NVIDIA DOCA DNS Filter

Application Guide

MLNX-15-060486 _v2.2.1 | October 2023

Table of Contents

Chapter 1. Introduction

Important: No updates were made to the DOCA DNS Filter application in DOCA 2.2. Please refer to DOCA 2.5 for a note regarding future updates.

Domain name system (DNS) translates domain names to IP addresses so browsers can load internet resources. Each device connected to the Internet has a unique IP address which other machines use to find the device.

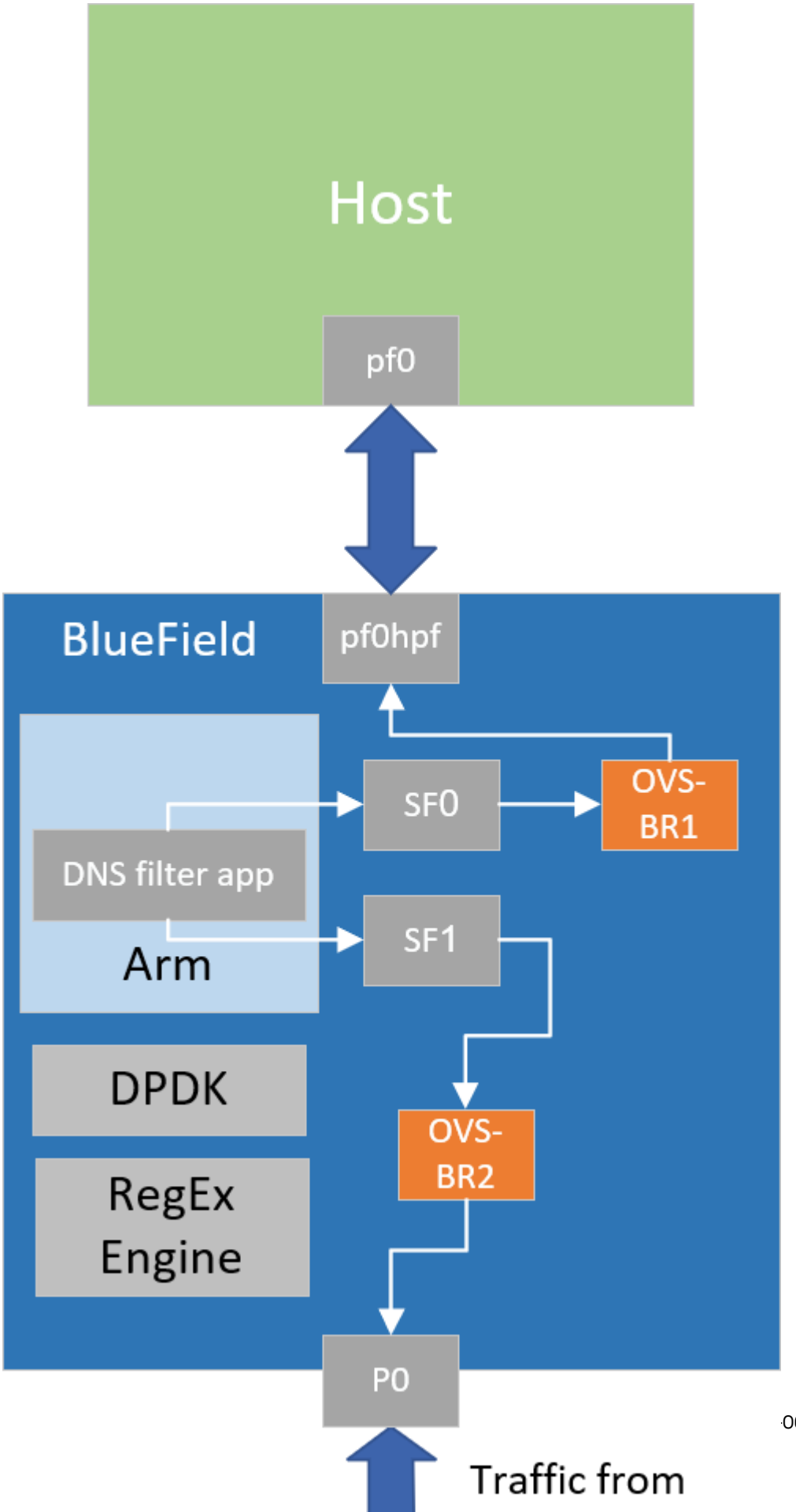
The DNS process includes several steps:

1. Once a user tries to log into a website using a browser, the user's device creates a DNS query and sends it to a DNS resolver.
2. The DNS resolver queries the DNS domain to get an IP address by searching its cache or sending the request to another DNS server.
3. Once a match is found, the DNS resolver returns the correct IP matching the DNS domain.
4. The user can log into the required website using the correct IP.

DNS filter is used to offload DNS requests from the host to the BlueField DPU Arm which allows reducing CPU overhead as Arm allows further DNS processing to be done. The application filters DNS requests according to a domain name allow/deny list.

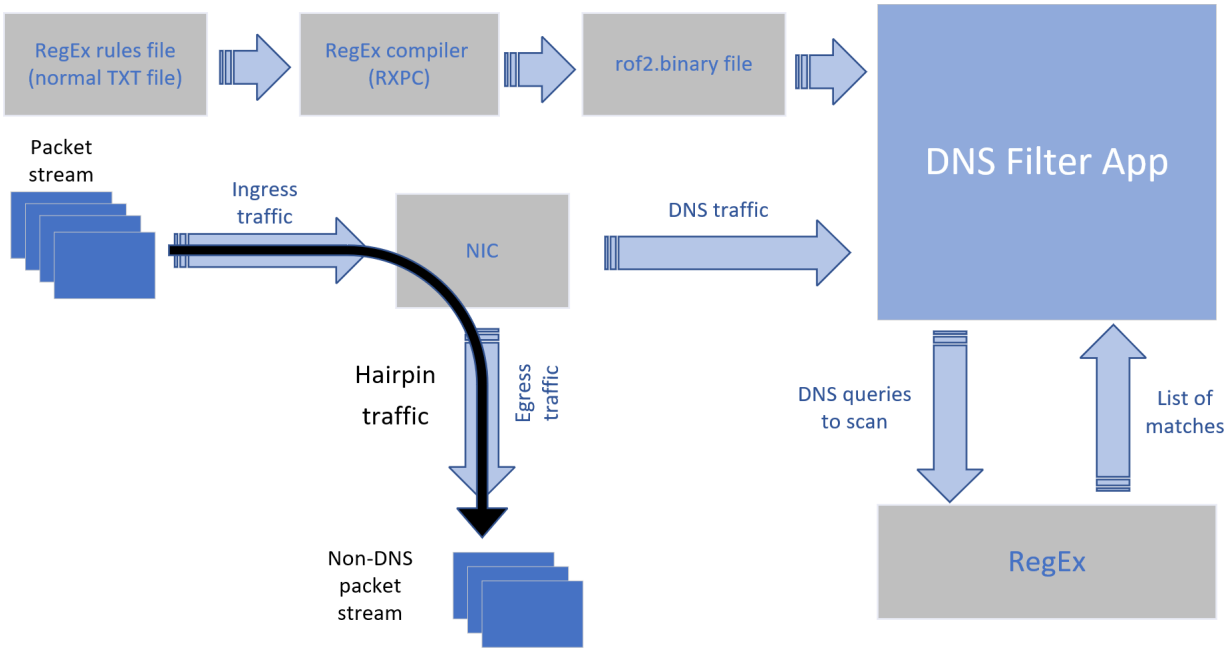
Chapter 2. System Design

The DNS filter application is designed to run as a "bump-on-the-wire" on the BlueField-2 DPU instance. The DPU intercepts the traffic coming (ingress traffic) from the wire and either passes it to the Arm (to filter the received packets according to the listing type) or forwards it to the egress port using hairpin. The decision is made by traffic classification.


Host
pf0
BlueField
pf0hpf
SF0
OVS-
BR1
DNS filter app
SF1
Arm
DPDK
OVS-
BR2
RegEx
Engine
P0
Traffic from

Chapter 3. Application Architecture

DNS Filter runs on top of DOCA Flow to classify DNS requests. It then uses the hardware RegEx engine to find matches according to the domain name listing rules (compiled regular expressions).



1. RegEx listing rules file is compiled into a `rof2.binary` file by the user.
2. The RegEx binary rules file is loaded into the RegEx engine.
3. Ingress packet types are identified using pipes which encapsulate flow rule matching patterns and actions.
4. The DNS filter application builds 3 pipes for each port (DNS drop pipe, DNS forward pipe, and hairpin pipe). Every pipe except the drop pipe includes exactly one entry. The drop pipe includes many entries in runtime and each entry represents a dropped packet 5-tuple. After app initialization and configuration, and before accepting any traffic, the pipe is empty.
5. The drop pipe matches DNS packets already blocked to drop them. The hairpin pipe matches every packet (no misses). The drop pipe serves as a root pipe, the

DNS forward pipe serves as a forwarding miss component to the drop pipe, and the hairpin pipe serves as a forwarding miss component to the DNS forward pipe.

Therefore, every received packet is checked first against the drop pipe. If there is a match, then it is dropped. Otherwise (miss case), it is checked against the DNS forward pipe. If there is a match there, it is forwarded to the Arm. Otherwise (another miss case), it is then forwarded to the hairpin pipe and matched.

Chapter 4. DOCA Libraries

This application leverages following DOCA libraries:

- DOCA Flow library
- DOCA RegEx library

Chapter 5. Configuration Flow

1. Parse application argument.
   ```
   doca_argp_init();
   ```
   a). Initialize arg parser resources.
   b). Register DOCA general flags.
   ```
   register_dns_filter_params();
   ```
   c). Register DNS filter 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, including mempool allocation.
   b). Initialize hairpin queues if needed.
   c). Binds hairpin queues of each port to its peer port.
4. DNS filter initialization.
   ```
   dns_filter_init();
   ```
   a). DOCA flow and DOCA flow port initialization.
   b). Creates hairpin pipe for both ports. This pipe includes one entry that matches every type of packet (no misses) and forwards it to the egress port through hairpin.
   c). Creates DNS forward pipe for both ports. The built pipe has one entry for matching DNS traffic and forwarding it to Arm. In addition, the hairpin pipe serves for forwarding if the DNS entry does not match (i.e., for each non-DNS packet, packets are hairpined).
   d). Creates drop pipe that serves as a root pipe for both ports. At the start, the pipe is empty. But as the application runs, it adds entries for dropped packets. In addition, the DNS forward pipe serves for forwarding if drop pipe entries do not match.
   e). DOCA RegEx initialization.
   f). Configure RegEx with the compiled rules file.

5. Processing and filtering DNS packets.
   ```
   dns_worker_lcores_run();
   ```
   a). All received packets on Arm are DNS packets, while non-DNS packets are forwarded to the egress port using hairpin allowing DNS packets to be filtered.
   b). Extract DNS queries.
   c). Send DNS queries as jobs to RegEx engine.
   d). Filter DNS packets according to RegEx responses.
   e). Block packet if needed by adding an entry to the DNS drop pipe.
6. DNS filter destroy.
   ```
   dns_filter_destroy();
   ```
   a). Free all allocated resources.
   b). Free all DOCA RegEx resources.
7. DPDK ports and queues destruction.
   ```
   dpdk_queues_and_ports_fini();
   ```
8. DPDK finish.
   ```
   dpdk_fini();
   ```
   Calls `rte_eal_destroy()` to destroy initialized EAL resources.
9. Arg parser destroy.
   ```
   doca_argp_destroy()
   ```
   a). Free DPDK resources.

Chapter 6. Dependencies

To run DNS Filter on the NVIDIA converged accelerator using a GPU device, you must build it using meson version 0.59.0 or higher. As such, DOCA's installation provides an updated meson version of 0.61.2.

You also need DPDK version 20.11.4.1 or higher which includes the `gpudev` library.

Chapter 7. Running the Application

1. Refer to the following documents:
   - NVIDIA DOCA Installation Guide for Linux 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 DNS filter example binary is located under `/opt/mellanox/doca/applications/dns_filter/bin/doca_dns_filter`. To build the applications together, run:

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

3. To build the DNS Filter application only:

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

   - Set `enable_all_applications` to `false`
   - Set `enable_dns_filter` to `true`

   b). Run the commands in step 2.

> Note: `doca_dns_filter` is created under `./build/dns_filter/src/`.

Application usage:

```
Usage: doca_dns_filter [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 program
 <CRITICAL=20, ERROR=30, WARNING=40, INFO=50, DEBUG=60>

Program Flags:
  -t, --type                        Set DNS listing type {allow, deny}
  -r, --rules <path>                Path to rules file (rof2.binary)
```

```
  -p, --pci-addr <address>          Set PCI address of the RXP engine to use
```

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

```
/opt/mellanox/doca/applications/dns_filter/bin/doca_dns_filter -h
```

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

```
/opt/mellanox/doca/applications/dns_filter/bin/doca_dns_filter -- -h
```

4. Running the application on BlueField:

   - To run the application, the RegEx-compiled rule files must be supplied to it. These files usually end with `*.rof2.binary`. To compile the example rules file, run:

     ```
     cd /opt/mellanox/doca/applications/dns_filter/bin/
     rxpc -V bf2 -f regex_rules.txt -p 0.01 -o /tmp/regex_rules
     ```

     The results of the `rxpc` are written to the `/tmp/` directory, each file with the prefix `regex_rules`.

     Note: For more information, refer to NVIDIA RXP Compiler Tool Guide.

   - Pre-run setup:

     a). The DNS Filter 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
     ```

     b). Make sure the RegEx engine is active:

     ```
     systemctl status mlx-regex
     ```

     If the status is inactive (`Active: failed`), run:

     ```
     systemctl start mlx-regex
     ```

   - CLI example for running the app:

     ```
     /opt/mellanox/doca/applications/dns_filter/bin/
     doca_dns_filter -a auxiliary:mlx5_core.sf.4,dv_flow_en=2 -a
      auxiliary:mlx5_core.sf.5,dv_flow_en=2 -- -l 60 -p 03:00.0 --rules /tmp/
     regex_rules.rof2.binary --type allow
     ```

     Note: The flags `-a auxiliary:mlx5_core.sf.4,dv_flow_en=2-a auxiliary:mlx5_core.sf.5,dv_flow_en=2` are necessary for proper usage of the application. Modifying them results in unexpected behavior as only 2 ports are supported. The subfunction number is arbitrary and configurable.

     Note: Sub-functions must be enabled according to the Scalable Function Setup Guide.

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

   ```
   /opt/mellanox/doca/applications/dns_filter/bin/doca_dns_filter -a
    03:00.3,dv_flow_en=2 -a 03:00.4,dv_flow_en=2 -c 0xff -- -l 60 -p 03:00.0 --
   rules /tmp/regex_rules.rof2.binary --type deny
   ```

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

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

```
doca_dns_filter --json [json_file]
```

   For example:

```
cd /opt/mellanox/doca/applications/dns_filter/bin
./doca_dns_filter --json /root/dns_filter_params.json
```

Chapter 8. Arg Parser DOCA Flags

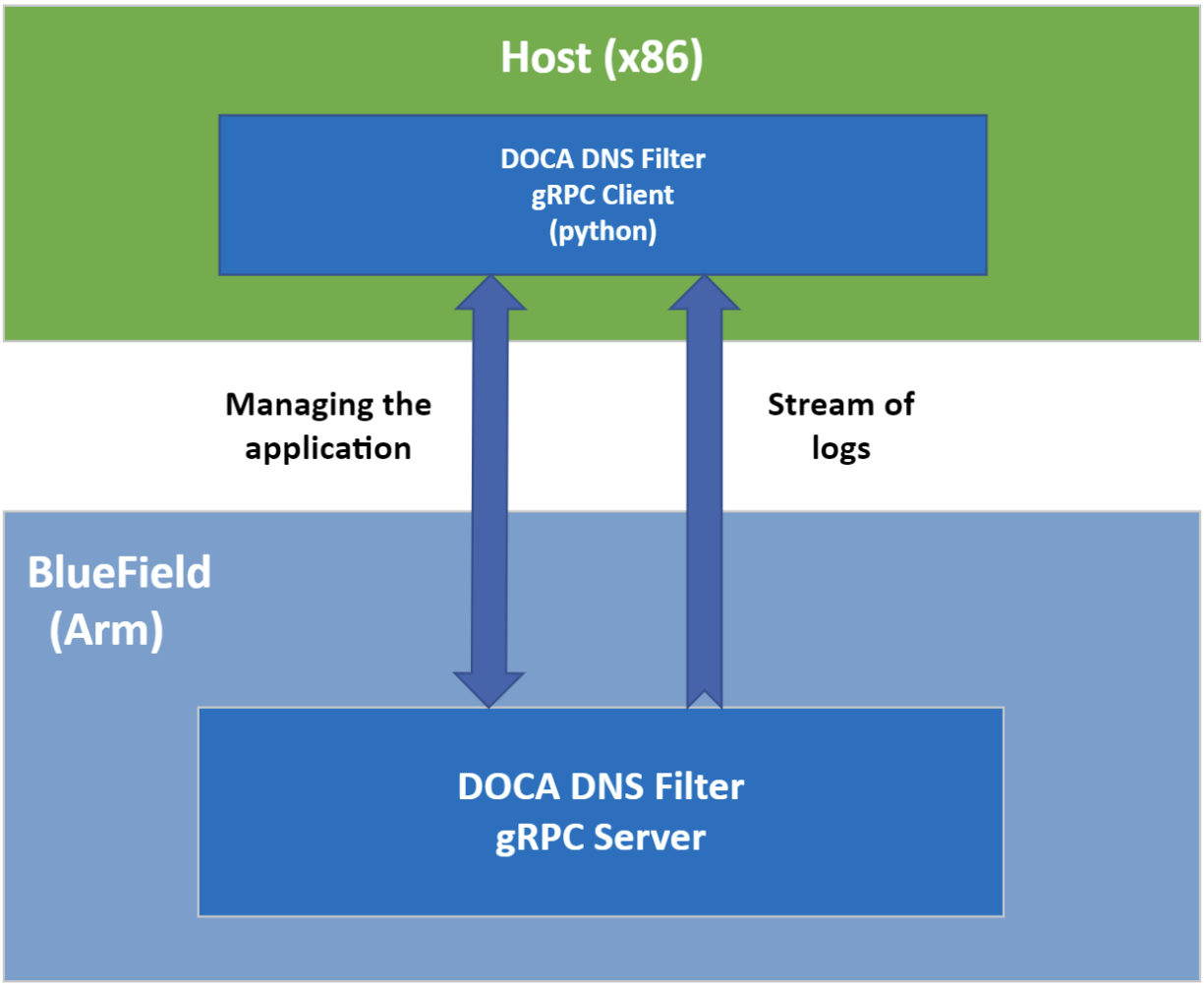
Refer to NVIDIA DOCA Arg Parser Programming Guide for more information.

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", "hws": true, }, { "device": "sf", "id": "5", "hws": true, }, ]`
General flags	`l`	`log-level`	Set the log level for the application: ▶ CRITICAL=20 ▶ ERROR=30 ▶ WARNING=40 ▶ INFO=50 ▶ DEBUG=60	`"log-level": 60`
	`v`	`version`	Print program version information	N/A

Flag Type	Short Flag	Long Flag/ JSON Key	Description	JSON Content
	`h`	`help`	Print a help synopsis	N/A
Program flags	`r`	`rules`	Path to rules file (`rof2.binary`) Note: `rules` is a mandatory flag.	`"rules": "/tmp/regex_rules.rof2.binary"`
	`t`	`type`	Set DNS listing type (`allow` or `deny`) Note: `type` is a mandatory flag.	`"type": "allow"`
	`p`	`pci-addr`	Set PCI address of the RXP engine to use Note: `pci-addr` is a mandatory flag.	`"pci-addr": "03:00.0"`

Chapter 9. Managing gRPC-Enabled Application from Host

For instructions on running the gRPC application server on the BlueField, refer to NVIDIA DOCA gRPC Infrastructure User Guide.



To run the Python client of the gRPC-enabled application:

```
./doca_dns_filter_gRPC_client.py -d/--debug <server address[:server port]>
```

For example:

```
/opt/mellanox/doca/applications/dns_filter/bin/grpc/client/
doca_dns_filter_gRPC_client.py 192.168.104.2
```

Chapter 10. Running Application on NVIDIA Converged Accelerator

This section details the steps necessary to run the DNS filter application on NVIDIA converged accelerator.

The DNS-filter application running on the converged accelerator has the same logic as described in previous sections of this page except for the extraction of DNS queries from the packets on the Arm. The extraction is done on the GPU side. The extracted queries are sent to the RegEx engine to check whether there is a match or not.

To make use of the GPU's capabilities, make sure to perform the following steps:

1. Refer to the NVIDIA DOCA Installation Guide for Linux for instructions on installing NVIDIA driver for CUDA and a CUDA-repo on your setup.
2. Create the sub-functions and configure the OVS according to Scalable Function Setup Guide.

10.1. Compiling and Running Application

A pre-compiled DNS filter application is included in the `doca-grpc` package for hosts based on Ubuntu 20.04 and Ubuntu 22.04.

For the DPU, or on hosts lacking the `doca-grpc` packages, users can still compile the application and run it. All the sources needed for building, compiling, and running the application with GPU support are found under `/opt/mellanox/doca/applications/dns_filter/src`.

To build and run the application, perform the following steps:

1. Setup CUDA paths:
```
export CPATH=/usr/local/cuda/targets/sbsa-linux/include:$CPATH
export LD_LIBRARY_PATH=/usr/local/nvidia/lib:/usr/local/nvidia/lib64:
$LD_LIBRARY_PATH
export PATH=/usr/local/nvidia/bin:/usr/local/cuda/bin:$PATH
```
2. To build the application with GPU support:

   a). Edit the `enable_gpu_support` flag to `true` in `/opt/mellanox/doca/applications/meson_option.txt`.

b). Compile application sources. Run:

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

`doca_dns_filter_gpu` is created under `./build/dns_filter/src/` alongside the regular `doca_dns_filter` binary which is compiled without the GPU support.

3. To run the application with GPU support:

   - Follow the instructions for compiling the RegEx rules file and pre-run setup in step 4 under section Running the Application.

     Note: Make sure the GPU's PCIe address is provided with the flags to the `gpudev` DPDK library.

   - Assuming the PCIe address of the GPU is `06:00`, the command to run the application is:

```
./build/dns_filter/src/doca_dns_filter_gpu -
a auxiliary:mlx5_core.sf.4,dv_flow_en=2  -a
 auxiliary:mlx5_core.sf.5,dv_flow_en=2  -a 06:00.0 -- -l 60 -p 03:00.0 -r /
tmp/regex_rules.rof2.binary -t allow
```

4. To run the application, follow the steps in Running the Application.

Chapter 11. References

- `/opt/mellanox/doca/applications/dns_filter/src/`
- `/opt/mellanox/doca/applications/dns_filter/src/grpc/dns_filter.proto`

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

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