NVIDIA DOCA Simple Forward VNF

Application

MLNX-15-060492 _v1.4 | September 2022

Table of Contents

Chapter 1. Introdution

Simple forward is a forwarding application that takes either VXLAN, GRE, or GTP traffic from a single RX port and transmits it on a single TX port.

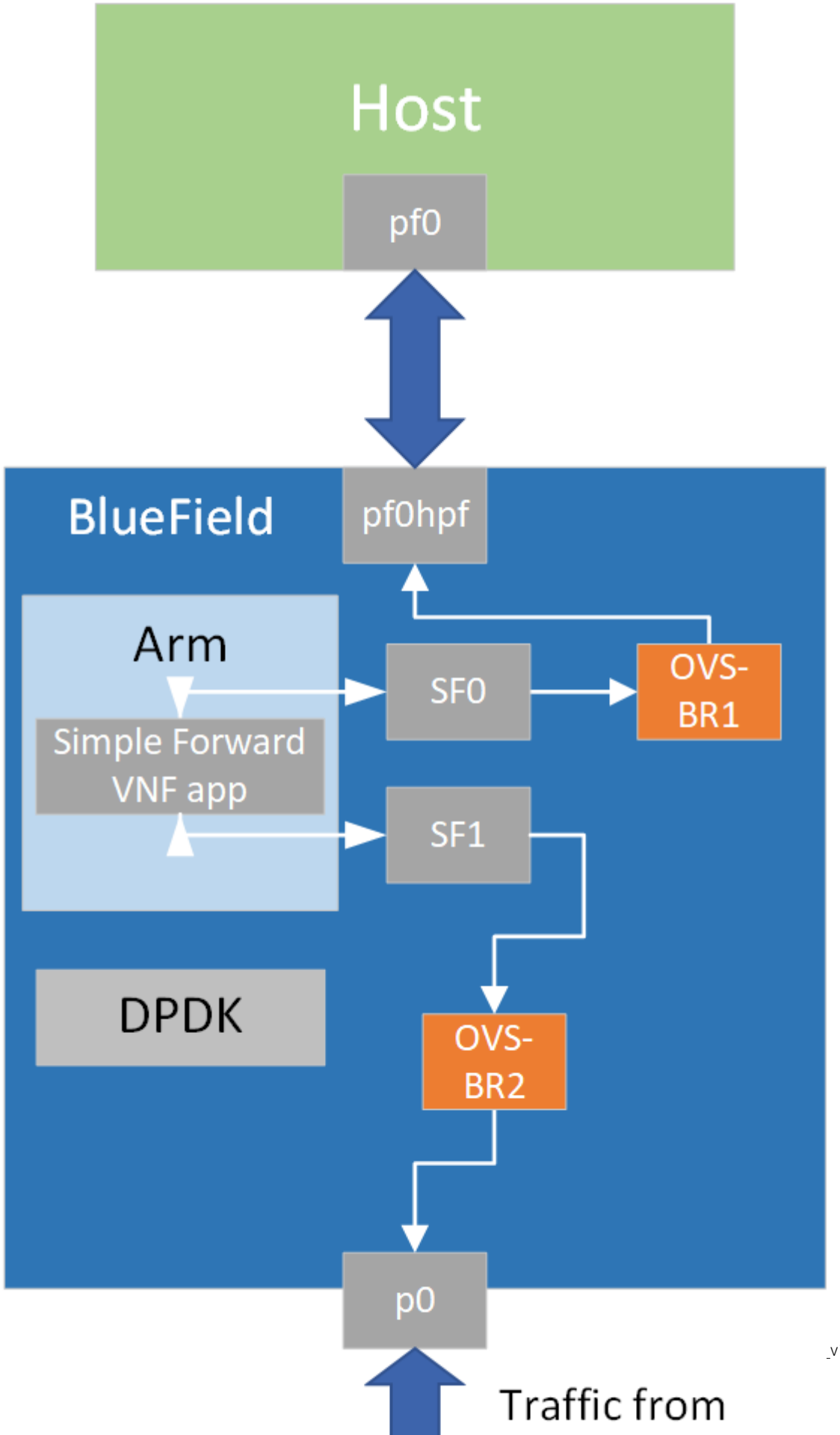
For every packet received on an RX queue on a given port, DOCA Simple Forward checks the packet's key, which consists of a 5-tuple. If it finds that the packet matches an existing flow, then it does not create a new one. Otherwise, a new flow is created with a FORWARDING component. Finally, the packet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

Chapter 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.


Host
pf0
BlueField
pf0hpf
Arm
Simple Forward VNF app
SF0
SF1
OVS-BR1
OVS-BR2
DPDK
p0
Traffic from

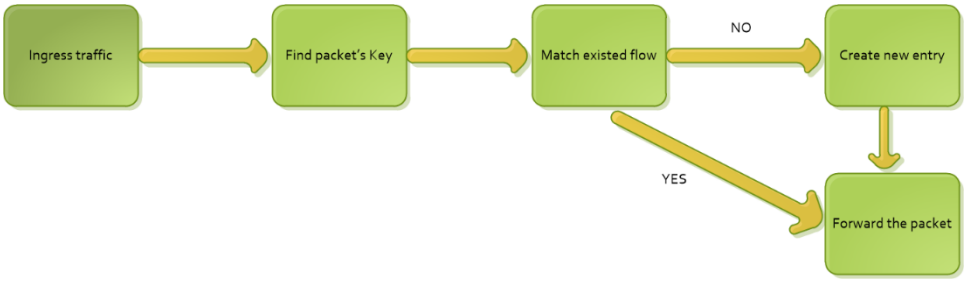
Chapter 3. Application Architecture

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 and new pipes are configured on the DPU
3. Forward the packet to the other port.

Chapter 4. DOCA Libraries

This application leverages the DOCA Flow Library.

Chapter 5. Configuration Flow

1. Parse application argument.
   ```
   doca_argp_init();
   ```
   a). Initialize arg parser resources.
   b). Register DOCA general flags.
   ```
   register_simple_fwd_params();
   ```
   c). Register application flags.
   ```
   doca_argp_start();
   ```
   d). Parse DPDK flags and invoke handler for calling the `rte_eal_init()` function.
   e). Parse app flags.
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();
   ```
   a). Initialize DPDK ports.
   b). Create mbuf pool using `rte_pktmbuf_pool_create`
   c). Driver initialization – use `rte_eth_dev_configure` to configure the number of queues
   d). Rx/Tx queue initialization – use `rte_eth_rx_queue_setup` and `rte_eth_tx_queue_setup` to initialize the queues
   e). Rx hairpin queue initialization – use `rte_eth_rx_hairpin_queue_setup` to initialize the queues
   f). Start the port using `rte_eth_dev_start`
4. Simple forward initialization.
   ```
   simple_fwd_init();
   ```
   a). `simple_fwd_create_ins` - create flow tables using `simple_fwd_ft_create`
   b). `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();
   ```
   a). Receive packets using `rte_eth_rx_burst` in a loop
   b). Process packets using `simple_fwd_process_offload`

c). 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();
```
   a). Parse the packet's `rte_mbuf` using `simple_fwd_pkt_info`.

   b). 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 closes port and cleans 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.

Chapter 6. Running Application

1. Refer to the following documents:
   - NVIDIA DOCA Installation Guide for details on how to install BlueField-related software.
   - NVIDIA DOCA Troubleshooting Guide for any issue you may encounter with the installation, compilation, or execution of DOCA applications.
2. The simple forward binary is located under `/opt/mellanox/doca/applications/simple_fwd_vnf/bin/doca_simple_fwd_vnf`. To build all the applications together, run:

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

3. To build the simple forward sample only:

   a). Edit the following flags in `/opt/mellanox/doca/applications/meson_option.txt`:

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

   b). Run the commands in step 2.

> **Note:** `doca_simple_fwd_vnf` is created under `./build/simple_fwd_vnf/src/`.

Application usage:

```
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 log level for the app <CRITICAL=0,
 DEBUG=4>
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
  -o, --hw-offload              Set hw offload
  -hq, --hairpinq               Set forwarding to hairpin queue
```

```
 -a, --age-thread                Start thread do aging
```

Note: For additional information on available flags for EAL, use `-h` before the `--` separator:

```
/opt/mellanox/doca/applications/simple_fwd_vnf/bin/doca_simple_fwd_vnf -h
```

Note: For additional information on the application, use `-h` after the `--` separator:

```
/opt/mellanox/doca/applications/simple_fwd_vnf/bin/doca_simple_fwd_vnf --
 -h
```

4. Running the application on BlueField:

   - Pre-run setup:

     The simple forward example is based on DPDK libraries. Therefore, the user is required to provide DPDK flags, and allocate huge pages.

     ```
     sudo echo 2048 > /sys/kernel/mm/hugepages/hugepages-2048kB/nr_hugepages
     ```

   - CLI example for running the app:

     ```
     /opt/mellanox/doca/applications/simple_fwd_vnf/bin/doca_simple_fwd_vnf -a
      auxiliary:mlx5_core.sf.4 -a auxiliary:mlx5_core.sf.5 -- -l 4
     ```

     Note: The flag `-a auxiliary:mlx5_core.sf.4 -a auxiliary:mlx5_core.sf.5` is mandatory for proper usage of the application. Modifying this flag results unexpected behavior as only 2 ports are supported. The SF number is arbitrary and configurable.

     Note: SFs must be enabled according to Scalable Function Setup 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 eswitch set pci/0000:03:00.0
      mode legacy
     /opt/mellanox/iproute2/sbin/devlink dev eswitch 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 eswitch set pci/0000:03:00.0
      mode switchdev
     /opt/mellanox/iproute2/sbin/devlink dev eswitch set pci/0000:03:00.1
      mode switchdev
     ```

     Note that if the encap mode is set to `basic` then the application fails upon initialization.

5. Running the application on the host, CLI example:

   ```
   /opt/mellanox/doca/applications/simple_fwd_vnf/bin/doca_simple_fwd_vnf -a 04:00.3
    -a 04:00.4 -- -l 4
   ```

   Note: Refer to section "Running DOCA Application on Host" in NVIDIA DOCA Virtual Functions User Guide.

6. To run `doca_simple_fwd_vnf` using a JSON file:

   ```
   doca_simple_fwd_vnf --json [json_file]
   ```

   For example:

   ```
   cd /opt/mellanox/doca/applications/simple_fwd_vnf/bin
   ```

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

Chapter 7. Arg Parser DOCA Flags

Refer to NVIDIA DOCA Arg Parser User Guide for more information.

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.	`"devices": [ {"device": "sf", "id": "4","sft": true}, {"device": "sf", "id": "5","sft": true}, ]`
General Flags	l	log-level	Set the log level for the application: ▶ CRITICAL=0 ▶ ERROR=1 ▶ WARNING=2 ▶ INFO=3 ▶ DEBUG=4	`"log-level": 4`
	v	version	Print program version information.	N/A
	h	help	Print a help synopsis.	N/A
Program Flags	t	stats-timer	Set interval to dump stats information.	`"stats-timer": 2`
	q	nr-queues	Set queues number.	`"nr-queues": 4`
	r	rx-only	Set RX only. When set, the packets	`"rx-only": false`

Flag Type	Short Flag	Long Flag/JSON Key	Description	JSON Content
			will not be sent to the TX queues.	
	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`

Chapter 8. References

- `/opt/mellanox/doca/applications/simple_fwd_vnf/src/simple_fwd_vnf.c`

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.

NVIDIA Corporation | 2788 San Tomas Expressway, Santa Clara, CA 95051
http://www.nvidia.com