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
An attack of the Denial of Service (DoS) type and its distributed modification (DDoS) is an attempt to harm the target system by making it inaccessible for end users and to interrupt company's business process. For instance, if a Web page is not accessible for too long, a lot of the users, who are potential clients, may never return to that page, considering that the company does not provide services anymore.
"DDoS Protector" application is being developed in order to prevent attacks by even distribution of requests and smart detection of abnormal activity. The application uses DPDK (Data Plane Development Kit) framework for fast packet processing. The main techniques that DPDK uses are Fast-Path (bypassing kernel) and PMD (Poll Mode Driver). Additionally, the DPDK framework includes Memory Managers to optimize the network buffer queues allocation and deallocation and HugePages. Enabling HugePages makes possible for the operating system to support greater memory pages size than the default ones have and it increases Translation look-aside buffer (TLB) hit ratio.
For load balancing is used RSS (Receive Side Scaling). This technology allows to distribute the incoming traffic quite evenly over CPUs. Since DPDK supports two ways of setting up RSS: legacy and via RTE flow, both of them are used for the application. However, the second one is declared as faster and we tend to replace legacy RSS configuration using "rte_ethdev.h" library with RSS set by "rte_flow" API, where possible. It seems to work well with the combinations of IPv4/IPv6 and TCP/UDP packet types and makes problems for Layer2-only ones.
Even though "rte_flow" is declared to have unambiguous behaviour over all Poll Mode Drivers, it is not a complete truth. Sometimes we faced unexpected NICs behaviour, but generally it is a powerful tool for acceleration purposes.

Keywords: DDoS mitigation — DPDK — Receive Side Scaling

Supplementary Material: Downloadable code

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

1. Introduction

A Denial of Service (DoS) attack is a cyber-attack, that has a goal to make target machine or network unavailable for ordinary users by flooding the system with an excessive amount of requests and consuming all the available resources of that system. In order to distinguish a DoS attack from the other connectivity errors, the following key features should be checked:

- unusually slow network performance (opening files or accessing web sites),
- unavailability of a particular web site,
- inability to access any web site.

A Distributed Denial of Service (DDoS) attack occurs when multiple systems flood the bandwidth or resources of a targeted system, usually one or more web servers. A DDoS attack uses more than one unique IP address or machines, often from thousands of hosts infected with malware. [1]

[1] https://en.wikipedia.org/wiki/Denial-of-service_attack#Distributed_DoS

All the attackers understand that they could generate many times more traffic if they used a botnet instead of one infected machine. A botnet refers to a group of devices, connected to Internet, which have been infected by malware. Each of bots is either running an infected device or its own sub-net of bots.

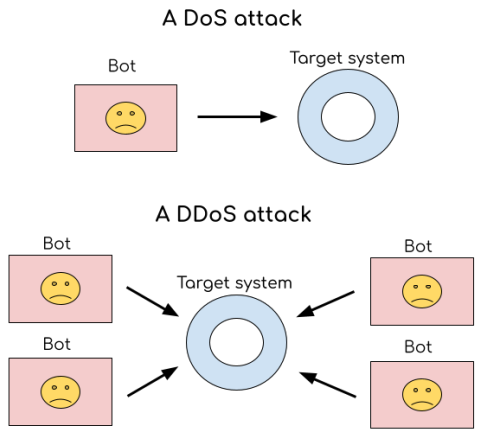

Figure 1. While a DoS attack is committed using a bot, a DDoS attack uses many bots, each of them may be a bot subnet. As in DDoS attack a large number of devices is usually involved, it is harder to prevent this type of cyber-attack.

All of DoS and DDoS attacks fall into any of three groups: volume based attacks, protocol attacks and application layer attacks. The first group includes UDP floods, ICMP floods, and other spoofed-packet floods. The attack's goal is to saturate the bandwidth of the attacked site. Protocol attacks include SYN floods, fragmented packet attacks, Ping of Death, Smurf DDoS and more. This type of attack consumes actual server resources, or those of intermediate communication equipment, such as firewalls and load balancers. The last group, application layer attacks include low-and-slow attacks, operation systems vulnerabilities and more. Comprised of seemingly legitimate and innocent requests, the goal of these attacks is to crash the web server.[2]

In one of the last and most infamous DDoS attacks have suffered Amazon Web Services(AWS) [3] in February 2020, and at its peak activity were sent 2.3 terabytes of data per second. According to Cisco, the number of attacks will grow from 7.9 million in 2018 to 15.4 million by 2023. [1]

2. DDoS Protector

Consequently, with the increase of number of DDoS attacks, their durability and speed, companies and organisations become interested in tools that will protect them, when it will be necessary. Pursuing the aim to protect Web services, "DDoS Protector" is being developed, focusing on such principles of DDoS mitigation as load balance and intelligent traffic routing.

In order to achieve load balance we use Receive Side Scaling (RSS), which is also known as multi-queue receive. It allows to distribute incoming traffic among multiple CPUs and increases the speed of its processing.

The problem of distinguishing requests coming from average users and attackers is not trivial, but we can start from acceptation those, which are coming from trustworthy sources. From the beginning those requests were divided into groups via software. But as we put the goal to increase the speed of packets processing, we have to delegate this to hardware. Of course, in future versions of "DDoS Protector" the problem of recognition suspicious activity will be solved, but for now we focus our attention on incoming traffic acceptance from trustworthy sources and making it as more efficient as possible.

3. Acceleration with DPDK

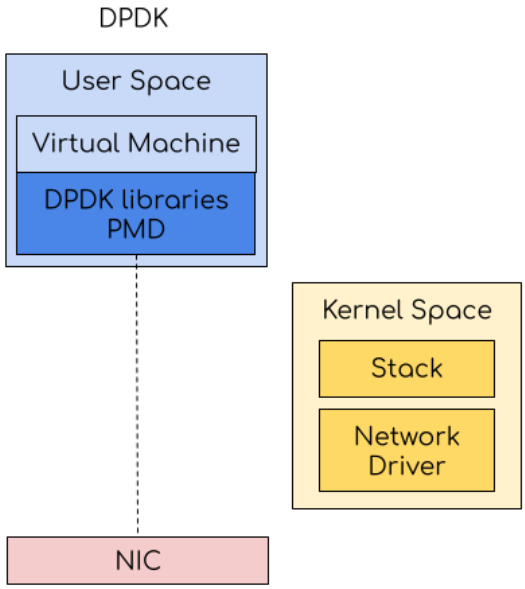

Figure 2. Fast packets processing using DPDK. The intercommunication between NIC and user space bypasses kernel space and makes possible avoiding unnecessary CPU cykles.

"DDoS Protector" application uses Data Plane Development Kit (DPDK) for fast packet processing. DPDK provides a set of libraries, which allow to bypass the kernel. This eliminates context switching when moving the frame between user space/kernel

[2] https://www.imperva.com/learn/ddos/ddos-attacks/
[3] https://aws.amazon.com/

space. Additionally, further gains are also obtained by negating the kernel stack/network driver, and the performance penalties they introduce.[2]

The second strength of DPDK is using Poll Mode Driver(PMD). The concept of PMD is refusing from CPUs' interrupts by Network Interface Controllers and dedicating a CPU for constant polling NICs for new incoming packets.

Moreover, DPDK libraries include such packet managers as a queue manager, which implements lockless queues, a buffer manager, which pre-allocates buffers with fixed size, and a memory manager, which allocates pools of objects in memory and uses a ring to store free objects.[2]

The DPDK also includes software examples that highlight best practices for software architecture, tips for data structure design and storage, application profiling and performance tuning utilities and tips that address common network performance deficits.[4]

4. Receive Side Scaling

Even packets distribution is the key to load balance, which can be achieved using RSS. Modern NICs support multiple queues for reception and transmission. For every incoming packet its hash is calculated, so then this packet is sent to a queue and is processed by a corresponding CPU. But it has a weakness. As hash is calculated using source and destination IP addresses, sending many packets from a single IP address or from some IP addresses, which will be mapped on the same queue may lead to imbalance and a congested core, while others will be idling. That is why a choosing a good hash function is so crucial. Generally, we tend to use symmetric hash keys for RSS to get symmetrical flows, because if the information about statistics on the different connections is shared between CPUs, it leads to undesirable locking and performance decrease.

Before DPDK v17.02 the only way to implement RSS was using "rte_ethdev.h" library. Since "DDoS Protector" uses a later version of DPDK, we are able to switch to "rte_flow" API. It is slightly higher-level than the legacy filtering framework which it encompasses and supersedes (including all functions and filter types) in order to expose a single interface with an unambiguous behavior that is common to all poll-mode drivers (PMDs).[3] Setting up RSS via "rte_flow" will be preferred in future as it is constantly developing and new features are added.

The "rte_flow" API allows to set RSS hash key manually or choosing a hash function. In DPDK v19.08 the symmetric toeplitz function was introduced.

[4]https://en.wikipedia.org/wiki/Data_Plane_Development_Kit

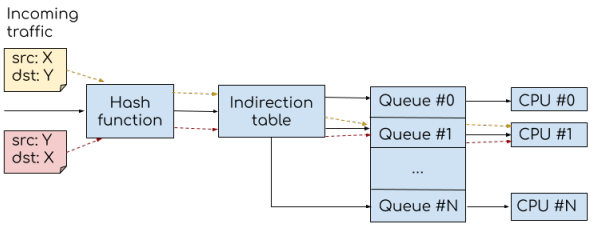

Figure 3. Symmetric Receive Side Scaling. Two packets with swapped source and destination IP addresses will come to the same queue thanks to an appropriate hash function. Note: The queue number and the corresponding CPU number do not have to be the same.

Our tests have proved that while working with Intel XL710 (i40e) only setting a symmetric key using legacy RSS is not enough for symmetric packets distribution. At the same time, using "SYMMETRIC_TOEPLITZ" works well with this type of card.

On the other side, we should be more careful with the "types" field using "rte_flow". Unlike global RSS settings used by other DPDK APIs, unsetting the types field does not disable RSS in a flow rule. Doing so instead requests safe unspecified "best-effort" settings from the underlying PMD, which depending on the flow rule, may result in anything ranging from empty (single queue) to all-inclusive RSS.[3] According to our tests, leaving the "types" field empty has the following results:

- on *MCX516A-CCAT (ConnectX-5 100GbE)* with *mlx5_core* driver packets are distributed as if using all-inclusive RSS,
- on *82599ES 10-Gigabit SFI/SFP+* and *X552 10 GbE SFP+* with *ixgbe* driver all the packets are sent to the queue number 0,
- *X710 for 10GbE SFP+* with *i40e* driver requres individual values for each rule. For example, it will not accept "ipv6-tcp" types setting together with "ETH/IPV4/TCP" pattern.

Since all the Poll Mode Drivers require special attention, a single way of setting up RSS with "rte_flow" was redesigned and now acquires the form of setting it up separately for each PMD.

One more constraint that "rte_flow" brings in is the moment, when RSS may be set up. Since DPDK v20.08 "rte_flow" rules should be created only after starting port, while legacy RSS mode was set before it. Destroying the rules also should be done before

stopping port, because it is not guaranteed that the rules will be destroyed automatically and it may lead to unexpected conflicts during new rules addition.

Another noteworthy technology for traffic management is Virtual Machine Device Queues (VMDq) presented by Intel. VMDq is a silicon level technology that offloads the network I/O management burden from the Virtual Machine Monitor, also called Hypervisor.[5] On NICs, which support VMDq, classification of packets based on Layer 2 and VLAN information is implemented in hardware. Network Controllers use round-robin algorithm to transmit queued packets to the wire. Although DPDK allows to combine VMDq with Receive Side Scaling, for now we did not fully investigate this technology and even DPDK example application did not demonstrate its use cases, as it ignored all the incoming traffic.

5. Host traffic acceleration

As we want to increase the speed of packets processing, as a general rule, moving responsibilities from software to hardware is a perspective idea. So, we have decided to begin with the *"host traffic"* component. It is responsible for the detection, whether the incoming packets belong to host traffic (packets which are sent to "dcpro-fwd0" interface) or not. This group of packets includes broadcast, multicast, ARP, NDP and packets with specific IP addresses and VLANs.

Since "rte_flow" allows to create rules that would distinguish packets with a specific pattern on each layer and mark them, the concept seems to be simple: to define patterns and to mark corresponding packets with a number and then let RSS do their distribution over cores. However, we have faced the problem of differences between cards with the different drivers. For instance, *82599ES 10-Gigabit SFI/SFP+* with *ixgbe* driver does not support the combination of actions MARK + RSS, meanwhile combination QUEUE + RSS is acceptable.

The most suitable for the acceleration has turned out to be *MCX516A-CCAT (ConnectX-5 100GbE)* with *mlx5_core*. Additionally to the required actions MARK and RSS, it supports groups, actions JUMP, DROP and COUNT, which will be useful for future mitigation.

However, even *mlx5_core* is not perfect. Trying to match broadcast and multicast packets and then to apply RSS action on them has become a stumbling stone. Firstly, its PMD does not accept the pattern for multicast packets such as *"pattern eth dst spec 01:00:00:00:00:00 dst mask 01:00:00:00:00:00 / end"* and fails with a weird error: *"port_flow_complain(): Caught PMD error type 1 (cause unspecified): hardware refuses to create flow: File exists"* even if it is the only rule, that was added. Secondly, even if we try to circumvent this behaviour and instead of the rule mentioned above we add patterns like *"pattern eth dst spec 01:00:00:00:00:00 dst mask 01:00:00:00:00:00 / ipv4 / end"* and *"pattern eth dst spec 01:00:00:00:00:00 dst mask 01:00:00:00:00:00 / ipv6 / end"* it will not help. This time nothing interferes the rules' creation and it may look like the problem is solved, while it is not.

It has turned out that we are not the only ones, who were trying to implement this use case of "rte_flow" rules. Unfortunately, using the *mlx5_core* driver has revealed that even if the rules, which patterns were mentioned above are created, none of the incoming packets will match them. It is a known bug in DPDK PMD [6] and hopefully soon it will be fixed.

At the present moment we have not achieved the aim of moving responsibility for distinguishing host traffic from software to hardware yet. Its skeleton is done, but it does not support the full functionality that it should have.

Furthermore, it is not efficient to estimate, whether a component is working properly or not without unit tests. This is what I am currently working on. These tests will help us to determine, whether current implementation of the "host traffic" component is a reference solution and whether something about broadcast and multicast rules will be improved, when we switch to a newer version of DPDK.

6. Conclusions

In conclusion, I would like to emphasize, that DDoS attacks are a real threat to all systems and underestimating it may lead to inevitable nuisances. All of the mitigation stages, such as detection, rerouting, filtering and analysis should be taken into account.

This paper focuses on the problem of increasing speed of load balance and filtering, because time is extremely precious for the mitigation. The system may be down in several minutes, when it is subjected to an attack, but recovering may take some hours. However, fast filtration saves the situation.

The Data Plane Development Kit library "rte_flow.h" allows to implement filtering rules that distinguish between the traffic from malicious users and legitimate traffic. Despite the fact that this API is intentionally developing to become a single interface with unambiguous behaviour for all Poll Mode Drivers, currently the differences in NICs behaviour, managed by their

[5] https://www.intel.com/content/dam/www/public/us/en/documents/white-papers/vmdq-technology-paper.pdf

[6] https://bugs.dpdk.org/show_bug.cgi?id=563

separate drivers are too significant for setting them up identically.

7. Future work

Since the made accelerations mentioned above were only the first stage and there is a large amount of work that is planned for future, the further plans include designing an infrastructure for "DDoS Protector", that will allow to accelerate even more the application using "rte_flow". There is a tendency to accelerate components and add some new ones, which will allow "rte_flow" to be used without having to resolve the problem of implementation of various solutions because of any specific details. Furthermore, we need to solve the problem of traffic that do not match any of our rules. What if it is a new case, which we did not took into consideration? Should it be allowed or dropped? Can we apply hardware acceleration on this traffic or should we investigate every packet via software? We need to find answers for these questions as soon as possible and it will be a part of this work's continuation.

Acknowledgements

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

References

[1] CISCO. *Cisco Annual Internet Report (2018–2023)*. 2020. Available at: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html.

[2] DONATO, R. *What is DPDK?* May 2018. Available at: https://www.packetflow.co.uk/what-is-dpdk/#fn1.

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