Dataplane Stack

Release 22.06

Arm Limited.

May 25, 2023

CONTENTS

CHAPTER

ONE

CONTENTS

1.1 Overview

1.1.1 Introduction

The Dataplane Stack provides users with solution reference implementations of high performance user space networking functionalities which would run on various Arm platforms.

End users interested in running these networking functions on Arm servers may include networking software developers, Arm partners and their customers, CPU performance analysts, CPU designers, and marketing team.

Target Platforms

Supported platforms

Current use cases have been validated in Ubuntu 20.04 distro on below machines:

- Ampere Altra (Neoverse-N1)

Future Platforms

The solutions will be ported to more platforms in the future. The planned platform includes:

- More silicon platforms - e.g., Neoverse-N2 server or embedded level CPUs, mainly focus on network functions and performance on Arm CPUs.
- Fixed Virtual Platforms - FVP are complete simulations of an Arm system, including processor, memory and peripherals. These are set out in a "programmer's view", which gives a comprehensive model on which to build and test users' software. FVP mainly focuses on verifying network functions and CPU behavior analysis in the early IP design stage.
- FPGA Emulator - RTL based CPU IP system emulator, mainly focuses on verifying network functions, performance benchmarking and CPU behavior analysis in the pre-silicon SOC design verification stage.

1.1.2 Solution Architecture

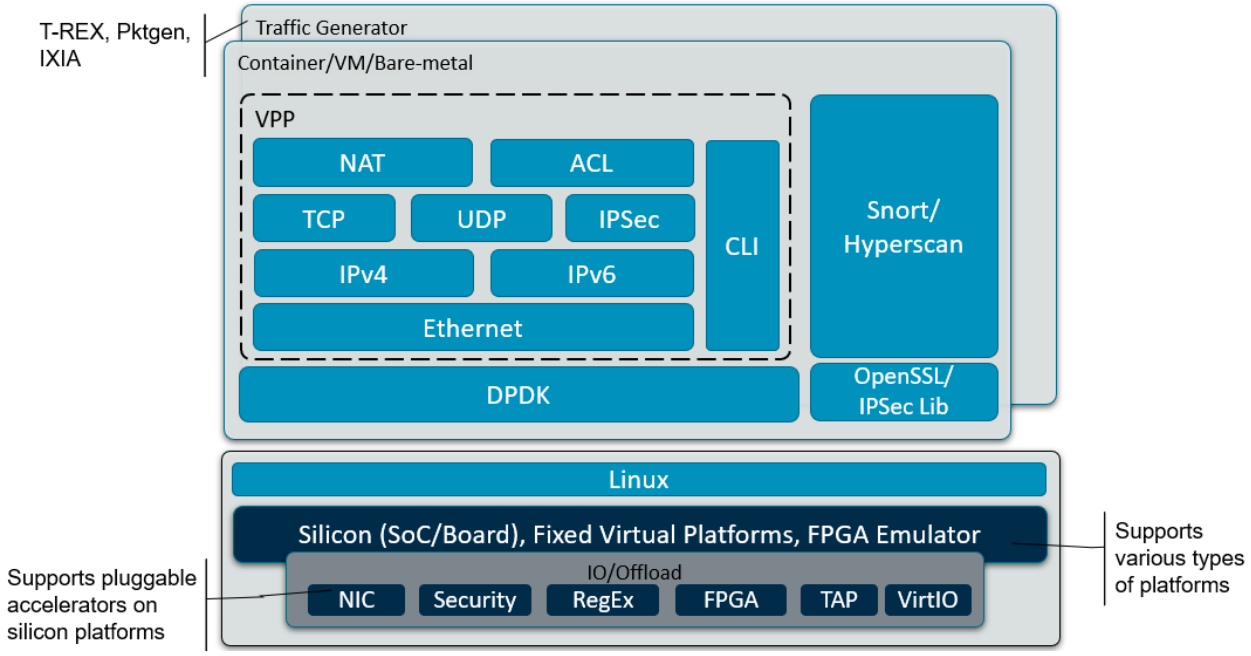


Fig. 1: Dataplane Stack solution architecture

This software mainly focuses on the dataplane network, as performance is of the primary concerns in dataplane implementation on COTS servers, e.g., Ampere Altra, AWS Gravition3, Alibaba Yitian 710.

This project provides the networking functionalities based on some well-known userspace network open source projects:

- DPDK: acronym of Data Plane Development Kit (DPDK), mostly runs in user space. It comprises of libraries to achieve high I/O performance and reach high packet processing rates, which are some of the most important features in the networking area. It supports multivendors and architectures, and runs on variety of CPU architectures, e.g., Arm, Power, x86. DPDK was created for the telecom/datacom infrastructure, but today, it's used almost everywhere, including the cloud, data centers, appliances, containers and more.
- VPP: acronym of Vector Packet Processor (VPP), is a fast and scalable layer 2-4 multi-platform network stack that provides out-of-the-box production quality switch/router functionality. It runs in Linux user space on multiple architectures including ARM, Power and x86 architectures. The benefits of this implementation of VPP are its high performance, proven technology, its modularity and flexibility, and rich feature set.
- Snort: is the foremost Open Source Intrusion Prevention System (IPS) in the world. Snort IPS uses a series of rules that help define malicious network activity and uses those rules to find packets that match against them and generates alerts for users.

1.1.3 Use Cases

Supported Usecases

Currently, below usecase/features have been implemented and supported:

- VPP based IPv4 forwarding
- DPDK L3fwd based IPv4/IPv6 forwarding

Future Usecases

The solutions will gradually expand with more functions and features.

The functions below are classified from functionality perspective, instead of from software component or implementation perspective. For example, IPv4 routing function could be implemented using VPP L3 or DPDK L3 features, and the firewall function could be provided using VPP ACL/Classifier or Snort3. Certain use case has a large scope and may be implemented with multiple software components and instances.

The implementation mechanism of each network function below will be described in detail in later chapters in this documentation.

- **L2 Switching**

  This is a typical L2 network function, including MAC learning, bridging, cross-connect, flooding, VLAN, etc.

- **IPv4 Routing**

  Forward packets to the next-hop IP by looking up the destination IP in the packets.

- **IPv6 Routing**

  IPv6 is almost identical to IPv4 routing under CIDR. The major difference is the addresses are 128-bit IPv6 addresses instead of 32-bit IPv4 addresses.

- **NAT**

  Network Address Translation (NAT) is a method of mapping an IP address space into another by modifying network address information in the IP header of packets while they are in transit across a traffic routing device. This software covers the basic NAT functions, e.g., SNAT, DNAT.

- **VxLAN**

  Virtual eXtensible Local Area Network (VXLAN) is an encapsulation protocol that provides data center connectivity. It uses tunneling to stretch Layer 2 connections over an underlying Layer 3 network. In data centers, VXLAN is the most commonly used protocol to create overlay networks that sit on top of the physical network, enabling the use of virtual networks.

- **IPSec**

  Internet Protocol Security (IPSec) is a framework of techniques used to secure the connection over the network communication. The typical encryption/decryption algorithms are covered in the implementation.

- **Firewall**

  A firewall is a network security device that monitors incoming and outgoing network traffic and permits or blocks data packets based on a set of security rules. The typical security applications are demonstrated based on some VPP and Snort3 security implementations.

- **TCP Termination**

  This function is primarily based on VPP host-stack, and we investigate the implementations with some other open source TCP/IP user software.

- **SSL Proxy**

  The SSL proxies control Secure Sockets Layer – SSL traffic -to ensure secure transmission of data between a client and a server. The SSL proxy is transparent, which means it performs SSL encryption and decryption between the client and the server.

- **Wireless Mid-haul or Back-haul**

  Backhaul is better known than fronthaul. It refers to the connections between a mobile network and a wired network that backhaul traffic from disparate cell sites to a mobile switching telephone office.

1.1.4 Implement Consideration

This software aims to provide high-throughput packet processing software stack and solutions to solve customer and partner use case in networking applications. There are several considerations from technical perspective in the implementation process:

- **Integrate DPDK, VPP and SNORT/Hyperscan**

  As mentioned above, this software provides network functions based on some open source projects, e.g., DPDK, VPP, Snort3 and scripts to combine them together for a complete solution.

- **Execute on Arm**

  The software is mainly implemented, validated, optimized and deployed on Arm AArch64 architecture and platforms. The target platforms are listed in later section.

- **Optimize for Arm architectures**

  Some of above open source projects are well-supported in community, including arch-specific compilation, community CI/CD, distro packages, arch-specific optimization, etc. While some of the projects are yet to be supported.

  That's why the software has developed some scripts to resolve compilation issues on Arm platforms, compile project source code with arch-specific features, apply optimization patches done by Arm OSS Networking, deploy optimal parameters tuned for specific Arm platforms, etc.

- **Integrate security libraries, e.g., OpenSSL, IPSec**

  A group of protocols are supported in established use cases to set up encrypted connections with the typical security libraries between devices. It helps keep data sent over public networks secure.

- **Support hardware offloads**

  The solution supports use cases that allow to offload some router features onto the underneath hardware. This allows reaching wire speeds when routing packets, which simply would not be possible with the CPU.

- **Validate with multiple traffic generator**

  Currently, quickstart and user guide use IXIA traffic generator to validate solution use cases. Some software packet generators are planned to be supported and provided along with the solution software in the future, e.g., TRex, DPDK Pktgen, Scapy.

1.1.5 Purposes

The network functions that the project provided serves multiple purposes of:

- Showcase the integration of various components and act as poof of concept to all stakeholders who cares about network function feasibility on Arm.
- Allow for performance analysis/optimization with a solution that is close to customers' production deployment.
- Provide customers with a out-of-the-box reference design for rapid design modeling.

1.1.6 Repository Structure

The code repository are stored in Arm GitLab arm-reference-solutions group, with the link, https://gitlab.arm.com/arm-reference-solutions/dataplane-stack.

```
├── CHANGELOG.rst - Change notes for each releases
├── doc           - Sphinx based documents
├── LICENSE.rst   - License file of this project
├── MAINTAINERS   - Maintainer recording of specific module/use-cases
├── Makefile      - Compiling various components and documentation
├── setup.sh      - Script to set up use case environment
├── README.md
└── tools         - Tools used to manage the project
```

1.1.7 License

The software is provided under an Apache 2.0 license (more details in Apache 2.0).

Contributions to the project should follow the same license.

1.1.8 Contributions

This project has not put in place a process for contributions currently. While proposals on new features, new use-cases or any improvement are welcome always. You are welcome to create Merge-Requests in project's Gitlab repository dataplane-stack to demonstrate your ideas and suggestions. We will sincerely investigate your proposals and provide feedback to you.

For bug reports, please submit an Issue via GitLab repository dataplane-stack. Please provide as much information as possible:

- Hardware information, including details on CPU, server, memory, disk, NIC, etc.
- System configuration, like distro type and version, kernel version, command line parameters, NUMA, huge-page setting
- Tools, e.g., compiler version, script tool version
- Use case topology and testing method

1.1.9 Feedback and Support

To provide feedback or to request support, please contact project maintainer by email at Lijian Zhang <Lijian.Zhang@arm.com>, or check 'MAINTAINERS' file in top directory of repository for module/feature/use-case specific maintainers.

Arm licensees may also contact Arm via their partner managers to request support.

1.2 Quickstart Guide

1.2.1 Introduction

Welcome to the Dataplane Stack reference solution quickstart guide. This guide provides instructions on how to fetch source code, build the source code and run sample applications.

By following the steps in this quickstart guide to the end, you will set up two user space programs for high-throughput packet forwarding. The programs accept packets from NIC port 0 and forward them out on the same NIC port based on their destination IP address.

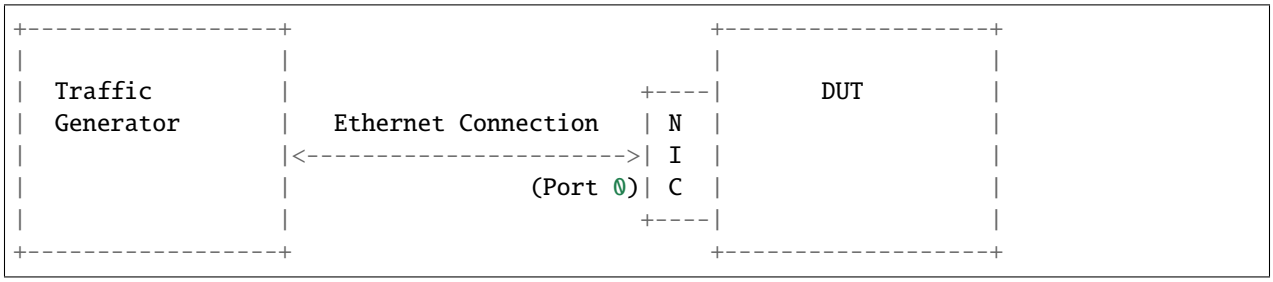
Users and Essential Skills

The reference solutions are targeted for a networking software development or performance analysis engineer who has in-depth networking knowledge, but does not know about Arm necessarily.

Mastering knowledge on certain user space open source networking projects, e.g., DPDK, VPP, ODP, will help gain deeper understanding of this guide and the reference solutions more easily.

1.2.2 Setup

The sample applications described in this guide require the following setup.



As shown, the Device Under Test (DUT) should have at least one NIC connected to the traffic generator on port 0. The user can use any traffic generator. The DUT is also used to download the solution repository and build the code. Cross compilation is not supported currently.

1.2.3 Tested Platforms

The sample applications are tested on the following platforms.

DUT

- Ampere Altra (Neoverse-N1)
  - Ubuntu 20.04.3 LTS (Focal Fossa)
  - Kernel 5.17.0-051700-generic

NIC

- Mellanox ConnectX-5
  - OFED driver: MLNX_OFED_LINUX-5.4-3.1.0.0
  - Firmware version: 16.30.1004 (MT_0000000013).

Note: To use Mellanox NIC, install OFED driver and update NIC firmware by following the guidance in *FAQ*.

1.2.4 Preparing the DUT

Requirements

The DUT needs to have a minimum hardware configuration as below.

- Processor: Minimum 1 GHz and 4 CPU cores
- Hard Drive: Minimum 32 GB
- Memory (RAM): Minimum 8 GB
- Network Interface Controller: Minimum 10G port connecting to Traffic Generator
- Connection to internet to download the source code and dependent packages

This documentation assumes the user has installed Ubuntu 20.04 (Focal) on the DUT.

- Admin (root) privileges are required to run the software and set up the DUT.
- Access to the internet is mandatory for downloading solution source code and installing all dependent packages and libraries.
- Scripts are provided to install the dependent packages and libraries.
- Mellanox OFED driver is installed and NIC firmware is updated.
- gcc 9.4.0 or newer version is required to compile the software.
- The provided scripts must be run in a bash shell.

The following utilities must be available on the DUT:

- git
- curl
- python

- python3

To configure Git, run:

```
git config --global user.email "you@example.com"
git config --global user.name "Your Name"
```

Follow the instructions provided in git-repo to install the `repo` tool manually.

Download Source Code

Create a new folder that will be the workspace, henceforth referred to as <nw_ds_workspace> in these instructions:

```
mkdir <nw_ds_workspace>
cd <nw_ds_workspace>
export NW_DS_RELEASE=refs/tags/NW-DS-2022.06.30
```

Note: Sometimes new features and additional bug fixes are made available in the git repositories, but are not tagged yet as part of a release. To pick up these latest changes, remove the `-b <release tag>` option from the `repo init` command below. However, please be aware that such untagged changes may not be formally verified and should be considered unstable until they are tagged in an official release.

To clone the repository, run the following commands:

```
repo init \
    -u https://git.gitlab.arm.com/arm-reference-solutions/arm-reference-solutions-
↪manifest.git \
    -b ${NW_DS_RELEASE} \
    -m dataplane-stack.xml
repo sync
```

Setup

This solution includes a `setup.sh` bash script responsible for the setup process.

The setup script:

- Installs and upgrades the required packages
- Configures platform level parameters required to run the applications

The affected packages and parameters can be found in `setup.sh`.

To set up the DUT:

```
cd <nw_ds_workspace>/dataplane-stack
sudo ./setup.sh
```

Build

This solution uses Makefile to build all the components.

The Makefile:

- Builds DPDK and the L3fwd sample application
- Builds VPP

To build Dataplane Stack, run the following on DUT:

```
cd <nw_ds_workspace>/dataplane-stack
make all
```

Note: The compilation might take some time to complete.

Reboot

After setting up DUT and building the software, reboot the DUT. This ensures the setup changes are reflected before running the sample applications.

Get NIC Information

Identify the interface and PCIe address of the NIC port attached to the traffic generator. `sudo ethtool --identify <interface name>` will help identify which NIC port is associated with a given interface name. `sudo lshw -c net -businfo` will identify the PCIe address for the interface.

For example, if `enP1p1s0f0` is attached to the traffic generator, then running `lshw -c net -businfo` will show the PCIe address as `0001:01:00.0`:

```
$ sudo lshw -c net -businfo
Bus info          Device      Class      Description
=========================================================
pci@0000:07:00.0  eth0        network    RTL8111/8168/8411 PCI Express Gigabit Ethernet␣
↪Controller
pci@0001:01:00.0  enP1p1s0f0  network    MT27800 Family [ConnectX-5]
pci@0001:01:00.1  enP1p1s0f1  network    MT27800 Family [ConnectX-5]
```

1.2.5 DPDK L3fwd

Bind NIC to Proper Linux Driver

For NICs that support bifurcated drivers, like Mellanox NICs, please skip this step.

For other NICs to be used by DPDK, the NIC needs to be bound to the appropriate driver. In practice, `vfio-pci` driver is sufficient. Before using `vfio-pci`, be sure to load the kernel module with `modprobe vfio-pci`. For more information, review DPDK's Linux Drivers Guide.

To bind the NIC to the appropriate driver, run:

```
cd <nw_ds_workspace>/dataplane-stack
sudo modprobe vfio-pci # ensure kernel module is loaded
sudo components/dpdk/usertools/dpdk-devbind.py -b vfio-pci <pcie_address>
```

For example, to bind PCIe address `0000:06:00.1` to `vfio-pci`:

```
sudo modprobe vfio-pci # ensure kernel module is loaded
sudo components/dpdk/usertools/dpdk-devbind.py -b vfio-pci 0000:06:00.1
```

Run

To run DPDK L3fwd application:

```
cd <nw_ds_workspace>/dataplane-stack
sudo components/dpdk/build/examples/dpdk-l3fwd -n 4 -l 2 -a <pcie_address> -- -P -p 0x1 -
↪-config='(0,0,2)'
```

For example, to run `dpdk-l3fwd` using `0001:01:00.0`:

```
cd <nw_ds_workspace>/dataplane-stack
sudo components/dpdk/build/examples/dpdk-l3fwd -n 4 -l 2 -a 0001:01:00.0 -- -P -p 0x1 --
↪config='(0,0,2)'
```

Test

For example, the typical output contains:

```
Initializing port 0 ... Creating queues: nb_rxq=1 nb_txq=1...
Address:98:03:9B:71:24:2E, Destination:02:00:00:00:00:00, Allocated mbuf pool on socket 0
LPM: Adding route 198.18.0.0 / 24 (0) [0001:01:00.0]
LPM: Adding route 2001:200:: / 64 (0) [0001:01:00.0]
txq=2,0,0
```

These logs show port 0 has MAC address `98:03:9B:71:24:2E` with PCIe address `0001:01:00.0` on the DUT. 1 IPv4 route matching the subnet `198.18.0.0/24` is added.

Configure the traffic generator to send packets to the NIC port, using the MAC and IP address displayed in the logs. In this example, use a destination MAC address of `98:03:9B:71:24:2E` and a destination IP of `198.18.0.21`. Then, `dpdk-l3fwd` will forward those packets out on port 0.

Stop

Stop the `dpdk-l3fwd` process with Control-C or `kill`. Next, if the NIC had been bound to a different Linux driver, rebind it to its original driver. Find the original driver by running `dpdk-devbind.py -s`, and notice the `unused=` part of the PCIe address.

For example, sample output from `dpdk-devbind.py -s` may look like:

```
cd <nw_ds_workspace>/dataplane-stack
sudo components/dpdk/usertools/dpdk-devbind.py -s
```

(continues on next page)

(continued from previous page)

```
Network devices using DPDK-compatible driver
============================================
0000:07:00.0 'Ethernet Controller XL710 for 40GbE QSFP+ 1583' drv=vfio-pci unused=i40e
...
```

In this example, bind `0000:07:00.0` to the `i40e` Linux driver using the following command

```
cd <nw_ds_workspace>/dataplane-stack
sudo components/dpdk/usertools/dpdk-devbind.py -b i40e 0000:07:00.0
```

1.2.6 VPP L3fwd

Note: Currently, the instructions provided in this section work with Mellanox NICs only.

Run

To run VPP:

```
cd <nw_ds_workspace>/dataplane-stack
sudo ./components/vpp/build-root/install-vpp-native/vpp/bin/vpp unix {interactive}
```

Note: It is possible that VPP may throw warnings and errors during initialization. These can be ignored safely.

Configure VPP with L3 interface and routes in VPP command prompt, note the interface name `enP1p1s0f0` below is obtained from above `lshw` command:

```
vpp# create interface rdma host-if enP1p1s0f0 name eth0
vpp# set interface mac address eth0 00:11:22:33:44:55
vpp# set interface ip address eth0 1.1.1.2/24
vpp# set interface state eth0 up
vpp# ip route add 198.18.0.0/24 via 1.1.1.1
vpp# set ip neighbor eth0 1.1.1.1 02:00:00:00:00:00
```

Test

After running the above command, configure the traffic generator to send packets to port 0 with a destination MAC address of `00:11:22:33:44:55` and an IP in the subnet `198.18.0.0/24`. `vpp` will forward those packets out on port 0.

Stop

To stop VPP, enter `quit` in VPP command line prompt:

```
vpp# quit
```

1.2.7 Changelog & Known Issues

To check newly added features, feature changes, and known issues in each of the releases, please refer to *CHANGELOG*.

1.3 User Guide

Welcome to the Dataplane Stack reference solution user guide. This guide provides the detailed guidelines to end users on how to download, build and execute the solution on Arm platforms. Guidelines on experimenting with various parameters that affect performance of the uses cases are described in detail. This guide is intended to describe complex and practical uses cases requiring complex test setup.

1.3.1 Users and Essential Skills

The reference solutions are targeted for a networking software development or performance analysis engineer who has in-depth networking knowledge, but does not know about Arm necessarily.

Using this guide requires in-depth knowledge on networking use cases. Mastering knowledge on certain user space open source networking projects, e.g., DPDK, VPP, ODP, will help gain deeper understanding of this guide and the reference solutions more easily.

1.3.2 Tested Platforms

The sample applications are tested on the following platforms.

DUT

- Ampere Altra (Neoverse-N1)
  - Ubuntu 20.04.3 LTS (Focal Fossa)
  - Kernel 5.17.0-051700-generic

NIC

- Mellanox ConnectX-5
  - OFED driver: MLNX_OFED_LINUX-5.4-3.1.0.0
  - Firmware version: 16.30.1004 (MT_0000000013).

Note: To use Mellanox NIC, install OFED driver and update NIC firmware by following the guidance in *FAQ*.

1.3.3 Preparing the DUT

The DUT is also used to download the solution repository and build the code. Cross compilation is not supported currently.

Requirements

The DUT needs to have a minimum hardware configuration as below.

- Processor: Minimum 1 GHz and 4 CPU cores
- Hard Drive: Minimum 32 GB
- Memory (RAM): Minimum 8 GB
- Network Interface Controller: Minimum 10G port connecting to Traffic Generator
- Ethernet connection to internet

This documentation assumes the user has installed Ubuntu 20.04 (Focal) on the DUT.

- Admin (root) privileges are required to run the software and set up the DUT.
- Access to the internet is mandatory for downloading solution source code and installing all dependent packages and libraries.
- Scripts are provided to install the dependent packages and libraries.
- Mellanox OFED driver is installed and NIC firmware is updated.
- gcc 9.4.0 or newer version is required to compile the software.
- The provided scripts must be run in a bash shell.

The following utilities must be available on the DUT:

- git
- curl
- python
- python3

To configure Git, run:

```
git config --global user.email "you@example.com"
git config --global user.name "Your Name"
```

Follow the instructions provided in git-repo to install the `repo` tool manually.

Download Source Code

Create a new folder that will be the workspace, henceforth referred to as `<nw_ds_workspace>` in these instructions:

```
mkdir <nw_ds_workspace>
cd <nw_ds_workspace>
export NW_DS_RELEASE=refs/tags/NW-DS-2022.06.30
```

Note: Sometimes new features and additional bug fixes will be made available in the git repositories and will not yet have been tagged as part of a release. To pick up these latest changes, remove the `-b <release tag>` option from the `repo init` command below. However, please be aware that such untagged changes have not yet been formally verified and should be considered unstable until they are tagged in an official release.

To clone the repository, run the following commands:

```
repo init \
    -u https://git.gitlab.arm.com/arm-reference-solutions/arm-reference-solutions-
↪manifest.git \
    -b ${NW_DS_RELEASE} \
    -m dataplane-stack.xml
repo sync
```

Setup

This solution includes a `setup.sh` Bash script responsible for the setup process.

The setup script:

- Installs and upgrades the following packages

  `net-tools build-essential manpages-dev libnuma-dev python python3-venv cmake meson pkg-config python3-pyelftools lshw`

- Configures platform level parameters required to run the applications

Specifically, it overwrites `GRUB_CMDLINE_LINUX` in `/etc/default/grub` to

- reserve one 1G hugepage (via `hugepagesz=1G hugepages=1`)
- reserve 512 2M hugepages (via `hugepagesz=2M hugepages=512`)
- set IOMMU into passthrough mode (via `iommu.passthrough=1`)
- isolate CPUs 2 and 3 from the Linux scheduler (via `isolcpus=2-3`)
- isolate CPUs 2 and 3 from processing RCU callbacks (via `rcu_nocbs=2-3`)
- set CPUs 2 and 3 to omit scheduling clock ticks (via `nohz_full=2-3`)
- disable cpufreq and cpuidle subsystems (via `cpufreq.off=1` and `cpuidle.off=1`)

To isolate more or different CPUs, edit `setup.sh` accordingly. To help speedup compilation, ensure that there are sufficient number of CPUs (at least 4 if possible) that are not isolated.

```
Hugepages help prevent TLB misses and are commonly used by networking
applications to manage memory. The larger the hugepage, the greater the
performance increase due to fewer TLB misses and fewer page table walks.

1GB hugepages are easiest to reserve at boot time, as reservation during
run time is likely to not be possible. Run time reservation would require
1GB of available contiguous memory, which typically is not available.
```

```
The combination of the kernel parameters mentioned above ensure that the
CPUs run only the desired application. They never process kernel RCU
callbacks, don't generate schedling ticks as there is only one process
```

(continues on next page)

(continued from previous page)

```
running on them, and are isolated from running any processes other than
the ones pinned to them (the desired application).
```

To set up the DUT:

```
cd <nw_ds_workspace>/dataplane-stack
sudo ./setup.sh
```

Build

This solution uses Makefile to build all the components.

The Makefile:

- Builds DPDK and the L3fwd sample applications
- Builds VPP

To build Dataplane Stack, run the following on DUT:

```
cd <nw_ds_workspace>/dataplane-stack
make all
```

It is also possible to compile the components individually by specifying `make dpdk` or `make vpp`. Run `make help` to view a list of all Makefile targets.

The above mentioned `make` commands can also be used to rebuild after modifying the code.

Reboot

After setting up DUT and building the software, reboot the DUT. This ensures the setup changes are reflected before running the sample applications.

Get NIC Interface

Next, identify the interface(s) on the NIC(s) connected to the traffic generator.

```
$ sudo ethtool --identify <interface>
will typically blink a light on the NIC to help identify the
physical port associated with the interface.
```

1.3.4 Porting/Integrating to another Arm platform

Although the solution is tested on limited hardware platforms, the solution might work just fine on other Arm platforms. However, such platforms should support ArmV8 architecture at least and should be supported by the underlying components.

1.3.5 Use Cases

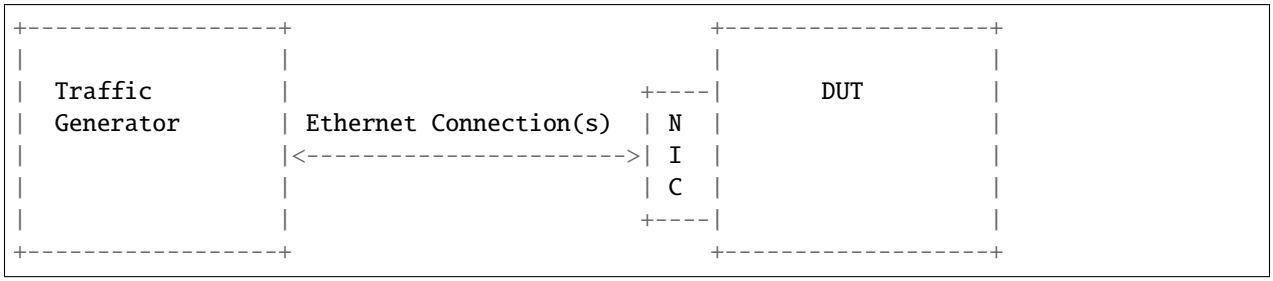
DPDK IPv4 L3fwd

Introduction

The `dpdk-l3fwd` sample application demonstrates the use of the hash, LPM and FIB based lookup methods provided in DPDK to implement packet forwarding using poll mode or event mode PMDs for packet I/O. The instructions provided in this guide do not cover all the features of this sample application. Users can refer to dpdk-l3fwd user guide to learn and experiment additional features.

Test Setup

This guide assumes the following setup:



As shown, the Device Under Test (DUT) should have at least one NIC port connected to the traffic generator. The user can use any traffic generator.

Get NIC PCIe Address

Identify the PCIe addresses of the NIC ports attached to the traffic generator. Once the interface is known, then `dpdk-devbind.py` can identify the matching PCIe address:

```
cd <nw_ds_workspace>/dataplane-stack
sudo components/dpdk/usertools/dpdk-devbind.py -s
```

The output may look like:

```
Network devices using kernel driver
===================================
0000:07:00.0 'RTL8111/8168/8411 PCI Express Gigabit Ethernet Controller 8168' if=enp7s0␣
↪drv=r8169 unused=vfio-pci *Active*
0001:01:00.0 'MT28800 Family [ConnectX-5 Ex] 1019' if=enP1p1s0f0 drv=mlx5_core␣
↪unused=vfio-pci
0001:01:00.1 'MT28800 Family [ConnectX-5 Ex] 1019' if=enP1p1s0f1 drv=mlx5_core␣
↪unused=vfio-pci
```

In this example output, if the interface `enP1p1s0f0` is connected to the traffic generator, then the corresponding PCIe address is `0001:01:00.0`.

Bind NIC to Proper Linux Driver

For NICs that support bifurcated drivers, like Mellanox NICs, please skip this step.

For other NICs to be used by DPDK, the NIC needs to be bound to the appropriate driver. In practice, the `vfio-pci` driver is sufficient. Before using `vfio-pci`, be sure to load the kernel module with `modprobe vfio-pci`. For more information, review DPDK's Linux Drivers Guide.

To bind the NIC to the appropriate driver, run:

```
cd <nw_ds_workspace>/dataplane-stack
sudo modprobe vfio-pci # ensure kernel module is loaded
sudo components/dpdk/usertools/dpdk-devbind.py -b vfio-pci <pcie_address>
```

For example, to bind PCIe address `0000:06:00.1` to `vfio-pci`:

```
sudo modprobe vfio-pci # ensure kernel module is loaded
sudo components/dpdk/usertools/dpdk-devbind.py -b vfio-pci 0000:06:00.1
```

Run

To run DPDK L3fwd application:

```
cd <nw_ds_workspace>/dataplane-stack
sudo components/dpdk/build/examples/dpdk-l3fwd [EAL Options] -- [L3fwd App Options]
```

Refer to DPDK documentation for supported EAL Options and L3fwd App Options.

For the configuration provided in `setup.sh` and a test setup with port 0 connected to traffic generator, use the following command to run the application:

```
cd <nw_ds_workspace>/dataplane-stack
sudo components/dpdk/build/examples/dpdk-l3fwd -n 4 -l 2 -a <pcie_address> -- -P -p 0x1 -
↪-config='(0,0,2)'
```

For example, to run `dpdk-l3fwd` using `0001:01:00.0`:

```
cd <nw_ds_workspace>/dataplane-stack
sudo components/dpdk/build/examples/dpdk-l3fwd -n 4 -l 2 -a 0001:01:00.0 -- -P -p 0x1 --
↪config='(0,0,2)'
```

Test

The typical output for the above command contains:

```
Initializing port 0 ... Creating queues: nb_rxq=1 nb_txq=1...
Address:98:03:9B:71:24:2E, Destination:02:00:00:00:00:00, Allocated mbuf pool on socket 0
LPM: Adding route 198.18.0.0 / 24 (0) [0001:01:00.0]
LPM: Adding route 2001:200:: / 64 (0) [0001:01:00.0]
txq=2,0,0
```

These logs show port 0 has MAC address `98:03:9B:71:24:2E` with PCIe address `0001:01:00.0` on the DUT. An IPv4 route matching the subnet `198.18.0.0/24` is added.

Configure the traffic generator to send packets to the NIC port, using the MAC and IP address displayed in the logs. In this example, use a destination MAC address of `98:03:9B:71:24:2E` and a destination IP of `198.18.10.21`. Then, `dpdk-l3fwd` will forward those packets out on port 0.

Stop

Stop the `dpdk-l3fwd` process with Control-C or `kill`. Next, if the NIC had been bound to a different Linux driver, rebind it to its original driver. Find the original driver by running `dpdk-devbind.py -s`, and notice the `unused=` part of the PCIe address.

For example, sample output from `dpdk-devbind.py -s` may look like:

```
cd <nw_ds_workspace>/dataplane-stack
sudo components/dpdk/usertools/dpdk-devbind.py -s

Network devices using DPDK-compatible driver
============================================
0000:07:00.0 'Ethernet Controller XL710 for 40GbE QSFP+ 1583' drv=vfio-pci unused=i40e
...
```

In this example, bind `0000:07:00.0` to the `i40e` Linux driver using the following command.

```
cd <nw_ds_workspace>/dataplane-stack
sudo components/dpdk/usertools/dpdk-devbind.py -b i40e 0000:07:00.0
```

Suggested Experiments

The example provided above covers a very simple use case of the DPDK L3fwd application. Users are encouraged to experiment with various options provided by the application.

The users are also encouraged to try the following options to understand the performance and scalability possible with Arm platforms.

- Number of RX/TX ring descriptors: This can affect the performance in multiple ways. For example, if the DUT is capable of storing the incoming packets in system cache, the incoming packets can trash the system cache, reducing the overall performance. To understand how these affect the performance, experiment by changing the number of descriptors. Change `RTE_TEST_RX_DESC_DEFAULT` and `RTE_TEST_TX_DESC_DEFAULT` in file `l3fwd.h` and recompile DPDK.
- `--config`: This parameter assigns the NIC RX queues to CPU cores. It is possible that a single queue might not be able to saturate a single CPU core. One can experiment by assigning multiple queues to a single core. For example, the option `--config='(0,0,1),(0,1,1)'` assigns the queues 0 and 1 of port 0 to lcore 1. Ensure that Receive Side Scaling (RSS) distributes the packets equally to all the enabled queues by sending multiple flows of traffic.
- CPU Scalability: Add more ports to DUT and run the application on more CPU cores to understand how the performance scales with the addition of CPU cores. Ensure that Receive Side Scaling (RSS) distributes the packets equally to all the enabled queues by sending multiple flows of traffic.
- Route Scalability: Add additional routes and multiple flows of traffic that exercise these routes. Additional routes can be added such that the accessed data size is more than the available L1, L2 or system cache size.

  To change forwarding rules, edit the global constants in:

  - `main.c`: edit the `ipv4_l3fwd_route_array` or `ipv6_l3fwd_route_array` to adjust default routes for FIB or LPM lookups.

DPDK in this solution is built with all the sample applications enabled. The users can run other sample applications by following the instructions in DPDK's Sample Applications User Guide.

Resources

1. DPDK Linux Getting Started Guide on DPDK Drivers
2. DPDK User Guide on dpdk-l3fwd
3. DPDK's dpdk-devbind.py documentation
4. DPDK Poll Mode Drivers
5. DPDK Event Mode
6. MLNX_OFED Software Download

VPP IPv4 L3fwd

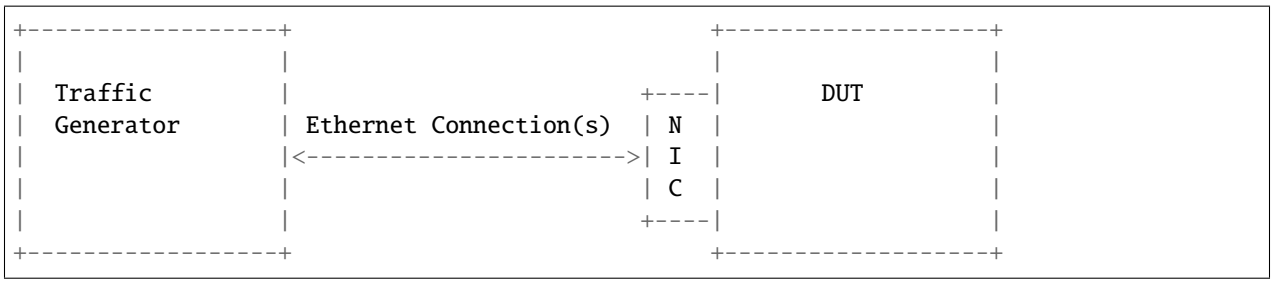
Introduction

VPP IPv4 L3fwd implements the typical routing function base on 32-bit IPv4 address. It forwards packets using Longest Prefix Match algorithm based on the mtrie forwarding table.

This guide explains in detail on how to use the VPP based IPv4 forwarding related use cases.

Test Setup

This guide assumes the following setup:



As shown, the Device Under Test (DUT) should have at least one NIC connected to the traffic generator. The user can use any traffic generator.

Run

Find out which interface is connected with traffic generator, `sudo ethtool --identify <interface>` will typically blink a light on the NIC to help identify the physical port associated with the interface.

Get interface name `enP1p1s0f0` from `lshw` command:

```
$ sudo lshw -c net -businfo
Bus info          Device      Class      Description
========================================================
pci@0000:07:00.0  eth0        network    RTL8111/8168/8411 PCI Express Gigabit Ethernet␣
↪Controller
```

(continues on next page)

(continued from previous page)

```
pci@0001:01:00.0  enP1p1s0f0  network    MT27800 Family [ConnectX-5]
pci@0001:01:00.1  enP1p1s0f1  network    MT27800 Family [ConnectX-5]
```

Start vpp with `interactive` command line arguments, and for more argument parameters, refer to VPP configuration reference:

```
cd <nw_ds_workspace>/dataplane-stack
sudo ./components/vpp/build-root/install-vpp-native/vpp/bin/vpp unix {interactive}
```

Typically we configure VPP with 1 packet flow and 10k packet flows. Both cases start with following common VPP command configuration:

```
# Same for different packet flow setups
vpp# create interface rdma host-if enP1p1s0f0 name eth0
vpp# set interface ip address eth0 1.1.1.2/30
vpp# set ip neighbor eth0 1.1.1.1 02:00:00:00:00:00
vpp# set interface state eth0 up
```

For more detailed usage on above commands, refer to following links,

- VPP rdma cli reference
- VPP set interface ip address reference
- VPP ip neighbor cli reference
- VPP set interface state reference

For 1 packet flow case:

```
# Add only one route entry here
vpp# ip route add 10.0.0.0/32 count 1 via 1.1.1.1 eth0
```

For 10k packet flows case:

```
# Add 10k route entries here
vpp# ip route add 10.0.0.0/32 count 10000 via 1.1.1.1 eth0
```

Refer to VPP ip route reference for more `ip route` options. To explore more on VPP's accepted commands, please review VPP cli reference.

Test

To display the current set of routes, use the command `show ip fib`. Here is a sample output for added routes:

```
vpp# show ip fib 10.0.0.1/32
ipv4-VRF:0, fib_index:0, flow hash:[src dst sport dport proto flowlabel ] epoch:0␣
↪flags:none locks:[adjacency:1, default-route:1, ]
10.0.0.1/32 fib:0 index:17 locks:2
  CLI refs:1 src-flags:added,contributing,active,
    path-list:[22] locks:20000 flags:shared,popular, uPRF-list:22 len:1 itfs:[1, ]
      path:[26] pl-index:22 ip4 weight=1 pref=0 attached-nexthop:  oper-flags:resolved,
        1.1.1.1 eth0
      [@0]: ipv4 via 1.1.1.1 eth0: mtu:9000 next:3 flags:[] 020000000000098039b6b62680800
```

(continues on next page)

(continued from previous page)

```
 forwarding:   unicast-ip4-chain
  [@0]: dpo-load-balance: [proto:ip4 index:19 buckets:1 uRPF:22 to:[0:0]]
    [0] [@5]: ipv4 via 1.1.1.1 eth0: mtu:9000 next:3 flags:[]␣
↪020000000000098039b6b62680800
```

Check the packet flow with IP destination 10.0.0.0/32, the next hop is resolved, packets will be forwarded to 1.1.1.1 via eth0.

To configure traffic generator for the destination MAC address, get the VPP interface MAC address via `show hardware-interfaces verbose`:

```
vpp# show hardware-interfaces verbose
              Name                Idx   Link  Hardware
eth0                               1     up   eth0
  Link speed: 40 Gbps
  RX Queues:
    queue thread         mode
    0     vpp_wk_0 (1)   polling
    1     vpp_wk_0 (1)   polling
  Ethernet address 02:fe:40:5e:73:e3
  netdev enP1p1s0f0 pci-addr 0001:01:00.0
```

For 1 packet flow case, configure your traffic generator to send packets with a destination MAC address of `02:fe:40:5e:73:e3` and an IP in the subnet `10.0.0.0/32`, then `vpp` will forward those packets out on eth0.

For 10000 packet flows case, configure your traffic generator to send packets with a destination MAC address of `02:fe:40:5e:73:e3` and an increasing destination IP address, increasing by 10000 times, starting from `10.0.0.0/32` with an increment of 1 in each increase, then `vpp` will forward those packets out on eth0.

Stop

To stop VPP, enter `quit` in VPP command line prompt:

```
vpp# quit
```

Suggested Experiments

Add another interface

To add another interface in VPP, for example `enP1p1s0f1` in *lshw output sample*.

Create another interface in VPP command line with different interface name:

```
vpp# create interface rdma host-if enP1p1s0f1 name eth1
vpp# set interface ip address eth1 3.3.3.2/30
vpp# set interface state eth1 up
```

New routes can be add to this interface afterwards:

```
vpp# ip route add 30.0.0.0/32 count 1 via 3.3.3.3 eth1
```

Start with configuration file

To start vpp with startup configuration file, refer to VPP starts with configuration file

Create a very simple startup.conf file:

```
cd <nw_ds_workspace>/dataplane-stack
cat <<EOF > startup.conf
unix {
        interactive
}
EOF
```

Instruct VPP to load this file with the -c option. For example:

```
sudo ./components/vpp/build-root/install-vpp-native/vpp/bin/vpp -c startup.conf
```

Add CPU cores to worker thread

To add more CPU cores for VPP data plane, configure vpp with more workers for better performance, refer to VPP configuration cpu section

```
cpu {
        main-core 1
        corelist-workers 2-3,18-19
}
```

Change number of descriptors in receive ring and transmit ring

To change number of descriptors in receive ring and transmit ring, increasing or reducing number can impact performance. Default is 1024, refer to VPP configuration num-rx-desc num-tx-desc

```
dpdk {
        dev default {
                num-rx-desc 512
                num-tx-desc 512
        }
}
```

Use faster DPDK vector PMDs

Disable multi-segment buffers, disable UDP / TCP TX checksum offload, needed to use faster DPDK vector PMDs, improves performance but disables Jumbo MTU support, refer to VPP configuration no-multi-seg

```
dpdk {
        no-multi-seg
        no-tx-checksum-offload
}
```

Use other types of device drivers

Besides Mellanox ConnectX-5, VPP supports NICs from other vendors as well. VPP is integrated with NICs using the following 2 methods:

- VPP native device drivers
- VPP dpdk device driver configuration

Resources

1. VPP configuration reference
2. VPP rdma cli reference
3. VPP set interface ip address reference
4. VPP ip neighbor cli reference
5. VPP set interface state reference
6. VPP ip route reference
7. VPP cli reference

This chapter provides the design specification for a user case/feature provided by the software. Each use case design specification may include information as:

- Explain the use case.
- Show in diagram the components involved.
- Show a diagram outlining any important data/control flows through the components.
- Explain any test steps that exist for the use case.

And the key components constituting the solution could also be described in details with information as:

- Explain how the solution is integrated with which components and which tools are involved (e.g.: DPDK, VPP)
- Show in diagram the components/projects/tools involved and its dependencies.

1.4 Solution Design

1.4.1 IPv4 - Based on DPDK L3fwd

Use Cases

Explain the use case.

Composition Block Diagram

Show in diagram the components involved.

Flow Diagram

Show a diagram outlining any important data/control flows through the components.

Command Line

Explain any test steps that exist for the use case.

Integration

Explain how the solution is integrated and which tools are involved.

Key Components

Show in diagram the components/projects/tools involved and its dependencies.

1.4.2 IPv4 - Based on VPP

Use Cases

Explain the use case.

Composition Block Diagram

Show in diagram the components involved.

Flow Diagram

Show a diagram outlining any important data/control flows through the components.

Command Line

Explain any test steps that exist for the use case.

Integration

Explain how the solution is integrated and which tools are involved.

Key Components

Show in diagram the components/projects/tools involved and its dependencies.

1.5 License

1.5.1 Apache-2.0

The software is provided under the Apache-2.0 license (below).

(continues on next page)

(continued from previous page)

```
   (an example is provided in the Appendix below).

   "Derivative Works" shall mean any work, whether in Source or Object
   form, that is based on (or derived from) the Work and for which the
   editorial revisions, annotations, elaborations, or other modifications
   represent, as a whole, an original work of authorship. For the purposes
   of this License, Derivative Works shall not include works that remain
   separable from, or merely link (or bind by name) to the interfaces of,
   the Work and Derivative Works thereof.

   "Contribution" shall mean any work of authorship, including
   the original version of the Work and any modifications or additions
   to that Work or Derivative Works thereof, that is intentionally
   submitted to Licensor for inclusion in the Work by the copyright owner
   or by an individual or Legal Entity authorized to submit on behalf of
   the copyright owner. For the purposes of this definition, "submitted"
   means any form of electronic, verbal, or written communication sent
   to the Licensor or its representatives, including but not limited to
   communication on electronic mailing lists, source code control systems,
   and issue tracking systems that are managed by, or on behalf of, the
   Licensor for the purpose of discussing and improving the Work, but
   excluding communication that is conspicuously marked or otherwise
   designated in writing by the copyright owner as "Not a Contribution."

   "Contributor" shall mean Licensor and any individual or Legal Entity
   on behalf of whom a Contribution has been received by Licensor and
   subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of
   this License, each Contributor hereby grants to You a perpetual,
   worldwide, non-exclusive, no-charge, royalty-free, irrevocable
   copyright license to reproduce, prepare Derivative Works of,
   publicly display, publicly perform, sublicense, and distribute the
   Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of
   this License, each Contributor hereby grants to You a perpetual,
   worldwide, non-exclusive, no-charge, royalty-free, irrevocable
   (except as stated in this section) patent license to make, have made,
   use, offer to sell, sell, import, and otherwise transfer the Work,
   where such license applies only to those patent claims licensable
   by such Contributor that are necessarily infringed by their
   Contribution(s) alone or by combination of their Contribution(s)
   with the Work to which such Contribution(s) was submitted. If You
   institute patent litigation against any entity (including a
   cross-claim or counterclaim in a lawsuit) alleging that the Work
   or a Contribution incorporated within the Work constitutes direct
   or contributory patent infringement, then any patent licenses
   granted to You under this License for that Work shall terminate
   as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the
```

(continues on next page)

(continued from previous page)

```
   Work or Derivative Works thereof in any medium, with or without
   modifications, and in Source or Object form, provided that You
   meet the following conditions:

   (a) You must give any other recipients of the Work or
       Derivative Works a copy of this License; and

   (b) You must cause any modified files to carry prominent notices
       stating that You changed the files; and

   (c) You must retain, in the Source form of any Derivative Works
       that You distribute, all copyright, patent, trademark, and
       attribution notices from the Source form of the Work,
       excluding those notices that do not pertain to any part of
       the Derivative Works; and

   (d) If the Work includes a "NOTICE" text file as part of its
       distribution, then any Derivative Works that You distribute must
       include a readable copy of the attribution notices contained
       within such NOTICE file, excluding those notices that do not
       pertain to any part of the Derivative Works, in at least one
       of the following places: within a NOTICE text file distributed
       as part of the Derivative Works; within the Source form or
       documentation, if provided along with the Derivative Works; or,
       within a display generated by the Derivative Works, if and
       wherever such third-party notices normally appear. The contents
       of the NOTICE file are for informational purposes only and
       do not modify the License. You may add Your own attribution
       notices within Derivative Works that You distribute, alongside
       or as an addendum to the NOTICE text from the Work, provided
       that such additional attribution notices cannot be construed
       as modifying the License.

   You may add Your own copyright statement to Your modifications and
   may provide additional or different license terms and conditions
   for use, reproduction, or distribution of Your modifications, or
   for any such Derivative Works as a whole, provided Your use,
   reproduction, and distribution of the Work otherwise complies with
   the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
   any Contribution intentionally submitted for inclusion in the Work
   by You to the Licensor shall be under the terms and conditions of
   this License, without any additional terms or conditions.
   Notwithstanding the above, nothing herein shall supersede or modify
   the terms of any separate license agreement you may have executed
   with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
   names, trademarks, service marks, or product names of the Licensor,
   except as required for reasonable and customary use in describing the
   origin of the Work and reproducing the content of the NOTICE file.
```

(continues on next page)

(continued from previous page)

```
7. Disclaimer of Warranty. Unless required by applicable law or
   agreed to in writing, Licensor provides the Work (and each
   Contributor provides its Contributions) on an "AS IS" BASIS,
   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
   implied, including, without limitation, any warranties or conditions
   of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
   PARTICULAR PURPOSE. You are solely responsible for determining the
   appropriateness of using or redistributing the Work and assume any
   risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
   whether in tort (including negligence), contract, or otherwise,
   unless required by applicable law (such as deliberate and grossly
   negligent acts) or agreed to in writing, shall any Contributor be
   liable to You for damages, including any direct, indirect, special,
   incidental, or consequential damages of any character arising as a
   result of this License or out of the use or inability to use the
   Work (including but not limited to damages for loss of goodwill,
   work stoppage, computer failure or malfunction, or any and all
   other commercial damages or losses), even if such Contributor
   has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
   the Work or Derivative Works thereof, You may choose to offer,
   and charge a fee for, acceptance of support, warranty, indemnity,
   or other liability obligations and/or rights consistent with this
   License. However, in accepting such obligations, You may act only
   on Your own behalf and on Your sole responsibility, not on behalf
   of any other Contributor, and only if You agree to indemnify,
   defend, and hold each Contributor harmless for any liability
   incurred by, or claims asserted against, such Contributor by reason
   of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

   To apply the Apache License to your work, attach the following
   boilerplate notice, with the fields enclosed by brackets "[]"
   replaced with your own identifying information. (Don't include
   the brackets!)  The text should be enclosed in the appropriate
   comment syntax for the file format. We also recommend that a
   file or class name and description of purpose be included on the
   same "printed page" as the copyright notice for easier
   identification within third-party archives.

Copyright 2022 Arm, Limited.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at
```

(continues on next page)

(continued from previous page)

```
   http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

1.5.2 SPDX Identifiers

In the software, individual files contain the SPDX identifiers instead of the full license text. Normally each file uses first line of the file to be SPDX tag. In case of *#!* scripts, SPDX tag can be placed in 2nd line of the file.

For example, to label a file as subject to the Apache-2.0 license, the following text would be used:

```
// SPDX-License-Identifier: Apache-2.0
```

or

```
#!/usr/bin/env bash
# SPDX-License-Identifier: Apache-2.0
```

This enables machine processing of license information based on the SPDX License Identifiers that are here available: http://spdx.org/licenses/

1.5.3 License for patches

The software also bundles patch files, which are applied to the sources of the various packages. Those patches are not covered by the license of this software. Instead, they are covered by the license of the software to which the patches are applied.

When said software is available under multiple licenses, the patches in this software are only provided under the publicly accessible licenses.

1.5.4 Exceptions

Some files in this software contain a different license statement. Those files are licensed under the license contained in the file itself.

The exceptional files and their licenses are recorded as below.

License	file
tools/check-git-log.sh	BSD-3-Clause
tools/git-log-fixes.sh	BSD-3-Clause

1.6 Changelog & Release Notes

1.6.1 Dataplane Stack v22.06

New Features

- Integrated and enabled DPDK L3Fwd sample application
- Integrated and enabled IPv4 forwarding with VPP

Changed

Initial version.

Tested platforms:

- DUT
  - Ampere Altra (Neoverse-N1)
    - Ubuntu 20.04.3 LTS (Focal Fossa)
    - Kernel 5.17.0-051700-generic
- NIC
  - Mellanox ConnectX-5
    - OFED driver: MLNX_OFED_LINUX-5.4-3.1.0.0
    - Firmware Version 16.30.1004 (MT_0000000013)

Main software components versions:

- DPDK v21.11
- VPP v22.02

Documentation:

- Initial documentation with README, overview, quickstart guide, user guide and FAQ.

Tools:

- Initialized QA checks to validate:
  - Sphinx based documentation generation
  - Dependent component compilation
  - Commit log format check
  - License-Identifier check

Limitations

- Users of this software stack must consider safety and security implications according to their own usage goals.
- Does not provide native traffic generator.

Resolved Issues

None.

Known Issues

None.

1.7 FAQ

1. Q: Is there any existing open source project with similar goals or with which this project may conflict or overlap? If yes, please identify the project and explain why we should create the ARM open source project anyway?

   A: Yes, there are some open source projects serve similar purpose, e.g., OPNFV. However, compared with the software we would like to create, OPNFV has a much larger scope for us to cover. Morever, we would like to simply the reference solutions, focus on combining some currently being contributed networking projects, e.g., DPDK, VPP, ODP, and provide reference solutions aligned with existing marketing requirements.

   Inside the project, we will apply some optimizations applicable on Arm platform only, to achieve better performance, and those architecture specific optimizations will not be likely accepted by upstream community.

2. Q: Could you please provide the project description of functionality or purpose?

   A: The network functions this software provided serves multiple purposes of,

   1. Showcase the integration of various components and act as poof of concept to all stakeholders.
   2. Allow for performance analysis/optimization with a solution that is close to customers' production deployment.
   3. Provide customers with a out-of-the-box reference design for rapid design modeling.

3. Q: What's "Fixed Virtual Platforms" and "FPGA Emulator"? And what are they used for?

   A: "Fixed Virtual Platforms" and "FPGA Emulator" are some pre-silicon simulation/emulation platforms, which could be used to develop and validate software for CPU features in early stage. Running the software on these platforms could introduce benefits on,

   1. External CPU/IP dimensioning according to different customer use-case
   2. Internal microarchitecture profiling and tuning using networking workload during CPU design phase

4. Q: Please explain dataplane and controlplane, and their differences?

   A: The terms "control plane" and "data plane" are all about the separation of responsibilities within a networking system. The two most commonly referenced compositions in networking are the control plane and the data plane. The control plane is the part of a network that controls how data is forwarded, while the data plane is the actual forwarding process.

   - **Control plane**: refers to the all functions and processes that determine which path to use to send the packet or frame. Control plane is responsible for populating the routing table, drawing network topology,

forwarding table and hence enabling the data plane functions. Control plane is the process of learning what we will do before sending the packet or frame.

- **Data plane**: refers to all the functions and processes that forward packets/frames from one interface to another based on control plane logic. Routing table, forwarding table and the routing logic constitute the data plane function. Data plane packet goes through the router, and incoming and outgoing of frames are done based on control plane logic. Data plane is moving the actual packets based on what we learned from control plane.

5. Q: What's user space Network?

   A: User space network software takes exclusive control of a network adapter, implements the whole NIC driver and develops packet processing framework completely in user space.

   There are several primary reasons to move the networking functionalities from the kernel to user space:

   - Reduce the number of context switches required to process packet data, as each syscall causes a context switch, which takes up time and resources.
   - Reduce the amount of software on the stack. Linux kernel provides abstractions, which are designed for general purpose and could be quite complicated in implementing packet processing. Customized implementation could remove unnecessary abstractions, simplify the logic, and improve performance.
   - User space drivers are easier to develop and debug than kernel drivers. Developing networking functions and getting it merge in mainline kernel would take considerable time and effort. Moreover, the function release would be bounded by Linux's release schedule. Finally, bugs in the source code may cause the kernel to crash.

   The user space networking ecosystems has really matured since some user space networking projects are open sourced, e.g., DPDK, VPP, Snort. A whole ecosystem of technologies developed based on user space network software has emerged.

6. Q: How to install Mellanox ConnectX-5 OFED driver and update NIC firmware?

   A: To use Mellanox NIC, firstly install the OFED driver MLNX_OFED, and then update NIC Firmware.

   The key steps are:

   - Download the OFED driver
   - Install OFED driver:

   ```
   $ sudo mount -o ro,loop MLNX_OFED_LINUX-5.4-3.1.0.0-ubuntu20.04-aarch64.iso /mnt
   $ sudo /mnt/mlnxofedinstall --upstream-libs --dpdk
   ```

   - Update firmware after OFED installation:

   ```
   $ wget https://www.mellanox.com/downloads/MFT/mft-4.20.0-34-arm64-deb.tgz
   $ tar xvf mft-4.20.0-34-arm64-deb.tgz
   $ cd mft-4.20.0-34-arm64-deb/
   $ sudo ./install.sh
   $ sudo mst start
   $ sudo mlxfwmanager --online -u -d <device PCIe address>
   ```