A fine-grained and high-performance telemetry system for R&E Network

Mazahir Hussain[1,2], Buseung Cho[1,2]

1. Korea Institute of Science and Technology Information, KREONet Center,

2. University of Science & Technology,

Daejeon, South Korea

[e-mail: mazahir.hussain@kisti.re.kr, bscho@kisti@re.kr]

*Corresponding author: Buseung Cho

Abstract

Traditional monitoring tools such as NetFlow, SNMP, sflow, and argus flow do not meet the requirements of network troubleshooting because of the limitations of providing fine-grained visibility in the network. Another bottleneck is handling the collection and processing of the packets with a line rate due to the system calls during I/O operations in Portable Operating System Interface (POSIX) compliant systems. In this project, we propose In-band network telemetry with Distributed Asynchronous Object Storage (DAOS) to collect customized telemetry data without the overhead of POSIX to enable easier network troubleshooting for high performance, and availability of the networking infrastructure. With the advancement of programmable switches, the P4 working group created In-band network telemetry (INT) for fine-grained telemetry, exporting directing from dataplane without the intervention of the overhead of the control plane.

Ⅰ. Introduction

Conventional network monitoring tools and protocols such as NetFlow, SNMP, sflow, argus flow, etc., are used by the network community for a long time to troubleshoot network devices and, are worth using to detect causes of performance degradations, anomaly detection, ensure performance, reliability, availability, and security [1]. However, this is not feasible with the traditional solutions due to the limitations of the packet mirroring techniques that use switched port analyzer (SPAN), Network test access point (TAP) along network packet broker (NPB). The limitations of the tools and Application Specific Integrated Circuits (ASICs) [2] of network hardware are causing issues for fine-grained visibility. With the adoption of the software-defined network (SDN), there are some freedoms in customization by accessing the forwarding plane that can be observed clearly in one of the instantiations of SDN, OpenFlow [3]. Another important factor of these conventional tools is the performance that draws limitations in the detection of microbursts [1].

Another issue in network telemetry is the collection and processing of network data with the line rate of the network infrastructure. The dedicated servers utilized for data storage and processing are full/partially Portable Operating System Interface (POSIX) compliant, which slows down the processing due to system calls on I/O operation [4].

In a previous study [5], RDMA over Converged Ethernet (RoCE) protocol is used with P4 language for in-band network telemetry to overcome the latency introduced by POSIX and they achieved a rate of around 20 million packets per second. They had to implement the RoCE protocol to use RDMA operations.

We propose a system equipped with In-band Network Telemetry (INT) and DAOS [4] storage stack. INT is a framework introduced by the p4 language consortium to collect and report the network state, directly from the data plane without the control plane's intervention, by inserting a small amount of information in the packet. DAOS is an open-source storage stack by Intel to overcome the limitations of traditional distributed storage to utilize the NVM technologies efficiently. We are proposing DAOS for the INT for two reasons, first to collect and store the data with the line rate as it is difficult to handle the data collection and monitoring with the traditional storage system with the more high-speed network being deployed such as 100Gbps ~, and secondly, the analysis of stored data using big data and artificial intelligence technologies that include a lot of reads/write operations.

Ⅱ. Design and Implementation

The first thing in the development of INT is to build an environment, there are two ways, virtual environment using Behavioral Model V2 (BMv2)[6] and real PISA-based hardware. First, we installed the SONiC (Software for Open Networking in the Cloud) community version then the Edgecore enterprise version as a network operating system (NOS) on the switches using Open Network Install Environment (ONIE) installer, however, we ended up after realizing that SONiC is not for P4 development because SONiC is built on top of any other Linux based distribution to use as it is for layer 2 or layer 3 networking. It does support programmable switches because SONiC works with Switch Abstraction Interface (SAI), but it is not for P4 development. Then, we installed Open Network Linux (ONL) which is based on Debian 9, we again failed to build the P4Studio Software development Environment (SDE) on top of Debian because of the kernel header mismatches as the current version of the P4studio SDE needs kernel version of 3.10.0-862 for CentOS 7, 4.18.0-240 for CentOS 8, 4.19.0-11 for Debian 10 and 4.14.151-OpenNetworkLinux for ONL as of P4Studio SDE 9.7.5. Prolific ways of the P4 development environment is to install a functional Linux-

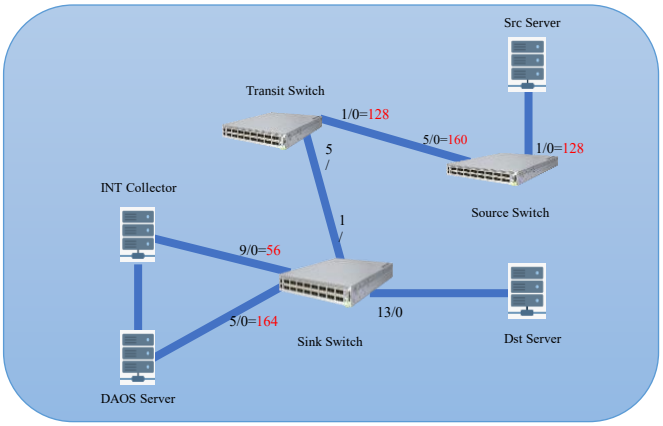

Figure 1: In-band telemetry with daos storage stack testbed

based operating system such as Ubuntu, CentOS, etc., and build P4Studio SDE on top of it. We have installed Ubuntu 20.04 on all the programmable switches. We are using three Wedge 100BF-32X in the current testbed for the project, a source switch that will add INT instruction in the packets, a transit switch that will follow the instruction set by the source switch to add INT data and a Sink Switch that add INT data according to the instructions set by source switch, send the data to mirroring ports/CPU ports, remove the INT header and send the packets to next hop. We have configured three servers, the source server to generate data and send it to the destination server, the destination server to receive data from the source server, the INT collector to collect INT-band telemetry data from the Sink switch, and the DAOS server to store the INT data in persistent memory and NVMe SSD, specifications of the daos servers are given in table 2 and the other servers are based on supermicro with Xeon® Gold 6246, 192GB RAM, Mellanox® ConnectX®-5, and 16TB NVMe SSD.

P4Studio SDE [8] need to be installed to build the P4 program and load the P4 binary on the switch ASIC. Using a YAML file, P4Studio SDE build and install on the switch. P4Studio SDE 9.7.0 is build for the profile, Board Support Path (BSP) files are required to build SDE for real hardware, and BSP is provided by the vendor of the ASIC, in our case Intel provides BSP files for Tofino.

First, we prepared and configured persistent memory and NVMe SSDs then created a pool which is a reservation of storage across a collection of targets in a distributed fashion, identifies by unique pool UUID using daos management command (dmg) tools.

Ⅲ. Testing and results

On real ASIC ports are not added by default during bf_switchd initialization and must be added manually using port manager (pm) commands. First, we need to add the port with the bandwidth and forward error correction, setting the auto-negotiation of link speeds, and then enable, once the ports are enabled it shows an ‘up’ status in operational status. To test the testbed, P4Studio SDE provides diagnostics APIs to test the components of switches.

We employed Scapy, a python library for packet generation to send packets from the source server, and shark to capture the packet on the destination server. Tshark is used to capture the coming packet from the source server on the port connected to the switch. The packets are generated using scapy. The simple layer 3 program forwarded the packets based on the IP address of the packets to the server. Figure 2 presents the result of the tshark captures on the destination server.

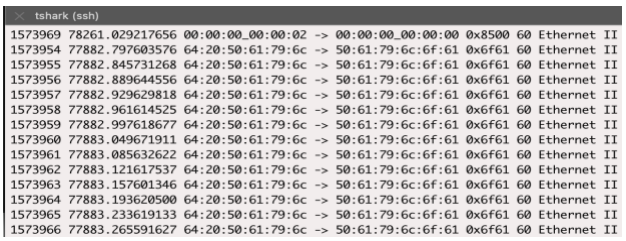

Figure 2: tshark packet captures on destination server

IV. Conclusion and future work

In this project, we are aiming to achieve INT-band network telemetry using state-of-the-art technologies to overcome the issues introduced by conventional tools and POSIX. So far, we have designed and tested the testbed for P4 development and DAOS storage stack, which was the most time-consuming task. In the future, we will design an INT mechanism based on the specification given by the P4 language consortium. Secondly, we will employ the DAOS spark APIs to create an analysis and visualization platform for better network troubleshooting and performance.

Acknowledgment

We would like to express our gratitude to the KREONet project for the Development and Establishment of National Large-scale Data Exchange Node. We also extend our thanks to all individuals who have directly or indirectly supported this work.

References

[1] Kfoury, Elie F., Jorge Crichigno, and Elias Bou-Harb. "An exhaustive survey on p4 programmable data plane switches: Taxonomy, applications, challenges, and future trends." IEEE Access 9 (2021).

[2] Kim, Daehyeok, et al. "Tea: Enabling state-intensive network functions on programmable switches." Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication. 2020.

[3] McKeown, Nick, et al. "OpenFlow: enabling innovation in campus networks." ACM SIGCOMM computer communication review (2008).

[4] Lofstead, Jay, et al. "DAOS and friends: a proposal for an exascale storage system." SC'16: Proceedings of the International Conference for High-Performance Computing, Networking, Storage and Analysis. IEEE, 2016.

[5] Beltman, Rutger, et al. "Using P4 and RDMA to collect telemetry data." 2020 IEEE/ACM Innovating the Network for Data-Intensive Science (INDIS).

[6] Fan, Chengze, et al. "NS4: A P4-driven network simulator." Proceedings of the SIGCOMM Posters and Demos. 2017. 105-107.

[7] “Barefoot P4 Studio.”https://www.barefootnetworks.com/products/brief-p4-studio/.