



NVIDIA DOCA Simple Forward VNF Application Guide

Table of contents

1. Introduction

2. System Design

3. Application Architecture

4. DOCA Libraries

5. Compiling the Application

6. Running the Application

7. Application Code Flow

8. References

This guide provides a Simple Forward implementation on top of NVIDIA® BlueField® DPU.

1. Introduction

Simple forward is a forwarding application that leverages the [DOCA Flow API](#) to take either VXLAN, GRE, or GTP traffic from a single RX port and transmits it on a single TX port.

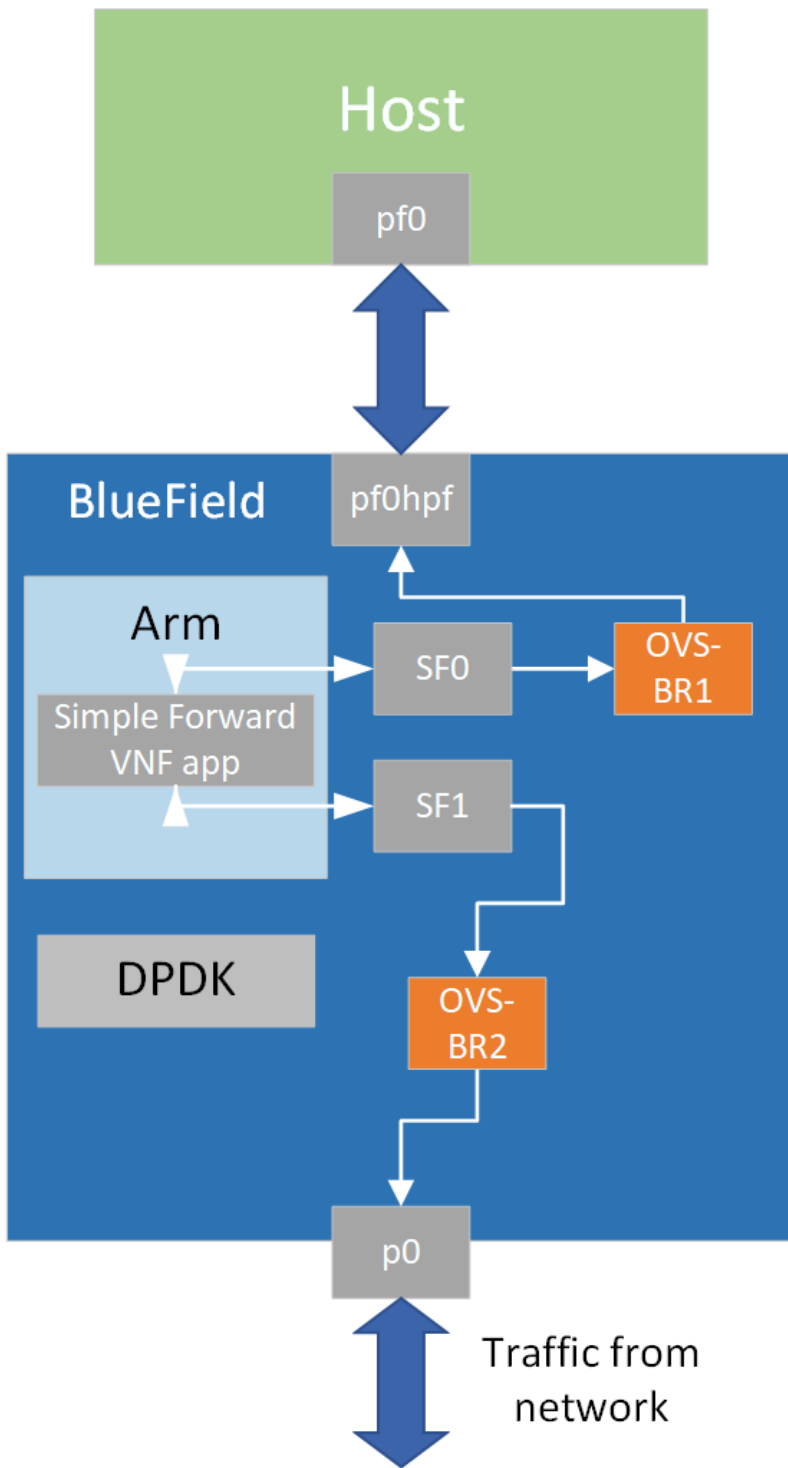
For every packet received on an RX queue on a given port, DOCA Simple Forward checks the packet's key, which consists of a 5-tuple. If it finds that the packet matches an existing flow, then it does not create a new one. Otherwise, a new flow is created with a FORWARDING component. Finally, the packet is forwarded to the TX queue of the egress port if the "rx-only" mode is not set.

The FORWARDING component type depends on the flags delivered when running the application. For example, if the `hairpinq` flag is provided, then the FORWARDING component would be hairpin. Otherwise, it would be RSS'd to software, and hence every VXLAN, GTP, or GRE packet would be received on RX queues.

Simple forward should be run with dual ports. By using a traffic generator, the RX port receives the VXLAN, GRE, or GTP packets and forwarding forwards them back to the traffic generator.

2. System Design

The following diagram illustrates simple forward's packet flows. It receives traffic coming from the wire and passes it to the other port.



3. Application Architecture

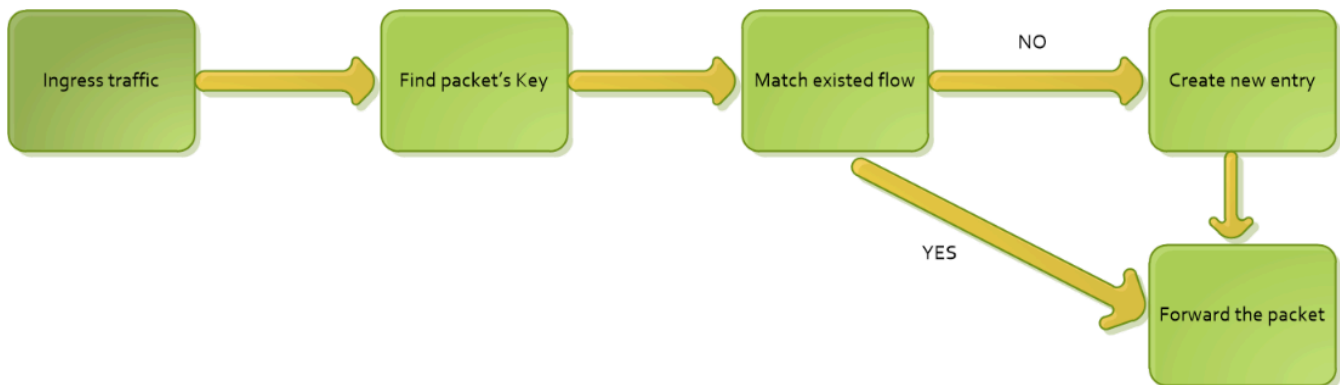
Simple forward first initializes DPDK, after which the application handles the incoming packets.

The following diagram illustrates the initialization process.



1. `Init_DPDK` – EAL init, parse argument from command line and register signal.
2. Start port – `mbuf_create`, `dev_configure`, rx/tx/hairpin queue setup and start the port.
3. `Simple_fwd INIT` – create flow tables, build default forward pipes.

The following diagram illustrates how to process the packet.



1. Based on the packet's info, find the key values (e.g. src/dst IP, src/dst port, etc).
2. Traverse the inner flow tables, check if the keys exist or not.
 - If yes, update inner counter
 - If no, a new flow table is added to the DPU
3. Forward the packet to the other port.

4. DOCA Libraries

This application leverages the following DOCA library:

- [DOCA Flow](#)

Refer to its respective programming guide for more information.

5. Compiling the Application

Info

Please refer to the [NVIDIA DOCA Installation Guide for Linux](#) for details on how to install BlueField-related software.

The installation of DOCA's reference applications contains the sources of the applications, alongside the matching compilation instructions. This allows for compiling the applications "as-is" and provides the ability to modify the sources, then compile a new version of the application.

Tip

For more information about the applications as well as development and compilation tips, refer to the [DOCA Applications](#) page.

The sources of the application can be found under the application's directory:

```
/opt/mellanox/doca/applications/simple_fwd_vnf/.
```

5.1 Compiling All Applications

All DOCA applications are defined under a single meson project. So, by default, the compilation includes all of them.

To build all the applications together, run:

```
cd /opt/mellanox/doca/applications/  
meson /tmp/build  
ninja -C /tmp/build
```

i Info

`doca_simple_fwd_vnf` is created under
`/tmp/build/simple_fwd_vnf/`.

5.2 Compiling Simple Forward Application Only

To directly build only the simple forward application:

```
cd /opt/mellanox/doca/applications/  
meson /tmp/build -Denable_all_applications=false -  
Denable_simple_fwd_vnf=true  
ninja -C /tmp/build
```

i Info

`doca_simple_fwd_vnf` is created under
`/tmp/build/simple_fwd_vnf/`.

Alternatively, users can set the desired flags in the `meson_options.txt` file instead of providing them in the compilation command line:

1. Edit the following flags in

`/opt/mellanox/doca/applications/meson_options.txt`:

- Set `enable_all_applications` to `false`
- Set `enable_simple_fwd_vnf` to `true`

2. Run the following compilation commands :

```
cd /opt/mellanox/doca/applications/  
meson /tmp/build  
ninja -C /tmp/build
```

Info

`doca_simple_fwd_vnf` is created under
`/tmp/build/simple_fwd_vnf/`.

5.3 Troubleshooting

Refer to the [NVIDIA DOCA Troubleshooting Guide](#) for any issue encountered with the compilation of the application .

6. Running the Application

6.1 Prerequisites

1. A FLEX profile number should be manually set to 3 on the system for the application to build the GRE, standard VXLAN and GRE pipes.

1. Set FLEX profile number to 3 from the DPU.

```
sudo mlxconfig -d <pcie_address> s  
FLEX_PARSER_PROFILE_ENABLE=3
```

2. Perform a [BlueField system reboot](#) for the `mlxconfig` settings to take effect.

i Info

Resetting the firmware can be done from the BlueField as well. For more information, refer to step 3.b of the "Upgrading Firmware" section of the [NVIDIA DOCA Installation Guide for Linux](#).

2. The Simple Forward application is based on DOCA Flow. Therefore, the user is required to allocate huge pages.

```
echo '2048' | sudo tee -a /sys/kernel/mm/hugepages/hugepages-2048kB/nr_hugepages
```

On some operating systems (RockyLinux, OpenEuler, CentOS 8.2) the default huge page size on the DPU (and Arm hosts) is larger than 2MB, and is often 512MB instead. One can find out the size of the huge pages using the following command:

```
$ grep -i huge /proc/meminfo

AnonHugePages:          0 kB
ShmemHugePages:        0 kB
FileHugePages:         0 kB
HugePages_Total:       4
HugePages_Free:        4
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:         524288 kB
Hugetlb:               6291456 kB
```

Given that the guiding principal is to allocate 4GB of RAM, in such cases instead of allocating 2048 pages, one should allocate the matching amount (8 pages):

```
echo '8' | sudo tee -a /sys/kernel/mm/hugepages/hugepages-524288kB/nr_hugepages
```

6.2 Application Execution

The simple forward application is provided in source form. Therefore, a compilation is required before the application can be executed.

1. Application usage instructions:

```
Usage: doca_simple_forward_vnf [DPDK Flags] -- [DOCA Flags]
[Program Flags]

DOCA Flags:
  -h, --help                Print a help synopsis
  -v, --version             Print program version
information
  -l, --log-level           Set the (numeric) log
level for the program <10=DISABLE, 20=CRITICAL, 30=ERROR,
40=WARNING, 50=INFO, 60=DEBUG, 70=TRACE>
  --sdk-log-level          Set the SDK (numeric) log
level for the program <10=DISABLE, 20=CRITICAL, 30=ERROR,
40=WARNING, 50=INFO, 60=DEBUG, 70=TRACE>
  -j, --json <path>       Parse all command flags
from an input json file

Program Flags:
  -t, --stats-timer <time> Set interval to dump
stats information
  -q, --nr-queues <num>   Set queues number
  -r, --rx-only            Set rx only
```

<code>-o, --hw-offload</code>	Set PCI address of the RXP engine to use
<code>-hq, --hairpinq queue</code>	Set forwarding to hairpin queue
<code>-a, --age-thread</code>	Start thread <code>do aging</code>

(i) Info

This usage printout can be printed to the command line using the `-h` (or `--help`) options:

```
./doca_simple_fwd_vnf -- -h
```

(i) Info

For additional information, refer to section "[Command Line Flags](#)".

2. CLI example for running the application on the BlueField:

```
./doca_simple_fwd_vnf -a auxiliary:mlx5_core.sf.4 -a  
auxiliary:mlx5_core.sf.5 -- -l 60
```

(i) Note

SFs must be enabled according to the [NVIDIA BlueField DPU Scalable Function User Guide](#).

Before creating SFs on a specific physical port, it is important to verify the encap mode on the respective PF FDB. The default mode is `basic`. To check the encap mode, run:

```
cat /sys/class/net/p0/compat/devlink/encap
```

In this case, disable encap on the PF FDB before creating the SFs by running:

```
/opt/mellanox/iproute2/sbin/devlink dev
switch set pci/0000:03:00.0 mode legacy
/opt/mellanox/iproute2/sbin/devlink dev
switch set pci/0000:03:00.1 mode legacy
echo none >
/sys/class/net/p0/compat/devlink/encap
echo none >
/sys/class/net/p1/compat/devlink/encap
/opt/mellanox/iproute2/sbin/devlink dev
switch set pci/0000:03:00.0 mode switchdev
/opt/mellanox/iproute2/sbin/devlink dev
switch set pci/0000:03:00.1 mode switchdev
```

If the encap mode is set to `basic` then the application fails upon initialization.

Note

The flag

```
-a auxiliary:mlx5_core.sf.4 -a
auxiliary:mlx5_core.sf.5
```

is mandatory for proper usage of the application.

1. Modifying this flag results unexpected behavior as only 2 ports are supported.
2. The SF number is arbitrary and configurable.

Note

The SF numbers must match the desired SF devices.

3. CLI example for running the application on the host:

```
./doca_simple_fwd_vnf -a 04:00.3 -a 04:00.4 -- -l 60
```

Note

The device identifiers must match the desired network devices.

Info

For more information, refer to section "Running DOCA Application on Host" in [NVIDIA DOCA Virtual Functions User Guide](#).

4. The application also supports a JSON-based deployment mode, in which all command-line arguments are provided through a JSON file:

```
./doca_simple_fwd_vnf --json [json_file]
```

For example:

```
./doca_simple_fwd_vnf --json ./simple_fwd_params.json
```

(i) Note

Before execution, ensure that the used JSON file contains the correct configuration parameters, and especially the PCIe addresses necessary for the deployment.

6.3 Command Line Flags

Flag Type	Short Flag	Long Flag/JSON Key	Description	JSON Content
DPDK Flags	a	devices	Add a PCIe device into the list of devices to probe.	<pre>"devices" : [{"device" : "sf", "id" : "4", "sft" : true } , {"device" : "sf", "id" : "5", "sft" : true } ,]</pre>

Flag Type	Short Flag	Long Flag/JSON Key	Description	JSON Content
General flags	h	help	Prints a help synopsis	N/A
	v	version	Prints program version information	N/A
	l	log-level	Set the log level for the application: <ul style="list-style-type: none"> • DISABLE=10 • CRITICAL=20 • ERROR=30 • WARNING=40 • INFO=50 • DEBUG=60 • TRACE=70 (requires compilation with <code>TRACE</code> log level support) 	<pre>"log-level" : 60</pre>
	N/A	sdk-log-level	Sets the log level for the program: <ul style="list-style-type: none"> • DISABLE=10 • CRITICAL=20 • ERROR=30 • WARNING=40 • INFO=50 • DEBUG=60 • TRACE=70 	<pre>"sdk-log-level" : 40</pre>
	j	json	Parse all command flags from an input JSON file	N/A
Program flags	t	stats-timer	Set interval to dump stats information.	<pre>"stats-timer" : 2</pre>
	q	nr-queues	Set queues number.	<pre>"nr-queues" : 4</pre>

Flag Type	Short Flag	Long Flag/JSON Key	Description	JSON Content
	r	rx-only	Set RX only. When set, the packets will not be sent to the TX queues.	"rx-only": false
	o	hw-offload	Set HW offload of the RXP engine to use.	"hw-offload": false
	hq	hairpinq	Set forwarding to hairpin queue.	"hairpinq": false
	a	age-thread	Start a dedicated thread that handles the aged flows.	"age-thread": false

i Info

Refer to [DOCA Arg Parser](#) for more information regarding the supported flags and execution modes.

6.4 Troubleshooting

Refer to the [NVIDIA DOCA Troubleshooting Guide](#) for any issue encountered with the installation or execution of the DOCA applications .

7. Application Code Flow

1. Parse application argument.

1. Initialize arg parser resources and register DOCA general parameters.

```
doca_argp_init();
```

2. Register application parameters.

```
register_simple_fwd_params();
```

3. Parse the arguments.

```
doca_argp_start();
```

1. Parse DPDK flags and invoke handler for calling the `rte_eal_init()` function.
2. Parse app parameters.

2. DPDK initialization.

```
dpdk_init();
```

Calls `rte_eal_init()` to initialize EAL resources with the provided EAL flags.

3. DPDK port initialization and start.

```
dpdk_queues_and_ports_init();
```

1. Initialize DPDK ports.

2. Create mbuf pool using `rte_pktmbuf_pool_create`.
3. Driver initialization – use `rte_eth_dev_configure` to configure the number of queues.
4. Rx/Tx queue initialization – use `rte_eth_rx_queue_setup` and `rte_eth_tx_queue_setup` to initialize the queues.
5. Rx hairpin queue initialization – use `rte_eth_rx_hairpin_queue_setup` to initialize the queues.
6. Start the port using `rte_eth_dev_start`.

4. Simple forward initialization.

```
simple_fwd_init();
```

1. `simple_fwd_create_ins` – create flow tables using `simple_fwd_ft_create`.
2. `simple_fwd_init_ports_and_pipes` – initialize DOCA port using `simple_fwd_init_doca_port` and build default pipes for each port.

5. Main loop.

```
simple_fwd_process_pkts();
```

1. Receive packets using `rte_eth_rx_burst` in a loop.
2. Process packets using `simple_fwd_process_offload`.
3. Transmit the packets on the other port by calling `rte_eth_tx_burst`. Or free the packet mbuf if `rx_only` is set to `true`.

6. Process packets.

```
simple_fwd_process_offload();
```

1. Parse the packet's `rte_mbuf` using `simple_fwd_pkt_info`.
2. Handle the packet using `simple_fwd_handle_packet`. If the packet's key does not match the existed the flow entry, create a new flow entry and PIPE using `simple_fwd_handle_new_flow`. Otherwise, increase the total packet's counter.

7. Simple forward destroy.

```
simple_fwd_destroy();
```

Simple forward close port and clean the flow resources.

8. DPDK ports and queues destruction.

```
dpdk_queues_and_ports_fini();
```

9. DPDK finish.

```
dpdk_fini();
```

Calls `rte_eal_destroy()` to destroy initialized EAL resources.

10. Arg parser destroy.

```
doca_argp_destroy();
```

- Free DPDK resources by call `rte_eal_cleanup()` function.

8. References

- `/opt/mellanox/doca/applications/simple_fwd_vnf/`
- `/opt/mellanox/doca/applications/simple_fwd_vnf/simple_fwd_params.`

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 (“NVIDIA”) makes no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assumes 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 and the NVIDIA logo are trademarks and/or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

© Copyright 2025, NVIDIA. PDF Generated on 10/09/2025