



# NVIDIA DOCA Emulated Devices

## User Guide



# NVIDIA DOCA 模拟设备

## 用户指南

# Table of Contents

|   |          |
|---|----------|
| <b>Chapter 1. Mediated Devices.....</b>               | <b>1</b> |
| 1.1. Related Configuration.....                       | 1        |
| <b>Chapter 2. VirtIO-net Emulated Devices.....</b>    | <b>3</b> |
| 2.1. VirtIO-net Controller.....                       | 3        |
| 2.1.1. SystemD Service.....                           | 3        |
| 2.1.2. User Frontend.....                             | 5        |
| 2.1.3. Controller Recovery.....                       | 5        |
| 2.2. VirtIO-net PF Devices.....                       | 6        |
| 2.2.1. VirtIO-net PF Device Configuration.....        | 7        |
| 2.2.2. Creating Hotplug VirtIO-net PF Device.....     | 7        |
| 2.3. VirtIO-net SR-IOV VF Devices.....                | 8        |
| 2.3.1. Virtio-net SR-IOV VF Device Configuration..... | 9        |
| 2.3.2. Creating Virtio-net SR-IOV VF Devices.....     | 9        |

# 目录

## 第 1 章. 中介设备 1

### 1.1. 相关配置 1

## 第二章 VirtIO-net 模拟设备 3

### 2.1. VirtIO-net 控制器 3 2.1.1. systemd 服务 3

#### 2.1.2. 用户前端 5

#### 2.1.3. 控制器恢复 5

### 2.2. 虚拟化 VirtIO-net PF 设备 6 2.2.1. 虚拟化 VirtIO-net PF 设备配置 7

#### 2.2.2. 创建热插拔 VirtIO-net PF 设备 7

### 2.3. 虚拟化 VirtIO-net SR-IOV VF 设备 8 2.3.1. 虚拟化 VirtIO-net SR-IOV VF 设备配置 9

#### 2.3.2. 创建虚拟化 VirtIO-net SR-IOV VF 设备 9

---

# Chapter 1. Mediated Devices

NVIDIA mediated devices deliver flexibility in allowing to create accelerated devices without SR-IOV on the BlueField system. These mediated devices support NIC and RDMA and offer the same level of ASAP2 offloads as SR-IOV VFs. Mediated devices are supported using mlx5 sub-function acceleration technology.

Two sub-function devices are created on the BlueField device upon boot (one per port if the port is in switchdev mode) using commands from `/etc/mellanox/mlnx-sf.conf`:

```
/sbin/mlnx-sf -a create -d 0000:03:00.0 -u 61a59715-aeec-42d5-be83-f8f42ba8b049 --
mac 12:11:11:11:11:11
/sbin/mlnx-sf -a create -d 0000:03:00.1 -u 5b198182-1901-4c29-97a0-6623f3d02065 --
mac 12:11:11:11:11:12
```

The help menu for `mlnx-sf` is presented below:

```
Usage: mlnx-sf [ OPTIONS ]
OPTIONS:
-a, -action,      --action <action>          Perform action
    action:      { enable | create | configure | remove | show | set_max_mdevs |
    query_mdevs_num }

-d, -device,      --device <device>          PCI device
    <domain>:<bus>:<device>.<func> (E.g.: 0000:03:00.0)
-m, -max_mdevs,  --max_mdevs <max_mdevs number> Set maximum number of MDEVs
-u, -uuid,       --uuid <uuid>              UUID to create SF with
-M, -mac,        --mac <MAC>                MAC to create SF with
-p, -permanent,  --permanent [<conf file>]    Store configuration to be used after
    reboot and/or driver restart. Default (/etc/mellanox/mlnx-sf.conf).
-V, -version,    --version                    Display script version and exit
-D, -dryrun,     --dryrun                     Display commands only
-v, -verbose,    --verbose                    Run script in verbose mode (print
    out every step of execution)
-h, -help,       --help                       Display help
```

`"/etc/mellanox/mlnx-sf.conf"` can be updated manually or using `"mlnx-sf"` tool with `"-p"` parameter.

## 1.1. Related Configuration

Interface names are configured using the UDEV rule under `/etc/udev/rules.d/82-net-setup-link.rules`.

```
SUBSYSTEM=="net", ACTION=="add", ATTR{phys_switch_id}!="", ATTR{phys_port_name}!="", \
\
    IMPORT{program}="/etc/infiniband/vf-net-link-name.sh $attr{phys_switch_id}
    $attr{phys_port_name}" \
    NAME="$env{NAME}", RUN+="/sbin/ethtool -L $env{NAME} combined 4"
# MDEV network interfaces
```

# 第 1 章. 中介设备

NVIDIA 中介设备在 BlueField 系统上允许创建无需 SR-IOV 的加速设备，提供了灵活性。这些中介设备支持网络接口卡(NIC)和 RDMA，并提供与 SR-IOV 虚拟功能(VFs)相同的 ASAP2 卸载级别。中介设备通过 mlx5 子功能加速技术得到支持。

启动时在 BlueField 设备上创建了两个子功能设备（如果端口处于 switchdev 模式，则每个端口一个）使用/etc/mellanox/mlnx-sf.conf 中的命令：

```
/sbin/mlnx-sf -a create -d 0000:03:00.0 -u 61a59715-aeec-42d5-be83-f8f42ba8b049 -mac 12:11:11:11:11:11
/sbin/mlnx-sf -a create -d 0000:03:00.1 -u 5b198182-1901-4c29-97a0-6623f3d02065 -mac 12:11:11:11:11:12
```

以下为 mlnx-sf 的帮助菜单：

用法：mlnx-sf [选项] 选项：-a, --action, --action <操作> 执行操作

```
action: { enable | create | configure | remove | show | set_max_mdevs | query_mdevs_num }

-d, -device, --device <device> PCI 设备 <domain>:<bus>:<device>.<func> (例如：0000:03:00.0) -m, -max_mdevs, --max_mdevs <max mdevs number> 设置 MDEVs 的最大数量 -u, -uuid, --uuid <uuid> 创建 SF 的 UUID -M, -mac, --mac <MAC> 创建 SF 的 MAC -p, -permanent, --permanent [<conf file>] 将配置存储到在重启和/或驱动程序重启后使用的配置文件中。默认 (/etc/mellanox/mlnx-sf.conf)。-V, -version, --version 显示脚本版本并退出 -D, -dryrun, --dryrun 仅显示命令 -v, -verbose, --verbose 以详细模式运行脚本（打印执行的每一步）-h, -help, --help 显示帮助
"/etc/mellanox/mlnx-sf.conf" 可以手动更新或使用带有 "p" 参数的 "mlnx-sf" 工具。
```

## 1.1. 相关配置

接口名称是通过在 /etc/udev/rules.d/82-net- 目录下的 UDEV 规则进行配置的 setup-link.rules 。

```
SUBSYSTEM=="net", ACTION=="add", ATTR{phys_switch_id}!="", ATTR{phys_port_name}!="", \
IMPORT{program}="/etc/infiniband/vf-net-link-name.sh $attr{phys_switch_id} $attr{phys_port_name}" \
名称="$env{NAME}", 运行+="/sbin/ethtool -L $env{NAME} combined 4" # MDEV 网络接口
```

```
ACTION=="add", SUBSYSTEM=="net", DEVPATH=="/devices/  
pci0000:00/0000:00:00.0/0000:01:00.0/0000:02:02.0/0000:03:00.0/61a59715-aeec-42d5-  
be83-f8f42ba8b049/net/eth[0-9]", NAME="p0m0"  
ACTION=="add", SUBSYSTEM=="net", DEVPATH=="/devices/  
pci0000:00/0000:00:00.0/0000:01:00.0/0000:02:02.0/0000:03:00.1/5b198182-1901-4c29-97a0-6623f3d0206  
net/eth[0-9]", NAME="p1m0"
```

NVMe SNAP uses p0m0 as its default interface. See `/etc/nvme_snap/sf1.conf`.

```
ACTION=="add", SUBSYSTEM=="net", DEVPATH=="/devices/ pci0000:00/0000:00:00.0/0000:01:00.0/0000:02:02.0/0000:03:00.0/61a59715-aeec-42d5be83-f8f42ba8b049/net/eth[0-9]", NAME="p0m0" ACTION=="add", SUBSYSTEM=="net", DEVPATH=="/devices/ pci0000:00/0000:00:00.0/0000:01:00.0/0000:02:02.0/0000:03:00.1/5b198182-1901-4c29-97a0-6623f3d02065 net/eth[0-9]", NAME="p1m0"
```

NVMe SNAP 使用 p0m0 作为其默认接口。请参阅/etc/nvme\_snap/sf1.conf。



---

## Chapter 2. VirtIO-net Emulated Devices

VirtIO emulated devices enables users to create VirtIO-net emulated PCIe devices in the system where the NVIDIA® BlueField®-2 DPU is connected. This is done by the virtio-net-controller software module present in the DPU. Virtio-net emulated devices allow users to hot plug up to 16 virtio-net PCIe PF Ethernet NIC devices or 504 virtio-net PCI VF Ethernet NIC devices in the host system where the DPU is plugged in.



**Note:** Currently, only virtio specifications v1.0 and above are supported.

DPU software also enables users to create virtio block PCIe PF and SR-IOV PCIe VF devices. This is covered in the *NVIDIA BlueField SNAP and virtio-blk SNAP Documentation*.

### 2.1. VirtIO-net Controller

Virtio-net-controller is a systemd service running on the DPU, with a user interface frontend to communicate with the background service. An SF representor is created for each virtio-net device created on the host. Virtio-net controller only uses an SF number  $\geq 1000$ .



**Note:** It is important to note that since the controller provides hardware resources and ACKs the request from the host's VirtIO driver, in order to reboot the DPU and host OS, it is necessary to reboot the host OS first, and only then reboot the DPU.



**Note:** SF representor name is determined by udev rules. The default name is in the format of `<prefix><pf_num><sf_num>`. For example: `en3f0pf0sf1001`.

#### 2.1.1. SystemD Service

Controller systemd service is enabled by default and runs automatically if `VIRTIO_NET_EMULATION_ENABLE` is true from `mlxconfig`.

1. To check controller service status, run:  

```
systemctl status virtio-net-controller.service
```
2. To reload the service, make sure to unload virtio-net/virtio-pcie drivers on host. Then run:  

```
systemctl restart virtio-net-controller.service
```
3. To monitor log output of the controller service, run:  

```
journalctl -u virtio-net-controller
```

# 第二章. VirtIO-net 模拟设备


VirtIO 模拟设备使用户能够在连接 NVIDIA® BlueField®-2 DPU 的系统中创建 VirtIO-net 模拟 PCIe 设备。这是通过 DPU 中存在的 virtio-netcontroller 软件模块实现的。VirtIO-net 模拟设备允许用户在插入 DPU 的主机系统中热插拔最多 16 个 virtio-net PCIe PF 以太网网卡设备或 504 个 virtio-net PCI VF 以太网网卡设备。


 注意：目前仅支持 virtio 规范 v1.0 及以上版本。

DPU 软件还允许用户创建 virtio 块 PCIe PF 和 SR-IOV PCIe VF 设备。这在 NVIDIA BlueField SNAP 和 virtio-blk SNAP 文档中有详细说明。

## 2.1. VirtIO-net 控制器

Virtio-net 控制器是运行在 DPU 上的 systemd 服务，具有与后台服务通信的用户界面前端。对于主机上创建的每个 virtio-net 设备，都会创建一个 SF 表示器。Virtio-net 控制器仅使用 SF 编号 ≥ 1000。

 注意：由于控制器提供硬件资源并确认来自主机 VirtIO 驱动程序的要求，因此为了重启 DPU 和主机操作系统，必须首先重启主机操作系统，然后才能重启 DPU。

 注意：SF 代表者名称由 udev 规则确定。默认名称的格式为 <prefix> <pf\_num> <sf\_num>。例如：en3f0pf0sf1001。

### 2.1.1. 系统服务

控制器 systemd 服务默认启用，如果从 mlxconfig 中 VIRTIO\_NET\_EMULATION\_ENABLE 为 true，则自动运行。

1. 检查控制器服务状态，请运行：`systemctl status virtio-net-controller.service`
2. 要重新加载服务，请确保在主机上卸载 virtio-net/virtio-pcie 驱动程序。然后运行：`systemctl restart virtio-net-controller.service`
3. 要监控控制器服务的日志输出，请运行：`journalctl -u 虚拟 io 网络控制器`

The controller service has an optional configuration file which allows users to customize several parameters. The configuration file should be defined on the DPU at the following path `/opt/mellanox/mlnx_virtnet/virtnet.conf`.

This file will be read every time the controller starts. Dynamic change of `virtnet.conf` is not supported. It is defined as a JSON format configuration file. The currently supported options are:

- ▶ `ib_dev_p0` – RDMA device (e.g., `mlx5_0`) used to create SF on port 0. This port is the EMU manager when `is_lag` is 0. Default value is `mlx5_0`.
- ▶ `ib_dev_p1` – RDMA device (e.g., `mlx5_1`) used to create SF on port 1. Default value is `mlx5_1`.
- ▶ `ib_dev_lag` – RDMA LAG device (e.g., `mlx5_bond_0`) used to create SF on LAG. Default value is `mlx5_bond_0`. This port is EMU manager when `is_lag` is 1. `ib_dev_lag` and `ib_dev_p0/ib_dev_p1` cannot be configured simultaneously.
- ▶ `ib_dev_for_static_pf` – the RDMA device (e.g., `mlx5_0`) which the static VirtIO PF is created on
- ▶ `is_lag` – whether or not LAG is used. Note that if LAG is used, make sure to use the correct IB dev for static PF.
- ▶ `pf_mac` – base MAC address for static PFs. MACs are automatically assigned with the following pattern: `pf_mac` → `pf_0`, `pf_mac+1` → `pf_1`, etc.



**Note:** Note that the controller does not validate the MAC address (other than its length). The user must ensure MAC is valid and unique.

- ▶ `recovery` – specifies whether recovery is enabled. If unspecified, recovery is enabled by default. To disable it, set `recovery` to 0.
- ▶ `sf_pool_percent` – determines the initial SF pool size as the percentage of `PF_TOTAL_SF` of `mlxconfig`. Valid range: [0, 100]. For instance, if the value is 5, it means an SF pool with 5% of `PF_TOTAL_SF` is created. 0 means no SF pool is reserved beforehand (default).



**Note:** `PF_TOTAL_SF` is shared by all applications. User must ensure the percent request is guaranteed or else the controller will not be able to reserve the requested SFs resulting in failure.

- ▶ `sf_pool_force_destroy` – specifies whether to destroy the SF pool. When set to 1, the controller destroys the SF pool when stopped/restarted (and the SF pool is recreated if `sf_pool_percent` is not 0 when starting), otherwise it does not. Default value is 0.

For example, the definition below has all static PFs using `mlx5_0` (port 0) as the data path device in a non-lag configuration.

```
{
  "ib_dev_p0": "mlx5_0",
  "ib_dev_p1": "mlx5_1",
  "ib_dev_for_static_pf": "mlx5_0",
  "is_lag": 0,
  "pf_mac": "00:11:22:33:44:55",
  "recovery": 1,
  "sf_pool_percent": 0,
  "sf_pool_force_destroy": 0
}
```

控制器服务有一个可选的配置文件，允许用户自定义多个参数。该配置文件应在 DPU 上的以下路径定义

/opt/mellanox/mlnx\_virtnet/virtnet.conf.

此文件将在控制器启动时每次读取。不支持 virtnet.conf 的动态更改。它被定义为 JSON 格式的配置文件。当前支持选项包括：

ib\_dev\_p0 – 用于在端口 0 上创建 SF 的 RDMA 设备（例如，mlx5\_0）。当 is\_lag 为 0 时，此端口是 EMU 管理器。默认值为 mlx5\_0。

ib\_dev\_p1 – 用于在端口 1 上创建 SF 的 RDMA 设备（例如，mlx5\_1）。默认值为 mlx5\_1。

ib\_dev\_lag – 用于在 LAG 上创建 SF 的 RDMA LAG 设备（例如，mlx5\_bond\_0）。默认值为 mlx5\_bond\_0。当 is\_lag 为 1 时，此端口是 EMU 管理器。ib\_dev\_lag 和 ib\_dev\_p0/ib\_dev\_p1 不能同时配置。

ib\_dev\_for\_static\_pf – 静态 VirtIO PF 创建在其上的 RDMA 设备（例如，mlx5\_0）

是否使用 LAG。注意，如果使用 LAG，请确保使用正确的 IB 设备进行静态 PF。

静态 PF 的基准 MAC 地址。MAC 地址将自动按照以下模式分配：pf\_mac→pf\_0, pf\_mac+1→pf\_1, 等等。



注意：控制器不会验证 MAC 地址（除了长度之外）。用户必须确保 MAC 地址有效且唯一。

恢复 – 指定是否启用恢复。如果未指定，则默认启用恢复。要禁用它，请将恢复设置为 0。

sf\_pool\_percent – 确定初始 SF 池大小为 mlxconfig 中 PF\_TOTAL\_SF 的百分比。有效范围：[0, 100]。例如，如果值为 5，则表示创建一个占 PF\_TOTAL\_SF 5% 的 SF 池。

5% 的 PF\_TOTAL\_SF 被创建。0 表示事先不预留 SF 池（默认）。



注意：PF\_TOTAL\_SF 被所有应用程序共享。用户必须确保请求的百分比得到保证，否则控制器将无法保留请求的 SFs，从而导致失败。

sf\_pool\_force\_destroy – 指定是否销毁 SF 池。当设置为 1 时，控制器在停止/重启时销毁 SF 池（如果 sf\_pool\_percent 在启动时不是 0，则重新创建 SF 池），否则不销毁。默认值为 0。

例如，以下定义中所有静态 PF 使用 mlx5\_0（端口 0）作为非 LAG 配置的数据路径设备。

```
{ "ib_dev_p0": "mlx5_0", "ib_dev_p1": "mlx5_1",
  "ib_dev_for_static_pf": "mlx5_0", "is_lag": 0,
  "pf_mac": "00:11:22:33:44:55", "recovery": 1,
  "sf_pool_percent": 0, "sf_pool_force_destroy": 0
}
```



The following is an example for LAG configuration:

```
{
  "ib_dev_lag": "mlx5_bond_0",
  "ib_dev_for_static_pf": "mlx5_bond_0",
  "is_lag": 1,
  "pf_mac": "00:11:22:33:44:55",
  "recovery": 1,
  "sf_pool_percent": 0,
  "sf_pool_force_destroy": 0
}
```

## 2.1.2. User Frontend

To communicate with the service, a user frontend program, `virtnet`, is installed on the DPU.

Run the following command to check its usage:

```
# virtnet -h
usage: virtnet [-h] [-v] {hotplug,unplug,list,query,modify,log} ...

Nvidia virtio-net-controller command line interface v1.0.9

positional arguments:
  {hotplug,unplug,list,query,modify,log}
                                ** Use -h for sub-command usage
  hotplug                      hotplug virtnet device
  unplug                      unplug virtnet device
  list                        list all virtnet devices
  query                      query all or individual virtnet device(s)
  modify                      modify virtnet device
  log                         set log level

optional arguments:
  -h, --help                  show this help message and exit
  -v, --version               show program's version number and exit
```

Note that each positional argument has its own help menu as well. For example:

```
# virtnet log -h
usage: virtnet log [-h] -l {info,err,debug}
optional arguments:
  -h, --help                  show this help message and exit
  -l {info,err,debug}, --level {info,err,debug}
                              log level: info/err/debug
```

To operate a particular device, either the VUID or device index can be used to locate the device. Both attributes can be fetched from command "`virtnet list`". For example, to modify the MAC of a specific VF, you may run either of the following commands:

```
# virtnet modify -p 0 -v 0 device -m 0C:C4:7A:FF:22:98
```

Or:

```
# virtnet modify -u <VUID-string> device -m 0C:C4:7A:FF:22:98
```

## 2.1.3. Controller Recovery

It is possible to recover the control and data planes if communications are interrupted so the original traffic can resume.

Recovery depends on the JSON files stored in `/opt/mellanox/mlnx_virtnet/recovery` where there is a file that corresponds to each device (either PF or VF). The following is an example of the data stored in these files:

```
{
  "port_ib_dev": "mlx5_0",
```

以下是一个 LAG 配置的示例：

```
{ "ib_dev_lag": "mlx5_bond_0", "ib_dev_for_static_pf":  
  "mlx5_bond_0", "is_lag": 1, "pf_mac":  
  "00:11:22:33:44:55", "recovery": 1, "sf_pool_percent":  
  0, "sf_pool_force_destroy": 0 }
```

## 2.1.2. 用户前端

与服务通信时，需要在 DPU 上安装用户前端程序 `virtnet`。运行以下命令以检查其使用情况：

```
virtnet -h 使用方法: virtnet [-h] [-v] {hotplug,unplug,list,query,modify,log} ...
```

```
Nvidia virtio-net-controller 命令行界面 v1.0.9
```

```
位置参数: {hotplug,unplug,list,query,modify,log}
```

```
          ** 使用 -h 查看子命令用法  
热插拔 热插拔 virtnet 设备 断开 断开 virtnet 设备 列表 列出所有 virtnet 设备 查询  
查询所有或单个 virtnet 设备 修改 修改 virtnet 设备 日志 设置日志级别
```

```
可选参数:
```

```
-h, --help 显示此帮助信息并退出 -v, --version 显示程序版本信息并退出
```

注意，每个位置参数都有自己的帮助菜单。例如：

```
# virtnet log -h 使用方法: virtnet log [-h] -l {info,err,debug} 可选参数: -  
h, --help 显示此帮助信息并退出 -l {info,err,debug}, --level  
{info,err,debug}
```

```
日志级别: info/err/debug
```

要操作特定设备，可以使用 VUID 或设备索引来定位设备。这两个属性都可以从"virtnet list"命令中获取。例如，要修改特定 VF 的 MAC 地址，可以运行以下任一命令：

```
# virtnet modify -p 0 -v 0 device -m 0C:C4:7A:FF:22:98
```

Or:

```
# virtnet modify -u <VUID-string> device -m 0C:C4:7A:FF:22:98
```

## 2.1.3. 控制器恢复

如果通信中断，则可以恢复控制面和数据面，以便原始流量可以继续。

恢复依赖于存储在 `/opt/mellanox/mlnx_virtnet/recovery` 中的 JSON 文件，其中每个设备（无论是 PF 还是 VF）都对应一个文件。以下是在这些文件中存储的数据示例：

```
{ "port_ib_dev": "mlx5_0",
```

```

"pf_id": 0,
"function_type": "pf",
"bdf_raw": 26624,
"device_type": "hotplug",
"mac": "0c:c4:7a:ff:22:93",
"pf_num": 0,
"sf_num": 2000,
"mq": 1
}

```

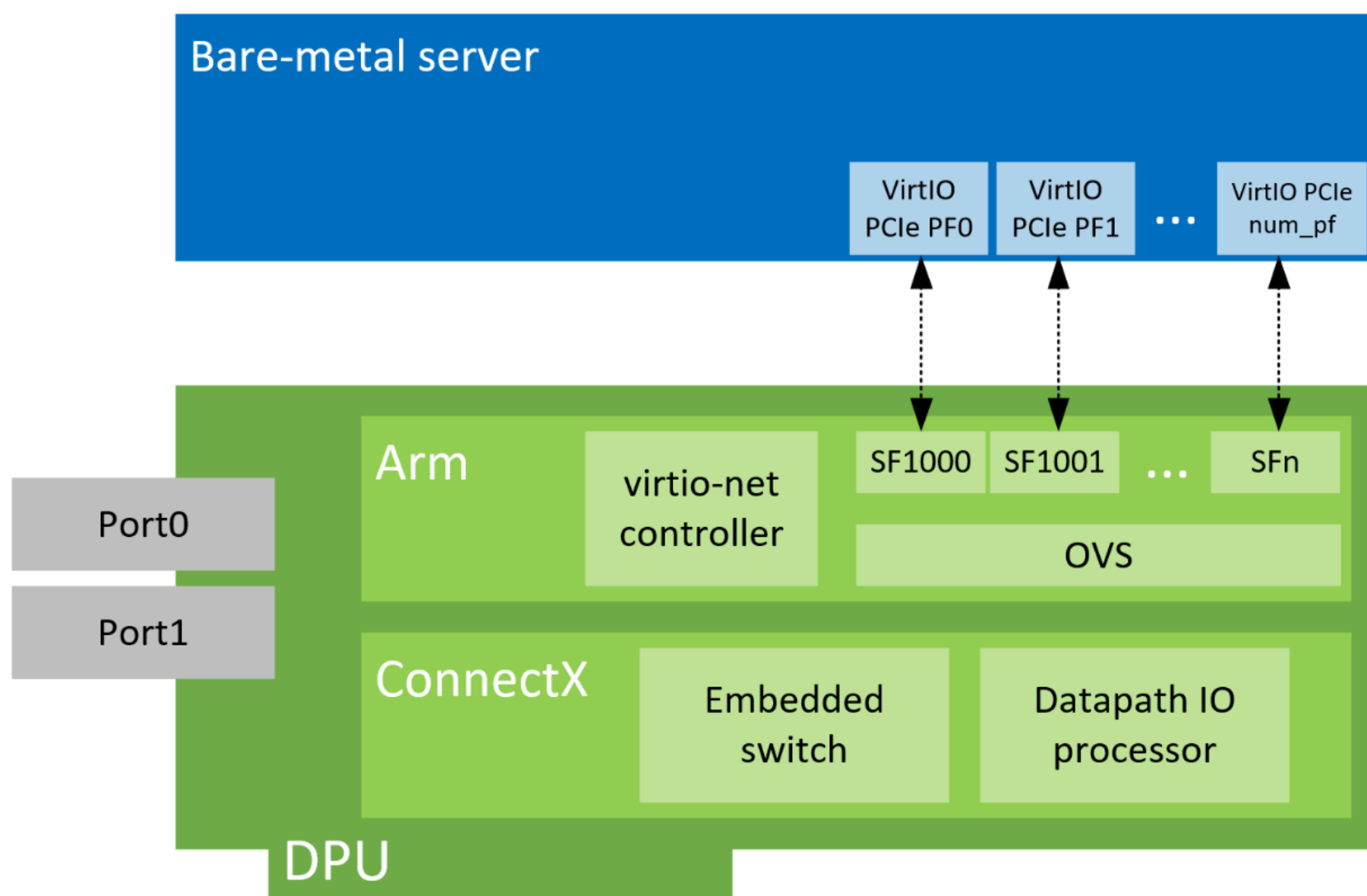
These files should not be modified under normal circumstances. However, if necessary, advanced users may tune settings to meet their requirements. Users are responsible for the validity of the recovery files and should only perform this when the controller is not running.



**Note:** Controller recovery is enabled by default and does not need user configuration or intervention unless a system reset is needed or BlueField configuration is changed (i.e. any of the mlxconfig options `PCI_SWITCH_EMULATION_NUM_PORT`, `VIRTIO_NET_EMULATION_NUM_VF`, or `VIRTIO_NET_EMULATION_NUM_PF`). To this end, the files under `/opt/mellanox/mlnx_virtnet/recovery` must be deleted.

## 2.2. VirtIO-net PF Devices

This section covers managing virtio-net PCIe PF devices using virtio-net-controller.



```
"pf_id": 0, "function_type": "pf",
"bdf_raw": 26624, "device_type":
"hotplug", "mac": "0c:c4:7a:ff:22:93",
"pf_num": 0, "sf_num": 2000, "mq": 1 }
```

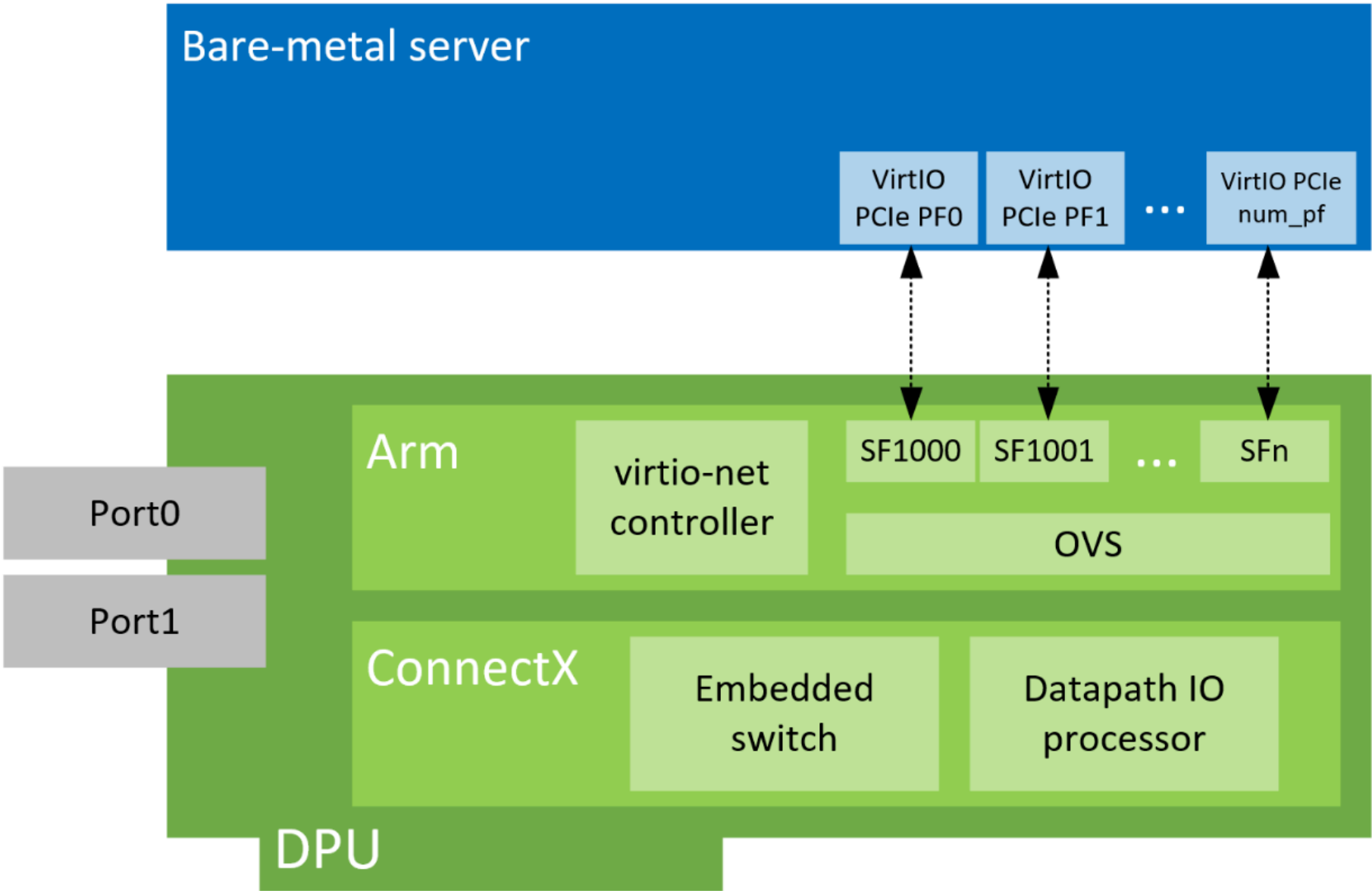
在正常情况下，这些文件不应被修改。但是，如果需要，高级用户可以调整设置以满足其需求。用户应负责恢复文件的有效性，并且只有在控制器未运行时才应执行此操作。



注意：控制器恢复默认启用，无需用户配置或干预，除非需要系统重置或更改 BlueField 配置（即 mlxconfig 选项 PCI\_SWITCH\_EMULATION\_NUM\_PORT、VIRTIO\_NET\_EMULATION\_NUM\_VF 或 VIRTIO\_NET\_EMULATION\_NUM\_PF 中的任何一项）。为此，必须删除/opt/mellanox/mlnx\_virtnet/recovery 下的文件。

## 2.2. VirtIO-net PF 设备

本节介绍使用 virtio-net 控制器管理 virtio-net PCIe PF 设备。





## 2.2.1. VirtIO-net PF Device Configuration

1. Run the following command on the DPU:

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

2. Add the following kernel boot parameters to the Linux boot arguments:

```
intel_iommu=on iommu=pt pci=realloc
```

3. Cold reboot the host system.

4. Apply the following configuration on the DPU:

```
$ mst start

$ mlxconfig -d /dev/mst/mt41686_pciconf0 s PF_BAR2_ENABLE=0 PER_PF_NUM_SF=1

$ mlxconfig -d /dev/mst/mt41686_pciconf0 s \
PCI_SWITCH_EMULATION_ENABLE=1 \
PCI_SWITCH_EMULATION_NUM_PORT=16 \
VIRTIO_NET_EMULATION_ENABLE=1 \
VIRTIO_NET_EMULATION_NUM_VF=0 \
VIRTIO_NET_EMULATION_NUM_PF=0 \
VIRTIO_NET_EMULATION_NUM_MSIX=10 \
VIRTIO_NET_EMULATION_VENDOR_ID=0x1af4 \
VIRTIO_NET_EMULATION_DEVICE_ID=0x1041 \
VIRTIO_NET_EMULATION_CLASS_CODE=0x028000 \
ECPF_ESWITCH_MANAGER=1 \
ECPF_PAGE_SUPPLIER=1 \
SRIOV_EN=0 \
PF_SF_BAR_SIZE=10 \
PF_TOTAL_SF=64

$ mlxconfig -d /dev/mst/mt41686_pciconf0.1 s \
PF_SF_BAR_SIZE=10 \
PF_TOTAL_SF=64
```

5. Cold reboot the host system a second time.

## 2.2.2. Creating Hotplug VirtIO-net PF Device

VirtIO emulated network PCIe devices are created and destroyed using virtio-net-controller application console. When this application is terminated, all created virtio-net emulated devices are hot unplugged.

1. Create a hotplug virtio-net device. Run:

```
virtnet hotplug -i mlx5_0 -f 0x0 -m 0C:C4:7A:FF:22:93 -t 1500 -n 3 -s 1024
```



**Note:** The maximum number of virtio-net queues is bound by the minimum of the following numbers:

- ▶ VIRTIO\_NET\_EMULATION\_NUM\_MSIX from the command `mlxconfig -d <mst_dev> q`
- ▶ max\_virtq from the command `virtnet list`

This creates one hotplug virtio-net device with MAC address 0C:C4:7A:FF:22:93, MTU 1500, and 3 VirtIO queues with a depth of 1024 entries. This device is uniquely identified by its index. This index is used to query and update device attributes. If the device is created successfully, an output appears similar to the following:

```
{
  "bdf": "85:00.0",
```

## 2.2.1.     VirtIO-net PF 设备配置

- 1. 在 DPU 上运行以下命令：  
mlxconfig -d /dev/mst/mt41686\_pciconf0 s INTERNAL\_CPU\_MODEL=1
- 2. 将以下内核引导参数添加到 Linux 引导参数中：  
intel\_iommu=on iommu=pt pci=realloc
- 3. 对主机系统进行冷重启。
- 4. 在 DPU 上应用以下配置：

```
$ mst start

$ mlxconfig -d /dev/mst/mt41686_pciconf0 s PF_BAR2_ENABLE=0 PER_PF_NUM_SF=1

$ mlxconfig -d /dev/mst/mt41686_pciconf0 s
PCI_SWITCH_EMULATION_ENABLE=1
PCI_SWITCH_EMULATION_NUM_PORT=16
VIRTIO_NET_EMULATION_ENABLE=1 VIRTIO_NET_EMULATION_NUM_VF=0

VIRTIO_NET_EMULATION_NUM_PF=0
VIRTIO_NET_EMULATION_NUM_MSIX=10
VIRTIO_NET_EMULATION_VENDOR_ID=0x1af4
VIRTIO_NET_EMULATION_DEVICE_ID=0x1041
VIRTIO_NET_EMULATION_CLASS_CODE=0x028000
ECPF_ESWITCH_MANAGER=1
ECPF_PAGE_SUPPLIER=1 \
SRIOV_EN=0 \ PF_SF_BAR_SIZE=10
\ PF_TOTAL_SF=64


$ mlxconfig -d /dev/mst/mt41686_pciconf0.1 s \
PF_SF_BAR_SIZE=10 \ PF_TOTAL_SF=64
```

- 5. 再次冷重启主机系统。

## 2.2.2.     创建热插拔 VirtIO-net PF 设备

VirtIO 虚拟网络 PCIe 设备通过 virtio-net-controller 应用程序控制台创建和销毁。当此应用程序终止时，所有创建的 virtio-net 虚拟设备都会进行热插拔断开。

- 1. 创建热插拔的 virtio-net 设备。运行：virtnet hotplug -i mlx5\_0 -f 0x0 -m 0C:C4:7A:FF:22:93 -t 1500 -n 3 -s 1024



注意：virtio-net 队列的最大数量受以下数字中的最小值限制：

- 从命令 `mlxconfig -d <mst_dev> q` 中获取 `VIRTIO_NET_EMULATION_NUM_MSIX`

从命令行

这将创建一个具有 MAC 地址 0C:C4:7A:FF:22:93、MTU 1500 和 3 个深度为 1024 条目的 VirtIO 队列的热插拔 virtio-net 设备。该设备通过其索引唯一标识。此索引用于查询和更新设备属性。如果设备创建成功，输出将类似于以下内容：

```
{ "bdf": "85:00.0", }
```

```

"vuid": "VNETS1D0F0",
"id": 3,
"sf_rep_net_device": "en3f0pf0sf2000",
"mac": "0C:C4:7A:FF:22:93"
}

```

2. Add the representor port of the device to the OVS bridge and bring it up. Run:

```

ip link set dev en3f0pf0sf1001 up
ovs-vsctl add-port <bridge> en3f0pf0sf1001

```

Once steps 1-3 are completed, virtio-net device should be available in the host system.

3. To query all the device configurations of virtio-net device that you created, run:

```
virtnet query -p 1
```

4. To list all the virtio-net devices, run:

```
virtnet list
```

5. To modify device attributes, for example, changing its MAC address, run:

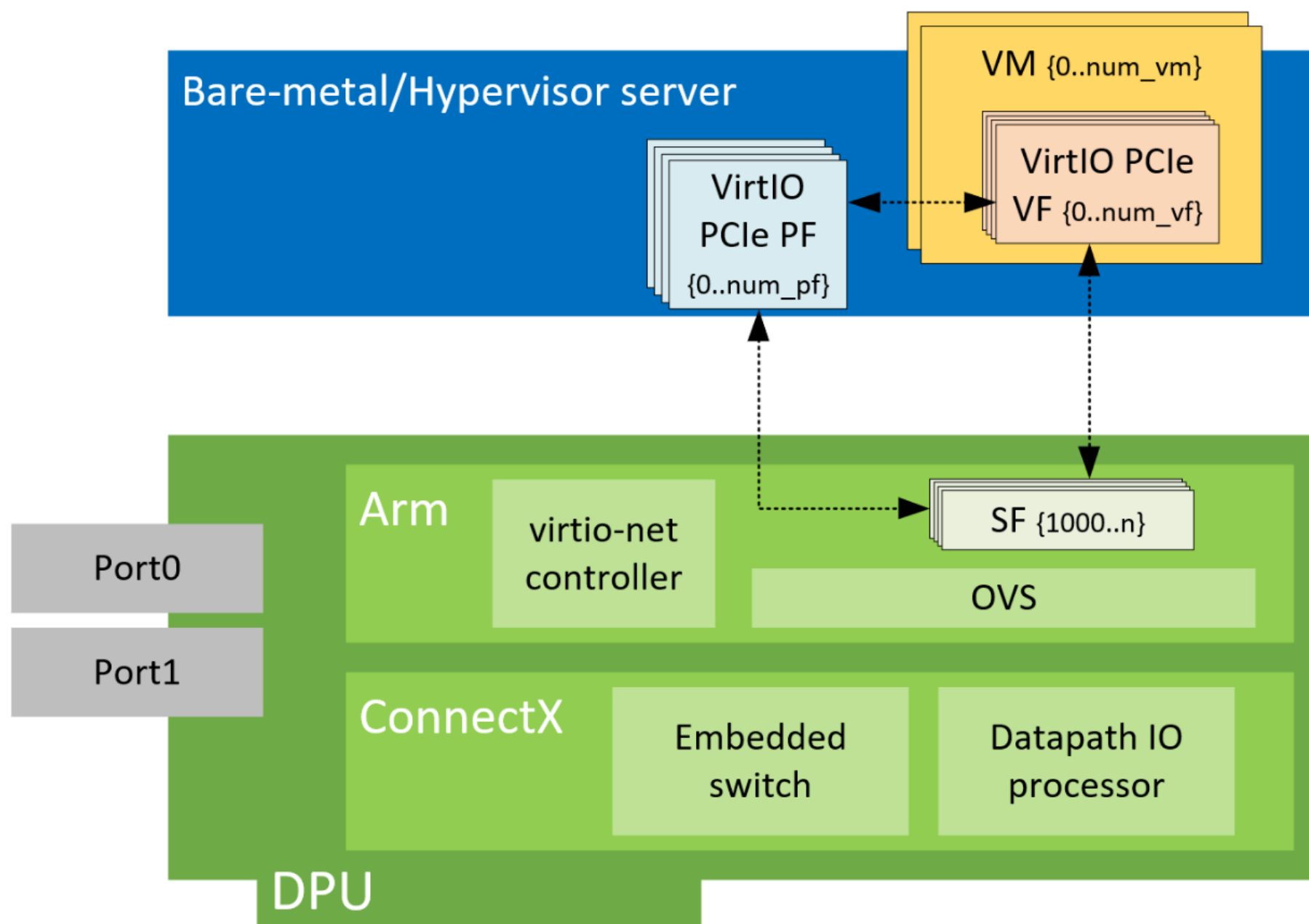
```
virtnet modify -p 0 device -m 0C:C4:7A:FF:22:98
```

6. Once usage is complete, to hot-unplug a VirtIO net device, run:

```
virtnet unplug -p 1
```

## 2.3. VirtIO-net SR-IOV VF Devices

This section covers managing virtio-net PCIe SR-IOV VF devices using virtio-net-controller.



```
{ "vuid": "VNETS1D0F0", "id": 3, "sf_rep_net_device":  
  "en3f0pf0sf2000", "mac": "0C:C4:7A:FF:22:93" }
```

2. 将设备的代表端口添加到 OVS 桥并启动。运行：

```
ip link set dev en3f0pf0sf1001 up ovs-vsctl add-port  
<bridge> en3f0pf0sf1001
```

完成步骤 1-3 后，virtio-net 设备应在主机系统中可用。

3. 要查询您创建的 virtio-net 设备的所有设备配置，请运行：

```
virtnet query -p 1
```

4. 要列出所有 virtio-net 设备，请运行：

```
virtnet 列表
```

5. 要修改设备属性，例如更改其 MAC 地址，请运行：

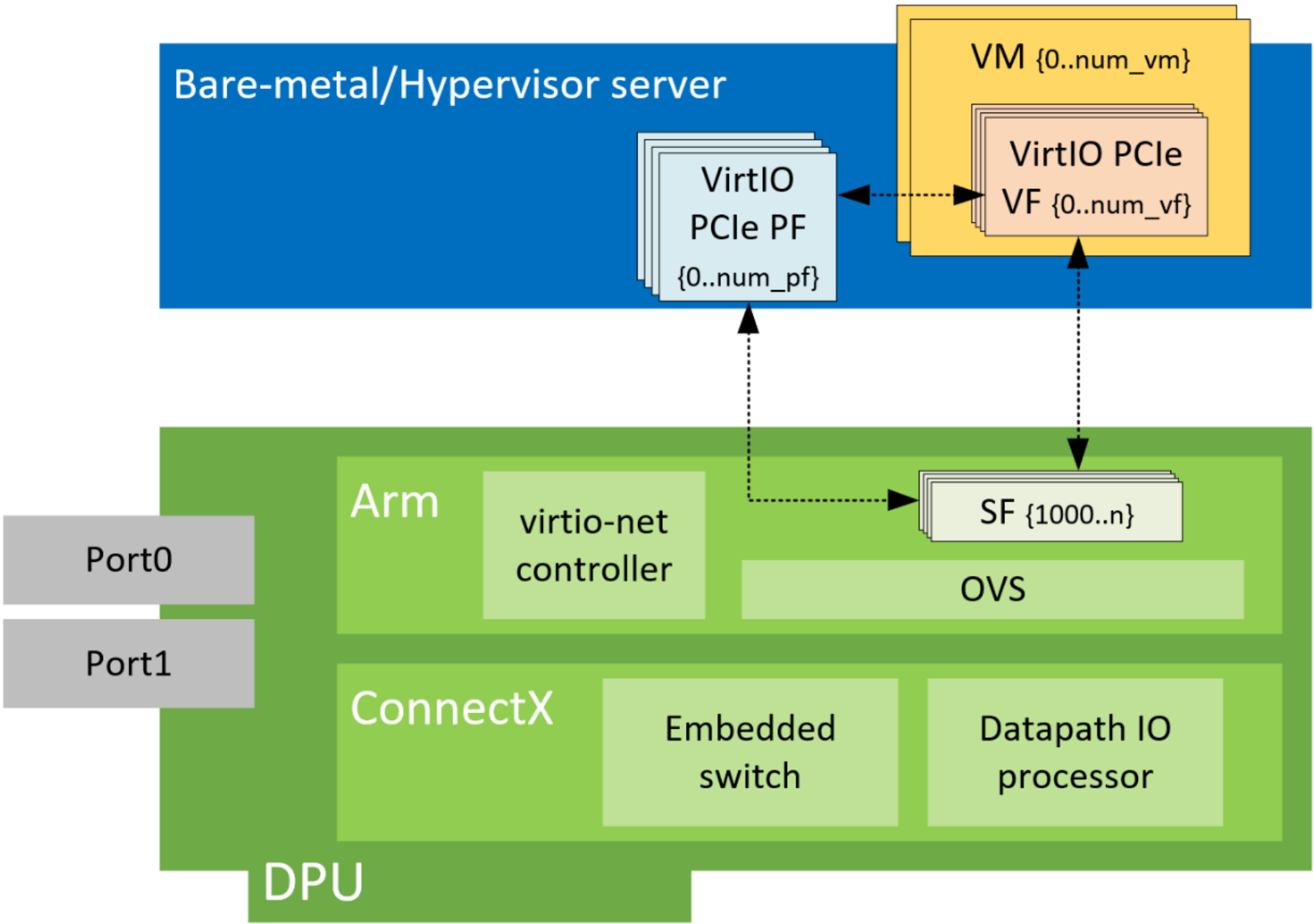
```
virtnet modify -p 0 device -m 0C:C4:7A:FF:22:98
```

6. 使用完毕后，要热插拔 VirtIO 网络设备，请运行：

```
virtnet 断开连接 -p 1
```

## 2.3. VirtIO-net SR-IOV 虚拟功能设备

本节介绍使用 virtio-net 控制器管理 virtio-net PCIe SR-IOV 虚拟功能设备。



## 2.3.1. Virtio-net SR-IOV VF Device Configuration



**Note:** Virtio-net SR-IOV VF is only supported with statically configured PF, hot-plugged PF is not currently supported.

1. On the DPU, make sure virtio-net-controller service is enabled so that it starts automatically. Run:  

```
systemctl status virtio-net-controller.service
```
2. On the x86 host, enable SR-IOV. Please refer to [MLNX\\_OFED documentation](#) under Features Overview and Configuration > Virtualization > Single Root IO Virtualization (SR-IOV) > Setting Up SR-IOV for instructions on how to do that. Make sure the parameters `intel_iommu=on iommu=pt pci=realloc` exist in `grub.conf` file.
3. It is recommended to add `pci=assign-busses` to the boot command line when creating more than 127 VFs. Without this option, the following errors might appear from host and the virtio driver will not probe these devices.  

```
pci 0000:84:00.0: [1af4:1041] type 7f class 0xffffffff
pci 0000:84:00.0: unknown header type 7f, ignoring device
```
4. Run the following command on the DPU:  

```
mst start && mlxconfig -d /dev/mst/mt41686_pciconf0 s INTERNAL_CPU_MODEL=1
```
5. Cold reboot the host system.
6. Apply the following configuration on the DPU in three steps to support up to 125 VFs per PF (500 VFs in total).

a). `mst start && mlxconfig -d /dev/mst/mt41686_pciconf0 s PF_BAR2_ENABLE=0  
PER_PF_NUM_SF=1`

b). `mlxconfig -d /dev/mst/mt41686_pciconf0 s \  
PCI_SWITCH_EMULATION_ENABLE=1 \  
PCI_SWITCH_EMULATION_NUM_PORT=16 \  
VIRTIO_NET_EMULATION_ENABLE=1 \  
VIRTIO_NET_EMULATION_NUM_VF=126 \  
VIRTIO_NET_EMULATION_NUM_PF=4 \  
VIRTIO_NET_EMULATION_NUM_MSIX=4 \  
VIRTIO_NET_EMULATION_VENDOR_ID=0x1af4 \  
VIRTIO_NET_EMULATION_DEVICE_ID=0x1041 \  
VIRTIO_NET_EMULATION_CLASS_CODE=0x028000 \  
ECPF_ESWITCH_MANAGER=1 \  
ECPF_PAGE_SUPPLIER=1 \  
SRIOV_EN=1 \  
PF_SF_BAR_SIZE=8 \  
PF_TOTAL_SF=508 \  
NUM_OF_VFS=0`

c). `mlxconfig -d /dev/mst/mt41686_pciconf0.1 s PF_TOTAL_SF=1 PF_SF_BAR_SIZE=8`

7. Cold reboot the host system.

## 2.3.2. Creating Virtio-net SR-IOV VF Devices

The virtio-net-controller application console must be kept alive to maintain the functionality of the static PF and its VFs.

1. On the host, make sure the static virtio network device presents. Run:

```
# lspci | grep -i virtio
85:00.3 Network controller: Red Hat, Inc. Virtio network device
```



## 2.3.1. Virtio-net SR-IOV VF 设备配置



注意：Virtio-net SR-IOV VF 仅支持静态配置的 PF，目前不支持热插拔 PF。

1. 在 DPU 上，请确保 virtio-net-controller 服务已启用，以便自动启动。运行：

```
systemctl status virtio-net-controller.service
```

2. 在 x86 主机上启用 SR-IOV。请参考 MLNX\_OFED 文档中的功能概述和配置 > 虚拟化 > 单根 IO 虚拟化 (SRIOV) > 设置 SR-IOV 的说明，了解如何进行操作。确保参数

intel\_iommu=on iommu=pt pci=realloc 存在于 grub.conf 文件中。

3. 建议在创建超过 127 个 VF 时，将 pci=assign-busses 添加到引导命令行中。没有此选项，可能会从主机和 virtio 驱动程序中出现以下错误。

```
pci 0000:84:00.0: [1af4:1041] 类型 7f 类 0xffffffff pci 0000:84:00.0: 未知头部类型 7f, 忽略设备
```

4. 在 DPU 上运行以下命令：

```
mst start && mlxconfig -d /dev/mst/mt41686_pciconf0 s INTERNAL_CPU_MODEL=1
```

5. 对主机系统进行冷重启。

6. 在 DPU 上分三步应用以下配置，以支持每个 PF 最多 125 个 VF (总计 500 个 VF) 。

a). 

```
/dev/mst/mt41686_pciconf0 s PF_BAR2_ENABLE=0
```

b). 

```
/dev/mst/mt41686_pciconf0 s \
PCI_SWITCH_EMULATION_ENABLE=1 \
PCI_SWITCH_EMULATION_NUM_PORT=16 \
VIRTIO_NET_EMULATION_ENABLE=1 \
VIRTIO_NET_EMULATION_NUM_VF=126 \
VIRTIO_NET_EMULATION_NUM_PF=4 \
VIRTIO_NET_EMULATION_NUM_MSIX=4 \
VIRTIO_NET_EMULATION_VENDOR_ID=0x1af4 \
VIRTIO_NET_EMULATION_DEVICE_ID=0x1041 \
VIRTIO_NET_EMULATION_CLASS_CODE=0x028000 \
ECPF_PAGE_SIZE=1 \
SRIOV_EN=1 \ PF_SF_BAR_SIZE=8 \
PF_TOTAL_SF=508 \ NUM_OF_VFS=0
```

c). 

```
/dev/mst/mt41686_pciconf0.1 s PF_TOTAL_SF=1 PF_SF_BAR_SIZE=8
```

7. 冷重启主机系统。

## 2.3.2. 创建 Virtio-net SR-IOV 虚拟功能设备。

虚拟 io 网络控制器应用程序控制台必须保持活跃，以维持静态 PF 及其 VF 的功能。

1. 在主机上，确保静态虚拟 io 网络设备存在。运行：# lspci | grep -i virtio

```
85:00.3 网络控制器: 红帽公司 Virtio 网络设备
```

2. On the host, make sure virtio\_pci and virtio\_net are loaded. Run:

```
# lsmod | grep virtio
```

The net device should be created:

```
# ethtool -i p7p3
driver: virtio_net
version: 1.0.0
firmware-version:
expansion-rom-version:
bus-info: 0000:85:00.3
supports-statistics: no
supports-test: no
supports-eprom-access: no
supports-register-dump: no
supports-priv-flags: no
```

3. To create SR-IOV VF devices on the x86 host, run:

```
# echo 2 > /sys/bus/pci/drivers/virtio-pci/0000\:85\:00.3/sriov_numvfs
```



**Note:** When the number of VFs created is high, SR-IOV enablement may take several minutes.

2 VFs should be created from the x86 host:

```
# lspci | grep -i virt
85:00.3 Network controller: Red Hat, Inc. Virtio network device
85:04.5 Network controller: Red Hat, Inc. Virtio network device
85:04.6 Network controller: Red Hat, Inc. Virtio network device
```

4. From the DPU virtio-net controller, run the following command to get VF information.

```
# virnet list
{
  "vf_id": 0,
  "parent_pf_id": 0,
  "function_type": "VF",
  "vuid": "VNETS0D0F2VF1",
  "bdf": "83:00.6",
  "sf_num": 3000,
  "sf_parent_device": "mlx5_0",
  "sf_rep_net_device": "en3f0pf0sf3000",
  "sf_rep_net_ifindex": 19,
  "sf_rdma_device": "mlx5_7",
  "sf_vhca_id": "0x192",
  "msix_config_vector": "0x0",
  "num_msix": 10,
  "max_queues": 4,
  "max_queues_size": 256,
  "net_mac": "5A:94:07:04:F6:1C",
  "net_mtu": 1500
},
```

You may use the pci-bdf to match the PF/VF on the x86 host to the information showing on DPU.

To query all the device configurations of the virtio-net device of that VF, run:

```
$ virtnet query -p 0 -v 0
```

Add the corresponding SF representor to the OVS bridge and bring it up. Run:

```
# ovs-vsctl add-port <bridge> en3f0pf0sf1004
# ip link set dev en3f0pf0sf1004 up
```

2. 在主机上，请确保已加载 virtio\_pci 和 virtio\_net。运行：

```
# lsmod | grep virtio
```


网络设备应该被创建：

```
# ethtool -i p7p3
驱动程序: virtio_net
版本: 1.0.0
固件版本: 扩展固件版本: 总线信息: 0000:85:00.3 支持统计: 否 支持测试: 否

不支持 EEPROM 访问: 否 不支持寄存器转储: 否 不支持私有标志: 否
```

3. 在 x86 主机上创建 SR-IOV 虚拟功能 (VF) 设备，请运行：

```
# echo 2 > /sys/bus/pci/drivers/virtio-pci/0000:85:00.3/sriov_numvfs
```



注意：当创建的 VF 数量较多时，SR-IOV 启用可能需要几分钟。

应从 x86 主机创建 2 个虚拟函数：

```
# lspci | grep -i virt 85:00.3 网络控制器: 红帽公司 Virtio 网络设备 85:04.5 网络控制器: 红帽公司 Virtio 网络设备 85:04.6 网络控制器: 红帽公司 Virtio 网络设备
```

4. 从 DPU 的 virtio-net 控制器运行以下命令以获取虚拟函数信息。

```
# virnet list
{
  "vf_id": 0,
  "parent_pf_id": 0, "function_type": "VF", "vuid":
  "VNETS0D0F2VF1", "bdf": "83:00.6", "sf_num": 3000,
  "sf_parent_device": "mlx5_0", "sf_rep_net_device":
  "en3f0pf0sf3000", "sf_rep_net_ifindex": 19,
  "sf_rdma_device": "mlx5_7", "sf_vhca_id": "0x192",
  "msix_config_vector": "0x0", "num_msix": 10,
  "max_queues": 4, "max_queues_size": 256, "net_mac":
  "5A:94:07:04:F6:1C", "net_mtu": 1500 }
```

您可以使用 pci-bdf 来匹配 x86 主机上的 PF/VF 与 DPU 上显示的信息。

要查询该 VF 的 virtio-net 设备的所有设备配置，请运行：

```
$ virtnet 查询 -p 0 -v 0
```

将相应的 SF 表示器添加到 OVS 桥并启动。运行：

```
# ovs-vsctl add-port <桥> en3f0pf0sf1004 # ip link set dev en3f0pf0sf1004 up
```



Now the VF should be functional.



**Note:** When port MTU (p0/p1 of the DPU) is changed after the controller is started, you must restart controller service. It is not recommended to use jumbo MTUs because that may lead to performance degradation.

5. To destroy SR-IOV VF devices on the host, run:

```
# echo 0 > /sys/bus/pci/drivers/virtio-pci/0000\:85\:00.3/sriov_numvfs
```



**Note:** When the command returns from the host OS, it does not necessarily mean the controller finished its operations. Look at controller log from the DPU and make sure you see a log like below before removing VirtIO kernel modules or recreate VFs.

```
# virtio-net-controller[3544]: [INFO]
  virtnet.c:617:virtnet_device_vfs_unload: PF(0): Unload (4) VFs finished
```

Once VFs are destroyed, created SFs from the DPU side are not destroyed but are saved into the SF pool to be reused later.

现在 VF 应该已经可用。



注意：在控制器启动后更改端口 MTU（DPU 的 p0/p1 端口）后，必须重新启动控制器服务。不建议使用巨帧 MTU，因为这可能会导致性能下降。

#### 5. 要销毁主机上的 SR-IOV VF 设备，请运行：

```
# echo 0 > /sys/bus/pci/drivers/virtio-pci/0000:85:00.3/sriov_numvfs
```



注意：当命令从主机操作系统返回时，这并不一定意味着控制器已经完成了其操作。请查看 DPU 的控制台日志，确保您看到以下日志内容后再移除 VirtIO 内核模块或重新创建 VF。

```
# virtio-net-controller[3544]: [INFO] virtnet.c:617:virtnet_device_vfs_unload: PF(0): 卸载 (4) VF 完成
```

一旦 VF 被销毁，从 DPU 侧创建的 SF 不会销毁，而是保存到 SF 池中以供以后重用。

## Notice

This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. NVIDIA Corporation nor any of its direct or indirect subsidiaries and affiliates (collectively: "NVIDIA") make no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assume no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice.

Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer ("Terms of Sale"). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document.

NVIDIA products are not designed, authorized, or warranted to be suitable for use in medical, military, aircraft, space, or life support equipment, nor in applications where failure or malfunction of the NVIDIA product can reasonably be expected to result in personal injury, death, or property or environmental damage. NVIDIA accepts no liability for inclusion and/or use of NVIDIA products in such equipment or applications and therefore such inclusion and/or use is at customer's own risk.

NVIDIA makes no representation or warranty that products based on this document will be suitable for any specified use. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer's sole responsibility to evaluate and determine the applicability of any information contained in this document, ensure the product is suitable and fit for the application planned by customer, and perform the necessary testing for the application in order to avoid a default of the application or the product. Weaknesses in customer's product designs may affect the quality and reliability of the NVIDIA product and may result in additional or different conditions and/or requirements beyond those contained in this document. NVIDIA accepts no liability related to any default, damage, costs, or problem which may be based on or attributable to: (i) the use of the NVIDIA product in any manner that is contrary to this document or (ii) customer product designs.

No license, either expressed or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this document. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA.

Reproduction of information in this document is permissible only if approved in advance by NVIDIA in writing, reproduced without alteration and in full compliance with all applicable export laws and regulations, and accompanied by all associated conditions, limitations, and notices.

THIS DOCUMENT AND ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL NVIDIA BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA's aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms of Sale for the product.

## Trademarks

NVIDIA, the NVIDIA logo, and Mellanox are trademarks and/or registered trademarks of Mellanox Technologies Ltd. and/or NVIDIA Corporation in the U.S. and in other countries. The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a world-wide basis. Other company and product names may be trademarks of the respective companies with which they are associated.

## Copyright

© 2022 NVIDIA Corporation & affiliates. All rights reserved.



注意

本文件仅提供信息目的，不得视为对产品特定功能、状态或质量的保证。英伟达公司及其所有直接或间接的子公司和关联公司（统称为“英伟达”）不对本文件中包含信息的准确性或完整性做出任何陈述或保证，并对本文件中可能存在的任何错误不承担任何责任。英伟达不对使用此类信息或由此使用可能导致的任何侵犯第三方专利或其他权利的行为承担任何责任。本文件不是开发、发布或交付任何材料（以下定义）、代码或功能的承诺。

英伟达保留在任何时候不经通知对本文档进行更正、修改、增强、改进和任何其他更改的权利。

客户在下单前应获取最新相关信息，并应核实此类信息是否最新且完整。

NVIDIA 产品销售均适用订单确认时提供的 NVIDIA 标准销售条款和条件，除非经 NVIDIA 授权代表和客户签署的单独销售协议另有约定（“销售条款”）。NVIDIA 明确反对在购买本文件中引用的 NVIDIA 产品时适用任何客户的通用条款和条件。本文件不会直接或间接产生任何合同义务。

NVIDIA 产品未设计、授权或保证适用于医疗、军事、航空、太空或生命维持设备，也不适用于 NVIDIA 产品故障或故障可能导致人身伤害、死亡、财产或环境损害的应用。NVIDIA 不对在上述设备或应用中使用 NVIDIA 产品承担任何责任，因此此类使用由客户自行承担风险。

NVIDIA 不对基于本文件的产品的适用性作出任何声明或保证。NVIDIA 不一定会对每个产品的所有参数进行测试。评估和确定本文件中包含的任何信息的适用性、确保产品适合客户计划的应用，并执行必要的应用测试以避免应用或产品的故障，是客户唯一的责任。客户产品设计中存在的缺陷可能会影响 NVIDIA 产品的质量和可靠性，并可能导致超出本文件所包含的额外或不同条件和要求。NVIDIA 不对以下任何违约、损害、费用或问题承担责任：（i）以违反本文件的方式使用 NVIDIA 产品；（ii）客户产品设计。

本文档下不授予任何形式的 NVIDIA 专利权、版权或其他 NVIDIA 知识产权许可。NVIDIA 关于第三方产品或服务的发布信息不构成 NVIDIA 授予使用此类产品或服务的许可、保证或认可。使用此类信息可能需要第三方根据第三方的专利或其他知识产权获得许可，或根据 NVIDIA 的专利或其他知识产权获得 NVIDIA 的许可。

本文档中的信息仅可在事先获得 NVIDIA 书面批准的情况下进行复制，且不得更改，必须完全遵守所有适用的出口法律和法规，并附上所有相关条件、限制和通知。

本文档以及所有 NVIDIA 设计规范、参考板、文件、图纸、诊断、清单和其他文档（统称为“材料”）均按“原样”提供。NVIDIA 不对材料作出任何明示、暗示、法定或其他保证，并明确排除所有侵权、适销性和特定用途适用性的默示保证。除非法律禁止，否则在任何情况下，NVIDIA 不对因使用本文档而产生的任何损害（包括但不限于任何直接、间接、特殊、偶然、惩罚性或后果性损害，无论损害产生的原因和责任理论如何），即使 NVIDIA 已被告知可能发生此类损害，也不承担任何责任。尽管客户可能因任何原因遭受损害，但 NVIDIA 对本文档中描述的产品对客户的累积和总责任应按照产品销售条款限制。

商标

NVIDIA、NVIDIA 标志和 Mellanox 是 Mellanox Technologies Ltd.和/或 NVIDIA Corporation 在美国及其他国家的商标和/或注册商标。注册商标 Linux 是根据 Linux Foundation 的许可使用，Linux Foundation 是 Linus Torvalds 的独家许可人，Linus Torvalds 是全球范围内该标志的所有者。其他公司和产品名称可能是与之相关的公司的商标。

版权

© 2022 NVIDIA Corporation 及子公司。保留所有权利。

