Host-side Interface Configuration

Table of contents

Virtual Ethernet Interface

RShim Support for Multiple DPUs

- Multi-board Management Example
- Configuring Management Interface on Host
- Configuring BlueField DPU Side

Permanently Changing Arm-side MAC Address

OOB Ethernet Interface

- OOB Interface MAC Address
- Supported ethtool Options for OOB Interface
- IP Address Configuration for OOB Interface

The NVIDIA® BlueField® DPU registers on the host OS a "DMA controller" for DPU management over PCIe. This can be verified by running the following:

```
#  lspci -d 15b3: | grep 'SoC Management Interface'
27:00.2 DMA controller: Mellanox Technologies MT42822 BlueField-2 SoC Management Interface (rev
01)
```

A special driver called RShim must be installed and run to expose the various BlueField management interfaces on the host OS. Refer to section "Install RShim on Host" for information on how to obtain and install the host-side RShim driver.

When the RShim driver runs properly on the host side, a sysfs device, `/dev/rshim0/*`, and a virtual Ethernet interface, `tmfifo_net0`, become available. The following is an example for querying the status of the RShim driver on the host side:

```
# systemctl status rshim
   rshim.service - rshim driver for BlueField SoC
     Loaded: loaded (/lib/systemd/system/rshim.service; disabled; vendor preset: enabled)
     Active: active (running) since Tue 2022-05-31 14:57:07 IDT; 1 day 1h ago
       Docs: man:rshim(8)
    Process: 90322 ExecStart=/usr/sbin/rshim $OPTIONS (code=exited, status=0/SUCCESS)
   Main PID: 90323 (rshim)
      Tasks: 11 (limit: 76853)
     Memory: 3.3M
     CGroup: /system.slice/rshim.service
             90323 /usr/sbin/rshim
May 31 14:57:07 ...  systemd[1]: Starting rshim driver for BlueField SoC...
May 31 14:57:07  ... systemd[1]: Started rshim driver for BlueField SoC.
May 31 14:57:07  ... rshim[90323]: Probing pcie-0000:a3:00.2(vfio)
May 31 14:57:07  ... rshim[90323]: Create rshim pcie-0000:a3:00.2
May 31 14:57:07  ... rshim[90323]: rshim pcie-0000:a3:00.2 enable
May 31 14:57:08  ... rshim[90323]: rshim0 attached
```

If the RShim device does not appear, refer to section "RShim Troubleshooting and How-Tos".

Virtual Ethernet Interface

On the host, the RShim driver exposes a virtual Ethernet device called tmfifo_net0. This virtual Ethernet can be thought of as a peer-to-peer tunnel connection between the host and the DPU OS. The DPU OS also configures a similar device. The DPU OS's BFB images are customized to configure the DPU side of this connection with a preset IP of 192.168.100.2/30. It is up to the user to configure the host side of this connection. Configuration procedures vary for different OSs.

The following example configures the host side of tmfifo_net0 with a static IP and enables IPv4-based communication to the DPU OS:

```
#  ip addr add dev tmfifo_net0 192.168.100.1/30
```

Note

For instructions on persistent IP configuration of the tmfifo_net0 interface, refer to step "Assign a static IP to tmfifo_net0" under "Updating Repo Package on Host Side".

Logging in from the host to the DPU OS is now possible over the virtual Ethernet. For example:

```
ssh ubuntu@192.168.100.2
```

RShim Support for Multiple DPUs

Multiple DPUs may connect to the same host machine. When the RShim driver is loaded and operating correctly, each board is expected to have its own device directory on sysfs, /dev/rshim<N>, and a virtual Ethernet device, tmfifo_net<N>.

The following are some guidelines on how to set up the RShim virtual Ethernet interfaces properly if multiple DPUs are installed in the host system.

There are two methods to manage multiple tmfifo_net interfaces on a Linux platform:

- Using a bridge, with all tmfifo_net<N> interfaces on the bridge – the bridge device bares a single IP address on the host while each DPU has unique IP in the same subnet as the bridge
- Directly over the individual tmfifo_net<N> – each interface has a unique subnet IP and each DPU has a corresponding IP per subnet

Whichever method is selected, the host-side tmfifo_net interfaces should have different MAC addresses, which can be:

- Configured using ifconfig. For example:

  ```
  $ ifconfig tmfifo_net0 192.168.100.1/24 hw ether 02:02:02:02:02:02
  ```

- Or saved in configuration via the /udev/rules as can be seen later in this section.

In addition, each Arm-side tmfifo_net interface must have a unique MAC and IP address configuration, as BlueField OS comes uniformly pre-configured with a generic MAC, and 192.168.100.2. The latter must be configured in each DPU manually or by DPU customization scripts during BlueField OS installation.

Multi-board Management Example

This example deals with two BlueField DPUs installed on the same server (the process is similar for more DPUs).

This example assumes that the RShim package has been installed on the host server.

Configuring Management Interface on Host

Note

This example is relevant for CentOS/RHEL operating systems only.

1. Create a bf_tmfifo interface under /etc/sysconfig/network-scripts. Run:

```
vim /etc/sysconfig/network-scripts/ifcfg-br_tmfifo
```

2. Inside `ifcfg-br_tmfifo`, insert the following content:

```
DEVICE="br_tmfifo"
BOOTPROTO="static"
IPADDR="192.168.100.1"
NETMASK="255.255.255.0"
ONBOOT="yes"
TYPE="Bridge"
```

3. Create a configuration file for the first BlueField DPU, `tmfifo_net0`. Run:

```
vim /etc/sysconfig/network-scripts/ifcfg-tmfifo_net0
```

4. Inside `ifcfg-tmfifo_net0`, insert the following content:

```
DEVICE=tmfifo_net0
BOOTPROTO=none
ONBOOT=yes
NM_CONTROLLED=no
BRIDGE=br_tmfifo
```

5. Create a configuration file for the second BlueField DPU, `tmfifo_net1`. Run:

```
DEVICE=tmfifo_net1
BOOTPROTO=none
ONBOOT=yes
NM_CONTROLLED=no
BRIDGE=br_tmfifo
```

6. Create the rules for the `tmfifo_net` interfaces. Run:

```
vim /etc/udev/rules.d/91-tmfifo_net.rules
```

7. Restart the network for the changes to take effect. Run:

```
# /etc/init.d/network restart
Restarting network (via systemctl):            [  OK  ]
```

Configuring BlueField DPU Side

BlueField DPUs arrive with the following factory default configurations for tmfifo_net0.

Address	Value
MAC	00:1a:ca:ff:ff:01
IP	192.168.100.2

Therefore, if you are working with more than one DPU, you must change the default MAC and IP addresses.

Updating RShim Network MAC Address

Note

This procedure is relevant for Ubuntu/Debian (`sudo` needed), and CentOS BFBs. The procedure only affects the `tmfifo_net0` on the Arm side.

1. Use a Linux console application (e.g. screen or minicom) to log into each BlueField. For example:

```
# sudo screen /dev/rshim<0|1>/console 115200
```

2. Create a configuration file for `tmfifo_net0` MAC address. Run:

```
# sudo vi /etc/bf.cfg
```

3. Inside `bf.cfg`, insert the new MAC:

```
NET_RSHIM_MAC=00:1a:ca:ff:ff:03
```

4. Apply the new MAC address. Run:

```
sudo bfcfg
```

5. Repeat this procedure for the second BlueField DPU (using a different MAC address).

Info

Arm must be rebooted for this configuration to take effect. It is recommended to update the IP address before you do that to avoid unnecessary reboots.

Note

For comprehensive list of the supported parameters to customize bf.cfg during BFB installation, refer to section "bf.cfg Parameters".

Updating IP Address

For Ubuntu:

1. Access the file 50-cloud-init.yaml and modify the tmfifo_net0 IP address:

```
sudo vim /etc/netplan/50-cloud-init.yaml

          tmfifo_net0:
             addresses:
             - 192.168.100.2/30    ===>>>    192.168.100.3/30
```

2. Reboot the Arm. Run:

```
sudo reboot
```

3. Repeat this procedure for the second BlueField DPU (using a different IP address).

Info

Arm must be rebooted for this configuration to take effect. It is recommended to update the MAC address before you do that to avoid unnecessary reboots.

For CentOS:

1. Access the file ifcfg-tmfifo_net0. Run:

```
# vim /etc/sysconfig/network-scripts/ifcfg-tmfifo_net0
```

2. Modify the value for IPADDR:

```
IPADDR=192.168.100.3
```

3. Reboot the Arm. Run:

```
reboot
```

   Or perform netplan apply.

4. Repeat this procedure for the second BlueField DPU (using a different IP address).

Info

Arm must be rebooted for this configuration to take effect. It is recommended to update the MAC address before you do that to avoid unnecessary reboots.

Permanently Changing Arm-side MAC Address

Note

It is assumed that the commands in this section are executed with root (or `sudo`) permission.

The default MAC address is `00:1a:ca:ff:ff:01`. It can be changed using `ifconfig` or by updating the UEFI variable as follows:

1. Log into Linux from the Arm console.
2. Run:

```
$ "ls /sys/firmware/efi/efivars".
```

3. If not mounted, run:

```
$ mount -t efivarfs none /sys/firmware/efi/efivars
$ chattr -i /sys/firmware/efi/efivars/RshimMacAddr-8be4df61-93ca-11d2-aa0d-00e098032b8c
$ printf "\x07\x00\x00\x00\x00\x1a\xca\xff\xff\x03" > \
  /sys/firmware/efi/efivars/RshimMacAddr-8be4df61-93ca-11d2-aa0d-00e098032b8c
```

The `printf` command sets the MAC address to `00:1a:ca:ff:ff:03` (the last six bytes of the `printf` value). Either reboot the device or reload the tmfifo driver for the change to take effect.

The MAC address can also be updated from the server host side while the Arm-side Linux is running:

1. Enable the configuration. Run:

```
# echo "DISPLAY_LEVEL 1" > /dev/rshim0/misc
```

2. Display the current setting. Run:

```
# cat /dev/rshim0/misc
```

```
DISPLAY_LEVEL   1 (0:basic, 1:advanced, 2:log)
BOOT_MODE       1 (0:rshim, 1:emmc, 2:emmc-boot-swap)
BOOT_TIMEOUT    300 (seconds)
DROP_MODE       0 (0:normal, 1:drop)
SW_RESET        0 (1: reset)
DEV_NAME        pcie-0000:04:00.2
DEV_INFO        BlueField-2(Rev 1)
PEER_MAC        00:1a:ca:ff:ff:01 (rw)
PXE_ID          0x00000000 (rw)
VLAN_ID         0 0 (rw)
```

3. Modify the MAC address. Run:

```
$ echo "PEER_MAC  xx:xx:xx:xx:xx:xx" > /dev/rshim0/misc
```

For more information and an example of the script that covers multiple DPU installation and configuration, refer to section "Installing Full DOCA Image on Multiple DPUs" of the *NVIDIA DOCA Installation Guide*.

OOB Ethernet Interface

The OOB interface is a gigabit Ethernet interface which provides TCP/IP network connectivity to the Arm cores. This interface is named `oob_net0` and is intended to be used for management traffic (e.g. file transfer protocols, SSH, etc). The Linux driver that controls this interface is named `mlxbf_gige.ko`, and is automatically loaded upon boot. This interface can be configured and monitored by use of standard tools (e.g. ifconfig, ethtool, etc). The OOB interface is subject to the following design limitations:

- Only supports 1Gb/s full-duplex setting
- Only supports GMII access to external PHY device
- Supports maximum packet size of 2KB (i.e. no support for jumbo frames)

The OOB interface can also be used for PXE boot. This OOB port is not a path for the boot stream. Any attempt to push a BFB to this port will not work. Please refer to How to use the UEFI boot menu for more information about UEFI operations related to the OOB interface.

OOB Interface MAC Address

The MAC address to be used for the OOB port is burned into Arm-accessible UPVS EEPROM during the manufacturing process. This EEPROM device is different from the SPI Flash storage device used for the NIC firmware and associated NIC MACs/GUIDs. The value of the OOB MAC address is specific to each platform and is visible on the board-level sticker.

Warning

It is not recommended to reconfigure the MAC address from the MAC configured during manufacturing.

If there is a need to re-configure this MAC for any reason, follow these steps to configure a UEFI variable to hold new value for OOB MAC.:

Note

The creation of an OOB MAC address UEFI variable will override the OOB MAC address defined in EEPROM, but the change can be reverted.

1. Log into Linux from the Arm console.
2. Issue the command `ls /sys/firmware/efi/efivars` to show whether efivarfs is mounted. If it is not mounted, run:

   ```
   mount -t efivarfs none /sys/firmware/efi/efivars
   ```

3. Run:

```
chattr -i /sys/firmware/efi/efivars/OobMacAddr-8be4df61-93ca-11d2-aa0d-00e098032b8c
```

4. Set the MAC address to 00:1a:ca:ff:ff:03 (the last six bytes of the printf value).

```
printf "\x07\x00\x00\x00\x00\x1a\xca\xff\xff\x03" > /sys/firmware/efi/efivars/OobMacAddr-
8be4df61-93ca-11d2-aa0d-00e098032b8c
```

5. Reboot the device for the change to take effect.

To revert this change and go back to using the MAC as programmed during manufacturing, follow these steps:

1. Log into UEFI from the Arm console, go to "Boot Manager" then "EFI Internal Shell".
2. Delete the OOB MAC UEFI variable. Run:

```
dmpstore -d OobMacAddr
```

3. Reboot the device by running "reset" from UEFI.
4. Log into Linux from the Arm console.
5. Issue the command `ls /sys/firmware/efi/efivars` to show whether efivarfs is mounted. If it is not mounted, run:

```
mount -t efivarfs none /sys/firmware/efi/efivars
```

6. Run:

```
chattr -i /sys/firmware/efi/efivars/OobMacAddr-8be4df61-93ca-11d2-aa0d-00e098032b8c
```

7. Reconfigure the original MAC address burned by the manufacturer in the format `aa\bb\cc\dd\ee\ff`. Run:

```
printf "\x07\x00\x00\x00\x00\<original-MAC-address>" > /sys/firmware/efi/efivars/OobMacAddr-
8be4df61-93ca-11d2-aa0d-00e098032b8c
```

8. Reboot the device for the change to take effect.

Supported ethtool Options for OOB Interface

The Linux driver for the OOB port supports the handling of some basic ethtool requests: get driver info, get/set ring parameters, get registers, and get statistics.

To use the ethtool options available, use the following format:

```
$ ethtool [<option>] <interface>
```

Where `<option>` may be:

- `<no-argument>` – display interface link information
- `-i` – display driver general information
- `-S` – display driver statistics
- `-d` – dump driver register set
- `-g` – display driver ring information
- `-G` – configure driver ring(s)
- `-k` – display driver offload information
- `-a` – query the specified Ethernet device for pause parameter information

- `-r` – restart auto-negotiation on the specified Ethernet device if auto-negotiation is enabled

For example:

```
$ ethtool oob_net0
Settings for oob_net0:
        Supported ports: [ TP ]
        Supported link modes:   1000baseT/Full
        Supported pause frame use: Symmetric
        Supports auto-negotiation: Yes
        Supported FEC modes: Not reported
        Advertised link modes:  1000baseT/Full
        Advertised pause frame use: Symmetric
        Advertised auto-negotiation: Yes
        Advertised FEC modes: Not reported
        Link partner advertised link modes:  1000baseT/Full
        Link partner advertised pause frame use: Symmetric
        Link partner advertised auto-negotiation: Yes
        Link partner advertised FEC modes: Not reported
        Speed: 1000Mb/s
        Duplex: Full
        Port: Twisted Pair
        PHYAD: 3
        Transceiver: internal
        Auto-negotiation: on
        MDI-X: Unknown
        Link detected: yes
```

```
$ ethtool -i oob_net0
driver: mlxbf_gige
version:
firmware-version:
expansion-rom-version:
bus-info: MLNXBF17:00
supports-statistics: yes
supports-test: no
supports-eeprom-access: no
supports-register-dump: yes
supports-priv-flags: no
```

```
# Display statistics specific to BlueField-2 design (i.e. statistics that are not shown in the output of
"ifconfig oob0_net")
$ ethtool -S oob_net0
NIC statistics:
    hw_access_errors: 0
    tx_invalid_checksums: 0
    tx_small_frames: 1
    tx_index_errors: 0
    sw_config_errors: 0
    sw_access_errors: 0
    rx_truncate_errors: 0
    rx_mac_errors: 0
    rx_din_dropped_pkts: 0
    tx_fifo_full: 0
    rx_filter_passed_pkts: 5549
    rx_filter_discard_pkts: 4
```

IP Address Configuration for OOB Interface

The files that control IP interface configuration are specific to the Linux distribution. The udev rules file (`/etc/udev/rules.d/92-oob_net.rules`) that renames the OOB interface to `oob_net0` and is the same for Yocto, CentOS, and Ubuntu:

```
SUBSYSTEM=="net", ACTION=="add", DEVPATH=="/devices/platform/MLNXBF17:00/net/eth[0-9]",
NAME="oob_net0"
```

The files that control IP interface configuration are slightly different for CentOS and Ubuntu:

- CentOS configuration of IP interface:
    - Configuration file for `oob_net0`: `/etc/sysconfig/network-scripts/ifcfg-oob_net0`
    - For example, use the following to enable DHCP:

      ```
      NAME="oob_net0"
      ```

```
DEVICE="oob_net0"
NM_CONTROLLED="yes"
PEERDNS="yes"
ONBOOT="yes"
BOOTPROTO="dhcp"
TYPE=Ethernet
```

    - For example, to configure static IP use the following:

    ```
    NAME="oob_net0"
    DEVICE="oob_net0"
    IPV6INIT="no"
    NM_CONTROLLED="no"
    PEERDNS="yes"
    ONBOOT="yes"
    BOOTPROTO="static"
    IPADDR="192.168.200.2"
    PREFIX=30
    GATEWAY="192.168.200.1"
    DNS1="192.168.200.1"
    TYPE=Ethernet
    ```

- For Ubuntu configuration of IP interface, refer to section "Default Network Interface Configuration".