BKM to Enable Linux Sample XDP Programs and Run Developed XDP Module

By: Rohan Tabish

Mentor: Muhammad Asim Jamshed

Manager: Ripan Das

Group: Data Center Group

# Introduction:

This documents lists the steps to enable express data path (XDP) [1] and how one can test sample Berkley Packet Filter (bpf) programs [2]. The main idea behind XDP is that it allows you to process the packets directly in the userspace without the need of passing it through the traditional IP stack in the kernel as shown in Figure 1 below. This enables fast processing by avoiding expensive copies. The same goal can be achieved using the Intel’s Data Plane Development Kit (DPDK). However, DPDK stack is not part of the linux kernel. Moreover, it requires support for hugepages. Table 2 below summarizes the requirements of the DPDK and XDP stack.

Table 1: Requirement for DPDK Vs XDP

|  |  |  |
| --- | --- | --- |
| **Requirements/Features** | **DPDK** | **XDP** |
| Installation of Extra SW to enable | √ | × |
| Dedicated HW | √ | × |
| Huge Page Installation | √ | × |
| Userspace packet processing | √ | √ |

The aforementioned goal is achieved by installing the bpf kernel probes into the network interface card (NIC) which maps packet descriptors of the NIC into the userspace memory. This allows us to process packets faster when compared to the traditional socket buffer (SKB) path where expensive copies between different layers of the IP stack in kernel as well as copying the final copy of packets to the userspace application kills the performance. The support for the XDP has been incorporated in the Linux Kernel.

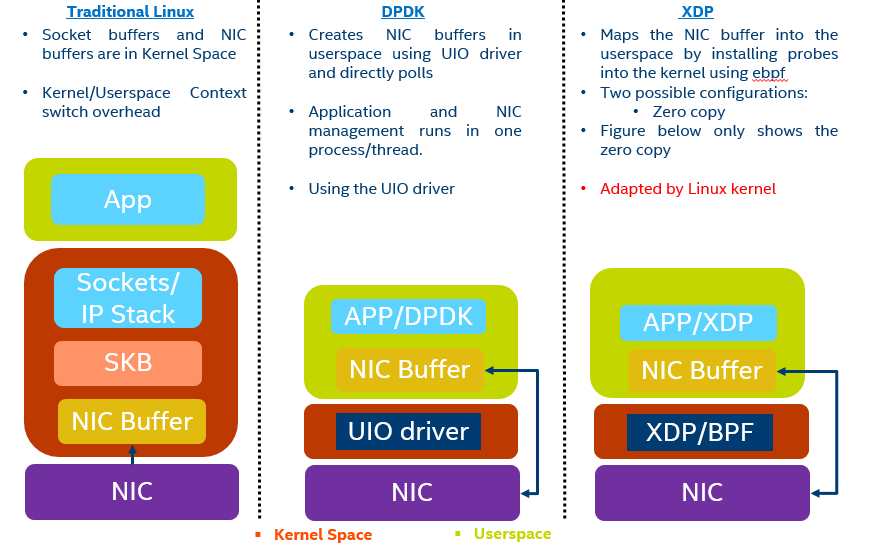


Figure 1: Comparison of Linux Stack Vs DPDK Vs XDP

For more information please follow [3]. The rest of the document is arranged as follows:

1. The first part of the document makes you familiar with the XDP module and shows how to run the sample applications and install the bpf libraries.
2. The second part of the document shows how to run the simple application that is based off of xdpsock\_user.c file.
3. The third part of the document shows how to install/run and benchmark the application using the Moongen simulator.

# Setting up the environment for XDP, Compiling and Testing:

The following list of steps can used to get started with the XDP.

1. The following setup was tested with the i40e driver. The firmware version was 5.0.5 and the card version 2.8.10-k. Please make sure that your NIC card has updated version. You can check of the NIC card using the following command. Two machines were directly connected with each other using a network cable. Both machines had i40e with the same firmware.

|  |
| --- |
| root@JF300-11A072T:/home/rohan# ethtool -i ens1f1  driver: i40e  version: 2.8.10-k  firmware-version: 5.05 0x800029eb 1.1313.0  expansion-rom-version:  bus-info: 0000:18:00.1  supports-statistics: yes  supports-test: yes  supports-eeprom-access: yes  supports-register-dump: yes  supports-priv-flags: yes |

1. First setup the environment with all the tools.

|  |
| --- |
| **For Ubuntu:**  # apt-get install -y make gcc libssl-dev bc libelf-dev libcap-dev clang gcc-multilib llvm libncurses5-dev git pkg-config libmnl bison flex graphviz  **For Fedora:**  # yum install llvm7.0-devel.x86\_64  # yum install clang  # yum install binutiles-devel  # yum install binutils-devel  # yum install readline-devel  # yum install llvm  # yum install gmp-devel |

1. Linux kernel version 4.1 or newer is required for BPF. This setup was tested using kernel version linux-5.1.2. In order to compile the kernel with the XDP and BPF support make sure the following options are enabled in the .config file after you make a copy of current running kernel config in step e.

|  |
| --- |
| 1. Download the linux-5.1.2.tar.xz in the /usr/src folder 2. tar -xvf linux-5.1.2.tar.xz 3. cd linux-5.1.2 4. cp -v /boot/config-$(uname -r) .config 5. Make sure that following options in the config file are selected   CONFIG\_CGROUP\_BPF=y  CONFIG\_BPF=y  CONFIG\_BPF\_SYSCALL=y  CONFIG\_NET\_SCH\_INGRESS=m  CONFIG\_NET\_CLS\_BPF=m  CONFIG\_NET\_CLS\_ACT=y  CONFIG\_BPF\_JIT=y  CONFIG\_LWTUNNEL\_BPF=y  CONFIG\_HAVE\_EBPF\_JIT=y  CONFIG\_BPF\_EVENTS=y  CONFIG\_TEST\_BPF=m   1. make -j $(nproc) 2. make modules\_install 3. make install 4. grub-mkconfig -o /boot/grub/grub.cfg1 5. grubby --set-default /boot/vmlinuz-5.1.2 6. cd tools/bpf 7. make 8. make install 9. reboot |

1. After reboot you must have booted into the new kernel linux-5.1.2. Now you can play with the samples under the samples/bpf folder in linux-5.1.2 kernel under /usr/src. Go to directory and run make. It will make all the samples programs there.

|  |
| --- |
| * 1. cd /usr/src/linux-5.12/samples/bpf   2. make |

1. There are plenty of samples to get you started but a few interesting ones are xdp\_rxq\_info and xdpsock. In order for the xdp to work properly and receive packets. You have to setup the filters accordingly. The filters can be setup using the ethtool. The xdp\_rxq\_info can help us see what queue the packets are being currently sent to.

|  |
| --- |
| ./xdp\_rxq\_info --dev ens1f1 --action XDP\_DROP  Running XDP on dev:ens1f1 (ifindex:3) action:XDP\_DROP options:no\_touch  XDP stats CPU pps issue-pps  XDP-RX CPU total 0  RXQ stats RXQ:CPU pps issue-pps |

1. In order to receive packets in a particular queue you can use the ethtool filter to tell NIC to forward packets to a particular queue. You can install (b), view (c) and delete (d) filter using ethtool. The filter in c will filter all udp packets with dst-port 4242 and sr-port 4242 to queue 16.

|  |
| --- |
| 1. ifconfig <devname> <ip> up 2. ethtool -N <devname> flow-type udp4 src-port 4242 dst-port 4242 action 16 3. ethtool –u <devname> 4. ethtool -N <devname> delete <filter number> |

1. After setting up the filter, you can run the xdpsock program, This program has various options such as tx\_only, rx\_drop, l2fwd. It also allows you to either use the XDP interface or the normal socket buffer interface. The box below shows all the available options. Feel free to play.

|  |
| --- |
| # ./xdpsock --help  Usage: xdpsock [OPTIONS]  Options:  -r, --rxdrop Discard all incoming packets (default)  -t, --txonly Only send packets  -l, --l2fwd MAC swap L2 forwarding  -i, --interface=n Run on interface n  -q, --queue=n Use queue n (default 0)  -p, --poll Use poll syscall  -S, --xdp-skb=n Use XDP skb-mod  -N, --xdp-native=n Enfore XDP native mode  -n, --interval=n Specify statistics update interval (default 1 sec).  -z, --zero-copy Force zero-copy mode.  -c, --copy Force copy mode. |

1. For instance if you want to run rxdrop only using the xdp native mode with zero copy. You can start is as follows in the box below.

|  |
| --- |
| # ./xdpsock -r -i ens1f1 -N -z |

1. If you cannot launch the program or if your program abruptly exits. You will lose the access to the Ethernet device. However, you can use the following command to reset the ethernet device and release the network interface.

|  |
| --- |
| # ip link set dev ens1f1 xdp off |

# Running the XDP module Application:

1. You can pull the latest code from the gitlab link below:

|  |
| --- |
| # git clone <https://gitlab.devtools.intel.com/rohantab/mtcpxdpm.git> |

2. The code has app.c where the main function is defined. There is xdp\_module.c which has all the information related xdp module. The box below shows these files.

|  |
| --- |
| # cd mtcpxdpm  # ls  app.c include Makefile README.md xdp\_module.c xdp\_module.h  # make |

3. Running the make in the mtcpxdpm folder generates app file. In order to run the application three parameters need to be passed. You can refer to README for usage. You can also print usage by running app as show in box below.

|  |
| --- |
| # ./app  Usage: ./app <r|t|l> <device> <queue number>  r: rx, t: tx, l: send back to same interface |

4. To run the application in the rx mode in device ens1f0 and queue 0, one can run the command as shown in box below. The app once started will continuously print the statistics. You can either need to setup the filter (hint: can be setup using ethtool as show in first part of the document) or pass the correct queue number (can be found by running the sample bpf program as shown in first part of the BKM) to the app to receive the incoming packets.

|  |
| --- |
| # ./app r ens1f0 0  \*\*\*\*\* Completed xdp\_init\_handle \*\*\*\*\*  sock0@ 1000171661 0 0  pps pkts 1.00  rx 0 0  tx 0 0 |

# Benchmark the XDP Code Using MoonGen Packet Generator:

1. For my setup, I have four connected for 10GbE cables directly between JF300-11A072T (**username**: rohan, **password** intel123, ip 10.54.43.49) and NCC1 (**username**: root, **password**: intel123, ip 10.242.51.165)

2. The JF300-11A072T runs the xdp code and the MoonGen can be found under /root/workspace\_rohan/MoonGen. You can also download MoonGen from here [[4]](#_References:). Steps to compile can be found on the website at [4].

3. To start the MoonGen you can follow the steps in the box below /root/workspace\_rohan/MoonGen. The first argument is the binary file the second argument is the .lua file which holds information about what kind of packets to forward mac address etc. The third and fourth argument are the Ethernet ports. MoonGen binary atleast needs two ports. Even if you are using one port, you still have to pass.

|  |
| --- |
| # ./build.sh  # ./bind-interfaces.sh  # ./build/MoonGen examples/l2-load-latency.lua 7 9 |

4. Once you run the MoonGen it will starts printing the statics of the packets like the one shown in box below. It transmits on both the ports and waits to receive packets. This process continues until you hit Control + C

|  |
| --- |
| [Device: id=7] RX: 0.00 Mpps, 0 Mbit/s (0 Mbit/s with framing)  [Device: id=9] RX: 0.00 Mpps, 0 Mbit/s (0 Mbit/s with framing)  [Device: id=7] TX: 10.59 Mpps, 5421 Mbit/s (7115 Mbit/s with framing)  [Device: id=9] TX: 10.28 Mpps, 5266 Mbit/s (6911 Mbit/s with framing) |

5. When you hit the Control + C it dumps the statistics to a file as well as prints the average and stddev on the screen. In the box below they are saved in histogram,.csv and also printed when I pressed Control + C

|  |
| --- |
| [INFO] Saving histogram to 'histogram.csv'  [Device: id=7] RX: 0.00 (StdDev nan) Mpps, 0 (StdDev nan) Mbit/s (0 Mbit/s with framing), total 0 packets with 0 bytes (incl. CRC)  [Device: id=9] RX: 0.00 (StdDev nan) Mpps, 0 (StdDev nan) Mbit/s (0 Mbit/s with framing), total 0 packets with 0 bytes (incl. CRC)  [Device: id=7] TX: 10.59 (StdDev nan) Mpps, 5421 (StdDev nan) Mbit/s (7115 Mbit/s with framing), total 29000515 packets with 1856032960 bytes (incl. CRC)  [Device: id=9] TX: 10.28 (StdDev nan) Mpps, 5266 (StdDev nan) Mbit/s (6911 Mbit/s with framing), total 28346976 packets with 1814206464 bytes (incl. CRC) |

6. Let us rerun the MoonGen example again. Let us login into the JF300-11A072T and go to folder /usr/src/linux-5.1.2/samples/bpf. Let us first run xdp\_rxq\_info. This will tell us which queue is receiving the packets from the MoonGen.

|  |
| --- |
| #./xdp\_rx\_info - - dev ens1f1  Running XDP on dev:ens1f1 (ifindex:4) action:XDP\_PASS options:no\_touch  XDP stats CPU pps issue-pps  XDP-RX CPU 11 6271046 0  XDP-RX CPU total 6271046 |

7. From the output in the box above we can see that the data is forwarded to the queue number 11. In order to start the rxdrop on queue number 11 and interface ens1f1 we can use the following command ./xdpsock -r -i ens1f1 -q 11

|  |
| --- |
| # ./xdpsock -r -i ens1f1 -q 11  sock0@ 1000170643 0ens1f1:11 rxdrop  pps pkts 1.00  rx 10,158,299 37,607,472  tx 0 0  sock0@ 1000171534 0ens1f1:11 rxdrop  pps pkts 1.00  rx 10,068,385 47,678,024  tx 0 0 |

8. Similarly we can run the xdpsock in the l2fwd mode using the command ./xdpsock -l -i ens1f1 -q 11 which gives the output as shown in box below.

|  |
| --- |
| sock0@ 1000168962 44163144ens1f1:11 l2fwd  pps pkts 1.00  rx 9,912,013 44,163,576  tx 9,911,885 44,163,576  ^C  sock0@ 137068396 45510140ens1f1:11 l2fwd  pps pkts 0.14  rx 9,824,030 45,510,396  tx 9,824,030 45,510,140 |

9. The MoonGen output when the l2fwd is running looks like something in the box below

|  |
| --- |
| [Device: id=7] RX: 10.02 Mpps, 5130 Mbit/s (6733 Mbit/s with framing)  [Device: id=9] RX: 0.00 Mpps, 0 Mbit/s (0 Mbit/s with framing)  [Device: id=7] TX: 9.85 Mpps, 5043 Mbit/s (6619 Mbit/s with framing)  [Device: id=9] TX: 10.02 Mpps, 5128 Mbit/s (6731 Mbit/s with framing)  [Device: id=7] RX: 10.01 Mpps, 5128 Mbit/s (6730 Mbit/s with framing)  [Device: id=9] RX: 0.00 Mpps, 0 Mbit/s (0 Mbit/s with framing)  [Device: id=7] TX: 9.84 Mpps, 5037 Mbit/s (6611 Mbit/s with framing)  [Device: id=9] TX: 10.02 Mpps, 5128 Mbit/s (6730 Mbit/s with framing)  [Device: id=7] RX: 10.03 Mpps, 5137 Mbit/s (6743 Mbit/s with framing) |

# Benchmarks

The numbers in Table 1 were measured on the XDP side. The corresponding numbers also were seen on MoonGen side but since MoonGen only shows up to two digits by rounding off. There is a little are error expected.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Benchmark | Packets Per Second (PPS)  1 Packet =  64 bytes | Packets Per Second (PPS)  1 Packet =  128 bytes | Packets Per Second (PPS)  1 Packet = 256 bytes | Packets Per Second (PPS)  1 Packet = 512 bytes | Packets Per Second (PPS)  1 Packet = 1024 bytes | Packets Per Second (PPS)  1 Packet = 1518 bytes |
| XDP tx\_push | 14,875,759 | 8,443,433 | 4,528,272 | 2,349,113 | 1,197,186 | 812,693 |
| XDP rxdrop | 9,849,441 | 8,108,052 | 4,528,036 | 2,349,049 | 1,197,223 | 812,464 |
| XDP l2fwd | 9,960,410 | 8,436,910 | 4,484,557 | 2,348,624 | 1,196,823 | 812,687 |

Table 2 shows the number measured on the MoonGen side.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Benchmark | Packets Per Second (MPPS)  1 Packet =  64 bytes | Packets Per Second (MPPS)  1 Packet =  128 bytes | Packets Per Second (MPPS)  1 Packet = 256 bytes | Packets Per Second (MPPS)  1 Packet = 512 bytes | Packets Per Second (MPPS)  1 Packet = 1024 bytes | Packets Per Second (MPPS)  1 Packet = 1518 bytes |
| XDP tx\_push | 14.92 | 8.44 | 4.55 | 2.35 | 1.20 | 0.81 |
| XDP rxdrop | 9.85 | 8.11 | 4.53 | 2.35 | 1.20 | 0.81 |
| XDP l2fwd | 9.96 | 8.45 | 4.49 | 2.35 | 1.20 | 0.81 |

# References:

1. [Express Data Path (XDP)](https://www.iovisor.org/technology/xdp)
2. [Berkley Packet Filter (BPF)](https://en.wikipedia.org/wiki/Berkeley_Packet_Filter)
3. [Compilation links for XDP](https://cilium.readthedocs.io/en/v1.1/bpf/)
4. [MoonGen](https://github.com/emmericp/MoonGen)