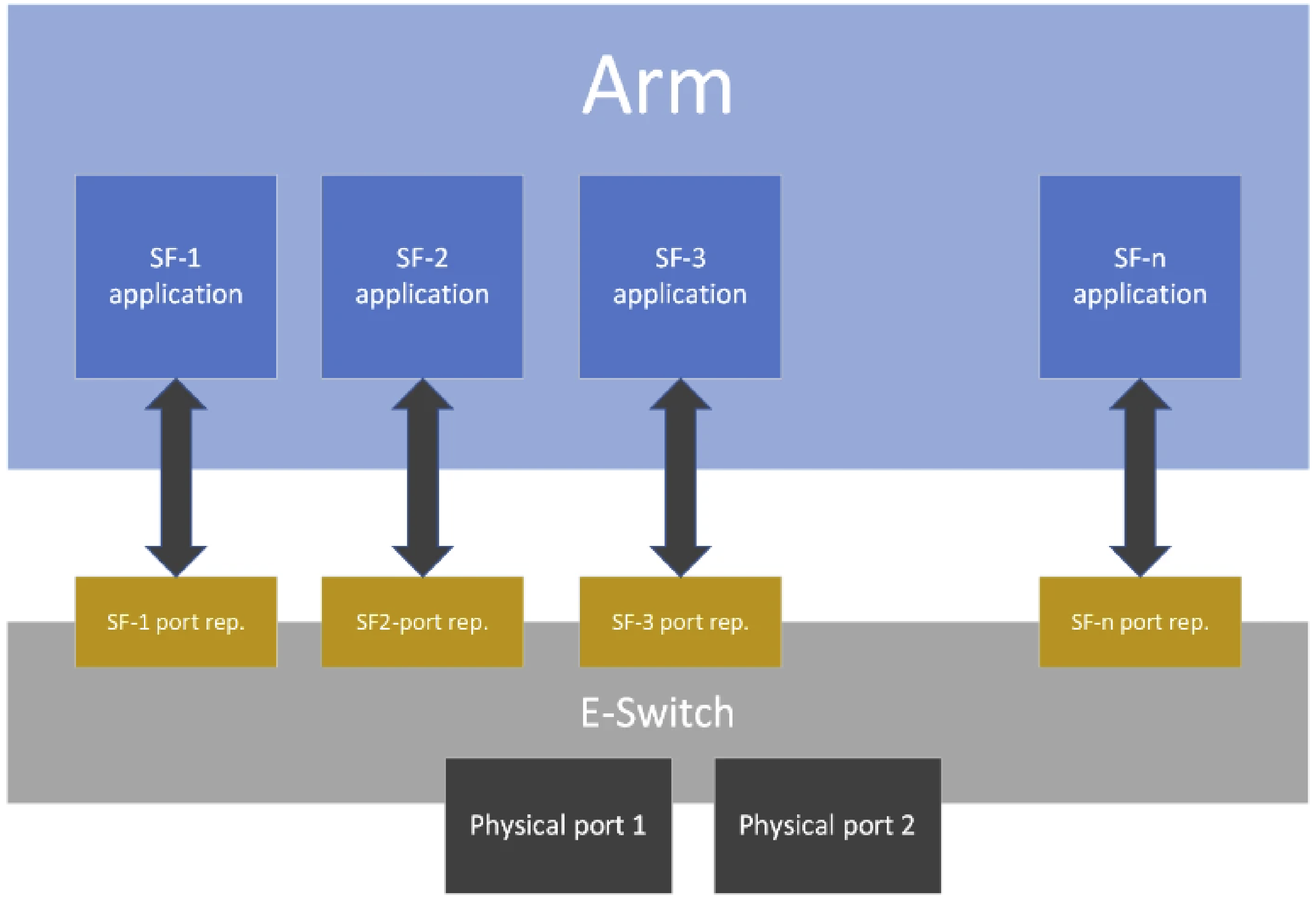
BF LAG Take aways

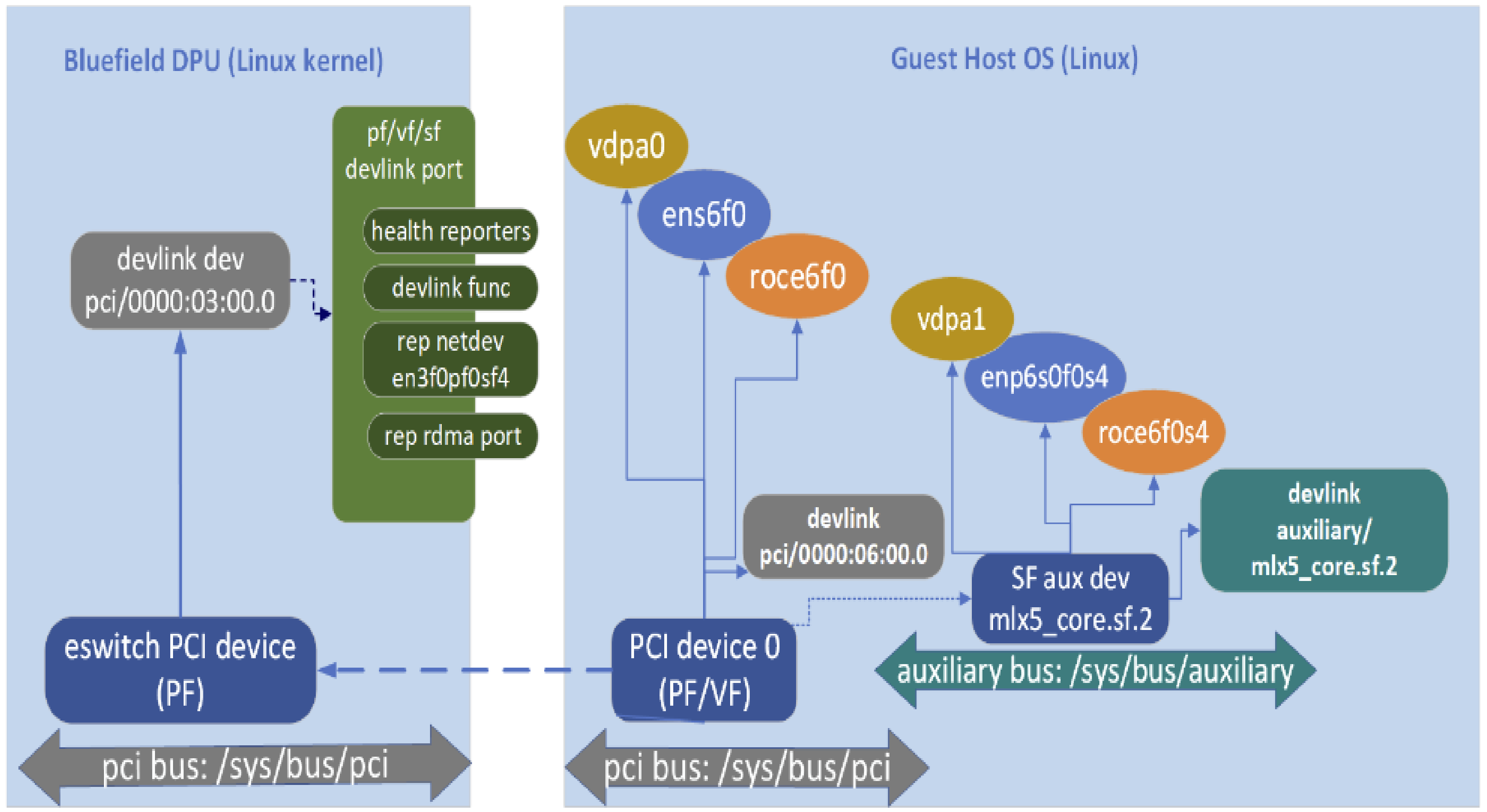
1. Bond/LAG 是否成功看mlx5_bond_0是否生成，bond运行需要在没有创建SRIOV或者OVS stop之前
2. 创建LAG之后，host上的PF, VF，DPU上的SF都具备了LAG属性，也就是能从两个uplink口收发数据
3. LAG的tx缺省是hash模式（5元组），可以选配queue_affinity模式
4. LAG的OVS中不需要向bridge中加入p0/p1，只需要加入bond master
5. DPU LAG 抓包
   - `tcpdump -i mlx5_bond_0  # need upgrade libpcap >= 1.10`
   - `ovs-tcpdump -i bond0`

Scalable Functions (SFs)

DPU Scalable Functions (SF)

- Scalable functions (SF) 功能相似于VF，提供的数量更多elastic networking interface (ENI)
- 从PF上衍生的轻量级的功能设备，有独立的硬件资源例如队列
- 不需要SRIOV支持，能和VF共存，并且能在DPU的ARM OS上或者HOST上创建
- 有eth device和对应的RDMA设备，DPU内对应representer用于OVS的flows



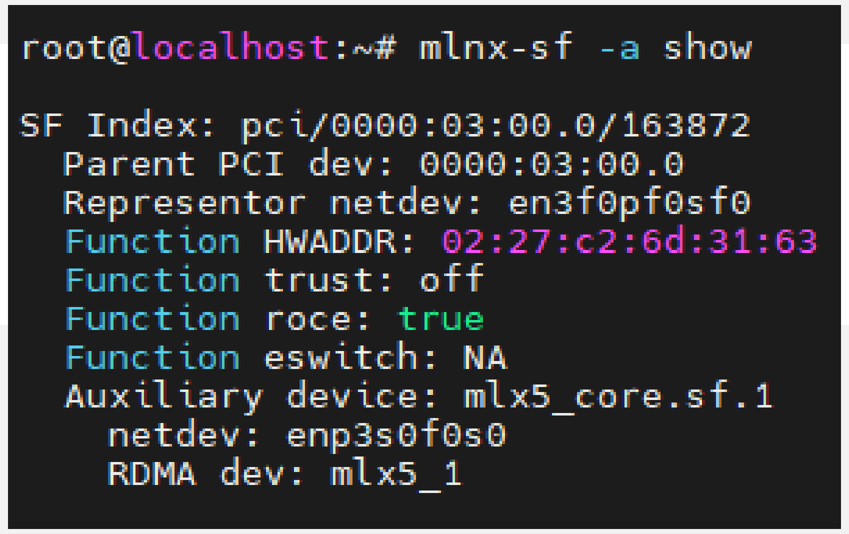


https://docs.nvidia.com/doca/sdk/nvidia+bluefield+dpu+scalable+function+user+guide/index.html

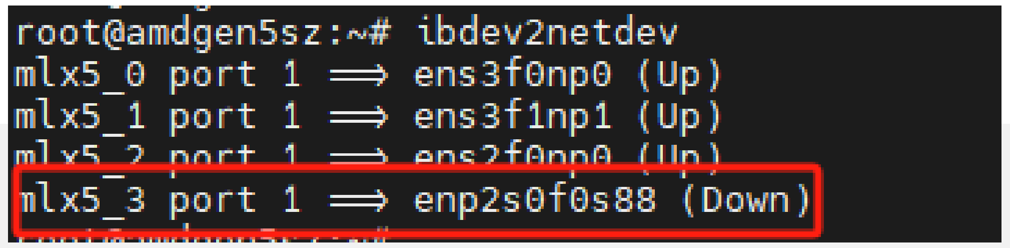
DPU Scalable Functions (SF)

- Create SF for DPU internal

FW config
```
    [DPU] mlxconfig -d 0000:03:00.0 s PF_BAR2_ENABLE=0 PER_PF_NUM_SF=1 PF_TOTAL_SF=236 PF_SF_BAR_SIZE=10
```
Create the SF
```
    [DPU] /opt/mellanox/iproute2/sbin/mlxdevm port add pci/0000:03:00.0 flavour pcisf pfnum 0 sfnum 5
```
Configure the SF
```
    [DPU] /opt/mellanox/iproute2/sbin/mlxdevm port function set pci/0000:03:00.0/163874 hw_addr 02:25:f2:8d:a2:5c trust on state active
```
Deploy the SF
```
    [DPU] echo mlx5_core.sf.3 > /sys/bus/auxiliary/drivers/mlx5_core.sf_cfg/unbind
    [DPU] echo mlx5_core.sf.3 > /sys/bus/auxiliary/drivers/mlx5_core.sf/bind
```
Delete SF
```
    [DPU] /opt/mellanox/iproute2/sbin/mlxdevm port function set pci/0000:03:00.0/163874 state inactive
    [DPU]/opt/mellanox/iproute2/sbin/mlxdevm port del pci/0000:03:00.0/163874
```


```
root@localhost:~# mlnx-sf -a show

SF Index: pci/0000:03:00.0/163872
  Parent PCI dev: 0000:03:00.0
  Representor netdev: en3f0pf0sf0
  Function HWADDR: 02:27:c2:6d:31:63
  Function trust: off
  Function roce: true
  Function eswitch: NA
  Auxiliary device: mlx5_core.sf.1
    netdev: enp3s0f0s0
    RDMA dev: mlx5_1
```

- Create SF for DPU host external


```
root@amdgen5sz:~# ibdev2netdev
mlx5_0 port 1 ==> ens3f0np0 (Up)
mlx5_1 port 1 ==> ens3f1np1 (Up)
mlx5_2 port 1 ==> ens2f0np0 (Up)
mlx5_3 port 1 ==> enp2s0f0s88 (Down)
```

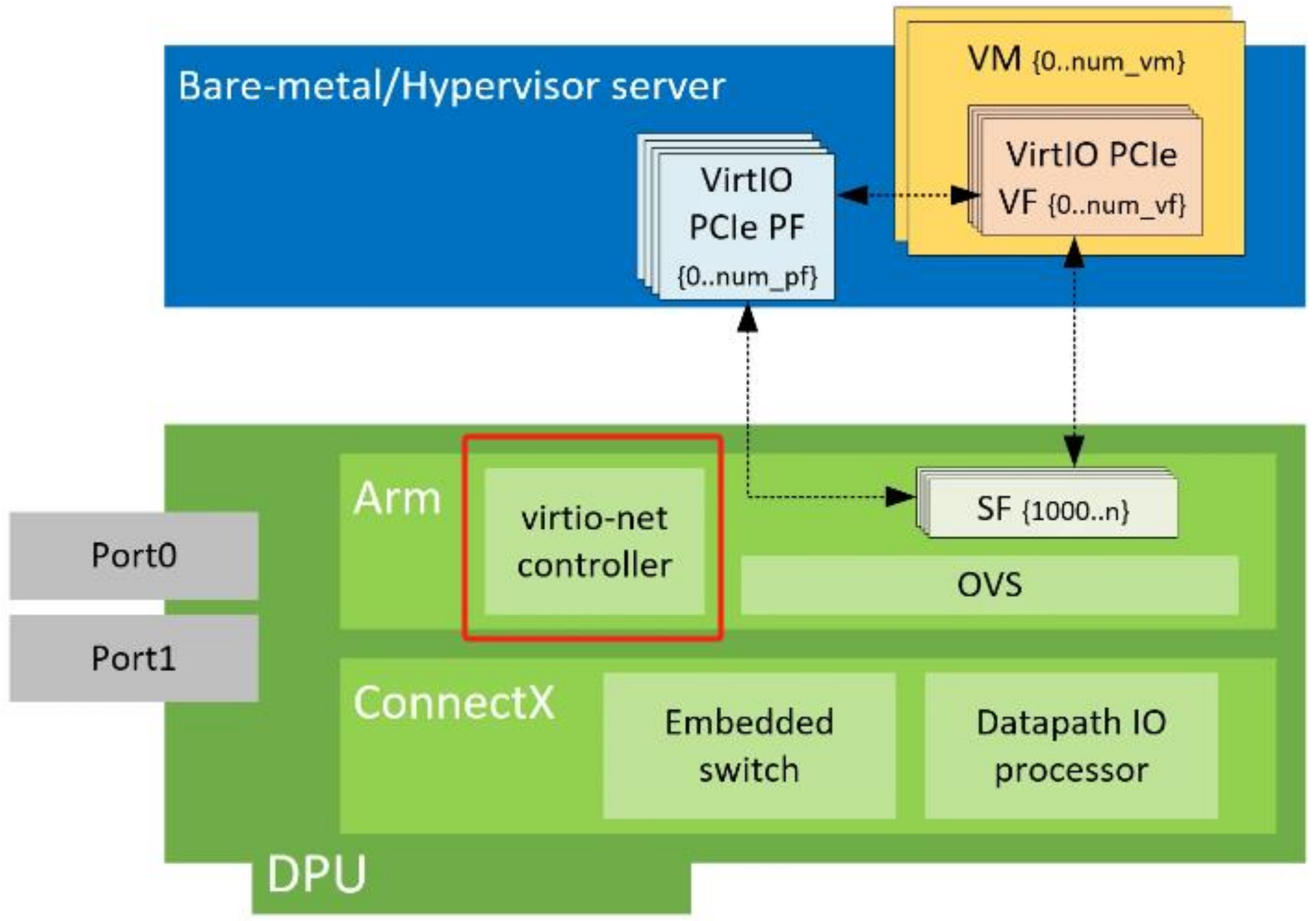
FW config (DPU and Host)
```
[DPU] mlxconfig -d 0000:03:00.0 -y s NUM_PF_MSIX_VALID=0
[DPU] mlxconfig -d 0000:03:00.0 -y s PF_NUM_PF_MSIX_VALID=1
[DPU] mlxconfig -d 0000:03:00.0 -y s PF_TOTAL_SF=32 PF_NUM_PF_MSIX=128 PF_BAR2_ENABLE=0 PER_PF_NUM_SF=1 PF_SF_BAR_SIZE=10
[Host] mlxconfig -d 0000:02:00.0 -y s PF_TOTAL_SF=32 PF_NUM_PF_MSIX=128 PF_BAR2_ENABLE=0 PER_PF_NUM_SF=1 PF_SF_BAR_SIZE=10
```

Create and active SF
```
[DPU]/opt/mellanox/iproute2/sbin/mlxdevm port add pci/0000:03:00.0 flavour pcisf pfnum 0 sfnum 88 controller 1
[DPU]/opt/mellanox/iproute2/sbin/mlxdevm port function set pci/0000:03:00.0/163840 hw_addr 02:25:f2:8d:aa:4c state active
```

Virtio-net Hotplug for Bare-metal

Virtio-net overview

1. Service **virtio-net-controller** manager virtio_net devices, must have initialization conf
2. Communicated through rpc, commands through **virtnet**
3. OVS or DPDK through rte_flow or TC to control all virtio-device flow rules for offload or data path
4. Each virtio_net device will map to one **SF for rep** for flow rule inserting interface or other such as QoS interface.



Note:

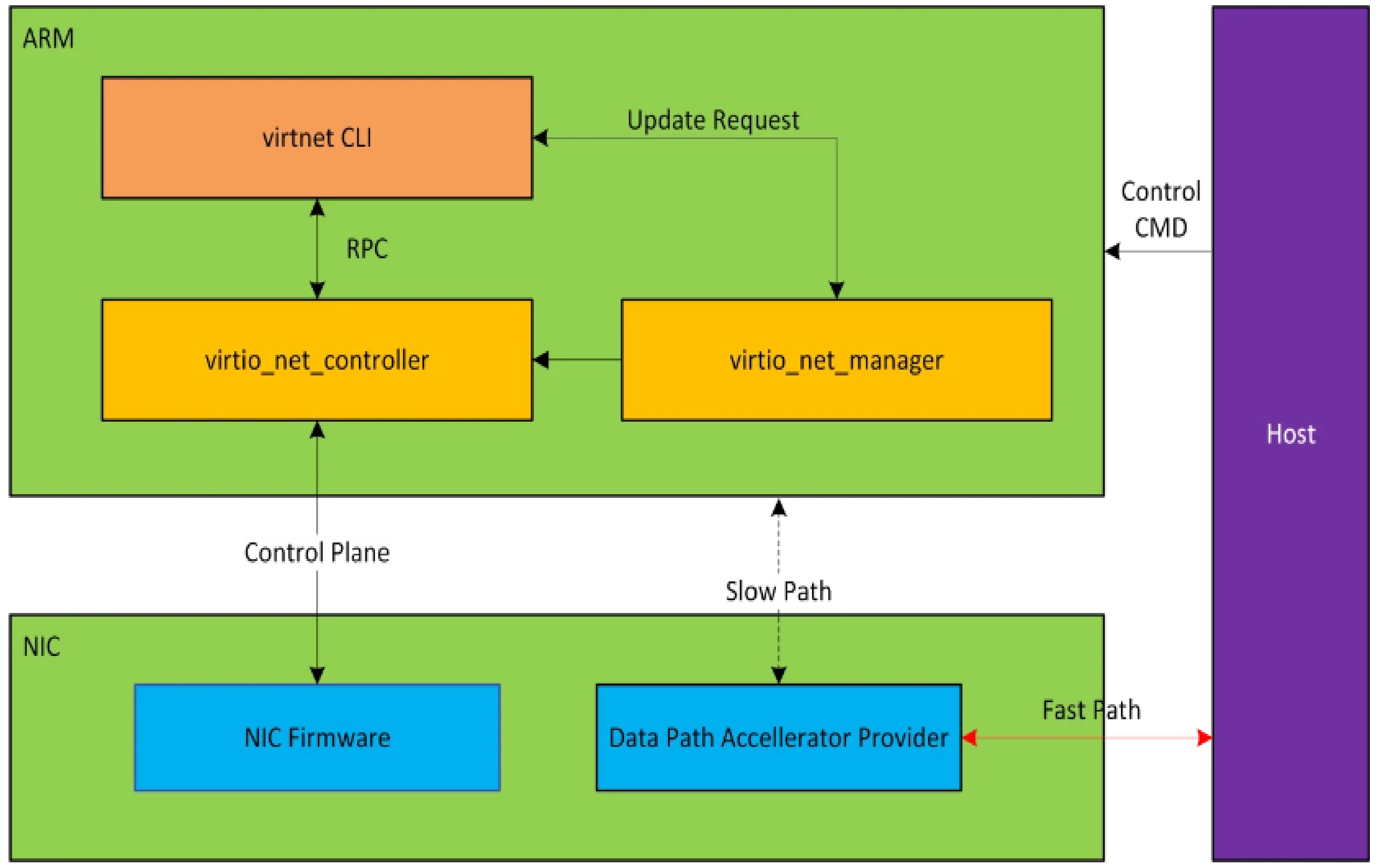
1. Hot-plug 动态virtio必须是基于PF，硬件限制最多31.
2. Static 可以是PF，也可以是VF，VF需要从host上创建
3. Static virtio_net 必须是先配置好backen，再加载host上的virtio_pci driver，否者会导致host kernel被挂住
4. OVS 或者DPDK或者TC必须配置好流规则后virtio才能通信

NVIDIA BlueField Virtio-net v1.9.0
https://docs.nvidia.com/networking/display/bluefieldvirtionetv190

Virtio-net how it works

Running Mechanism in DPU

1. virtio-net-controller service works as control center
2. virtnet cli is communicated with cmd “**virtnet**” through RPC
3. Virtio devices control messages exchange through FW path
4. RPC API can be integrated with upper applications
5. virt_net_manager is working for live update
6. Support fast recovery in case of crash
7. Static virtio-net is created from DPU power up
8. Hot-plug virtio-net is created dynamically
9. virtio-net capabilities : max number, max queues, features are limited by HW or FW setting.
10. Support virtio spec 1.x version (limited for 0.95)



Static PF	Hot-plug PF	VF
31	31	1008

Virtio-net hotplug – 1

1. FW config (on DPU)

```
mlxconfig -d 03:00.0 -y reset
mlxconfig -d 03:00.0 -y set \
    INTERNAL_CPU_MODEL=1 SRIOV_EN=0 \
    NUM_OF_VFS=0 NUM_OF_PF=0 \
    PCI_SWITCH_EMULATION_ENABLE=1 \
    PCI_SWITCH_EMULATION_NUM_PORT=32 \
    VIRTIO_NET_EMULATION_ENABLE=1 \
    VIRTIO_NET_EMULATION_NUM_VF=0 \
    VIRTIO_NET_EMULATION_NUM_PF=0 \
    VIRTIO_NET_EMULATION_NUM_MSIX=64 \
    LAG_RESOURCE_ALLOCATION=1 \
    PER_PF_NUM_SF=1 PF_TOTAL_SF=64 PF_SF_BAR_SIZE=10

mlxconfig -d 03:00.1 -y set \
    PER_PF_NUM_SF=1 PF_TOTAL_SF=2 PF_SF_BAR_SIZE=10
```

2. Enable virtio-net-controller service

```
systemctl enable virtio-net-controller
journalctl -u virtio-net-controller -n 100 -f
```

3. Edit /opt/mellanox/mlnx_virtnet/virtnet.conf

```
{
    "ib_dev_lag": "mlx5_bond_0",
    "ib_dev_for_static_pf": "mlx5_bond_0",
    "is_lag": 1,
    "recovery": 1,
    "sf_pool_percent": 0,
    "sf_pool_force_destroy": 0
}
```

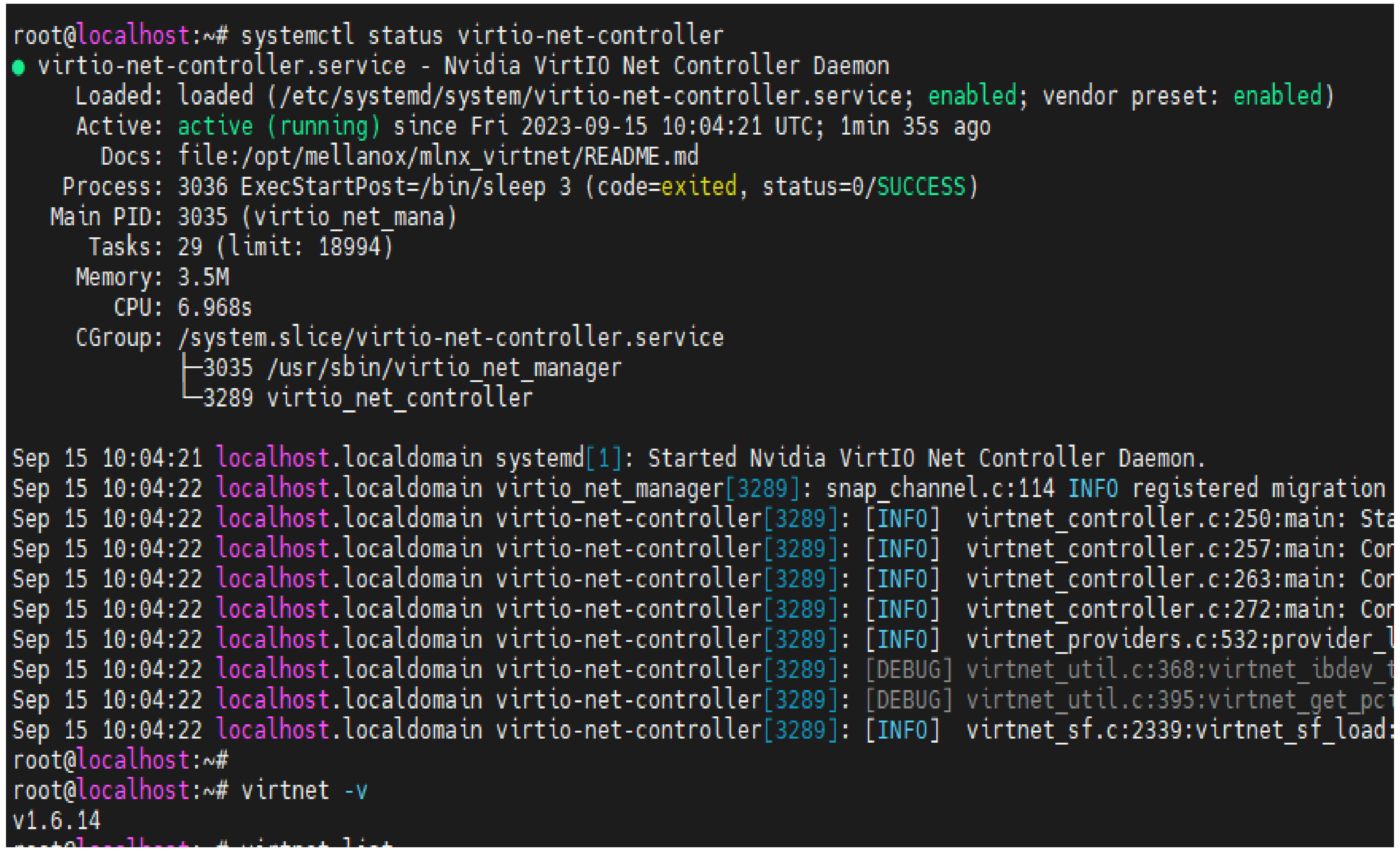
4. Powe cycle

```
[host]
Add "pci=realloc" into kernel boot grub options

[host]
ipmitool power cycle
```

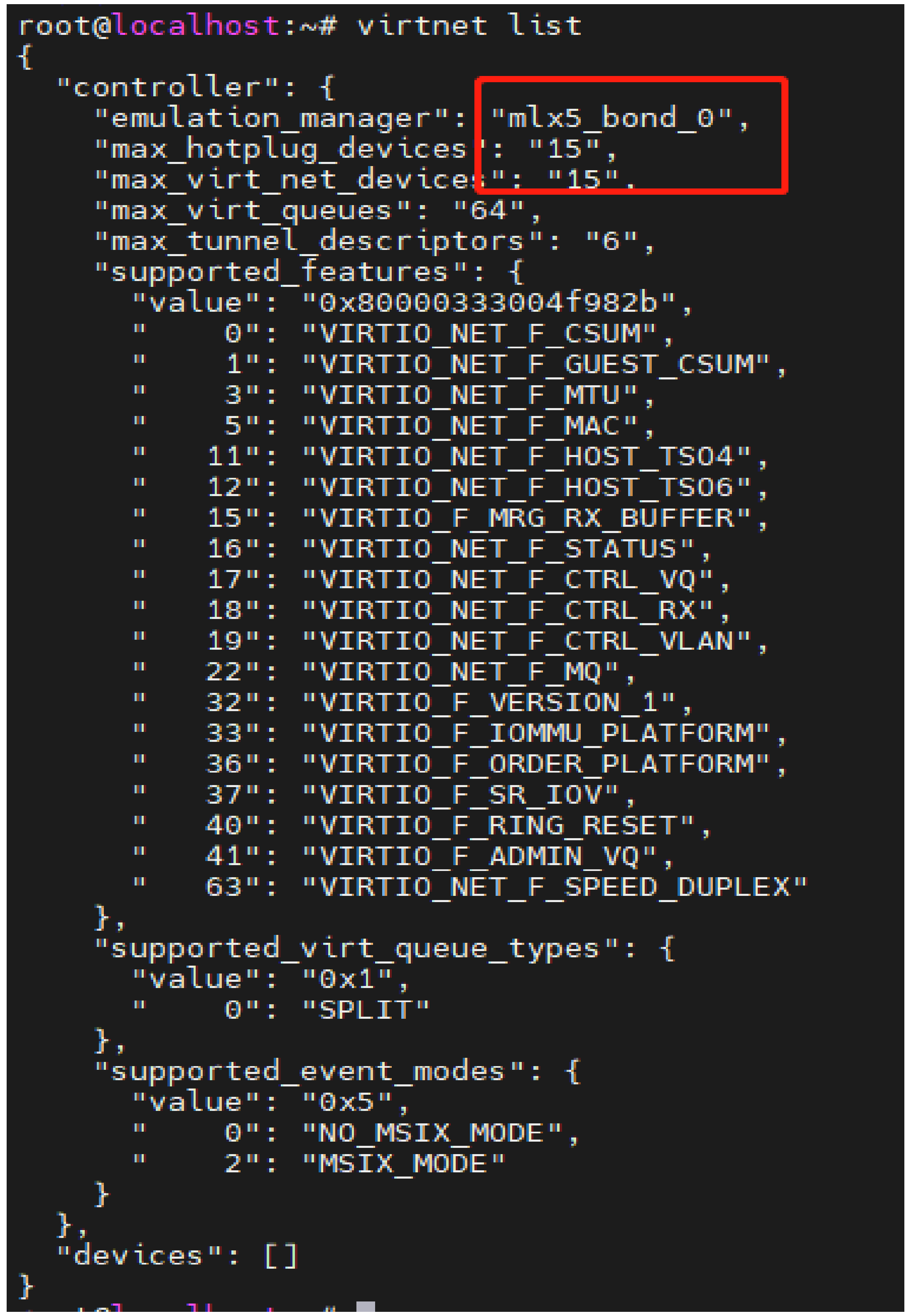
Virtio-net usage :
/opt/mellanox/mlnx_virtnet/README.md

Virtio-net hotplug – 2

5. Check virtio-net-controller service



```
root@localhost:~# systemctl status virtio-net-controller
● virtio-net-controller.service - Nvidia VirtIO Net Controller Daemon
     Loaded: loaded (/etc/systemd/system/virtio-net-controller.service; enabled; vendor preset: enabled)
     Active: active (running) since Fri 2023-09-15 10:04:21 UTC; 1min 35s ago
       Docs: file:/opt/mellanox/mlnx_virtnet/README.md
    Process: 3036 ExecStartPost=/bin/sleep 3 (code=exited, status=0/SUCCESS)
   Main PID: 3035 (virtio_net_mana)
      Tasks: 29 (limit: 18994)
     Memory: 3.5M
        CPU: 6.968s
     CGroup: /system.slice/virtio-net-controller.service
             ├─3035 /usr/sbin/virtio_net_manager
             └─3289 virtio_net_controller

Sep 15 10:04:21 localhost.localdomain systemd[1]: Started Nvidia VirtIO Net Controller Daemon.
Sep 15 10:04:22 localhost.localdomain virtio_net_manager[3289]: snap_channel.c:114 INFO registered migration
Sep 15 10:04:22 localhost.localdomain virtio-net-controller[3289]: [INFO]  virtnet_controller.c:250:main: Sta
Sep 15 10:04:22 localhost.localdomain virtio-net-controller[3289]: [INFO]  virtnet_controller.c:257:main: Con
Sep 15 10:04:22 localhost.localdomain virtio-net-controller[3289]: [INFO]  virtnet_controller.c:263:main: Con
Sep 15 10:04:22 localhost.localdomain virtio-net-controller[3289]: [INFO]  virtnet_controller.c:272:main: Con
Sep 15 10:04:22 localhost.localdomain virtio-net-controller[3289]: [INFO]  virtnet_providers.c:532:provider_l
Sep 15 10:04:22 localhost.localdomain virtio-net-controller[3289]: [DEBUG] virtnet_util.c:368:virtnet_ibdev_t
Sep 15 10:04:22 localhost.localdomain virtio-net-controller[3289]: [DEBUG] virtnet_util.c:395:virtnet_get_pci
Sep 15 10:04:22 localhost.localdomain virtio-net-controller[3289]: [INFO]  virtnet_sf.c:2339:virtnet_sf_load:
root@localhost:~#
root@localhost:~# virtnet -v
v1.6.14
```

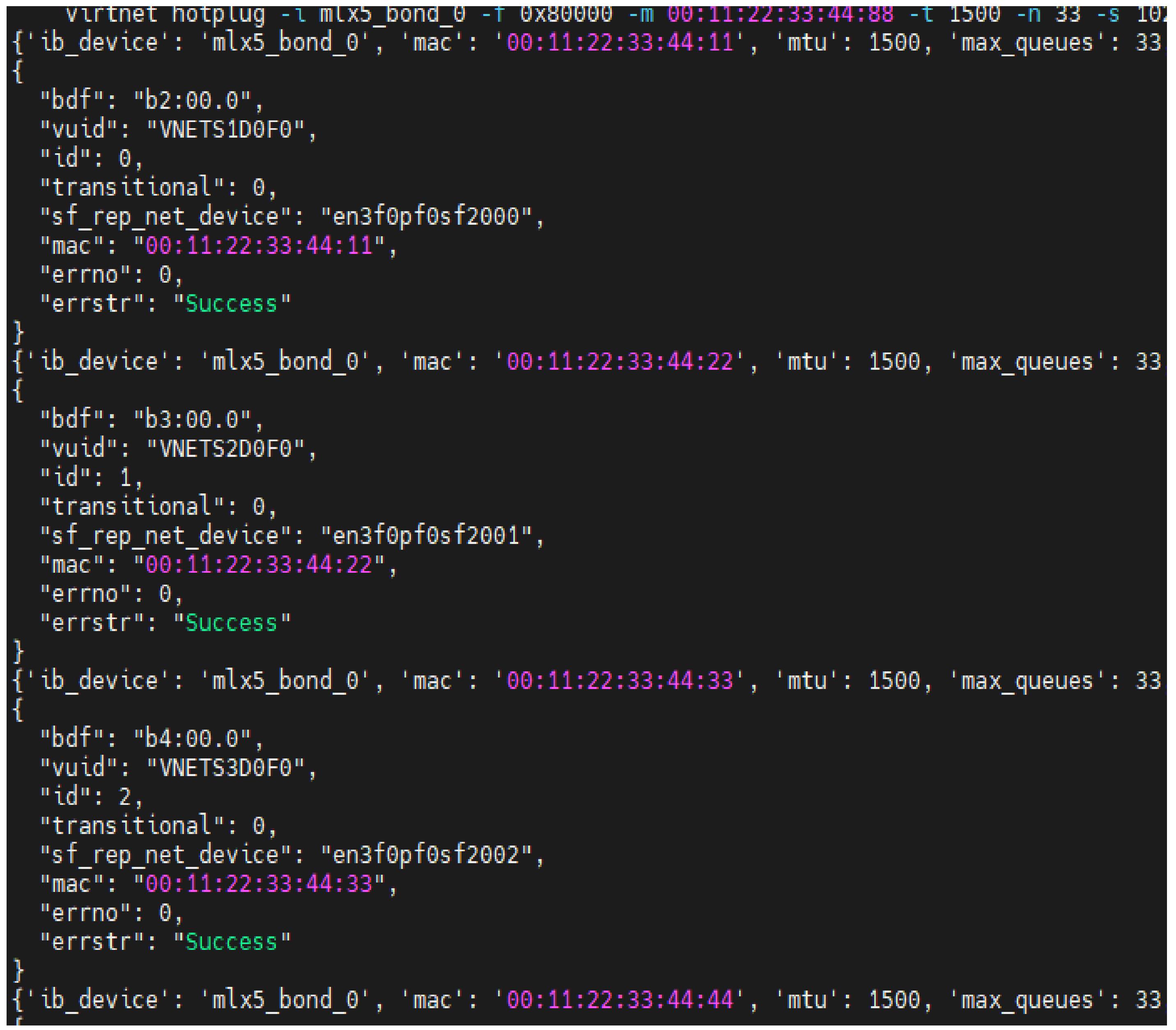


```
root@localhost:~# virtnet list
{
  "controller": {
    "emulation_manager": "mlx5_bond_0",
    "max_hotplug_devices": "15",
    "max_virt_net_devices": "15",
    "max_virt_queues": "64",
    "max_tunnel_descriptors": "6",
    "supported_features": {
      "value": "0x80000333004f982b",
      "    0": "VIRTIO_NET_F_CSUM",
      "    1": "VIRTIO_NET_F_GUEST_CSUM",
      "    3": "VIRTIO_NET_F_MTU",
      "    5": "VIRTIO_NET_F_MAC",
      "   11": "VIRTIO_NET_F_HOST_TSO4",
      "   12": "VIRTIO_NET_F_HOST_TSO6",
      "   15": "VIRTIO_F_MRG_RX_BUFFER",
      "   16": "VIRTIO_NET_F_STATUS",
      "   17": "VIRTIO_NET_F_CTRL_VQ",
      "   18": "VIRTIO_NET_F_CTRL_RX",
      "   19": "VIRTIO_NET_F_CTRL_VLAN",
      "   22": "VIRTIO_NET_F_MQ",
      "   32": "VIRTIO_F_VERSION_1",
      "   33": "VIRTIO_F_IOMMU_PLATFORM",
      "   36": "VIRTIO_F_ORDER_PLATFORM",
      "   37": "VIRTIO_F_SR_IOV",
      "   40": "VIRTIO_F_RING_RESET",
      "   41": "VIRTIO_F_ADMIN_VQ",
      "   63": "VIRTIO_NET_F_SPEED_DUPLEX"
    },
    "supported_virt_queue_types": {
      "value": "0x1",
      "    0": "SPLIT"
    },
    "supported_event_modes": {
      "value": "0x5",
      "    0": "NO_MSIX_MODE",
      "    2": "MSIX_MODE"
    }
  },
  "devices": []
}
```

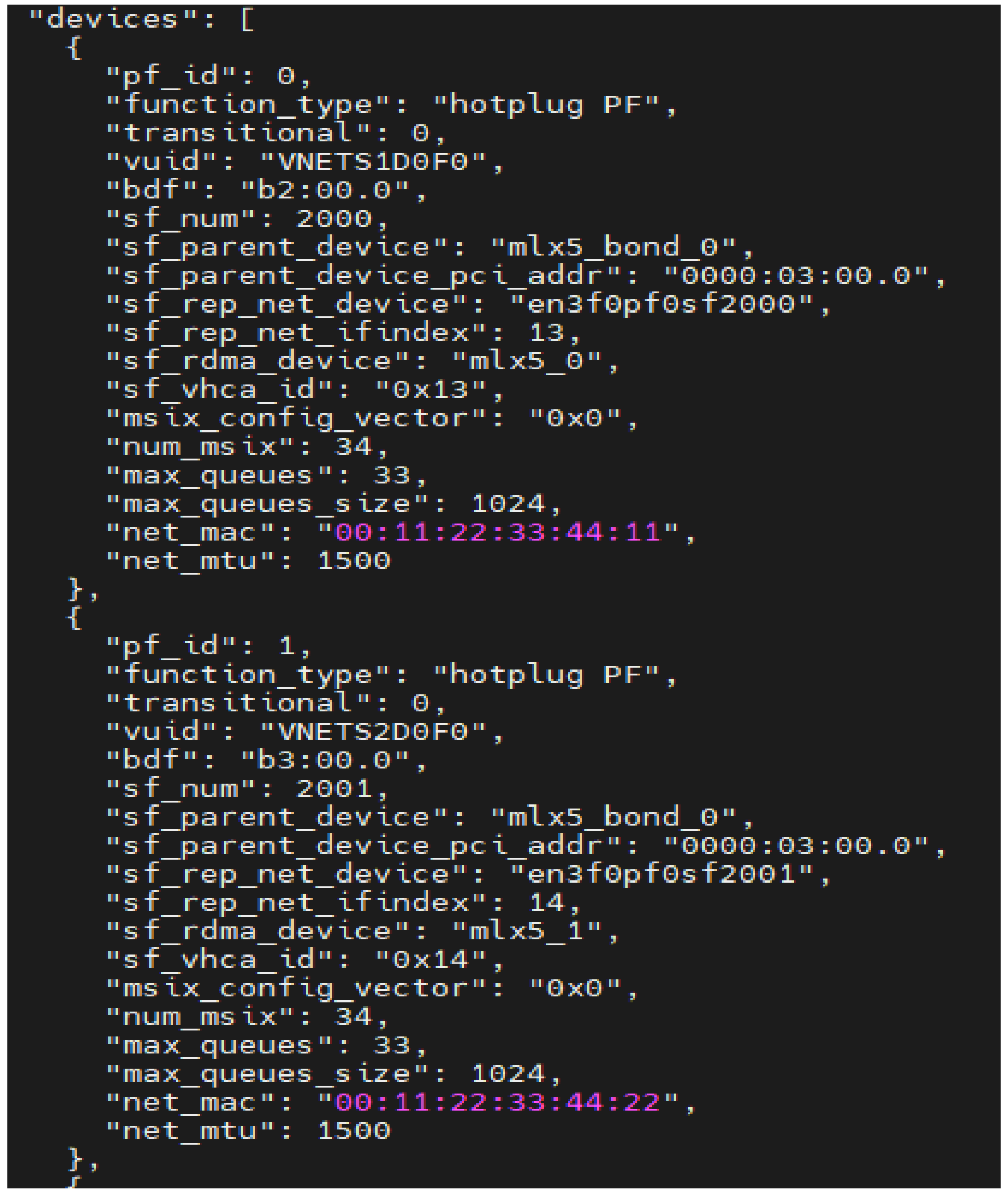
virtio-net hotplug – 3

6. Create virtio_net hotplug devices

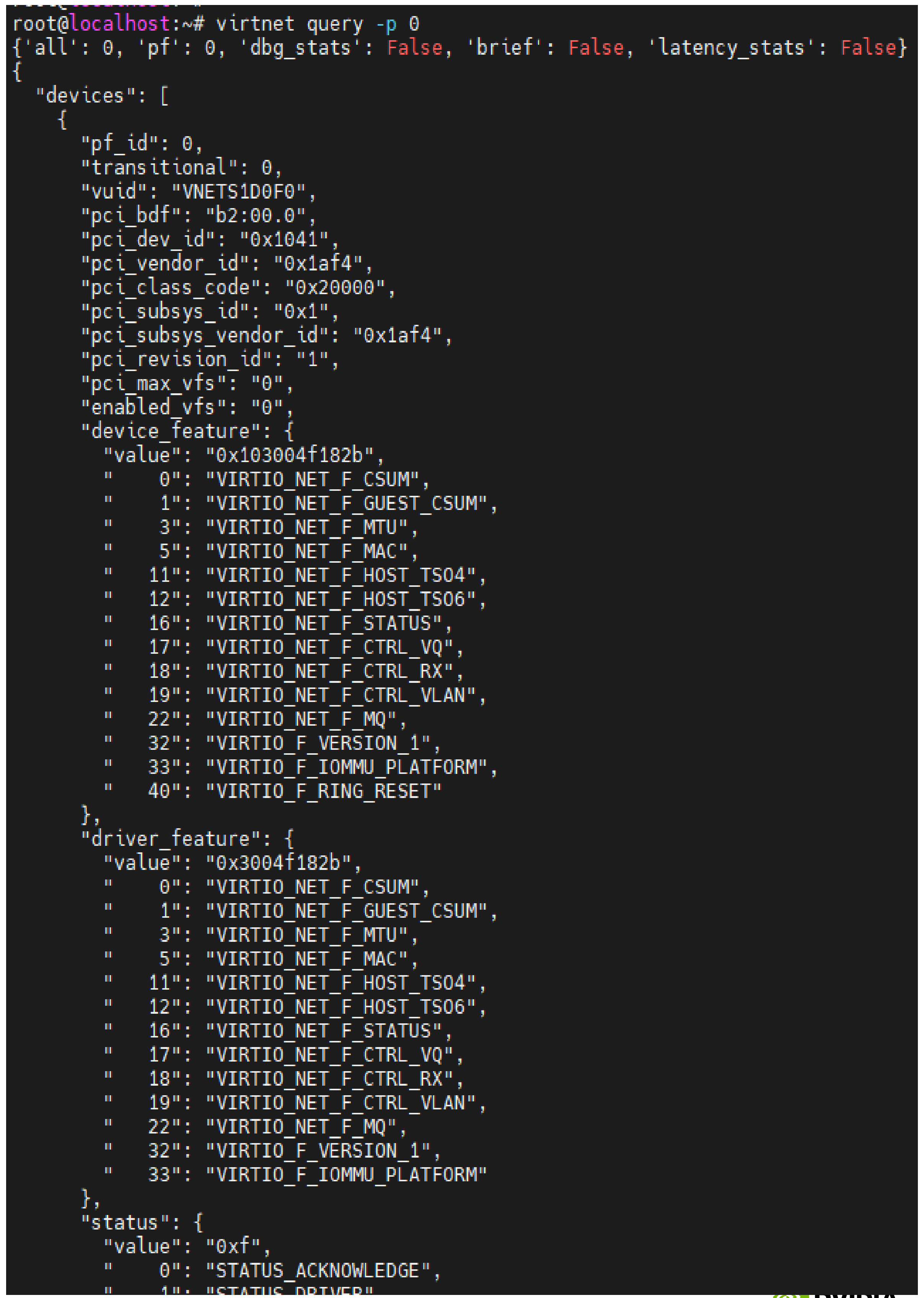
```
virtnet hotplug -i mlx5_bond_0 -f 0x80000 -m 00:11:22:33:44:11 -t 1500 -n 33 -s 1024
virtnet hotplug -i mlx5_bond_0 -f 0x80000 -m 00:11:22:33:44:22 -t 1500 -n 33 -s 1024
virtnet hotplug -i mlx5_bond_0 -f 0x80000 -m 00:11:22:33:44:33 -t 1500 -n 33 -s 1024
virtnet hotplug -i mlx5_bond_0 -f 0x80000 -m 00:11:22:33:44:44 -t 1500 -n 33 -s 1024
……
virtnet list
```



```
   virtnet hotplug -i mlx5_bond_0 -f 0x80000 -m 00:11:22:33:44:88 -t 1500 -n 33 -s 10
{'ib_device': 'mlx5_bond_0', 'mac': '00:11:22:33:44:11', 'mtu': 1500, 'max_queues': 33
{
  "bdf": "b2:00.0",
  "vuid": "VNETS1D0F0",
  "id": 0,
  "transitional": 0,
  "sf_rep_net_device": "en3f0pf0sf2000",
  "mac": "00:11:22:33:44:11",
  "errno": 0,
  "errstr": "Success"
}
{'ib_device': 'mlx5_bond_0', 'mac': '00:11:22:33:44:22', 'mtu': 1500, 'max_queues': 33
{
  "bdf": "b3:00.0",
  "vuid": "VNETS2D0F0",
  "id": 1,
  "transitional": 0,
  "sf_rep_net_device": "en3f0pf0sf2001",
  "mac": "00:11:22:33:44:22",
  "errno": 0,
  "errstr": "Success"
}
{'ib_device': 'mlx5_bond_0', 'mac': '00:11:22:33:44:33', 'mtu': 1500, 'max_queues': 33
{
  "bdf": "b4:00.0",
  "vuid": "VNETS3D0F0",
  "id": 2,
  "transitional": 0,
  "sf_rep_net_device": "en3f0pf0sf2002",
  "mac": "00:11:22:33:44:33",
  "errno": 0,
  "errstr": "Success"
}
{'ib_device': 'mlx5_bond_0', 'mac': '00:11:22:33:44:44', 'mtu': 1500, 'max_queues': 33
```



```
"devices": [
  {
    "pf_id": 0,
    "function_type": "hotplug PF",
    "transitional": 0,
    "vuid": "VNETS1D0F0",
    "bdf": "b2:00.0",
    "sf_num": 2000,
    "sf_parent_device": "mlx5_bond_0",
    "sf_parent_device_pci_addr": "0000:03:00.0",
    "sf_rep_net_device": "en3f0pf0sf2000",
    "sf_rep_net_ifindex": 13,
    "sf_rdma_device": "mlx5_0",
    "sf_vhca_id": "0x13",
    "msix_config_vector": "0x0",
    "num_msix": 34,
    "max_queues": 33,
    "max_queues_size": 1024,
    "net_mac": "00:11:22:33:44:11",
    "net_mtu": 1500
  },
  {
    "pf_id": 1,
    "function_type": "hotplug PF",
    "transitional": 0,
    "vuid": "VNETS2D0F0",
    "bdf": "b3:00.0",
    "sf_num": 2001,
    "sf_parent_device": "mlx5_bond_0",
    "sf_parent_device_pci_addr": "0000:03:00.0",
    "sf_rep_net_device": "en3f0pf0sf2001",
    "sf_rep_net_ifindex": 14,
    "sf_rdma_device": "mlx5_1",
    "sf_vhca_id": "0x14",
    "msix_config_vector": "0x0",
    "num_msix": 34,
    "max_queues": 33,
    "max_queues_size": 1024,
    "net_mac": "00:11:22:33:44:22",
    "net_mtu": 1500
  },
```

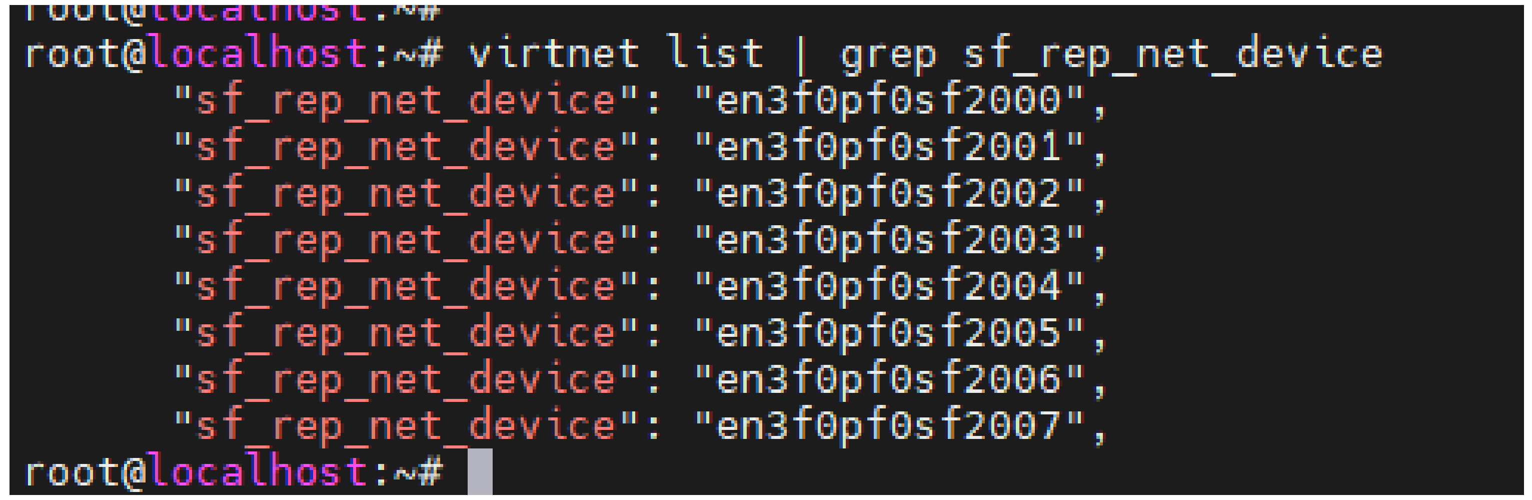


```
root@localhost:~# virtnet query -p 0
{'all': 0, 'pf': 0, 'dbg_stats': False, 'brief': False, 'latency_stats': False}
{
  "devices": [
    {
      "pf_id": 0,
      "transitional": 0,
      "vuid": "VNETS1D0F0",
      "pci_bdf": "b2:00.0",
      "pci_dev_id": "0x1041",
      "pci_vendor_id": "0x1af4",
      "pci_class_code": "0x20000",
      "pci_subsys_id": "0x1",
      "pci_subsys_vendor_id": "0x1af4",
      "pci_revision_id": "1",
      "pci_max_vfs": "0",
      "enabled_vfs": "0",
      "device_feature": {
        "value": "0x103004f182b",
        "    0": "VIRTIO_NET_F_CSUM",
        "    1": "VIRTIO_NET_F_GUEST_CSUM",
        "    3": "VIRTIO_NET_F_MTU",
        "    5": "VIRTIO_NET_F_MAC",
        "   11": "VIRTIO_NET_F_HOST_TSO4",
        "   12": "VIRTIO_NET_F_HOST_TSO6",
        "   16": "VIRTIO_NET_F_STATUS",
        "   17": "VIRTIO_NET_F_CTRL_VQ",
        "   18": "VIRTIO_NET_F_CTRL_RX",
        "   19": "VIRTIO_NET_F_CTRL_VLAN",
        "   22": "VIRTIO_NET_F_MQ",
        "   32": "VIRTIO_F_VERSION_1",
        "   33": "VIRTIO_F_IOMMU_PLATFORM",
        "   40": "VIRTIO_F_RING_RESET"
      },
      "driver_feature": {
        "value": "0x3004f182b",
        "    0": "VIRTIO_NET_F_CSUM",
        "    1": "VIRTIO_NET_F_GUEST_CSUM",
        "    3": "VIRTIO_NET_F_MTU",
        "    5": "VIRTIO_NET_F_MAC",
        "   11": "VIRTIO_NET_F_HOST_TSO4",
        "   12": "VIRTIO_NET_F_HOST_TSO6",
        "   16": "VIRTIO_NET_F_STATUS",
        "   17": "VIRTIO_NET_F_CTRL_VQ",
        "   18": "VIRTIO_NET_F_CTRL_RX",
        "   19": "VIRTIO_NET_F_CTRL_VLAN",
        "   22": "VIRTIO_NET_F_MQ",
        "   32": "VIRTIO_F_VERSION_1",
        "   33": "VIRTIO_F_IOMMU_PLATFORM"
      },
      "status": {
        "value": "0xf",
        "    0": "STATUS_ACKNOWLEDGE",
        "    1": "STATUS_DRIVER"
```

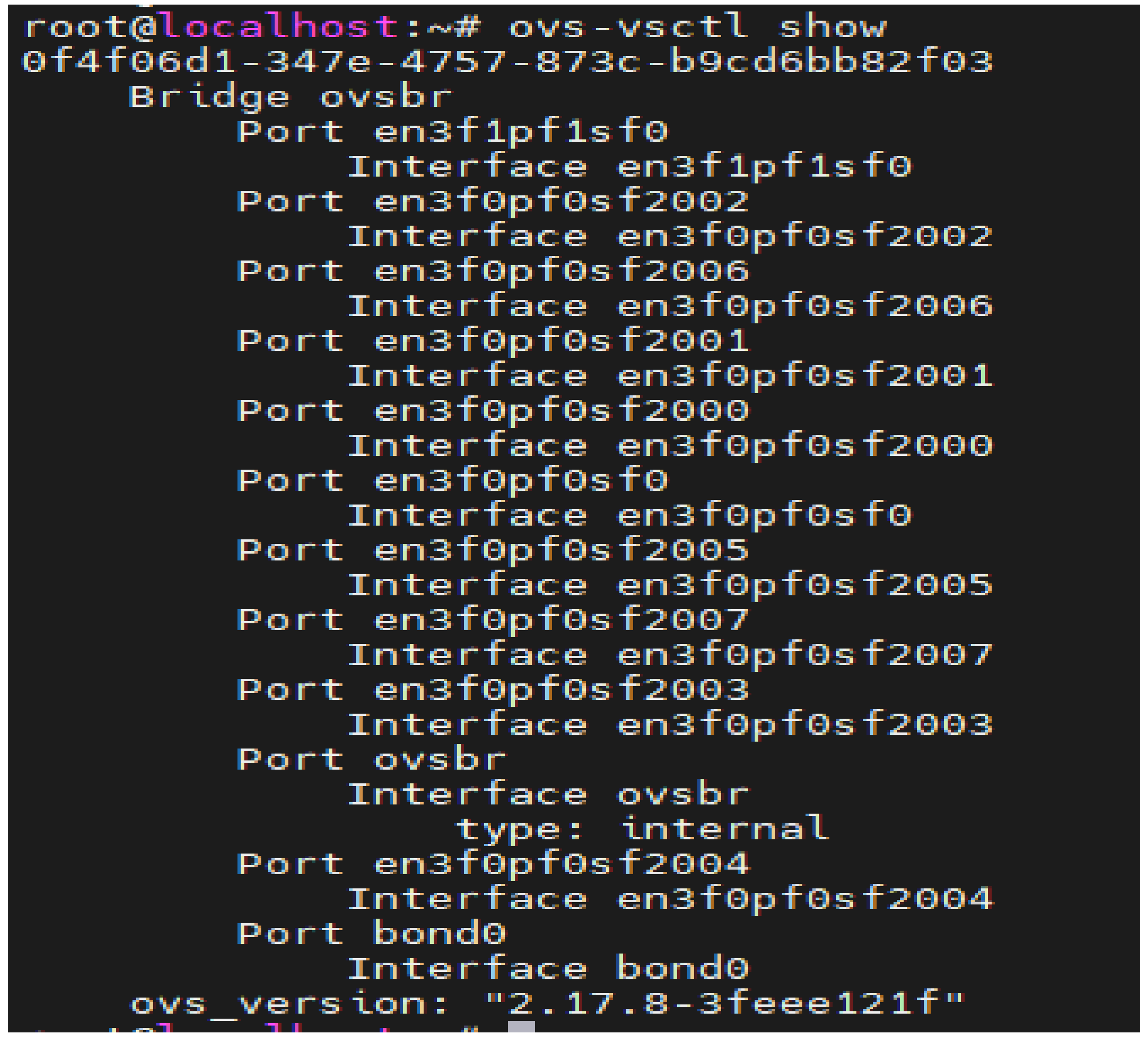
virtio-net hotplug – 4

7. virtio_net device sf rep into ovs for datapath offload

```
for i in {0..7};do ovs-vsctl add-port ovsbr en3f0pf0sf$((2000+i));done
ovs-vsctl show
```



```
root@localhost:~# virtnet list | grep sf_rep_net_device
        "sf_rep_net_device": "en3f0pf0sf2000",
        "sf_rep_net_device": "en3f0pf0sf2001",
        "sf_rep_net_device": "en3f0pf0sf2002",
        "sf_rep_net_device": "en3f0pf0sf2003",
        "sf_rep_net_device": "en3f0pf0sf2004",
        "sf_rep_net_device": "en3f0pf0sf2005",
        "sf_rep_net_device": "en3f0pf0sf2006",
        "sf_rep_net_device": "en3f0pf0sf2007",
root@localhost:~#
```

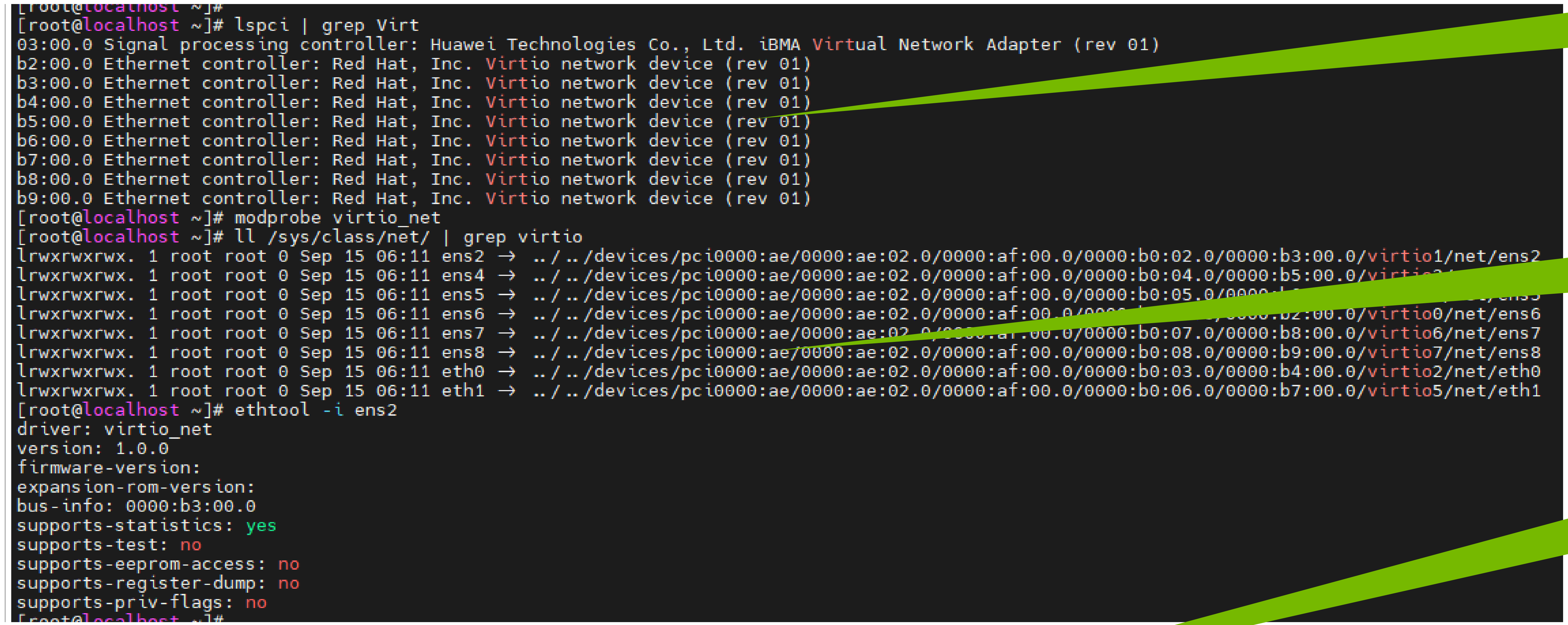


```
root@localhost:~# ovs-vsctl show
0f4f06d1-347e-4757-873c-b9cd6bb82f03
    Bridge ovsbr
        Port en3f1pf1sf0
            Interface en3f1pf1sf0
        Port en3f0pf0sf2002
            Interface en3f0pf0sf2002
        Port en3f0pf0sf2006
            Interface en3f0pf0sf2006
        Port en3f0pf0sf2001
            Interface en3f0pf0sf2001
        Port en3f0pf0sf2000
            Interface en3f0pf0sf2000
        Port en3f0pf0sf0
            Interface en3f0pf0sf0
        Port en3f0pf0sf2005
            Interface en3f0pf0sf2005
        Port en3f0pf0sf2007
            Interface en3f0pf0sf2007
        Port en3f0pf0sf2003
            Interface en3f0pf0sf2003
        Port ovsbr
            Interface ovsbr
                type: internal
        Port en3f0pf0sf2004
            Interface en3f0pf0sf2004
        Port bond0
            Interface bond0
    ovs_version: "2.17.8-3feee121f"
```

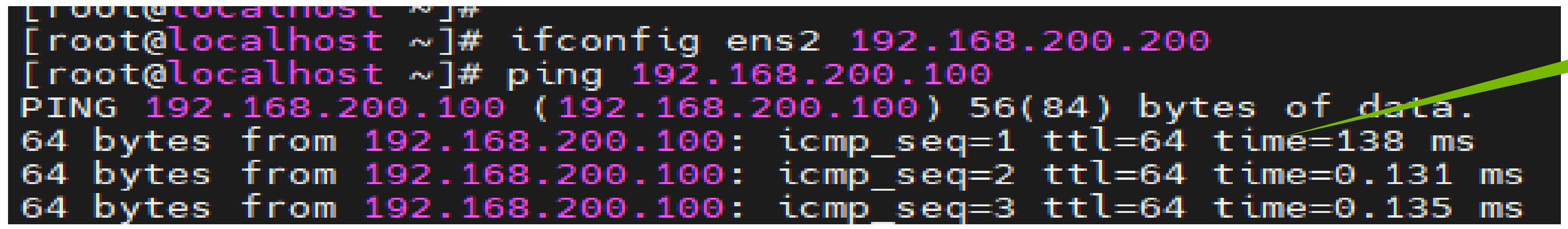
virtio-net hotplug – 5

8. verify host virtio_net devices and configure ip and ping to peer

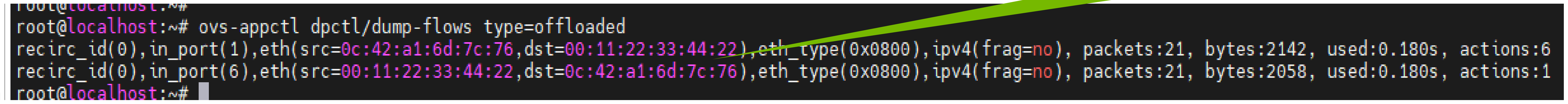
```
[host]# modprobe -v virtio-pci && modprobe -v virtio-net
```



```
[root@localhost ~]#
[root@localhost ~]# lspci | grep Virt
03:00.0 Signal processing controller: Huawei Technologies Co., Ltd. iBMA Virtual Network Adapter (rev 01)
b2:00.0 Ethernet controller: Red Hat, Inc. Virtio network device (rev 01)
b3:00.0 Ethernet controller: Red Hat, Inc. Virtio network device (rev 01)
b4:00.0 Ethernet controller: Red Hat, Inc. Virtio network device (rev 01)
b5:00.0 Ethernet controller: Red Hat, Inc. Virtio network device (rev 01)
b6:00.0 Ethernet controller: Red Hat, Inc. Virtio network device (rev 01)
b7:00.0 Ethernet controller: Red Hat, Inc. Virtio network device (rev 01)
b8:00.0 Ethernet controller: Red Hat, Inc. Virtio network device (rev 01)
b9:00.0 Ethernet controller: Red Hat, Inc. Virtio network device (rev 01)
[root@localhost ~]# modprobe virtio_net
[root@localhost ~]# ll /sys/class/net/ | grep virtio
lrwxrwxrwx. 1 root root 0 Sep 15 06:11 ens2 → ../../devices/pci0000:ae/0000:ae:02.0/0000:af:00.0/0000:b0:02.0/0000:b3:00.0/virtio1/net/ens2
lrwxrwxrwx. 1 root root 0 Sep 15 06:11 ens4 → ../../devices/pci0000:ae/0000:ae:02.0/0000:af:00.0/0000:b0:04.0/0000:b5:00.0/[illegible]
lrwxrwxrwx. 1 root root 0 Sep 15 06:11 ens5 → ../../devices/pci0000:ae/0000:ae:02.0/0000:af:00.0/0000:b0:05.0/0000[illegible]
lrwxrwxrwx. 1 root root 0 Sep 15 06:11 ens6 → ../../devices/pci0000:ae/0000:ae:02.0/0000:af:00.0/[illegible]:b2:00.0/virtio0/net/ens6
lrwxrwxrwx. 1 root root 0 Sep 15 06:11 ens7 → ../../devices/pci0000:ae/0000:ae:02.0/0000:af:00.0/0000:b0:07.0/0000:b8:00.0/virtio6/net/ens7
lrwxrwxrwx. 1 root root 0 Sep 15 06:11 ens8 → ../../devices/pci0000:ae/0000:ae:02.0/0000:af:00.0/0000:b0:08.0/0000:b9:00.0/virtio7/net/ens8
lrwxrwxrwx. 1 root root 0 Sep 15 06:11 eth0 → ../../devices/pci0000:ae/0000:ae:02.0/0000:af:00.0/0000:b0:03.0/0000:b4:00.0/virtio2/net/eth0
lrwxrwxrwx. 1 root root 0 Sep 15 06:11 eth1 → ../../devices/pci0000:ae/0000:ae:02.0/0000:af:00.0/0000:b0:06.0/0000:b7:00.0/virtio5/net/eth1
[root@localhost ~]# ethtool -i ens2
driver: virtio_net
version: 1.0.0
firmware-version:
expansion-rom-version:
bus-info: 0000:b3:00.0
supports-statistics: yes
supports-test: no
supports-eeprom-access: no
supports-register-dump: no
supports-priv-flags: no
[root@localhost ~]#
```



```
[root@localhost ~]#
[root@localhost ~]# ifconfig ens2 192.168.200.200
[root@localhost ~]# ping 192.168.200.100
PING 192.168.200.100 (192.168.200.100) 56(84) bytes of data.
64 bytes from 192.168.200.100: icmp_seq=1 ttl=64 time=138 ms
64 bytes from 192.168.200.100: icmp_seq=2 ttl=64 time=0.131 ms
64 bytes from 192.168.200.100: icmp_seq=3 ttl=64 time=0.135 ms
```



```
root@localhost:~#
root@localhost:~# ovs-appctl dpctl/dump-flows type=offloaded
recirc_id(0),in_port(1),eth(src=0c:42:a1:6d:7c:76,dst=00:11:22:33:44:22),eth_type(0x0800),ipv4(frag=no), packets:21, bytes:2142, used:0.180s, actions:6
recirc_id(0),in_port(6),eth(src=00:11:22:33:44:22,dst=0c:42:a1:6d:7c:76),eth_type(0x0800),ipv4(frag=no), packets:21, bytes:2058, used:0.180s, actions:1
root@localhost:~#
```

Check on host virtio-net pcie devices

Check on host virtio-net driver is loaded and ethernet devices is working

Run ping and traffic from host virtio_net

Check flows hw offloaded in DPU OVS

Virtio-net Static for Virtualization

virtio-net static pf/vf – 1

1. FW config

```
mlxconfig -d 03:00.0 -y reset
mlxconfig -d 03:00.0 -y set \
    INTERNAL_CPU_MODEL=1 SRIOV_EN=1 \
    PCI_SWITCH_EMULATION_ENABLE=0 \
    PCI_SWITCH_EMULATION_NUM_PORT=0 \
    VIRTIO_NET_EMULATION_ENABLE=1 \
    VIRTIO_NET_EMULATION_NUM_VF=126 \
    VIRTIO_NET_EMULATION_NUM_PF=4 \
    VIRTIO_NET_EMULATION_NUM_MSIX=32 \
    LAG_RESOURCE_ALLOCATION=1 \
    PER_PF_NUM_SF=1 PF_TOTAL_SF=512 \
    PF_SF_BAR_SIZE=8

mlxconfig -d 03:00.1 -y set \
    PER_PF_NUM_SF=1 PF_TOTAL_SF=2 \
    PF_SF_BAR_SIZE=8
```

2. Enable virtio-net-controller service

```
systemctl enable virtio-net-controller
```

3. Edit /opt/mellanox/mlnx_virtnet/virtnet.conf

```
{
    "ib_dev_lag": "mlx5_bond_0",
    "ib_dev_for_static_pf": "mlx5_bond_0",
    "is_lag": 1,
    "recovery": 1,
    "sf_pool_percent": 0,
    "sf_pool_force_destroy": 0
}
```

4. Powe cycle

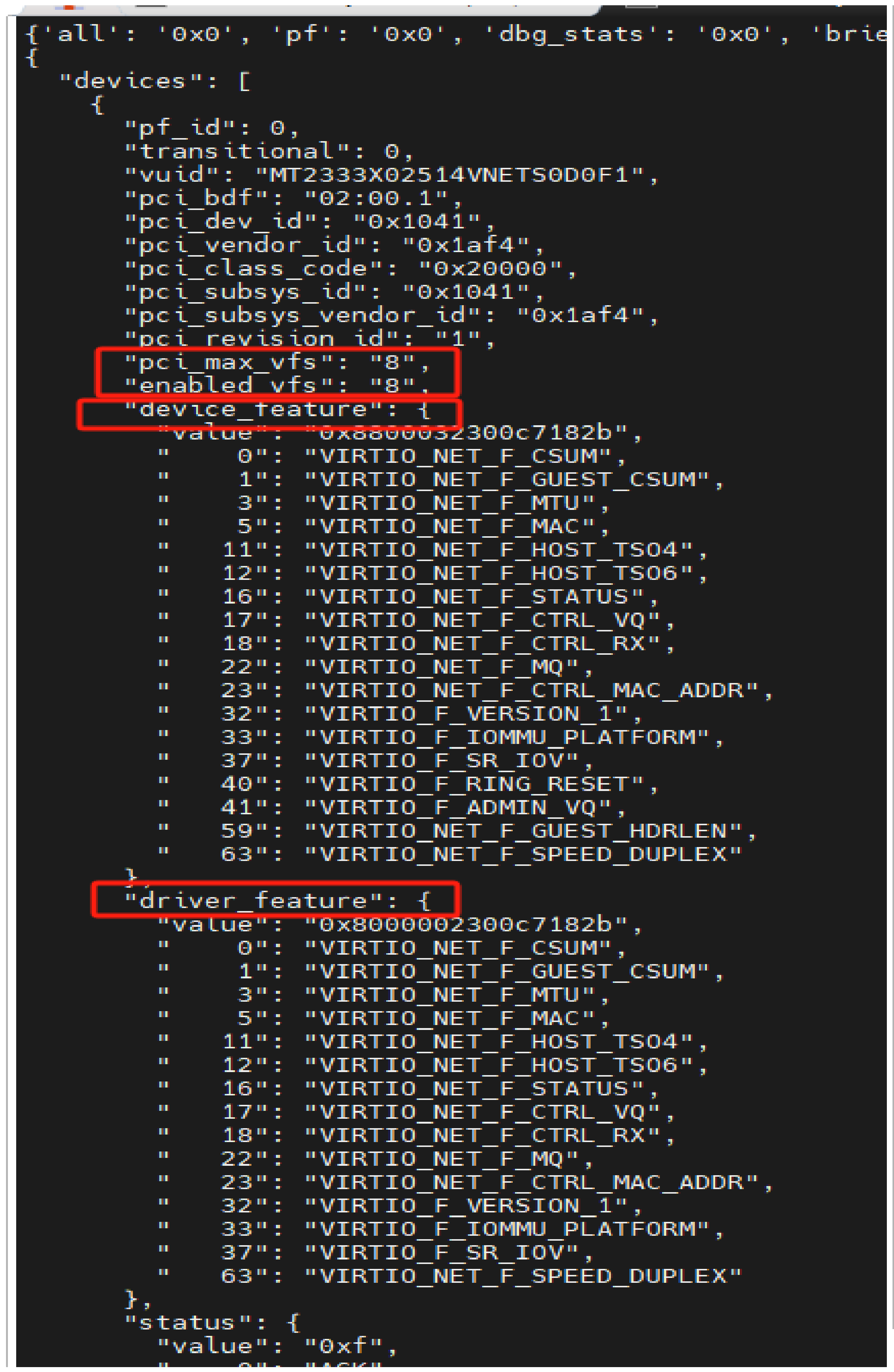
```
[host]
Add "pci=realloc intel_iommu=on iommu=pt" into
kernel boot grub options
Add "pci=assign-busses" in case > 127 VFs

[host]
ipmitool power cycle
```

virtio-net static pf/vf – 2

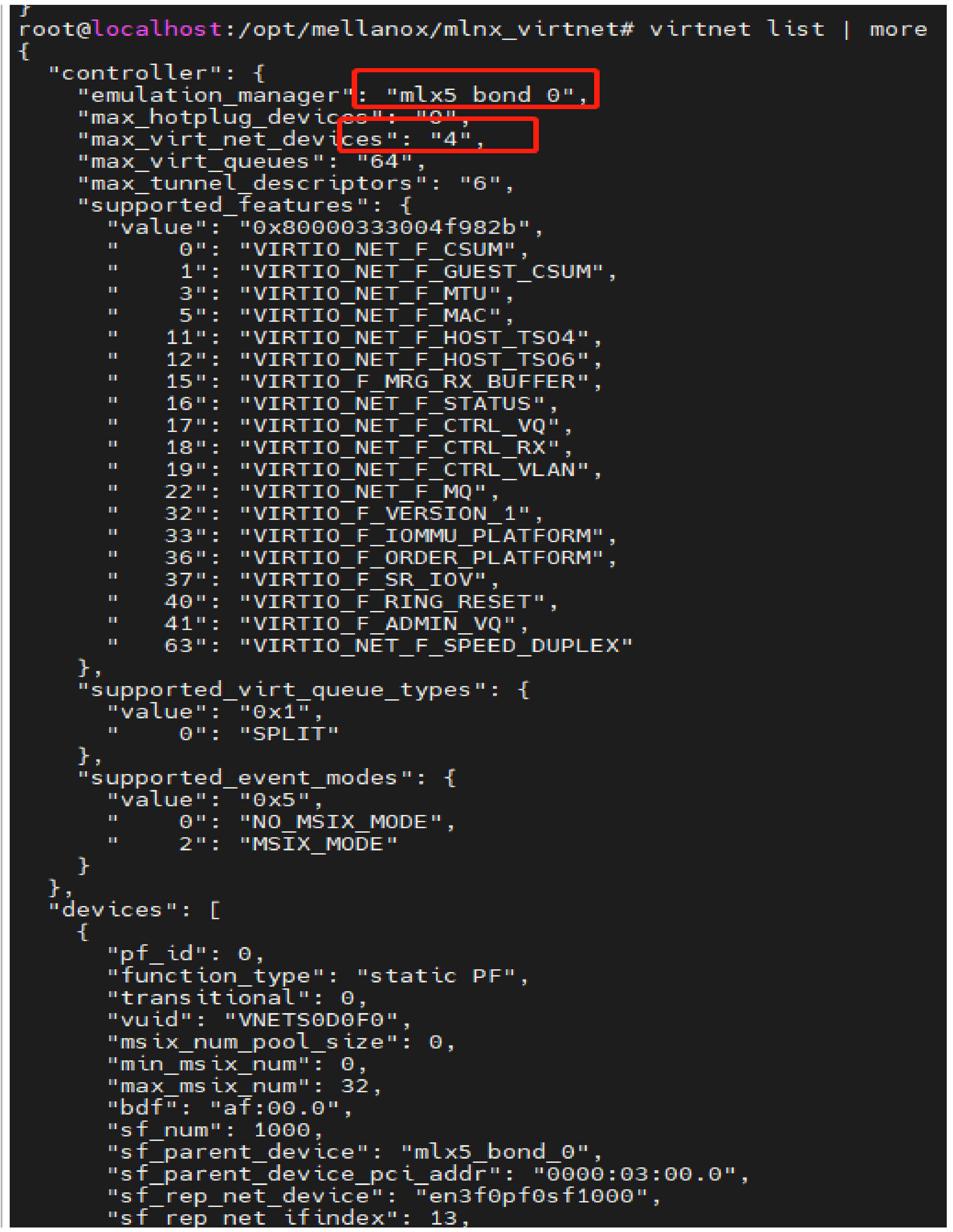
5. Check after power up

```
systemctl status
virtio-net-controller

virtnet list

virtnet query -p 0
```

- pci_max_vfs 是host virtio pci 设定
- msix 由
- Virtio Feature 由device和driver协商确定



```
{'all': '0x0', 'pf': '0x0', 'dbg_stats': '0x0', 'brie
{
  "devices": [
    {
      "pf_id": 0,
      "transitional": 0,
      "vuid": "MT2333X02514VNETS0D0F1",
      "pci_bdf": "02:00.1",
      "pci_dev_id": "0x1041",
      "pci_vendor_id": "0x1af4",
      "pci_class_code": "0x20000",
      "pci_subsys_id": "0x1041",
      "pci_subsys_vendor_id": "0x1af4",
      "pci_revision_id": "1",
      "pci_max_vfs": "8",
      "enabled_vfs": "8",
      "device_feature": {
        "value": "0x8800032300c7182b",
        "     0": "VIRTIO_NET_F_CSUM",
        "     1": "VIRTIO_NET_F_GUEST_CSUM",
        "     3": "VIRTIO_NET_F_MTU",
        "     5": "VIRTIO_NET_F_MAC",
        "    11": "VIRTIO_NET_F_HOST_TSO4",
        "    12": "VIRTIO_NET_F_HOST_TSO6",
        "    16": "VIRTIO_NET_F_STATUS",
        "    17": "VIRTIO_NET_F_CTRL_VQ",
        "    18": "VIRTIO_NET_F_CTRL_RX",
        "    22": "VIRTIO_NET_F_MQ",
        "    23": "VIRTIO_NET_F_CTRL_MAC_ADDR",
        "    32": "VIRTIO_F_VERSION_1",
        "    33": "VIRTIO_F_IOMMU_PLATFORM",
        "    37": "VIRTIO_F_SR_IOV",
        "    40": "VIRTIO_F_RING_RESET",
        "    41": "VIRTIO_F_ADMIN_VQ",
        "    59": "VIRTIO_NET_F_GUEST_HDRLEN",
        "    63": "VIRTIO_NET_F_SPEED_DUPLEX"
      },
      "driver_feature": {
        "value": "0x8000002300c7182b",
        "     0": "VIRTIO_NET_F_CSUM",
        "     1": "VIRTIO_NET_F_GUEST_CSUM",
        "     3": "VIRTIO_NET_F_MTU",
        "     5": "VIRTIO_NET_F_MAC",
        "    11": "VIRTIO_NET_F_HOST_TSO4",
        "    12": "VIRTIO_NET_F_HOST_TSO6",
        "    16": "VIRTIO_NET_F_STATUS",
        "    17": "VIRTIO_NET_F_CTRL_VQ",
        "    18": "VIRTIO_NET_F_CTRL_RX",
        "    22": "VIRTIO_NET_F_MQ",
        "    23": "VIRTIO_NET_F_CTRL_MAC_ADDR",
        "    32": "VIRTIO_F_VERSION_1",
        "    33": "VIRTIO_F_IOMMU_PLATFORM",
        "    37": "VIRTIO_F_SR_IOV",
        "    63": "VIRTIO_NET_F_SPEED_DUPLEX"
      },
      "status": {
        "value": "0xf",
```

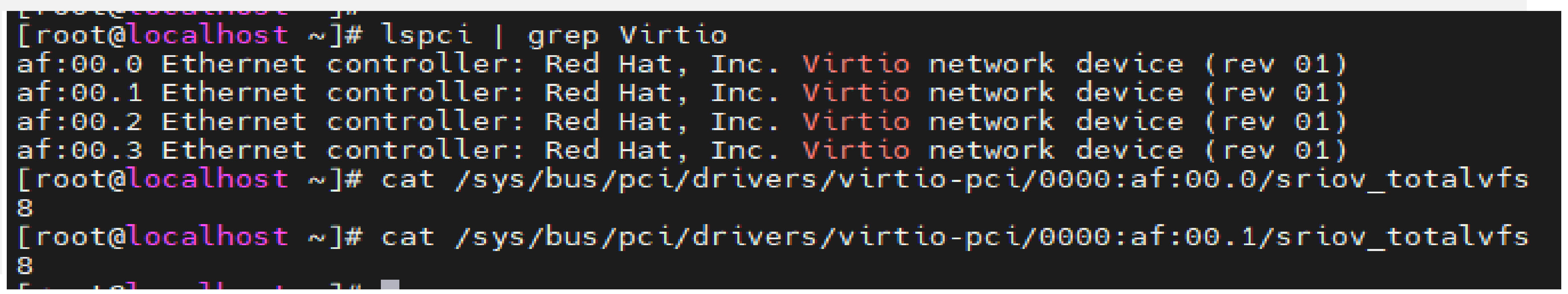


```
}
root@localhost:/opt/mellanox/mlnx_virtnet# virtnet list | more
{
  "controller": {
    "emulation_manager": "mlx5_bond_0",
    "max_hotplug_devices": "0",
    "max_virt_net_devices": "4",
    "max_virt_queues": "64",
    "max_tunnel_descriptors": "6",
    "supported_features": {
      "value": "0x80000333004f982b",
      "     0": "VIRTIO_NET_F_CSUM",
      "     1": "VIRTIO_NET_F_GUEST_CSUM",
      "     3": "VIRTIO_NET_F_MTU",
      "     5": "VIRTIO_NET_F_MAC",
      "    11": "VIRTIO_NET_F_HOST_TSO4",
      "    12": "VIRTIO_NET_F_HOST_TSO6",
      "    15": "VIRTIO_F_MRG_RX_BUFFER",
      "    16": "VIRTIO_NET_F_STATUS",
      "    17": "VIRTIO_NET_F_CTRL_VQ",
      "    18": "VIRTIO_NET_F_CTRL_RX",
      "    19": "VIRTIO_NET_F_CTRL_VLAN",
      "    22": "VIRTIO_NET_F_MQ",
      "    32": "VIRTIO_F_VERSION_1",
      "    33": "VIRTIO_F_IOMMU_PLATFORM",
      "    36": "VIRTIO_F_ORDER_PLATFORM",
      "    37": "VIRTIO_F_SR_IOV",
      "    40": "VIRTIO_F_RING_RESET",
      "    41": "VIRTIO_F_ADMIN_VQ",
      "    63": "VIRTIO_NET_F_SPEED_DUPLEX"
    },
    "supported_virt_queue_types": {
      "value": "0x1",
      "     0": "SPLIT"
    },
    "supported_event_modes": {
      "value": "0x5",
      "     0": "NO_MSIX_MODE",
      "     2": "MSIX_MODE"
    }
  },
  "devices": [
    {
      "pf_id": 0,
      "function_type": "static PF",
      "transitional": 0,
      "vuid": "VNETS0D0F0",
      "msix_num_pool_size": 0,
      "min_msix_num": 0,
      "max_msix_num": 32,
      "bdf": "af:00.0",
      "sf_num": 1000,
      "sf_parent_device": "mlx5_bond_0",
      "sf_parent_device_pci_addr": "0000:03:00.0",
      "sf_rep_net_device": "en3f0pf0sf1000",
      "sf_rep_net_ifindex": 13,
```

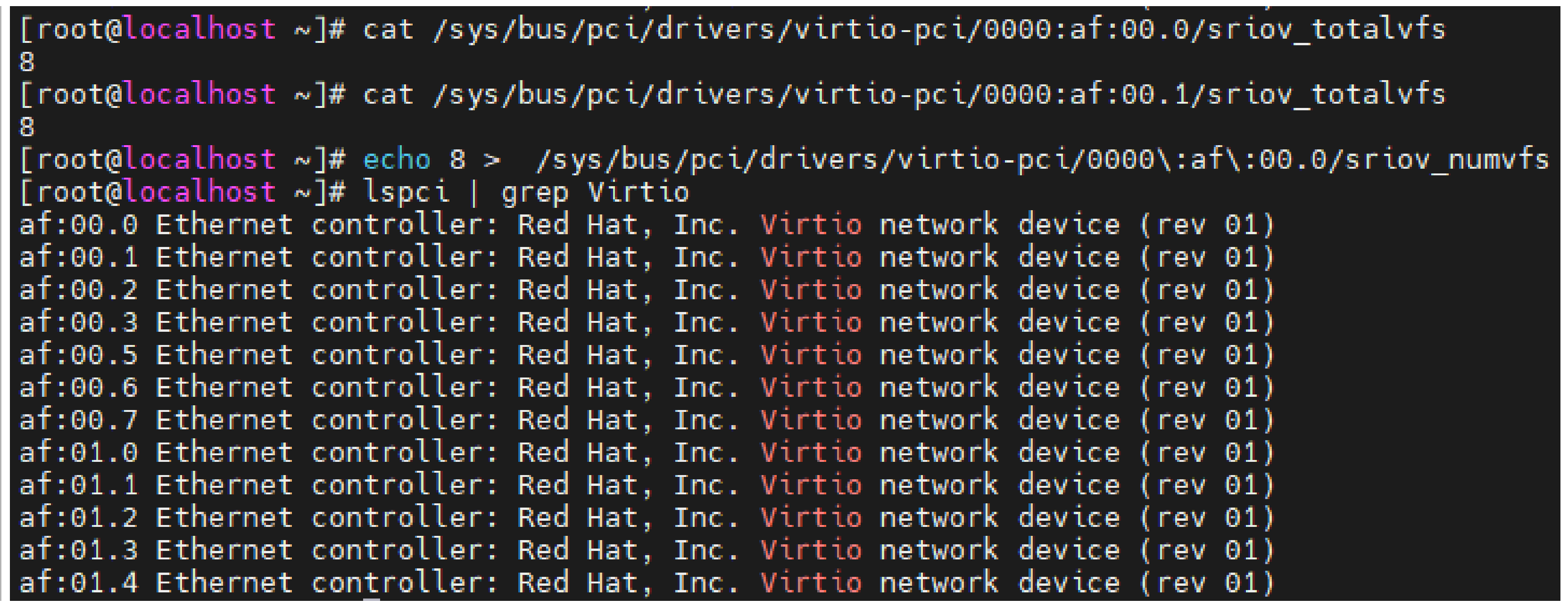
virtio-net static pf/vf – 3

6. Create virtio_net vf and modify mac and features

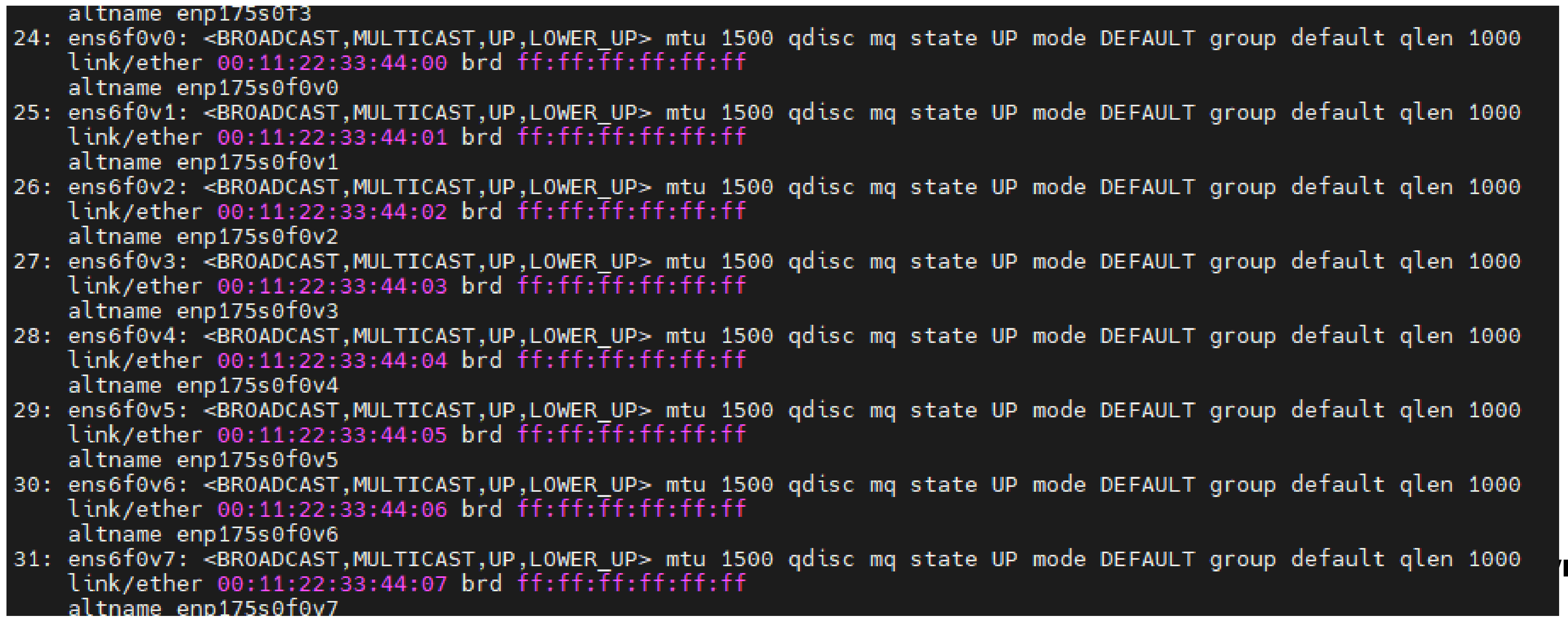
```
[host]  #static vf 是从host上 virtio-pci创建
    #echo 0 > /sys/bus/pci/drivers/virtio-pci/0000:af:00.0/sriov_drivers_autoprobe
    lspci | grep Virtio
    cat /sys/bus/pci/drivers/virtio-pci/0000:af:00.0/sriov_totalvfs
    cat /sys/bus/pci/drivers/virtio-pci/0000:af:00.1/sriov_totalvfs
    modprobe -r virtio_net
    echo 8 > /sys/bus/pci/drivers/virtio-pci/0000:af:00.0/sriov_numvfs
    echo 8 > /sys/bus/pci/drivers/virtio-pci/0000:af:00.1/sriov_numvfs
[dpu]
    for i in {0..7};do virtnet modify -p 0 -v $i device -m "00:11:22:33:44:0$((0+i))" -f 0x23004f182b;done
    virtnet list
    virtnet query -p 0 -v 0

[host]
   modprobe -v virtio_net
```



```
[root@localhost ~]# lspci | grep Virtio
af:00.0 Ethernet controller: Red Hat, Inc. Virtio network device (rev 01)
af:00.1 Ethernet controller: Red Hat, Inc. Virtio network device (rev 01)
af:00.2 Ethernet controller: Red Hat, Inc. Virtio network device (rev 01)
af:00.3 Ethernet controller: Red Hat, Inc. Virtio network device (rev 01)
[root@localhost ~]# cat /sys/bus/pci/drivers/virtio-pci/0000:af:00.0/sriov_totalvfs
8
[root@localhost ~]# cat /sys/bus/pci/drivers/virtio-pci/0000:af:00.1/sriov_totalvfs
8
```



```
[root@localhost ~]# cat /sys/bus/pci/drivers/virtio-pci/0000:af:00.0/sriov_totalvfs
8
[root@localhost ~]# cat /sys/bus/pci/drivers/virtio-pci/0000:af:00.1/sriov_totalvfs
8
[root@localhost ~]# echo 8 >  /sys/bus/pci/drivers/virtio-pci/0000\:af\:00.0/sriov_numvfs
[root@localhost ~]# lspci | grep Virtio
af:00.0 Ethernet controller: Red Hat, Inc. Virtio network device (rev 01)
af:00.1 Ethernet controller: Red Hat, Inc. Virtio network device (rev 01)
af:00.2 Ethernet controller: Red Hat, Inc. Virtio network device (rev 01)
af:00.3 Ethernet controller: Red Hat, Inc. Virtio network device (rev 01)
af:00.5 Ethernet controller: Red Hat, Inc. Virtio network device (rev 01)
af:00.6 Ethernet controller: Red Hat, Inc. Virtio network device (rev 01)
af:00.7 Ethernet controller: Red Hat, Inc. Virtio network device (rev 01)
af:01.0 Ethernet controller: Red Hat, Inc. Virtio network device (rev 01)
af:01.1 Ethernet controller: Red Hat, Inc. Virtio network device (rev 01)
af:01.2 Ethernet controller: Red Hat, Inc. Virtio network device (rev 01)
af:01.3 Ethernet controller: Red Hat, Inc. Virtio network device (rev 01)
af:01.4 Ethernet controller: Red Hat, Inc. Virtio network device (rev 01)
```

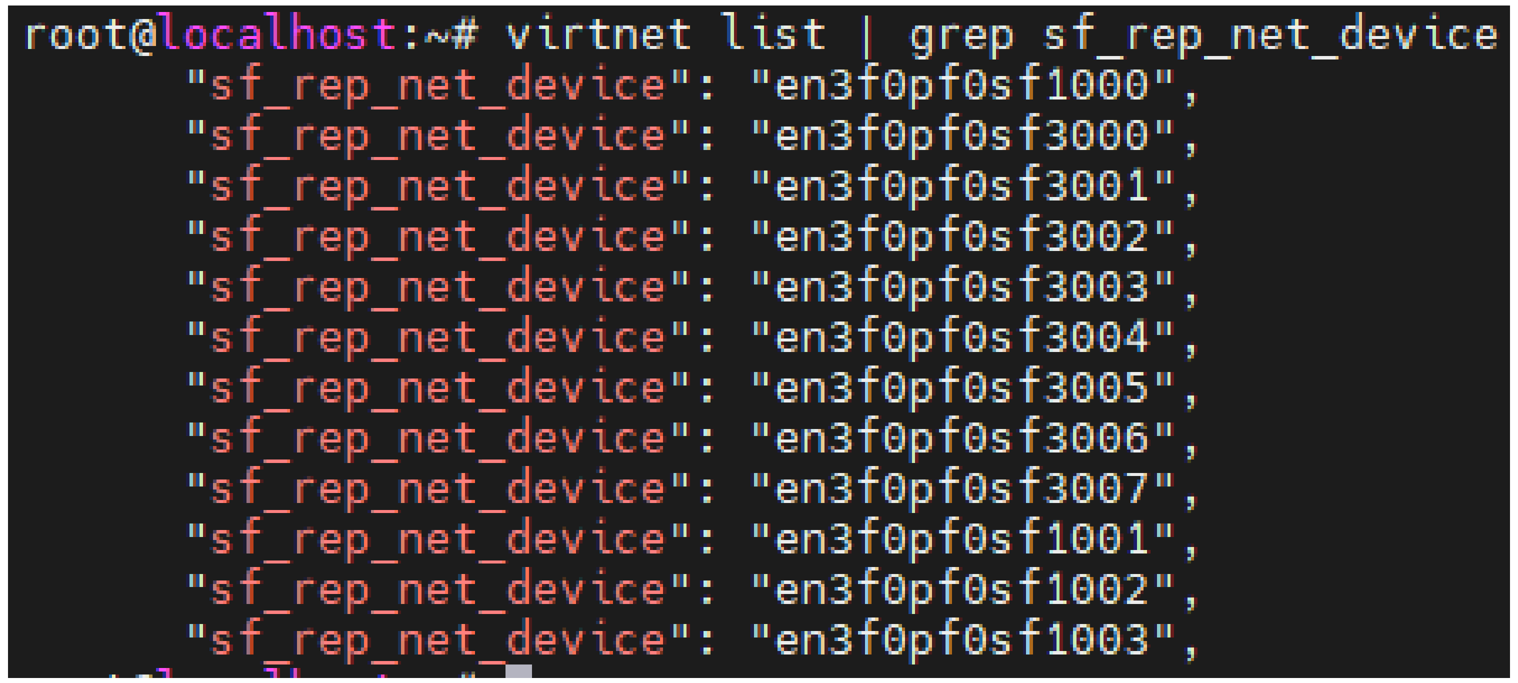


```
    altname enp175s0f3
24: ens6f0v0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default qlen 1000
    link/ether 00:11:22:33:44:00 brd ff:ff:ff:ff:ff:ff
    altname enp175s0f0v0
25: ens6f0v1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default qlen 1000
    link/ether 00:11:22:33:44:01 brd ff:ff:ff:ff:ff:ff
    altname enp175s0f0v1
26: ens6f0v2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default qlen 1000
    link/ether 00:11:22:33:44:02 brd ff:ff:ff:ff:ff:ff
    altname enp175s0f0v2
27: ens6f0v3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default qlen 1000
    link/ether 00:11:22:33:44:03 brd ff:ff:ff:ff:ff:ff
    altname enp175s0f0v3
28: ens6f0v4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default qlen 1000
    link/ether 00:11:22:33:44:04 brd ff:ff:ff:ff:ff:ff
    altname enp175s0f0v4
29: ens6f0v5: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default qlen 1000
    link/ether 00:11:22:33:44:05 brd ff:ff:ff:ff:ff:ff
    altname enp175s0f0v5
30: ens6f0v6: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default qlen 1000
    link/ether 00:11:22:33:44:06 brd ff:ff:ff:ff:ff:ff
    altname enp175s0f0v6
31: ens6f0v7: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default qlen 1000
    link/ether 00:11:22:33:44:07 brd ff:ff:ff:ff:ff:ff
    altname enp175s0f0v7
```

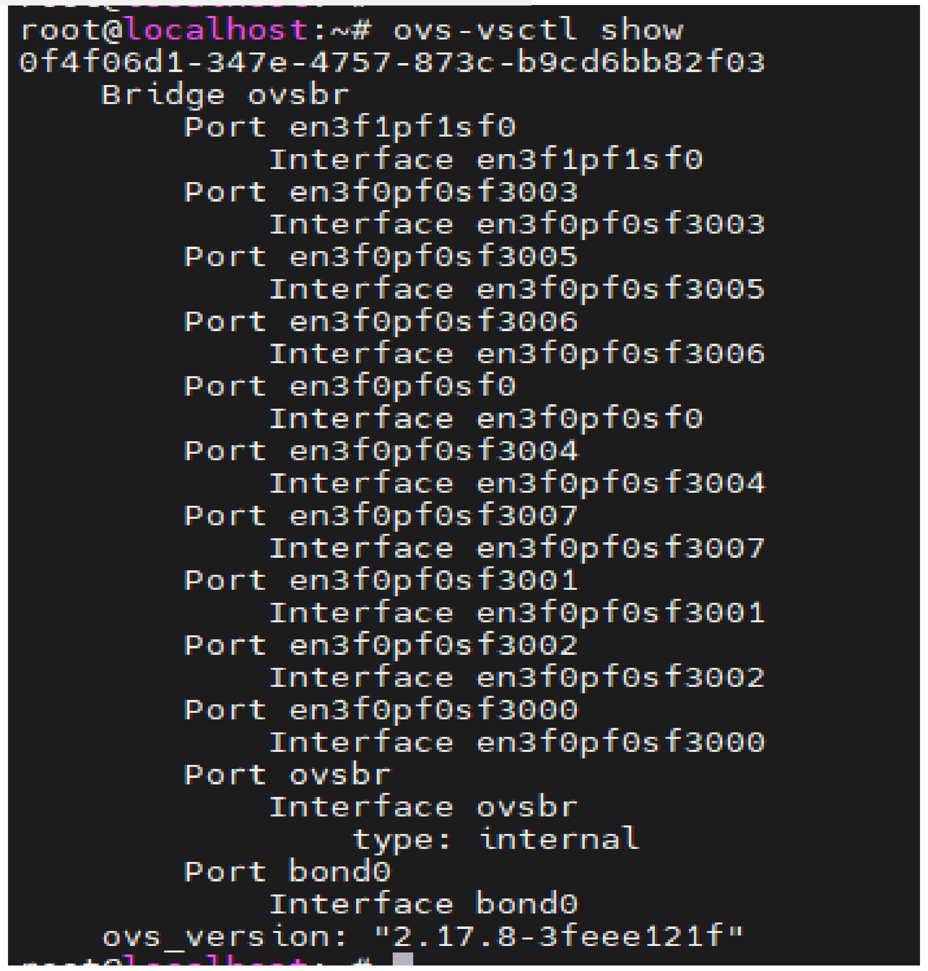
virtio-net static pf/vf – 4

5. ovs and ping verify

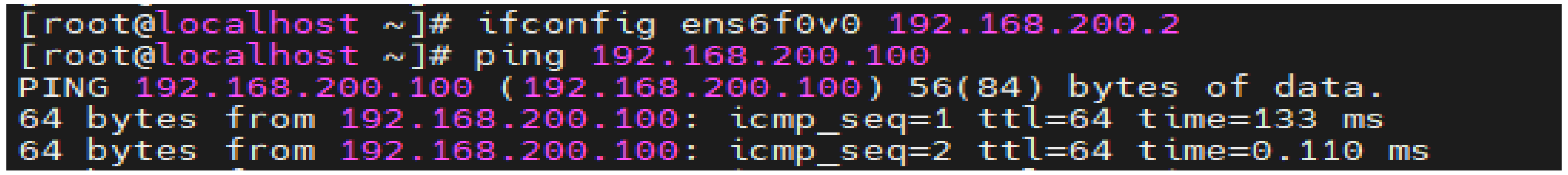
```
[dpu]
    virtnet list | grep sf_rep_net_device
    for i in {0..7};do ovs-vsctl add-port ovsbr en3f0pf0sf$((3000+i));done
    ovs-vsctl show
[host]
    ifconfig ens6f0v0 192.168.200.2
[dpu]
    ovs-appctl dpctl/dump-flows type=offloaded
```



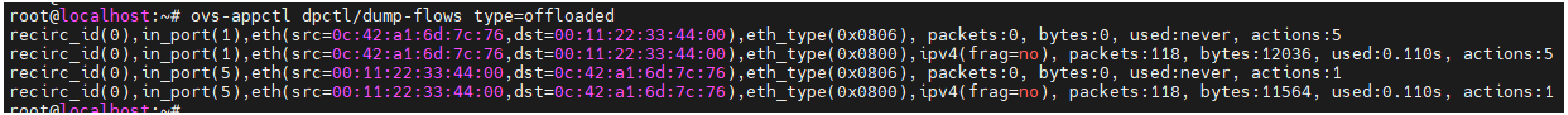
```
root@localhost:~# virtnet list | grep sf_rep_net_device
        "sf_rep_net_device": "en3f0pf0sf1000",
        "sf_rep_net_device": "en3f0pf0sf3000",
        "sf_rep_net_device": "en3f0pf0sf3001",
        "sf_rep_net_device": "en3f0pf0sf3002",
        "sf_rep_net_device": "en3f0pf0sf3003",
        "sf_rep_net_device": "en3f0pf0sf3004",
        "sf_rep_net_device": "en3f0pf0sf3005",
        "sf_rep_net_device": "en3f0pf0sf3006",
        "sf_rep_net_device": "en3f0pf0sf3007",
        "sf_rep_net_device": "en3f0pf0sf1001",
        "sf_rep_net_device": "en3f0pf0sf1002",
        "sf_rep_net_device": "en3f0pf0sf1003",
```



```
root@localhost:~# ovs-vsctl show
0f4f06d1-347e-4757-873c-b9cd6bb82f03
    Bridge ovsbr
        Port en3f1pf1sf0
            Interface en3f1pf1sf0
        Port en3f0pf0sf3003
            Interface en3f0pf0sf3003
        Port en3f0pf0sf3005
            Interface en3f0pf0sf3005
        Port en3f0pf0sf3006
            Interface en3f0pf0sf3006
        Port en3f0pf0sf0
            Interface en3f0pf0sf0
        Port en3f0pf0sf3004
            Interface en3f0pf0sf3004
        Port en3f0pf0sf3007
            Interface en3f0pf0sf3007
        Port en3f0pf0sf3001
            Interface en3f0pf0sf3001
        Port en3f0pf0sf3002
            Interface en3f0pf0sf3002
        Port en3f0pf0sf3000
            Interface en3f0pf0sf3000
        Port ovsbr
            Interface ovsbr
                type: internal
        Port bond0
            Interface bond0
    ovs_version: "2.17.8-3feee121f"
```



```
[root@localhost ~]# ifconfig ens6f0v0 192.168.200.2
[root@localhost ~]# ping 192.168.200.100
PING 192.168.200.100 (192.168.200.100) 56(84) bytes of data.
64 bytes from 192.168.200.100: icmp_seq=1 ttl=64 time=133 ms
64 bytes from 192.168.200.100: icmp_seq=2 ttl=64 time=0.110 ms
```



```
root@localhost:~# ovs-appctl dpctl/dump-flows type=offloaded
recirc_id(0),in_port(1),eth(src=0c:42:a1:6d:7c:76,dst=00:11:22:33:44:00),eth_type(0x0806), packets:0, bytes:0, used:never, actions:5
recirc_id(0),in_port(1),eth(src=0c:42:a1:6d:7c:76,dst=00:11:22:33:44:00),eth_type(0x0800),ipv4(frag=no), packets:118, bytes:12036, used:0.110s, actions:5
recirc_id(0),in_port(5),eth(src=00:11:22:33:44:00,dst=0c:42:a1:6d:7c:76),eth_type(0x0806), packets:0, bytes:0, used:never, actions:1
recirc_id(0),in_port(5),eth(src=00:11:22:33:44:00,dst=0c:42:a1:6d:7c:76),eth_type(0x0800),ipv4(frag=no), packets:118, bytes:11564, used:0.110s, actions:1
```

Virtio VF Dynamic MSIX

- 每个PF支持动态msi池，缺省是0，VF按固定值分配

```
[DPU]# mlxconfig -y -d 03:00.0 s VIRTIO_NET_EMULATION_NUM_MSIX=32 NUM_VF_MSIX=32
```

- 查询当前PF msix池的大小，初始化是0，因为msix被固定分配出去了，需要从已分配的VF中回收

```
[DPU]# virtnet list | grep -i '"pf_id": 0' -A 8 | grep -i msix_num_pool_size
```

- 查询VF可以分配的msix的范围

```
[DPU]# virtnet list | grep -i '"pf_id": 0' -A 8 | grep -i max_msix_num
[DPU]# virtnet list | grep -i '"pf_id": 0' -A 8 | grep -i min_msix_num
```

- 释放所有的VF的msix到PF的msix池，然后重新按需分配

```
[host]# echo <vf0_bdf> > /sys/bus/pci/drivers/virtio-pci/unbind
[host]# echo <vf1_bdf> > /sys/bus/pci/drivers/virtio-pci/unbind

[DPU]# virtnet modify -p 0 -v 0 device -n 0
[DPU]# virtnet modify -p 0 -v 1 device -n 0
[DPU]# virtnet list | grep -i '"pf_id": 0' -A 8 | grep -i msix_num_pool_size
```

- 重新分配VF的msix输出

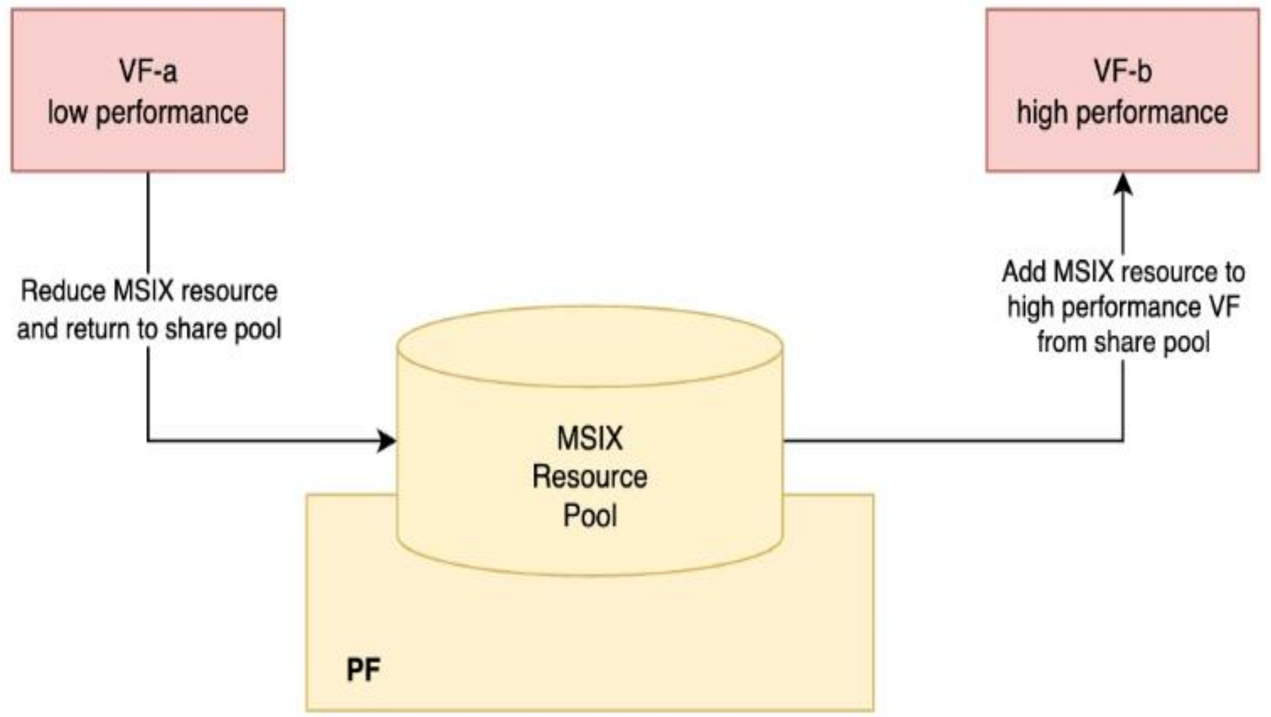
```
[DPU]# virtnet modify -p 0 -v 0 device -n 48
[DPU]# virtnet modify -p 0 -v 1 device -n 16
```

- 查询分配给VF的msix数

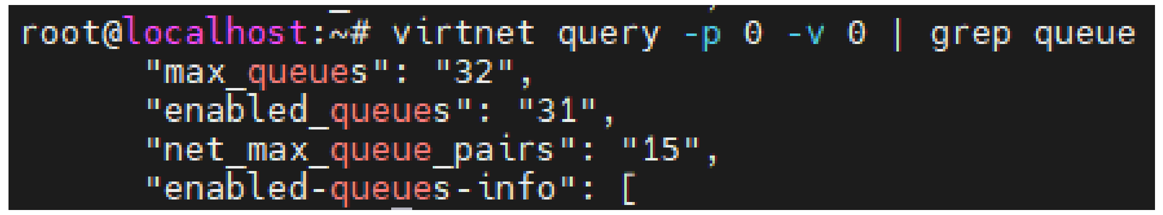
```
[DPU]# virtnet query -p 0 -v 0 | grep -i num_msix
[DPU]# virtnet query -p 0 -v 1 | grep -i num_msix
```

- 重新在host上加载VF的virtio驱动

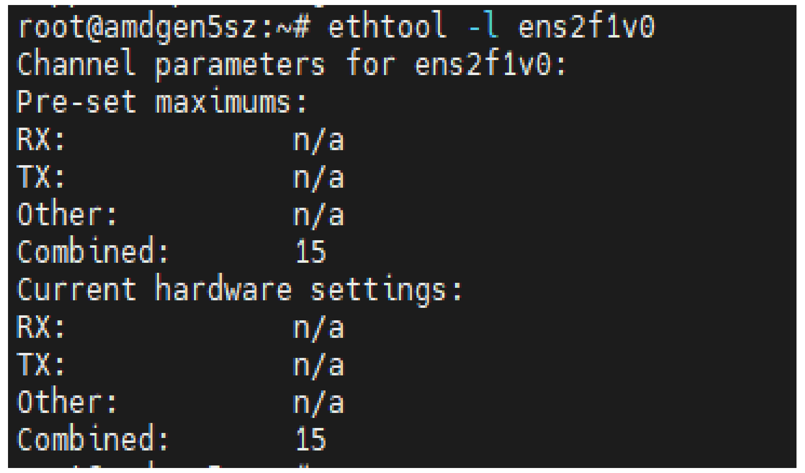
```
[host]# echo <vf0_bdf> > /sys/bus/pci/drivers/virtio-pci/bind
[host]# echo <vf1_bdf> > /sys/bus/pci/drivers/virtio-pci/bind
```



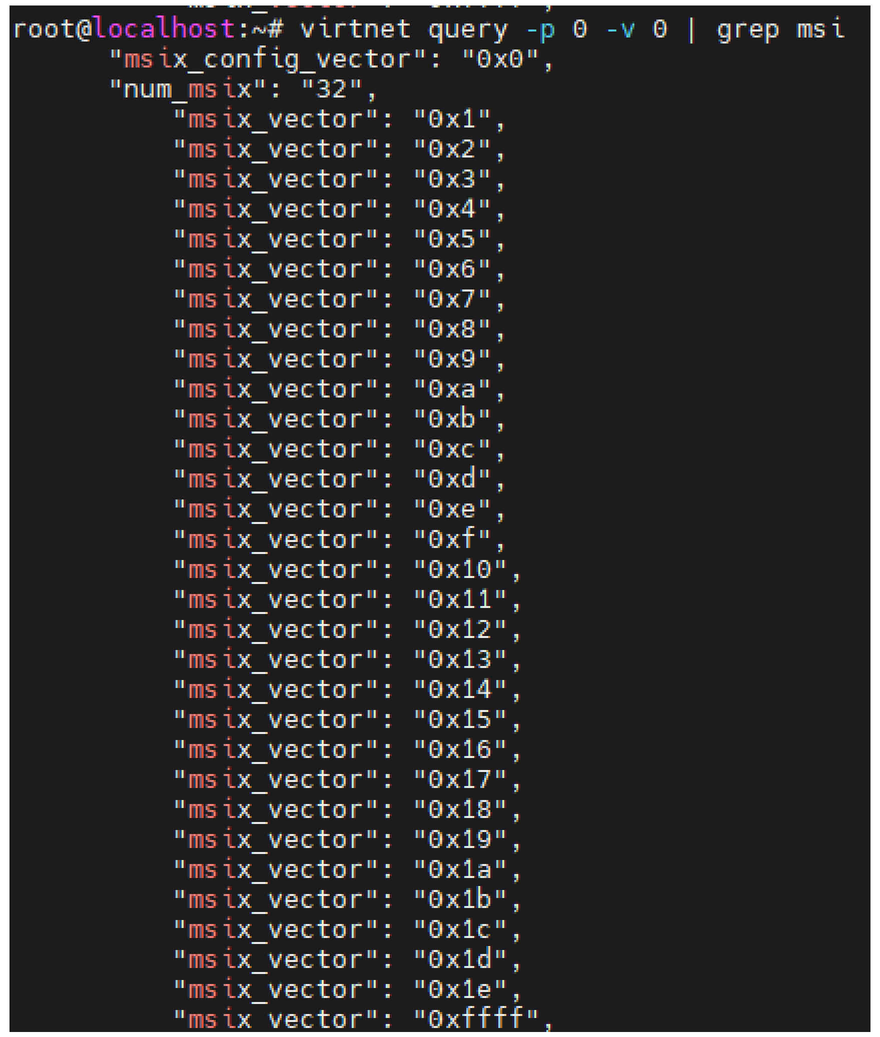
VF-a
low performance
VF-b
high performance
Reduce MSIX resource and return to share pool
Add MSIX resource to high performance VF from share pool
MSIX Resource Pool
PF



```
root@localhost:~# virtnet query -p 0 -v 0 | grep queue
        "max_queues": "32",
        "enabled_queues": "31",
        "net_max_queue_pairs": "15",
        "enabled-queues-info": [
```



```
root@amdgen5sz:~# ethtool -l ens2f1v0
Channel parameters for ens2f1v0:
Pre-set maximums:
RX:             n/a
TX:             n/a
Other:          n/a
Combined:       15
Current hardware settings:
RX:             n/a
TX:             n/a
Other:          n/a
Combined:       15
```



```
root@localhost:~# virtnet query -p 0 -v 0 | grep msi
      "msix_config_vector": "0x0",
      "num_msix": "32",
          "msix_vector": "0x1",
          "msix_vector": "0x2",
          "msix_vector": "0x3",
          "msix_vector": "0x4",
          "msix_vector": "0x5",
          "msix_vector": "0x6",
          "msix_vector": "0x7",
          "msix_vector": "0x8",
          "msix_vector": "0x9",
          "msix_vector": "0xa",
          "msix_vector": "0xb",
          "msix_vector": "0xc",
          "msix_vector": "0xd",
          "msix_vector": "0xe",
          "msix_vector": "0xf",
          "msix_vector": "0x10",
          "msix_vector": "0x11",
          "msix_vector": "0x12",
          "msix_vector": "0x13",
          "msix_vector": "0x14",
          "msix_vector": "0x15",
          "msix_vector": "0x16",
          "msix_vector": "0x17",
          "msix_vector": "0x18",
          "msix_vector": "0x19",
          "msix_vector": "0x1a",
          "msix_vector": "0x1b",
          "msix_vector": "0x1c",
          "msix_vector": "0x1d",
          "msix_vector": "0x1e",
          "msix_vector": "0xffff",
```

virtio-net run with ovs-dpdk example

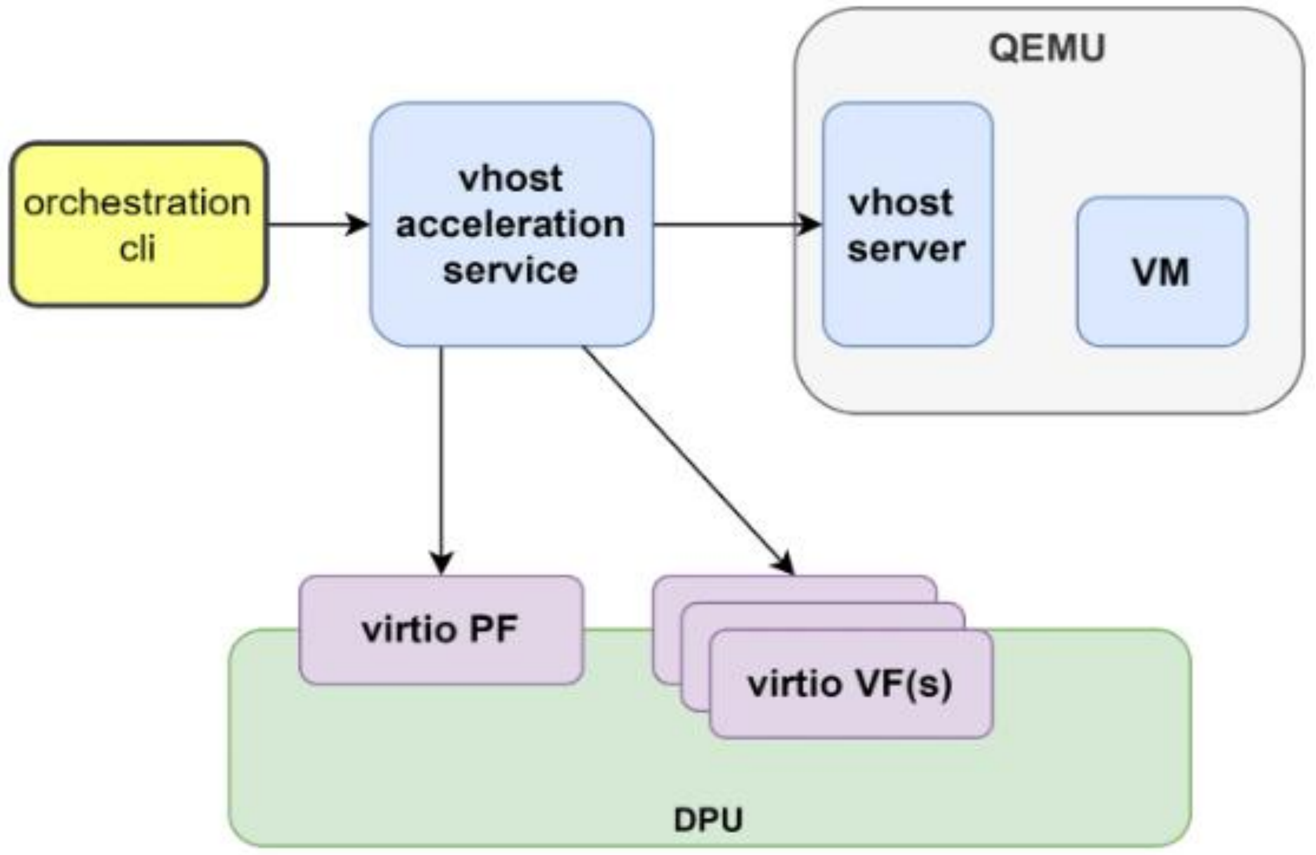
- LAG+Jumbo+vxlan+connection_tracking

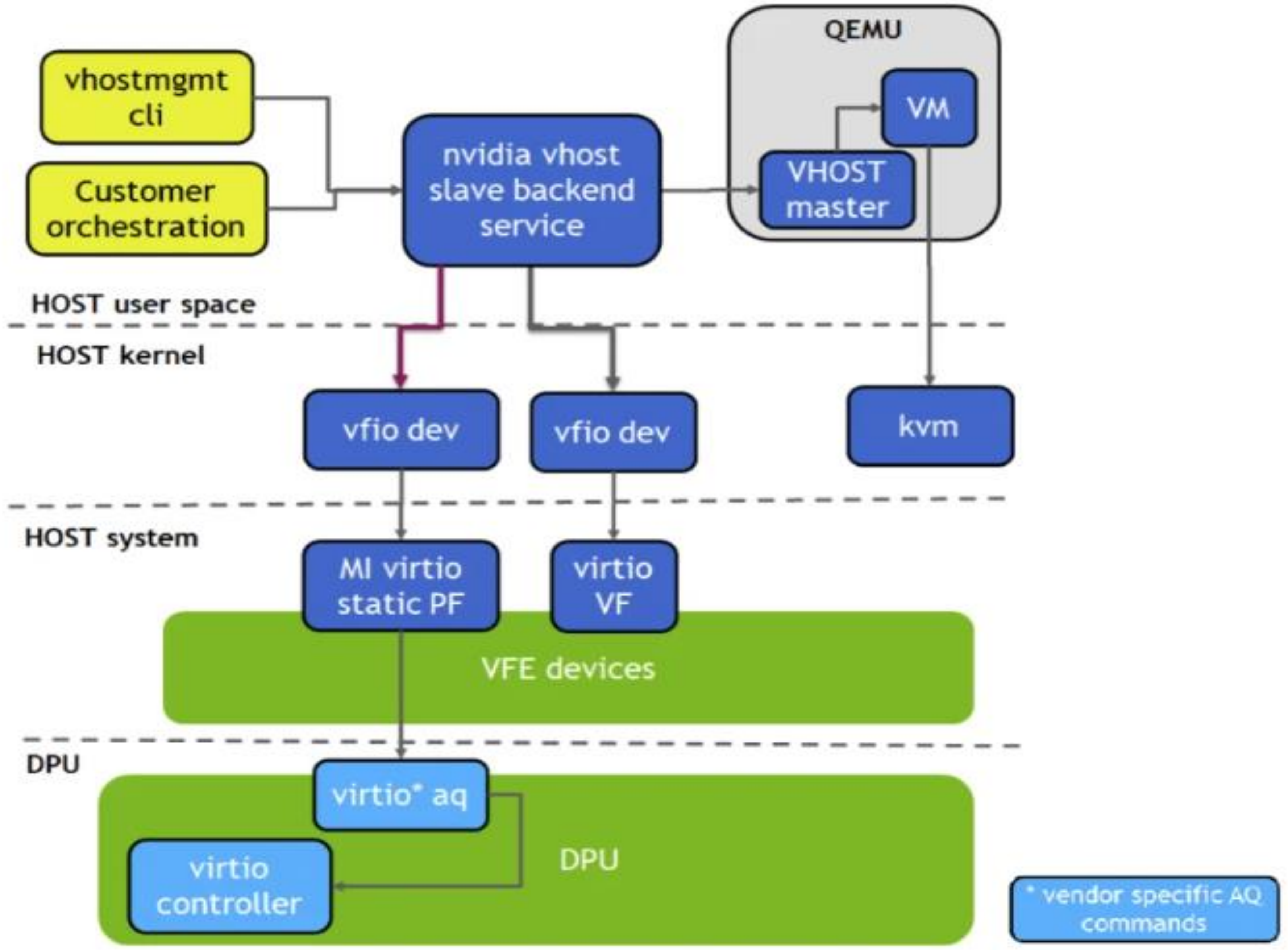
	Static PF	Hotplug PF	SR-IOV VF
SF Range	1000-999	2000-2999	3000 and above

```
echo 2048 > /sys/devices/system/node/node0/hugepages/hugepages-2048kB/nr_hugepages
ovs-vsctl add-br br-int -- set bridge br-int datapath_type=netdev
ovs-vsctl add-br br-ext -- set bridge br-ext datapath_type=netdev
ovs-vsctl --no-wait set o . other_config:dpdk-extra="-a 0000:03:00.0,representor=pf0sf[2000-
2030,65535],dv_xmeta_en=1,sys_mem_en=1 mtu_request=9000",
ovs-vsctl --no-wait set o . other_config:dpdk-init=true
ovs-vsctl --no-wait set o . other_config:hw-offload=true other_config:max-idle=60000
ovs-vsctl add-port br-ext bond0 -- set Interface bond0 type=dpdk options:dpdk-
devargs="0000:03:00.0,dv_xmeta_en=1,sys_mem_en=1" mtu_request=9000
ovs-vsctl add-port br-int hostpf -- set Interface hostpf type=dpdk options:dpdk-
devargs="0000:03:00.0,representor=sf65535,dv_xmeta_en=1,sys_mem_en=1" mtu_request=9000
ovs-vsctl add-port br-int vxlan0 -- set interface vxlan0 type=vxlan options:remote_ip=11.0.0.2 options:local_ip=11.0.0.1
options:key=5 mtu_request=9000

ovs-ofctl del-flows br-int
rep=hostpf
vxlan_rep=vxlan0
ovs-ofctl add-flow br-int "table=0, priority=10, arp,in_port=$rep, actions=$vxlan_rep "
ovs-ofctl add-flow br-int "table=0, priority=10, arp,in_port=$vxlan_rep, actions=$rep"
ovs-ofctl add-flow br-int "table=0, priority=10, icmp,in_port=$rep, actions=$vxlan_rep"
ovs-ofctl add-flow br-int "table=0, priority=10, icmp,in_port=$vxlan_rep, actions=$rep"
ovs-ofctl add-flow br-int "table=0, priority=50, ct_state=-trk, tcp, in_port=$rep, actions=ct(table=0)"
ovs-ofctl add-flow br-int "table=0, priority=50, ct_state=-trk, tcp, in_port=$vxlan_rep, actions=ct(table=0)"
ovs-ofctl add-flow br-int "table=0, priority=50, ct_state=+trk+new, tcp, in_port=$vxlan_rep actions=ct(commit),$rep"
ovs-ofctl add-flow br-int "table=0, priority=50, ct_state=+trk+new, tcp, in_port=$rep, actions=ct(commit),$vxlan_rep"
ovs-ofctl add-flow br-int "table=0, priority=50, ct_state=+trk+est, tcp, in_port=$vxlan_rep, actions=$rep"
ovs-ofctl add-flow br-int "table=0, priority=50, ct_state=+trk+est, tcp, in_port=$rep, actions=$vxlan_rep"
ovs-ofctl dump-flows br-int
```

Virtio live migration

- Virtio热迁移满足虚拟化场景
- Host需要运行backend service，利用vhost的机制和qemu接口
- Backend service利用静态PF的virtio 设备同DPU进行通信
- 有接口支持定制化适配客户的热迁移机制





https://docs.nvidia.com/networking/display/bluefieldvirtionetv190/live+migration

Q&A

Q&A

- 更改FW配置后，server不能启动，BIOS hang住
    - 可能原因是FW更改配置导致BIOS申请的资源不可用，BIOS会hang住，解决方法是方法进入DPU OS或者BMC，复位FW配置或者更改FW配置，掉电重启尝试
    - DPU的NIC mode如果配置virtio emulation会导致BIOS hang，这个可以通过DPU的BMC进入UEFI menu更改到DPU mode
- 在配置静态virtio-net时，OS的virtio 驱动crash，kernel hang
    - 可能原因是OS主动加载了virtio pci驱动，但是因为DPU上的virtio net没有配置完全，或者services没有启动，导致virito net的controll message没有响应，被hang住。
    - 解决方式是，DPU的virtnet先在host os启动之前配置好，或者host先禁止加载virtio pci驱动
- Virtnet 性能
    - virtio-net性能和队列数相关，单条队列的性能有上限，尽量使用多个队列多流提高性能，开启checksum，LRO、TSO 硬件卸载
    - 单virtio-net的最多队列数是31，((64-1)/2)
- Virtio host UEFI
    - Virtio(blk and net) uefi driver 包含在DPU的 exp rom里，不建议用BIOS自带的
    - Enable或者disable只能从host上设置，和DPU上设置是分开的

```
[Host]
root@amdgen5sz:~# mlxconfig -d 02:00.0 q | grep UEFI
        UEFI_HII_EN                                 True(1)
        UEFI_LOGS                                   DISABLED(0)
        EXP_ROM_VIRTIO_NET_UEFI_ARM_ENABLE          False(0)
        EXP_ROM_VIRTIO_NET_UEFI_x86_ENABLE          False(0)
        EXP_ROM_VIRTIO_BLK_UEFI_ARM_ENABLE          False(0)
        EXP_ROM_VIRTIO_BLK_UEFI_x86_ENABLE          False(0)
        EXP_ROM_NVME_UEFI_x86_ENABLE                True(1)
        EXP_ROM_UEFI_ARM_ENABLE                     True(1)
        EXP_ROM_UEFI_x86_ENABLE                     True(1)
```

NVIDIA®