BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

http://fit.vut.cz

Project practice
2020/2021

Extension of the application for DDoS mitigation

Elena Carasec*

Abstract
Distributed Denial of Service attacks are one of the most serious threats for companies. As organizations have grown more dependent on the Internet and web-based applications and services, availability has become as essential as electricity.
DDoS is not only a threat to retailers, financial services and gaming companies with an obvious need for availability. DDoS attacks also target the mission critical business applications that your organization relies on to manage daily operations, such as email, salesforce automation and many others.[6]
This is the reason why DDoS protector application is being developed and extended. The application uses DPDK framework, which is a set of libraries for fast packets processing. Data Plane Development Kit includes libraries with a bunch of functions named RTE flow. It helps to offload routing of the incoming traffic.
RTE Stands for “Run Time Environment”. DPDK Core functions/APIs starts with ‘rte‘ prefix. RTE flow stands for DPDK defined way for Flow Representation. RTE flow API’s/functions and structure objects can be used to program a packet forwarding rule into NIC-hardware itself.[2]
Another crucial part of developing an application is testing. Software Testing is Important because if there are any bugs or errors in the software, it can be identified early and can be solved before delivery of the software product. Properly tested software product ensures reliability, security and high performance which further results in time saving, cost effectiveness and customer satisfaction.[1]
In order to catch errors as they appear PyTest is used. PyTest is a powerful tool, which helps to make tests less verbose and more readable and at the same time more maintainable.
Keywords: DDoS mitigation — DPDK — RTE flow

Supplementary Material: Testpmd models — Pytest code

* xcaras00@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

This project has an aim to extend and add new features to the application for DDoS mitigation named DDoS protector, which is being developed by CESNET.

The main goal of this paper is to demonstrate progress in fast packets processing using DPDK framework. This project is a continuation of its first part.

1.1 What is DPDK?

DPDK is the Data Plane Development Kit that consists of libraries to accelerate packet processing workloads running on a wide variety of CPU architectures. Designed to run on x86, POWER and ARM processors, it runs mostly in Linux userland, with a FreeBSD port available for a subset of DPDK features. DPDK is licensed under the Open Source BSD License. DPDK Library Features:

- Receive and Send Packets Within the Minimum Number of CPU Cycles
- Develop Fast Packet Capture Algorithms (tcpdump-like)
- Run Third-Party Fast Path Stacks

Some packet processing functions have been benchmarked up to hundreds million frames per second, using 64-byte packets with a PCIe NIC. [3]

1.2 What are Testpmd scripts?

In order to check the behavior of functions from Network Interface Cards drivers' implementation Testpmd scripts were used. These scripts allow to create small programs in a short time period, which are prototypes for new modules to be written in C language. These programs are not influenced by the other C application modules and allow to test possible logical errors.

The essence of TestPMD is a DPDK Application implemented using the DPDK library, which is used to forward data packets between Ethernet ports. Through the command line of the TestPMD runtime, we can configure the data packet forwarding between ports and other functions supported by the network interface. In addition, we can also use TestPMD to try some different driver functions, such as: RSS, filters and Intel Ethernet Flow Director (Ethernet Flow Controller). TestPMD supports two configuration scenarios:

1. TestPMD connects two Ethernet ports to an external flow generator.
2. TestPMD connects two Ethernet ports into loopback mode, so that you can check the receiving and transmitting functions of network devices without an external traffic generator.

[8]

1.3 RTE flow rules

The most useful feature for acceleration is RTE flow rules, which allow to offload traffic management by grouping the incoming packets, tagging them, redirecting them, etc. This API provides a generic means to configure hardware to match specific ingress or egress traffic, alter its fate and query related counters according to any number of user-defined rules. It is named rte_flow after the prefix used for all its symbols, and is defined in rte_flow.h.

Matching can be performed on packet data (protocol headers, payload) and properties (e.g. associated physical port, virtual device function ID). Possible operations include dropping traffic, diverting it to specific queues, to virtual/physical device functions or ports, performing tunnel offloads, adding marks and so on. [5]

1.4 Testing with PyTest framework

We cannot neglect testing if we want to save time developing programs, because proving that a new module does not break old functionality is impossible without testing. In order to prove that DDoS protector uses PyTest. PyTest tops the list of the best automation framework (available in any language). PyTest is also used for unit testing, integration testing, and end-to-end testing like other Python frameworks.

Though PyUnit (unittest) is the default test framework in Python, developers and automation testers widely prefer PyTest since it is more feature-rich than PyUnit. Tests can consist of simple functions, or they can also take input parameters to support parameterized testing.

PyTest fixtures make it easy to execute tests on a different web browser and platform combinations. Parameterized PyTest fixtures can be executed across different input values.

As Python is a scripting language, the automation engineer need not worry about running a compiler for converting code into an executable. Python has a comprehensive standard library. The language-constructs and object-oriented approach help programmers to come up with easy-to-read code that does the intended job.

The simple naming nomenclature followed by test frameworks (e.g., Test functions in PyTest should start with test_) eases the job of identifying test functions. It is suited for small-scale and complex projects, making Python the best scripting language for test automation. [9]

2. Testpmd scripts

It is not always easy to find out if there is a bug in your program or in a NIC's driver. Sometimes we need prototypes to eliminate likelihood of bugs in other project modules and in drivers to narrow down number of lines of code, which may influence a problem.

DPDK allows to check NIC's behaviour correctness using *testpmd scripts*, which serves us as prototypes. As it covers all the features of DPDK RTE flow functionality, it is a convenient tool, that allows to implement a part of a future program in C language many times faster and gives confidence that eventual errors are caused by cards drivers' implementation, assuming there are no errors in the script itself. On the other side if the script works without problems and the program in C, which uses the appropriate DPDK RTE flow library functions does not, the probability to find a bug in the C program in future is approaching the limit of 100%.

Testpmd scripts have helped me to better understand how do groups and priorities work in case of Mellanox card MCX516A-CCAT (ConnectX-5 100GbE). As for groups attribute, everything behaves according to the DPDK documentation. "Flow rules can be grouped by assigning them a common group number... Group 0 is the default group and this is the only group

which flows are guarantee to matched against, all subsequent groups can only be reached by way of the JUMP action from a matched flow rule."[5] It has been experimentally confirmed and works completely as expected.

However, as for priorities attribute, it is slightly different. The documentation says, that "a priority level can be assigned to a flow rule, lower values denote higher priority, with 0 as the maximum. A flow which matches multiple rules in the same group will always matched by the rule with the highest priority in that group. If a packet is matched by several rules of a given group for a given priority level, the outcome is undefined. It can take any path, may be duplicated or even cause unrecoverable errors."[5] According to this information, the main conclusion should be that if there are two rules with same group number and different priority level, the packet, that matches both rules' patterns, will match the one with higher priority level. Unfortunately, the experiments with testpmd have shown, that it is not always the truth. Every time sending a broadcast IPv4 or IPv6 packet it will match the second rule in spite of its lower priority, which is inconsistent with the documentation.

```
# broadcast
flow create 0 group 0 ingress
  pattern
    eth dst is ff:ff:ff:ff:ff:ff /
    end
  actions
    mark id 1 / count /
    jump group 2 / end

# any
flow create 0
  group 0 priority 1 ingress
  pattern end
  actions
    jump group 1 / count / end
```

The same behaviour of packets and rules matching was noticed implementing the same logic in C.

Since Mellanox NICs do not refuse to create overlapping rules, it would be helpful to have the possibility to operate with rules' priorities, but unfortunately by now it seems that priorities do not have any effect on RTE flow rules.

3. Flow isolated mode

Setting priority attribute should help to control flow of packets that do not match any existing RTE flow rule, when isolated mode is turned on.

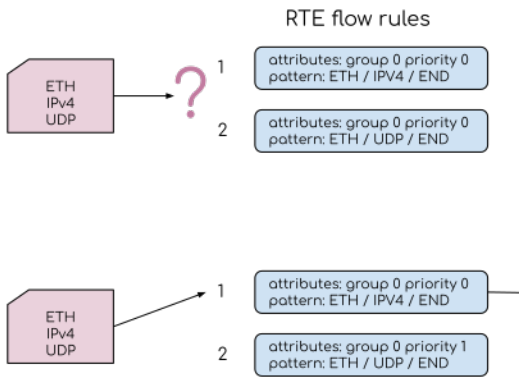

Figure 1. RTE flow attributes "group" and "priority" according to the DPDK v21.02 documentation.

By default isolated mode is off and packets, which have not matched any rule, end up in a queue defined by the NIC driver's implementation. This behaviour may be influenced by toggling flow isolated mode. When it is on, all the packets that do not match any rule are simply rejected and it is appropriate to have a default rule for this group of packets, which can be covered by one rule with void pattern. However without priorities all the incoming traffic may end in that rule.

Toggling isolated mode affects not just the behaviour of unmatched packets, but also has its side effects. Once effective, the following functionality has no effect on the underlying port and may return errors such as ENOTSUP ("not supported"):

1. Toggling promiscuous mode.
2. Toggling allmulticast mode.
3. Configuring MAC addresses.
4. Configuring multicast addresses.
5. Configuring VLAN filters.
6. Configuring global RSS settings.

According to the documentation, "existing flow rules or global configuration settings may prevent a driver from entering isolated mode."[5] As for mlx5 driver, since isolated mode should be entered before starting ports and all the rules should be created after it, it is not possible to cause inconsistency in it.

Nevertheless, toggling flow isolated mode brings more than just restrictions. When it is off, drivers functionality may not always be used to the maximum because of complexity since they have to check all the possible side effects when creating of destroying some rules. Sometimes even if an action is implemented in the driver, creating a rule with that action may cause error. The same problem is with patterns. For instance, trying to create multicast rule

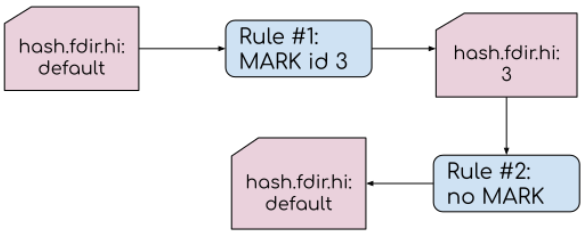

Figure 2. RTE flow MARK action. Changing packets' `hash.fdir.hi` field after matching rules with and without mark action.

```
flow create 0 ingress
    pattern eth
      dst spec 01:00:00:00:00:00
      dst mask 01:00:00:00:00:00 /
      end
    actions rss / end
```

causes error without isolation, but works without problems when isolated mode is on. Supposing it is not the only case, it is better to use isolation when working with RTE flow rules.

4. MARK action

Unfortunately, the DPDK documentation about RTE flow rules is not always comprehensive, probably, assuming some informations are intuitive. Notwithstanding, it may be misleading sometimes.

That was the case with RTE flow MARK action. What will be the value of packet's `hash.fdir.hi` field after matching another rule without mark action? Will it remain the same or will it be replaced by a default value?

The experiments have shown that if a packet matches a rule with mark action, this value in the `hash.fdir.hi` field is valid until it matches another rule. If the second rule has MARK action, the old value is expectedly replaced by a new one. If no MARK action is in the second rule, the tag is removed and the default value is set. There is no way to keep the value from the first rule after matching the second one.

Along with the structure mbuf's hash.fdir.hi field MARK action affects ol_flags by setting PKT_RX_FDIR and PKT_RX_FDIR_ID flags.

5. Host traffic tests using Pytest

Host traffic is a component of DDoS protector application, that divides incoming traffic into two flows: a part of it is redirected to the kernel and another part is processed by the other program modules. Only specific packets are redirected to the kernel: multicast (including broadcast), ARP, NDP and some specific packets with predefined VLANs and IPv4 or IPv6 addresses.

Accelerated host traffic manager should fulfill completely the functionality of host traffic component, but do it faster. To control it were introduced tests that cover all the necessary use cases. These tests send groups of packets and control if the types of packets mentioned above end in the queue leading to the kernel and on the other side test if some packets that do not meet these requirements are never redirected to the kernel.

One interesting difference between Mellanox and Intel drivers has been noticed. Assuming promiscuous mode is off, unicast packets with a random MAC address are blocked by Intel i40e card and at the same time Mellanox card accepts them and they are processed by the application. Probably the problem lies in linking cards, since Intel i40e X710 for 10GbE SFP+ is connected via loopback between two ports and Mellanox MCX516A-CCAT (ConnectX-5 100GbE) has practical loopback through the switch.

To test the functionality of this new module Pytest was used, because Pytest is a mature and full-featured testing framework, from small tests to large scale functional tests for applications and libraries alike and Pytest is simple to get started with. Pytest not only makes it simple to write tests, it has many command-line options that increase productivity, such as running just the last failing tests, or running a specific group of tests by name or because they're specially marked.[7]

6. Conclusion

In conclusion, I would like to emphasize the importance of testing, because it actually saves a big amount of time resources. If you want to develop a program and control its functionality, there is no way to omit testing. The pytest framework makes it easy to write small tests, yet scales to support complex functional testing for applications and libraries.[4]

Another great way to check the necessary functionality is prototyping. If we work with DPDK framework, Testpmd scripts are exactly what should be used. Thanks to these scripts it is possible to predict the behaviour of the new module by writing just a few lines of code.

7. Future work

Since the most part of host traffic acceleration is already done, it is an important part of DDoS protector application, which should be completely functional in a short time period. However, the application evolves

and there are definitely some other modules, which will benefit from their acceleration and I plan to contribute to it in my future work.

It is not excluded that this problem's solution will not have some unexpected side effects, that should also be eliminated.

Also I would like to contribute to the application using machine learning methods and write my bachelor's thesis about it. Machine learning methods may be quite helpful in networking, since the methods may be used for packets recognition, which may help to process faster packets from trustworthy sources and block faster packets from untrustworthy packets. For example, it may be realised by automatic flow rules creation if it is possible and pertinent. Probably, there are many more applications of the methods I do not know about by now, but I believe it will bring a lot of benefit to the DDoS protector.

Acknowledgements

I would like to thank my supervisors Ing. Jan Kučera and Ing. Jan Viktorin for their help.

References

[1] *What is Software Testing? Definition, Basics & Types*. Available at: https://www.guru99.com/software-testing-introduction-importance.html.

[2] BHARDWAJ, A. *DPDK RTE-FLOWS*. February 2020. Available at: https://www.iottrends.tech/blog/dpdk-rte-flows-rte-flow/.

[3] DPDK PROJECT. *DPDK Developer Quick Start Guide*. 2021. Available at: https://www.dpdk.org/.

[4] HOLGER KREKEL AND PYTEST-DEV TEAM. *Pytest: helps you write better programs*. 2015-2021. Available at: https://docs.pytest.org/en/6.2.x/.

[5] MELLANOX TECHNOLOGIES, 6WIND S.A. *Generic flow API (rte_flow)*. 2016. Available at: https://doc.dpdk.org/guides/prog_guide/rte_flow.html.

[6] NETSCOUT. *Why are DDoS attacks so dangerous?* 2021. Available at: https://www.netscout.com/what-is-ddos/why-is-ddos-dangerous.

[7] OLIVEIRA, B. *Why pytest?* August 2018. Available at: https://subscription.packtpub.com/book/web_development/9781789347562/1/ch01lvl1sec12/why-pytest.

[8] PROGRAMMERSOUGHT. *DPDK — TestPMD*. 2018-2021. Available at: https://www.programmersought.com/article/84056319581/.

[9] SHETH, H. *Why Python Is My Favourite For Test Automation?* October 2020. Available at: https://www.lambdatest.com/blog/python-automation-testing/.