2. Install DPDK and SPP

Before using SPP, you need to install DPDK. In this document, briefly described how to install and setup DPDK. Refer to DPDK documentation for more details. For Linux, see Getting Started Guide for Linux .

2.1. Required Packages

Installing packages for DPDK and SPP is almost the on Ubunu and CentOS, but names are different for some packages.

2.1.1. Ubuntu

To compile DPDK, required to install following packages.

```
$ sudo apt install libnuma-dev \
  libarchive-dev \
  build-essential
```

You also need to install linux-headers of your kernel version.

```
$ sudo apt install linux-headers-$(uname -r)
```

Some of secondary processes depend on external libraries and you failed to compile SPP without them.

SPP provides libpcap-based PMD for dumping packet to a file or retrieve it from the file. `spp_nfv` and `spp_pcap` use `libpcap-dev` for packet capture. `spp_pcap` uses `liblz4-dev` and `liblz4-tool` to compress PCAP file.

```
$ sudo apt install libpcap-dev \
  liblz4-dev \
  liblz4-tool
```

`text2pcap` is also required for creating pcap file which is included in `wireshark`.

```
$ sudo apt install wireshark
```

2.1.2. CentOS

Before installing packages for DPDK, you should add IUS Community repositories with yum command.

```
$ sudo yum install https://centos7.iuscommunity.org/ius-release.rpm
```

To compile DPDK, required to install following packages.

```
$ sudo yum install numactl-devel \
  libarchive-devel \
  kernel-headers \
  kernel-devel
```

Some of secondary processes depend on external libraries and you failed to compile SPP without them.

SPP provides libpcap-based PMD for dumping packet to a file or retrieve it from the file. `spp_nfv` and `spp_pcap` use `libpcap-dev` for packet capture. `spp_pcap` uses `liblz4-dev` and `liblz4-tool` to compress PCAP file. `text2pcap` is also required for creating pcap file which is included in `wireshark`.

```
$ sudo apt install libpcap-dev \
  libpcap \
  libpcap-devel \
  lz4 \
  lz4-devel \
  wireshark \
  wireshark-devel \
  libX11-devel
```

2.2. DPDK

Clone repository and compile DPDK in any directory.

```
$ cd /path/to/any
$ git clone http://dpdk.org/git/dpdk
```

Installing on Ubuntu and CentOS are almost the same, but packages are different.

PCAP is disabled by default in DPDK configuration. `CONFIG_RTE_LIBRTE_PMD_PCAP` and `CONFIG_RTE_PORT_PCAP` defined in config file `common_base` should be changed to `y` to enable PCAP.

```
# dpdk/config/common_base
CONFIG_RTE_LIBRTE_PMD_PCAP=y
...
CONFIG_RTE_PORT_PCAP=y
```

Compile DPDK with target environment.

```
$ cd dpdk
$ export RTE_SDK=$(pwd)
$ export RTE_TARGET=x86_64-native-linuxapp-gcc  # depends on your env
$ make install T=$RTE_TARGET
```

PCAP is disabled by default in DPDK configuration. `CONFIG_RTE_LIBRTE_PMD_PCAP` and `CONFIG_RTE_PORT_PCAP` defined in config file `common_base` should be changed to `y` to enable PCAP.

```
# dpdk/config/common_base
CONFIG_RTE_LIBRTE_PMD_PCAP=y
...
CONFIG_RTE_PORT_PCAP=y
```

Compile DPDK with target environment.

```
$ cd dpdk
$ export RTE_SDK=$(pwd)
$ export RTE_TARGET=x86_64-native-linuxapp-gcc  # depends on your env
$ make install T=$RTE_TARGET
```

2.3. Pyhton

Python3 and pip3 are also required if not installed.

```
# Ubuntu
$ sudo apt install python3 \
  python3-pip
```

For CentOS, you need to specify minor version of python3. Here is an example of installing python3.6.

```
# CentOS
$ sudo yum install python36 \
  python36-pip
```

SPP provides `requirements.txt` for installing required packages of Python3. You might fail to run `pip3` without sudo on some environments.

```
$ pip3 install -r requirements.txt
```

For some environments, `pip3` might install packages under your home directory `$HOME/.local/bin` and you should add it to `$PATH` environment variable.

2.4. SPP

Clone SPP repository and compile it in any directory.

```
$ cd /path/to/any
$ git clone http://dpdk.org/git/apps/spp
$ cd spp
$ make  # Confirm that $RTE_SDK and $RTE_TARGET are set
```

If you use `spp_mirror` in deep copy mode, you should comment out the definition of copy mode in Makefile of `spp_mirror` before. It is for copying full payload into a new mbuf. Default mode is shallow copy.

```
# src/mirror/Makefile
#CFLAGS += -Dspp_mirror_SHALLOWCOPY
```

Note

Before run make command, you might need to consider if using deep copy for cloning packets in `spp_mirror`. Comparing with shallow copy, it clones entire packet payload into a new mbuf and it is modifiable, but lower performance. Which of copy mode should be chosen depends on your usage.

2.5. Binding Network Ports to DPDK

Network ports must be bound to DPDK with a UIO (Userspace IO) driver. UIO driver is for mapping device memory to userspace and registering interrupts.

2.5.1. UIO Drivers

You usually use the standard `uio_pci_generic` for many use cases or `vfio-pci` for more robust and secure cases. Both of drivers are included by default in modern Linux kernel.

```
# Activate uio_pci_generic
$ sudo modprobe uio_pci_generic

# or vfio-pci
$ sudo modprobe vfio-pci
```

You can also use kmod included in DPDK instead of `uio_pci_generic` or `vfio-pci`.

```
$ sudo modprobe uio
$ sudo insmod kmod/igb_uio.ko
```

2.5.2. Binding Network Ports

Once UIO driver is activated, bind network ports with the driver. DPDK provides `usertools/dpdk-devbind.py` for managing devices.

Find ports for binding to DPDK by running the tool with `-s` option.

```
$ $RTE_SDK/usertools/dpdk-devbind.py --status

Network devices using DPDK-compatible driver
============================================
<none>

Network devices using kernel driver
===================================
0000:29:00.0 '82571EB ... 10bc' if=enp41s0f0 drv=e1000e unused=
0000:29:00.1 '82571EB ... 10bc' if=enp41s0f1 drv=e1000e unused=
0000:2a:00.0 '82571EB ... 10bc' if=enp42s0f0 drv=e1000e unused=
0000:2a:00.1 '82571EB ... 10bc' if=enp42s0f1 drv=e1000e unused=

Other Network devices
=====================
<none>
....
```

You can find network ports are bound to kernel driver and not to DPDK. To bind a port to DPDK, run `dpdk-devbind.py` with specifying a driver and a device ID. Device ID is a PCI address of the device or more friendly style like `eth0` found by `ifconfig` or `ip` command..

```
# Bind a port with 2a:00.0 (PCI address)
./usertools/dpdk-devbind.py --bind=uio_pci_generic 2a:00.0

# or eth0
./usertools/dpdk-devbind.py --bind=uio_pci_generic eth0
```

After binding two ports, you can find it is under the DPDK driver and cannot find it by using `ifconfig` or `ip`.

```
$ $RTE_SDK/usertools/dpdk-devbind.py -s

Network devices using DPDK-compatible driver
============================================
0000:2a:00.0 '82571EB ... 10bc' drv=uio_pci_generic unused=vfio-pci
0000:2a:00.1 '82571EB ... 10bc' drv=uio_pci_generic unused=vfio-pci

Network devices using kernel driver
===================================
0000:29:00.0 '...' if=enp41s0f0 drv=e1000e unused=vfio-pci,uio_pci_generic
0000:29:00.1 '...' if=enp41s0f1 drv=e1000e unused=vfio-pci,uio_pci_generic

Other Network devices
=====================
<none>
....
```

2.6. Confirm DPDK is setup properly

You can confirm if you are ready to use DPDK by running DPDK's sample application. `l2fwd` is good choice to confirm it before SPP because it is very similar to SPP's worker process for forwarding.

```
$ cd $RTE_SDK/examples/l2fwd
$ make
  CC main.o
  LD l2fwd
  INSTALL-APP l2fwd
  INSTALL-MAP l2fwd.map
```

In this case, run this application simply with just two options while DPDK has many kinds of options.

- `-l`: core list
- `-p`: port mask

```
$ sudo ./build/app/l2fwd \
  -l 1-2 \
  -- -p 0x3
```

It must be separated with `--` to specify which option is for EAL or application. Refer to L2 Forwarding Sample Application for more details.

2.7. Build Documentation

This documentation is able to be built as HTML and PDF formats from make command. Before compiling the documentation, you need to install some of packages required to compile.

For HTML documentation, install sphinx and additional theme.

```
$ pip3 install sphinx \
  sphinx-rtd-theme
```

For PDF, inkscape and latex packages are required.

```
# Ubuntu
$ sudo apt install inkscape \
  texlive-latex-extra \
  texlive-latex-recommended
```

```
# CentOS
$ sudo yum install inkscape \
  texlive-latex
```

You might also need to install `latexmk` in addition to if you use Ubuntu 18.04 LTS.

```
$ sudo apt install latexmk
```

HTML documentation is compiled by running make with `doc-html`. This command launch sphinx for compiling HTML documents. Compiled HTML files are created in `docs/guides/_build/html/` and You can find the top page `index.html` in the directory.

```
$ make doc-html
```

PDF documentation is compiled with `doc-pdf` which runs latex for. Compiled PDF file is created as `docs/guides/_build/html/SoftPatchPanel.pdf`.

```
$ make doc-pdf
```

You can also compile both of HTML and PDF documentations with `doc` or `doc-all`.

```
$ make doc
# or
$ make doc-all
```

Note

For CentOS, compilation PDF document is not supported.