Building a Traffic Policer for DDoS Mitigation on Top of Commodity Hardware

Erkin Kirdan, Daniel Raumer, Paul Emmerich, and Georg Carle
Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich
Munich, Germany
{kirdan, raumer, emmericp, carle}@in.tum.de

***Abstract*—Traffic policing is the process of ensuring that network traffic complies with its policies with methods like traffic shaping. As the distribution of sources involved in a DDoS attack differs significantly from the typical distribution of customers for web services, traffic shapers and policers can be used in DDoS mitigation. In the past, software-based middleboxes, like traffic shapers, easily became overloaded and therefore a vulnerability for DDoS attacks. Although recent advances in network stack design on commodity hardware increased the performance, the software on top of the network stack also needs to provide adequate throughput and scalability regarding the number of limited subnets. Therefore, we build a high-performance and scalable traffic policer called MoonPol and evaluated it in a DDoS mitigation scenario. MoonPol runs on any commodity hardware, takes advantage of the underlying framework, DPDK, and combines it with appropriate algorithms and data structures. Data structures for efficient lookups are implemented together with the token bucket algorithm to police a traffic of fine-grained IP address ranges. Benchmarking results show that the single core throughput of the policer running on a 3.2 GHz CPU, is 6.5 Mpps with limiting 1 Million subnets, i.e., 492 CPU cycles per packet. With 250K subnets of all countries in the world, the throughput is 6.66 Mpps.**

***Index Terms*—Traffic policing; DDoS mitigation; User space networking; Lua; DPDK**

I. Introduction

Traffic policing is the act of enforcing a traffic contract, e.g., via limiting the data rate of a particular traffic. The traffic under policing can be specified by the 5-tuple of a flow. Policers can run on network interconnect devices such as routers or switches. Traffic shaping has a subtle difference in the process: a traffic policer works as an ingress filter which limits packets on input ports whereas a traffic shaper accepts all packets, buffers them as much as buffers allow, and then applies the limit on the output port.

High-performance traffic policers can be utilised in applications such as guaranteeing QoS, prioritising particular flows or enhancing network security. In this paper, we focus on Distributed Denial-of-Service Attack (DDoS) mitigation. The idea of using a traffic policer as a DDoS mitigation solution comes from discouraging illegitimate load (e.g., an attack) while letting legitimate users continue to use the system. As an example, a data centre can launch a high-performance traffic policer that limits data rates of countries or maybe even subnets within countries according to the typical access pattern. Skewed distributions of legitimate access may stem from services only relevant in certain countries (e.g., a regional video platform or webshop) or from distributed services where DNS-based load balancing assigns the closest services. The origin of DDoS attacks is usually defined by the attacker-controlled network and therefore follows a different pattern. As an example, in Figure 1, SubnetA is limited to 1 Mpps. In case of an attack, the attacker inside this subnet is limited to 1 Mpps and users from other subnets are still served. However, for this scenario the throughput of the policer becomes an essential metric. The policer must not cause any bottleneck by its operational cost on the link.

978-1-5386-3779-1/18/$31.00 © 2018 IEEE

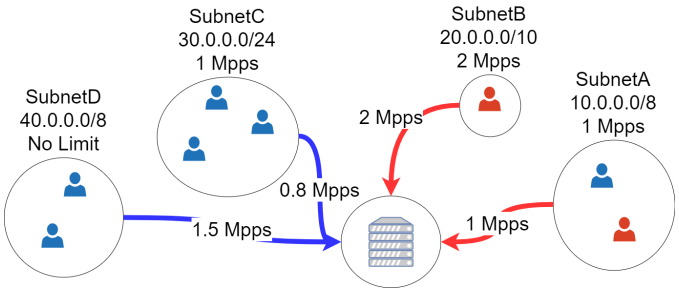

Fig. 1. Applying traffic policing to mittigate a DDoS attack

Example configurations for our use case are limiting data rates of countries, limiting data rates of all countries except the target countries, and limiting subnets with different data rates within countries. Considering there are 250K subnets owned by the countries in the world [1], the number of subnets limited by the policer without a significant decrease in the throughput becomes very important. Existing traffic policers may struggle because of the high number of subnets and the increase in the number of subnets creates an operational bottleneck, which decreases the throughput. Therefore, the following performance criteria need to be fulfilled by the traffic policer:

- Capability to limit up to 1 million subnets
- Throughput suitable for the physical connection capacity
- Low memory footprint to serve a high number of limiters

II. Related Work

Linux tc tool is a widely used application for traffic control such as shaping and policing. Research [2] shows that the throughput drops to zero after 200 IP address ranges which is a small number of subnets compared to our use case. In the same research, an algorithm is proposed to overcome the bottleneck caused by the linear search of tc tool. However, still, in 1 Gbit link and after 50K subnets, throughput starts to decrease. While Linux tc can be applied to end hosts as well as to the Linux routers for different purposes, other approaches that are designed to be applied for latency control [3] also exist. These are not applicable for DDoS defence.

Open vSwitch provides rate limiting through shaping or policing with Quality of Service support [4]. However, considering its forwarding capabilities under 2 Mpps [5], it does not fulfil the requirement of high throughput. The optional DPDK backend of Open vSwitch takes advantage of underlying DPDK framework, thus reaches up to 10 Mpps data rate in packet forwarding but without any rate limiting application. However, DPDK was already used in DDoS defence scenarios; e.g. Zhao [6] proposed to use DPDK as forwarder wherein only traffic that passed an anomaly detection is whitelisted. Rate limiting functionality also exists for less known high-speed packet IO frameworks. The snabb framework has also a rate limiter application, claimed to reach 20 Mpps data rate without any statement about the testing conditions [7]. In own measurements, we did not even achive line rate forwarding performance on a single core [8]. Additionally, Snabb only supports a subset of the NICs that are supported by DPDK.

Finally, Arbor Networks also offers high-performance DDoS mitigation solutions [9]. Apart from lack of any performance report or analysis, these solutions come as both hardware and software instalments. Moreover, their hefty price tag makes them suitable for large enterprises but not for those looking for small and practical software solutions that can run on any commodity hardware.

III. Approach

For our analysis, we implemented a traffic policer software as proof of concept and make it available as open source on GitHub [10].

A. Data Plane Development Kit

Data Plane Development Kit (DPDK) is an Open Source BSD licensed project by the Linux Foundation for fast packet processing through a set of libraries and drivers [11]. It overcomes the bottlenecks caused by the operating system's network stack by running the whole network interface card (NIC) driver in the userspace application. It supports a broad set of NICs from companies such as Cisco, Intel, and Broadcom [12]. In previous work we did a comparison of DPDK with netmap and pfring and analysed the sources for the performance increase in more detail [13].

B. Libmoon

Libmoon is a wrapper library for DPDK combining flexibility through a Lua scripting API with the packet processing performance of DPDK [14]. It is designed for prototyping DPDK applications due to its flexibility and quick turn-around cycles during development. Libmoon is used as the framework for our policer. Its packet processing performance is already proven with applications such as MoonGen [15]. Using Lua with libmoon as the scripting language for the implementation, the quick turn-around time between developing an idea, implementing, and running it without requiring a compilation step leverages innovation in networking.

C. Data Structures for Faster Lookup

Existing policers like Linux tc tool suffers from a scalability bottleneck caused by the linear search of policing rules [2]. During the packet processing, the source IP address of the incoming packet is searched through the list of subnets for matching. It limits the number of subnets that can be in the list to around 50-200 [2]. Considering countries having more than 80K subnets, it is not even possible to limit subnets of one country. To overcome this bottleneck of linear search, an algorithm that dedicates a fixed portion of the memory and uses it for subnet ID lookup is preferred. The use of a fixed portion of the memory can be negligible by today's hardware. It allows using the source IP address of the packet as an index to the table of subnet IDs.

IV. Design

A. Policing Policy

Considering our use case, policing based only on source IP addresses of packets is preferred. Therefore, limits are applied as per subnet.

B. Configuration

The configuration of the policer is done with a list of subnets with their limits. The hierarchical limiting of subnets is also possible with appending fine-grained subnets after coarse-grained subnets in the configuration file. Hence, applying a different limit to a subnet in a subnet is possible.

Limits are taken as packet per second. The rationale behind using packet per second instead of bit per second as a unit of limit is that the number of packets becomes more indicative metric to determine the size of a DDoS attack, for instance, a SYN flood, than the volume of the traffic in bits per second.

C. Subnet ID Lookup

Subnet ID lookup is done using a router data structure to achieve better scalability. DIR-24-8-BASIC [16] is a common data structure used for routing table lookup in routers. It is used to find next hop IDs, which are subnet IDs in our context.

The algorithm allocates a fixed amount of memory. In addition to that, it allocates memory proportional to the number of subnets having prefix 25 or more in the configuration file. Therefore, with inefficient and redundant use of two tables, it needs one memory lookup for the packets in subnets having

a prefix of 24 or less and one more memory lookup for the packets in subnets having a prefix of 25 or more. This design choice resolves the main scalability bottleneck caused by linear search as in Linux tc tool.

D. Limit Enforcement

After determining the subnet ID of the packet, a limit is enforced according to the traffic contract of the related subnet. The token bucket algorithm, which is also used in different policing/shaping applications such as Cisco [17] is implemented for limit enforcement.

According to the algorithm, a bucket is created for every subnet in the configuration file. Buckets hold tokens up to the bucket sizes. If a packet does not belong to any subnet under a rule, it is directly accepted. Otherwise, the bucket of the subnet which the packet belongs to is checked, and in case of no available tokens, the packet is dropped without any error message. If there is a token in the bucket of the subnet, one token is removed, and the packet is accepted. After a particular time, new tokens proportional to the time passed since the last addition and the limit of the subnet are added to the buckets.

E. Optimized Token Bucket Algorithm

DPDK works with active polling, i.e., during the operation, packets are polled, and processed in a run-to-completion manner. However, to maintain the token addition of buckets, the policer has to stop this loop at some point. Say, after polling n packets, the policer adds tokens to the buckets. Picking the value of n is critical for the quality and the throughput of the policer. If n is picked too large, then, some subnets may be depleted before token addition. This kind of token depletion causes square-shaped data rate for the subnet which leads to a poor quality traffic policer since the expected behaviour of a policer is limiting the data rate of a subnet precisely at the limit with a horizontal line. If n is picked too small, then, policer has to spend more time in token addition instead of polling and processing packets. This inefficient use of time causes a lower throughput in the policer. Therefore, there is an optimum value of n which is large enough to maximise the throughput without causing depletion in any bucket.

This optimum value is determined by several parameters. Apart from the link capacity, the number of subnets and the minimum limit in the configuration, the location of the network device in the network should also be taken into account. By the location, it is meant that a router is at the core of the network where packets from different sources are flowing may have a larger n value than a router at the edge of the network where packets from a small number of different sources are flowing. For our experiment and application, we have tested a couple of values for n and empirically determined 1 million as optimum value. Therefore, token management is handled, latest after each millionth packet.

F. Logging

Two different logging levels are possible to observe the behaviour of the policer wherein the first level, only dropped packets and in the second level, all accepted and dropped packets from subnets for a given interval are logged. As a result, disabling logging decreases memory usage and the second level logging needs more memory than the first level.

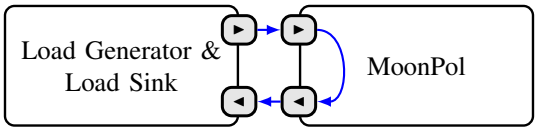

Fig. 2. Setup

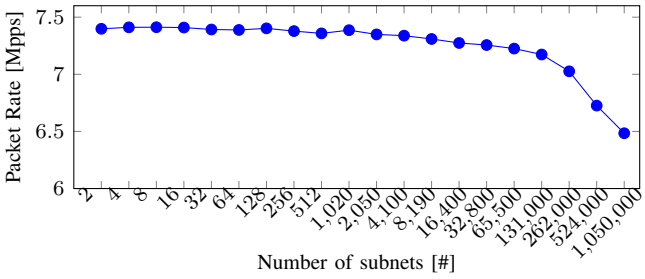

Fig. 3. Throughput w.r.t. to ne number of shaped subnets

V. Experiments & Results

The experiments are conducted on an Intel Xeon E3-1230 CPU with a frequency of 3.2 GHz where Hyper-Threading, TurboBoost and SpeedStep are disabled to make results reproducible. In two different experiments, the throughput of the increasing number of subnets and varying packet size are observed. The basic setup is shown in Figure 2. Two hosts, connected with a 10 Gbit/s link, are configured where one of them is load generator MoonGen, and the other is the traffic policer. Moongen [18] and the MoonPol [10] are both available on GitHub. After the policing, the policer host forwards packets back to the load generator.

A. Number of Subnets

The following experiment is conducted to test the scalability of the policer. The configuration files having all subnets of prefixes 2 up to 20 are generated. Thus, each configuration file contains 2 to 1 million distinct subnets.

The load is generated with packets having random source IP addresses and minimum sizes (64B). Completely random source IP address is a worst-case scenario far from reality since none of the real DDoS attacks can control such a vast range. However, to stress the implementation as much as possible, minimum packet size and random source IP addresses are preferred. Randomizing addresses decreases the cache utilisation to the minimum and triggers memory lookups as much as possible. Moreover, minimizing packet size brings more packets to be processed in a particular duration.

The results of the experiment are shown in Figure 3 where the data rate is in million packets per second (Mpps). The average data rate decreases from 7.4 Mpps and to 6.5 Mpps,

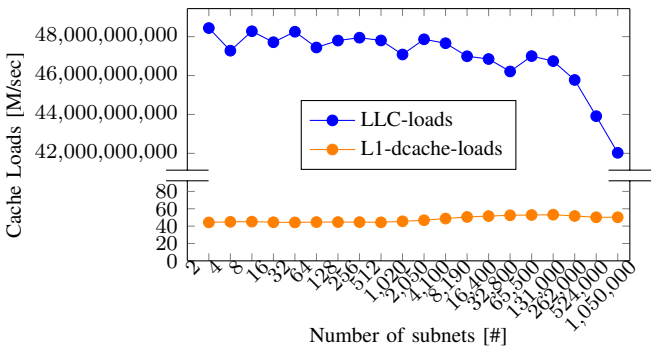

Fig. 4. Cache loads

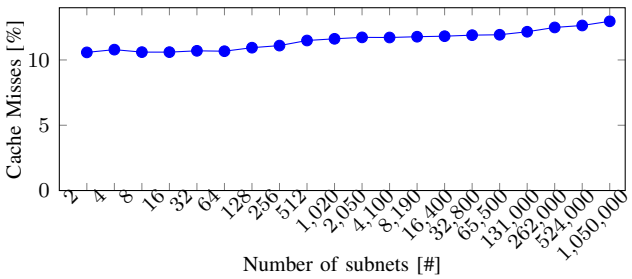

Fig. 5. Cache misses

i.e., CPU cycles per packet increases from 432 to 492, with the increasing number of subnets up to 1 million.

The decrease in the throughput has a similar trend with L1 cache loads as in Figure 4. Since in a large number of subnets each packet will fall into different subnet, cache miss rates increase as in Figure 5.

B. Packet Size

Packets having minimum sizes stress the CPU limit since the number of packets in a second is higher [19]. Increasing the packet size relaxes the CPU operation, and after one particular point, data rate regarding Mpps decreases since the link is saturated. To the find the crossing point of CPU limit and the link limit, where the link is saturated, the following experiment is conducted. The configuration file having 1 million subnets

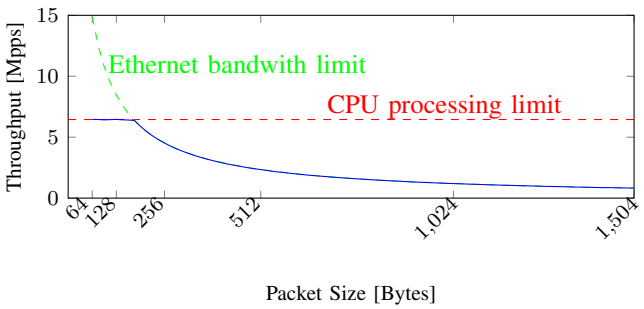

Fig. 6. Throughput with respect to packet size

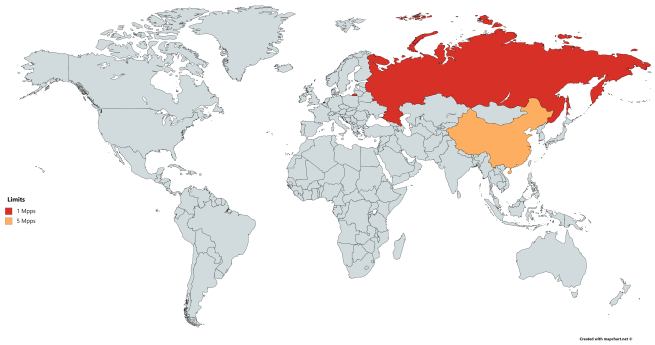

Fig. 7. Case study: setting limits to countries

TABLE I
RESULTS OF CASE STUDY

	Russia + China	All Countries
Number of Subnets	15K	250K
Data Rate [Mpps]	9.22	6.69
Memory [MB]	83	186

(all /20 subnets in IPv4) is given to the policer and a load of packets having random source IP addresses is generated. The link of 10 Gbit/s is saturated at around 170 Bytes per packet which can be seen in Figure 6 .

VI. CASE STUDY

We tested our implementation [10] within two possible case studies with actual subnets of the countries.

In the first case study, a configuration file having all subnets of two arbitrarily chosen countries, China and Russia, is generated. In total, they have 15K subnets. The limits of subnets arranged so that none of the packets will be discarded by the limit to see maximum achievable data rate in this use case.

In the second case study, a configuration file having all subnets of the world, which is 250 K, is generated. Again, limits are set so that no packet is discarded by the limit. The map in the Figure 7 shows the use case of prioritising target countries.

The subnets are fetched from GeoLite2 [1]. The results of the experiments are shown in Table I.

There are two crucial differences between the use case and benchmarking results regarding achieved data rate. First, the data rate in all subnets (with 250K subnets) is 6.66 Mpps whereas the data rate of 250K subnets is 7 Mpps according to benchmarking results. We prepared the configuration file for the benchmarking so that it covers all /18 subnets. Since none of the subnets has a prefix above 25, all packets trigger only one lookup for the subnet ID in the DIR24-8 data structure. However, naturally, countries have prefixes both above and below 25, which leads to the second lookup in the memory for the packets in subnets having a prefix of 25 or more. Thus, there is a difference between these two results caused by the second memory lookup in the real-life application. Second, the

data rate in two countries (with 15K subnets) is 9.22 Mpps whereas the data rate of 16K subnets is around 7.3 Mpps according to benchmarking results. Again, the configuration file for 16K subnets in benchmarking tests is generated with all /14 subnets. Therefore, each packet falls into one or another subnet which triggers memory lookup and token bucket algorithm etc. However, in the real-life application, most of the incoming packets having random source IP do not even belong to any of the subnets, which means, they are directly accepted without any need of further lookup. The lookup is only required for a small number of packets which randomly falls into subnets of limited ranges. Thus, in this case, a higher data rate is achieved then benchmarking results.

VII. Conclusion

In this study, a scalable and high-performance traffic policer is designed. It can be used in DDoS mitigation as well as other use cases of traffic shapers/policers or rate limiters. Its scalability regarding the number of subnets to be limited and throughput are analysed. It is shown that high scalability together with high throughput can be achieved with appropriate algorithm and data structure design combined with high-performance packet processors like DPDK.

The implemented policer can be launched in any commodity hardware having a DPDK supported NIC. When a list of subnets and their limits are given in a configuration file, the policer runs in the background and limits the traffic. These subnets can be up to 1 million finely grained limits.

Reproducible Research

The prototype can be found in [10]. To reproduce the benchmarking results, a configuration file generator is made available in the repository. Configuration files for both of the case studies presented in this paper are also available. After preparing the desired configuration file, the policer can be launched as explained in the readme file.

Acknoledgements

The authors would like to thank Andreas Blenk and Wolfgang Kellerer for the fruitful discussions. This paper is supported by SENDATE-Planets (16KIS0472).

References

[1] "GeoLite2 Free Downloadable Databases," https://dev.maxmind.com/geoip/geoip2/geolite2/, accessed: 14-Oct-2017.

[2] M. Goldstein, M. Reif, A. Stahl, and T. Breuel, "High performance traffic shaping for ddos mitigation," in *Proceedings of the 2008 ACM CoNEXT Conference*. ACM, 2008, p. 41.

[3] K. He, W. Qin, Q. Zhang, W. Wu, J. Yang, T. Pan, C. Hu, J. Zhang, B. Stephens, A. Akella, and Y. Zhang, "Low latency software rate limiters for cloud networks," in *Proceedings of the First Asia-Pacific Workshop on Networking*, ser. APNet'17. ACM, 2017, pp. 78–84.

[4] "Quality of Service (QoS) Open vSwitch 2.8.90 documentation," http://docs.openvswitch.org/en/latest/faq/qos/, accessed: 14-Oct-2017.

[5] P. Emmerich, D. Raumer, F. Wohlfart, and G. Carle, "Performance Characteristics of Virtual Switching," in *IEEE 3rd Int. Conf. on Cloud Netw. (CloudNet)*, 2014.

[6] X. Zhao, "Study on ddos attacks based on dpdk in cloud computing," *2017 3rd International Conference on Computational Intelligence & Communication Technology (CICT)*, 2017.

[7] "Snabb: Simple and fast packet networking," https://github.com/snabbco/snabb, accessed: 14-Oct-2017.

[8] W. Hahn, B. Gajic, F. Wohlfart, D. Raumer, P. Emmerich, S. Gallenmller, and G. Carle, "Feasibility of Compound Chained Network Functions for Flexible Packet Processing," in *International Workshop on 5G Enabling Technologies for the Internet of Things (GET-IoT) at the 23rd European Wireless (EW2017)*, Dresden, Germany, May 2017.

[9] "DDoS Protection & Prevention Products — Arbor Networks: DDoS Attack Solutions," https://www.arbornetworks.com/ddos-protection-products, accessed: 14-Oct-2017.

[10] "MoonPol GitHub repository," https://github.com/erkinkirdan/moonpol.

[11] "DPDK: Data Plane Development Kit," http://dpdk.org/, accessed: 14-Oct-2017.

[12] "DPDK: Data Plane Development Kit," http://dpdk.org/doc/nics, accessed: 14-Oct-2017.

[13] S. Gallenmüller, P. Emmerich, F. Wohlfart, D. Raumer, and G. Carle, "Comparison of Frameworks for High-Performance Packet IO," in *ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS 2015)*, Oakland, CA, USA, May 2015.

[14] "libmoon: library for fast and flexible packet processing with DPDK and LuaJIT," https://github.com/libmoon/libmoon, accessed: 14-Oct-2017.

[15] P. Emmerich, S. Gallenmüller, D. Raumer, F. Wohlfart, and G. Carle, "MoonGen: A Scriptable High-Speed Packet Generator," in *15th ACM SIGCOMM Conference on Internet Measurement (IMC'15)*, 2015.

[16] P. Gupta, S. Lin, and N. McKeown, "Routing lookups in hardware at memory access speeds," in *INFOCOM'98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3. IEEE, 1998, pp. 1240–1247.

[17] C. T. Notes, "Comparing traffic policing and traffic shaping for bandwidth limiting," *Document ID*, vol. 19645, pp. 22–42, accessed: 14-Oct-2017.

[18] "MoonGen GitHub repository," https://github.com/emmericp/MoonGen.

[19] P. Emmerich, D. Raumer, F. Wohlfart, and G. Carle, "Assessing Soft- and Hardware Bottlenecks in PC-based Packet Forwarding Systems," in *Fourteenth International Conference on Networks (ICN 2015), Best Paper Award*, Barcelona, Spain, Apr. 2015.