DeepDiag: Detailed NFV Performance Diagnosis

Junzhi Gong[1], Yuliang Li[1], Bilal Anwer[2], Aman Shaikh[2], Minlan Yu[1]
[1]Harvard University, [2]AT&T

CCS CONCEPTS

• **Networks → Middle boxes / network appliances**; **Network performance analysis**; *Network performance modeling*;

KEYWORDS

NFV, performance, diagnosis

ACM Reference Format:
Junzhi Gong[1], Yuliang Li[1], Bilal Anwer[2], Aman Shaikh[2], Minlan Yu[1] [1]Harvard University, [2]AT&T. 2019. DeepDiag: Detailed NFV Performance Diagnosis. In *SIGCOMM '19: ACM SIGCOMM 2019 Conference (SIGCOMM Posters and Demos '19), August 19–23, 2019, Beijing, China.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3342280.3342298

1 INTRODUCTION

In modern networks, NFV (network function virtualization) [1] has been replacing middleboxes, because NFV runs network functions on software platform, which achieves high flexibility in developing new functionalities, helps network operators to deploy network functions easily, and requires low cost in network function management [5]. However, compared to middleboxes, NFV is more error-prone, and traffic in NFV often experiences long tail latency or occasional packet drops. Usually, people simply blame this on "software" nature and neglect it. However, as the long tail latency becomes more significant in applications' performance, we need to better understand the NFV performance issue.

Performance issues in NFV stem from the "temporal resource contention" [3, 4]. There are various types of resource contentions that can impact NFV performance, including the input traffic queue of each NF, the CPU, the cache, *etc.* Each resource has different types of contenders. For example, the queue could be under contention when bursts of flows arriving into the queue, and the cache could be under contention when other applications have large cache usage.

To diagnose this issue, we need to understand **what resources are under contention, who are the contenders, and where those resource contentions come from.** This information can help operators to choose an appropriate method to solve the performance problem. For example, as shown in Figure 1, the NAT and Firewall share a cache line. If there is a burst of traffic arriving at the NAT, which causes the resource contention in the shared cache line, resulting in performance problems in Firewall. Then the operator can choose to isolate the cache line for NAT and Firewall, or choose to rate limit the input traffic arriving to NAT.

Diagnosis with NFV is challenging. First, there could be multiple resource contentions mixed together, and each could involve many contenders. As shown in Figure 1, each NF is bound with a queue, cache resource, and CPU resource. It is hard to figure out how each resource contention contributes to the problem. Second, a local view of diagnosis is not sufficient. As shown in Figure 1, if the problem happens in Firewall, then checking resource contentions within Firewall (queues in Firewall, cache and CPU) is not enough, because the root cause could come from NAT (*e.g.*, NAT sends out bursts of traffic), which is a hop away from Firewall. Third, the long latency of a packet is not only impacted by the resource contention during the processing of the packet, but also the contention before processing the packet. As shown in Figure 1, NAT has a long buffered queue due to CPU contention, and it sends out all buffered packets in a burst after the contention, resulting in problems in Firewall.

Our key insight is to monitor the queue of each NF. With the queuing information from all NFs, we can learn the experience of each packet, and how the traffic pattern changes across different NFs. With resource counters, we can also learn how each resource is under contention at any time. As a result, for each performance problem, we can learn which resource contentions contribute to the problem, and who are the contenders. Furthermore, we can also learn how the resource contention changes the traffic pattern across different NFs, and how the traffic pattern change impacts the performance of NFs.

To this end, we design DeepDiag, which is an NFV performance diagnosis system that can provide accurate and

SIGCOMM Posters and Demos '19, August 19–23, 2019, Beijing, China

ACM ISBN 978-1-4503-6886-5/19/08…$15.00
https://doi.org/10.1145/3342280.3342298

detailed diagnosis report. Operators can turn on DeepDiag on demand, which monitors light-weight information from the NFV chain. With the monitoring data, DeepDiag can perform offline diagnosis to identify relevant resource contentions, contenders, and how contentions propagate their impact to the performance problem. To help operators understand the diagnosis result, DeepDiag summarizes the result into an impact graph, which shows the quantified value of impact for each resource contention, and how the value of impact propagates across different components in the NFV chain.

2 SYSTEM DESIGN

When performance problems happen in an NFV chain, DeepDiag is responsible to figure out the following information: which resource contentions contribute to the problem, who are the contenders, and where those resource contentions come from. We design an impact graph model to show the diagnosis result. The impact graph consists of different components in the NFV chain, and the links between components describe how different components impact each other in different ways. Figure 1 shows an example of the impact graph. Each node is associated with a score, which is used to quantify how resource contentions impact this node. Each link is also associated with a score, which is used to quantify how different nodes impact each other. Since the traffic is composed of millions of packets, a brief view of the diagnosis result is needed. Therefore, DeepDiag also provides a list of rules, which look like "<when>, <which flow>, at <which hop>, suffers from problems, caused by <which resource contention>". We also give a score for each rule to quantify how the resource contention contributes to the problem.

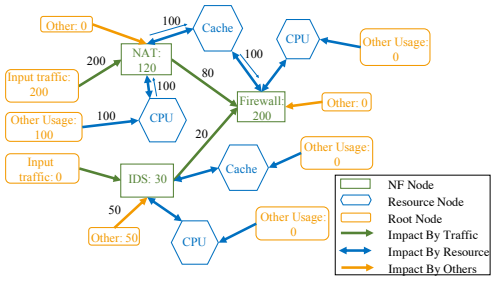

Figure 1: Example impact graph. ("Other" means other issues like code bugs and misconfigurations.)

2.1 Packet-level Queuing Information

The key idea of DeepDiag is to leverage packet-level queuing information. DeepDiag will dump the queuing information for each input packet for each NF, and the information includes: 1) five-tuple, 2) timestamp, 3) queue length, and 4) packet ID (which is used for identifying each packet). The packet-level queuing information is significant in DeepDiag. First, when performance problems happen, DeepDiag can learn how the input queue is built up. Second, the queuing information describes how the traffic pattern changes across different NFs, and such traffic pattern change shows the propagation of the impact of resource contentions.

2.2 Diagnosis Algorithm

As mentioned above, DeepDiag leverages packet-level queuing information to diagnose the performance problem. Therefore, DeepDiag will perform diagnosis on every packet suffering from performance problems (either long latency or packet drop). The diagnosis algorithm can be divided into 2 steps: 1) local score allocation and 2) score back-propagation.

Local score allocation. The goal of this step is to figure out where the long latency of the packet comes from, *i.e.*, how the long queue is built up. One key idea in this step is to consider the "*queuing interval*" rather than the actual queue content. Queuing interval means the time period from the time when the queue started to build up, to the time when the victim packet arrives. We consider queuing interval rather than current queue content because the impact of resource contentions could come from historical traffic. Another key idea in this step is to define an impact score that can be comparable across different resource contentions. We define the score based on the number of packets buffered in the queue due to different resource contentions. For example, either a burst of flows or CPU shortage can buffer packets in the queue, because the NF cannot process the high input load in time. In this way, the number of packets buffered in the queue represents how different types of resource contentions impact the performance. Therefore, we provide a general way to quantify the impact of different types of contentions.

Score back-propagation. The goal of this step is to figure out the root cause of the problem, and how the root cause propagate its impact to the victim. The key idea is to figure out the correct set of packets that help to propagate the impact. For example, if the impact is propagated by sending higher rate traffic, then we need to find out the set of packets in the higher rate traffic. If the impact is propagated through a shared resource, then we need to find out which NF is the contender, and which set of packets make the NF over-utilizes the resource. We run the score back-propagation recursively, until we back-propagate all scores to root nodes. At the end, we get an impact graph for a single victim packet.

Result summary. After getting the impact graph for each victim packet, we consider generating a list of rules that can summarize the diagnosis result. For each NF, we use AutoFocus [2] algorithm to find out the significant flow pattern. Based on packets in the flow pattern and their impact graph, we derive how all resource contentions propagate their impact to the flow pattern. Therefore, we can generate an accurate rule for victim packets.

REFERENCES

[1] [n. d.]. Network Functions Virtualisation. https://portal.etsi.org/NFV/NFV_White_Paper.pdf. ([n. d.]).

[2] Cristian Estan, Stefan Savage, and George Varghese. 2003. Automatically inferring patterns of resource consumption in network traffic. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*. ACM, 137–148.

[3] Bo Han, Vijay Gopalakrishnan, Lusheng Ji, and Seungjoon Lee. 2015. Network function virtualization: Challenges and opportunities for innovations. *IEEE Communications Magazine* 53, 2 (2015), 90–97.

[4] Bo Han, Vijay Gopalakrishnan, Gnanavelkandan Kathirvel, and Aman Shaikh. 2017. On the resiliency of virtual network functions. *IEEE Communications Magazine* 55, 7 (2017), 152–157.

[5] Zhijing Li, Zihui Ge, Ajay Mahimkar, Jia Wang, Ben Y Zhao, Haitao Zheng, Joanne Emmons, and Laura Ogden. 2018. Predictive Analysis in Network Function Virtualization. In *Proceedings of the Internet Measurement Conference 2018*. ACM, 161–167.