Application Switch using DPN for Improving TCP Based Data Center Applications

Shinnosuke Nirasawa
Kogakuin University
Shinjuku-ku, Tokyo, Japan
cm16035@ns.kogakuin.ac.jp

Masaki Hara
Kogakuin University
Shinjuku-ku, Tokyo, Japan
cm16037@ns.kogakuin.ac.jp

Akihiro Nakao
The University of Tokyo
Bunkyo-ku, Tokyo, Japan
nakao@iii.u-tokyo.ac.jp

Masato Oguchi
Ochanomizu University
Bunkyo-ku, Tokyo, Japan
oguchi@is.ocha.ac.jp

Shu Yamamoto
The University of Tokyo
Bunkyo-ku, Tokyo, Japan
shu@iii.u-tokyo.ac.jp

Saneyasu Yamaguchi
Kogakuin University
Shinjuku-ku, Tokyo, Japan
sane@cc.kogakuin.ac.jp

***Abstract*—Current network switches cannot be programmed and flexibly controlled. Then, developers of a data center application system, which is composed of software and computers connected with a network, are not able to optimize behavior of network switches on which the application is running. On the other hand, *Deeply Programmable Network* (DPN) switches can completely analyze packet payloads and be profoundly programmed. In our previous work, we introduced an application switch based on DPN. The switch was able to be deeply programed and developers could implement a part of functions of a data center application in the switch. The switch deeply analyzed packets, which is called *Deep Packet Inspection* (DPI), and provided some functions of the application in the switch. However, the switch did not manage connection and not support communication with TCP. In this paper, we proposed a method for constructing an application switch supporting TCP based communication. The method analyzes IP headers, TCP headers, and payloads of packets. When the switch detects a request which the switch supports, the switch replies according to its TCP session. We then introduce our implementation and evaluate performance of our application switch. Our evaluation has demonstrated that our switch has been able to improve performance of the data center applications.**

Keywords—component; formatting; style; styling; insert (key words)

I. INTRODUCTION

Large scale applications in data centers are composed of computers connected with a network. Traditional network switches cannot be flexibly controlled. Then, application developer cannot optimize network elements' behavior for improving application performance. On the other hand, Deeply Programmable Network (DPN) switches can completely analyze packet payloads and be profoundly programmed.

In our previous work [1], we introduced an application switch based on DPN. The switch was able to be deeply programed and developers could implement a part of functions of a data center application in the switch. The switch deeply analyzed packets, which is called *Deep Packet Inspection* (DPI), and provided some functions of the application in the switch. We then constructed sample data center applications with our application switch and demonstrated that our switch was effective for improving application performance. However, the switch did not manage connection and not support communication with TCP.

In this paper, we discuss a method for constructing an application switch supporting TCP based communication. For achieving this, analyzing and managing IP headers and TCP headers are required in addition to analyzing payloads of packets.

This paper is organized as follows. Section Ⅱ introduces related works. Section Ⅲ proposed a new application switch supporting TCP based communication. Section Ⅳ evaluates our method and demonstrates that our switch is effective for improving application performance. Section Ⅴ discusses difference with another existing method. Section Ⅵ concludes this work.

II. RELATED WORK

In this section, we introduce application switch using DPN and its related works.

A. DPI and DPN

While traditional network elements cannot be flexibly controlled by applications on the network, DPN [2] switches can be profoundly programmed and controlled. DPN is a network in which data planes can be programmed. *FLARE* [3] is typical network architecture for the network.

In traditional network, every network element has both of control plane and data plane functions. In this case, controlling the entire network is not easy. In the case of *Software-Defined Network* (SDN), data plane function is separated from network elements and logically consolidated. With this consolidation, controlling the whole network is easily achieved and network can be managed flexibly. An SDN switch, including its flow table, can be managed through software using SDN protocol like OpenFlow. However, elements in a deeper part in the device, such as its CPU and storage device, cannot be controlled with SDN. Unlike traditional network element, an SDN switch can analyze the layer 4 headers in packets. However, payloads of packets cannot be analyzed.

In contrast, even elements in a deeper part in the devices can be controlled with DPN. Thus, network management utilizing network element's devices can be realized. In addition, the entire packet, including its payload, can be analyzed. This enables profound analyses and flexible control. In DPN, data plane can be programed using Click Modular Router [4] and C++ language.

DPI is a method for filtering packets. The method analyses the payload of a packet for detecting spam, malicious software or other attacks. Then, the network element doing DPI determines whether the packet passes, and collects statistical information. In a case of a packet in Internet, a packet has Ethernet header and trailer, IP header, TCP header, and payload. It optionally has an application protocol header, such as HTTP header, in the payload. Common IP routers and OpenFlow switches monitor IP headers and L4 header, TCP and UDP header, respectively. DPI usually means analyzing payloads.

B. *Optimization of Switch's behaviors*

In this subsection, we introduce an *Application Switch*. In order to improve performance of data center applications, we have proposed to customize and optimize network switches deeply [5][6]. Traditional network switches cannot be programed and controlled. Developers optimize only computers and application software in them like "without DPN" in Fig. 1. On the other hand, DPS switches can be programed and controlled. Developers then can optimize behaviors of both of computers and switches like "with DPN" in Fig. 2. In usual cases, a network switch is placed at the center of computers. We then argue that putting an important function of the application of the data center in the center of the system, i.e. the switch, is an effective method for improving the data center application. For example, placing data cache at the center, which is the most important place is expected to improve performance effectively.

In the first work [5], we proposed to control packet forwarding considering application behavior. The switch deeply inspected packets and analyzed the requests from the clients. The switch then transmitted them to the optimal computer based on application behavior. The work demonstrated that analyzing requests in packets and forwarding the requests to a computer which had the request data in its cache remarkably improved data the I/O bounded database application. The method supported only UDP based application.

In the second work [6], we proposed to process some functions of data center application in the switch. The switch

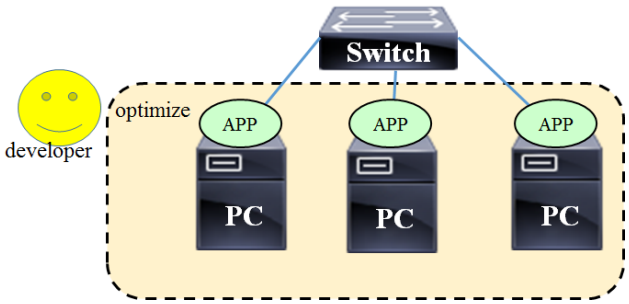

Fig. 1. Without DPN

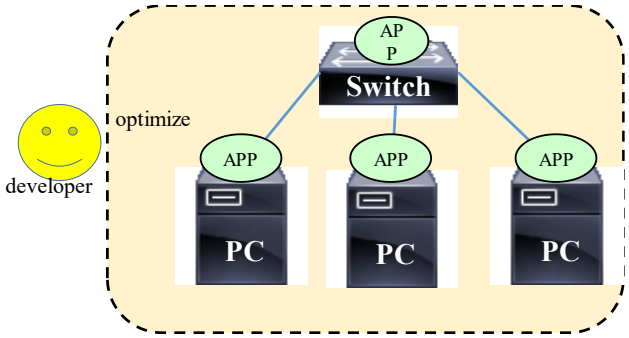

Fig. 2. With DPN

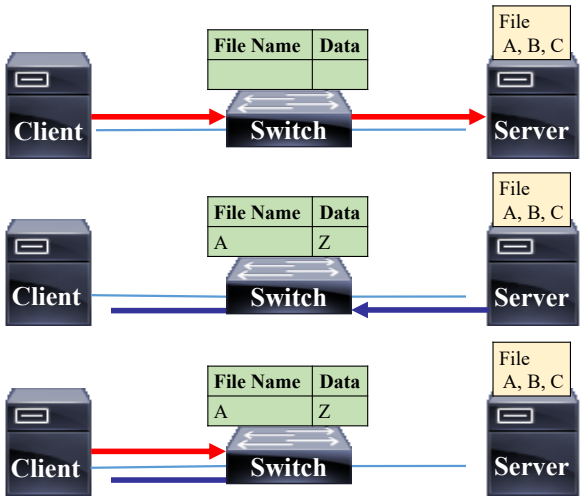

Fig. 3. Overview of the proposed method

deeply inspects each packet including a request for the data center application and interpreted the request. If the switch detected that it could reply the request, the switch created its reply and sent it the client without forwarding the request to the server. The switch supported only application using connection-less UDP communication because it could not take over a connection information, such as serial number and identification number.

We think our method is effective in the situation wherein the service provider utilized their data center, or on-premise system, exclusively and has permission to manage their network system including computers and network elements. In some large scale services, profound optimization of the system for improving

performance is required and our method is useful for such situations.

C. TCP connection

A TCP/IP communication manages the following numbers. Each connection is identified with the source and destination IP addresses and source and destination port numbers. IP header and TCP header have check sums which are calculated using various values, such as sequence numbers, in the packet. The sequence number and Ack number are updated at every packet sending and receiving. By updating these values, a TCP connection certifies that all the data are transmitted in correct order. Every sequence number is set as the Ack number of the previous packet with the opposite direction. The Ack value is set as the sum of the sequence number of the previous packet with the opposite direction and the size of the packet.

For handouting a TCP connection, these values should be succeeded and kept managed.

III. Application Switch for TCP Based Applications

In this section, we propose a method for applying the concept of application switch. In this section, we propose a method for applying the concept of application switch to TCP based applications.

Fig. 3 illustrates the overview of the proposed scheme. In a usual case, a TCP connection is established between the client and the server with 3-way handshake. Naturally, a request from the client is transmitted to the server and processed by the server using the connection. In the case of application switch, a TCP connection between the client and the server is sometimes handovered to the switch. The switch then succeeds the connection and sends a reply instead of the server.

The switch must keep managing the sequence number and acknowledge number of a handovered connection. For succeeding, the switch has to monitor connections which pass through it. In addition, the switch must succeed and manage these numbers after begin handovered. The switch inspects every packets and sometimes creates a packet instead of the server according to the monitored sequence number.

IV. Evaluation

For measuring performance, we have implemented an application switch and data center application using the switch. Fig. 4 illustrates our experimental system. The server computers have application software on it and the switch also has some functions implementation on it. The application is a simple web service. The client sends a request for a document and the system, which includes both of the computers and the switch, returns the requested document.

We have compared performance of application with and without application switch. Without application switch, the client outputs an HTTP request and the switch only forwards the request packet to the server. The server then processes the request and replies the request. With the application switch, a request packet from the client is forward to the server or processes in the switch. Cases of being forwarded are very similar to cases without application switch. In cases of being

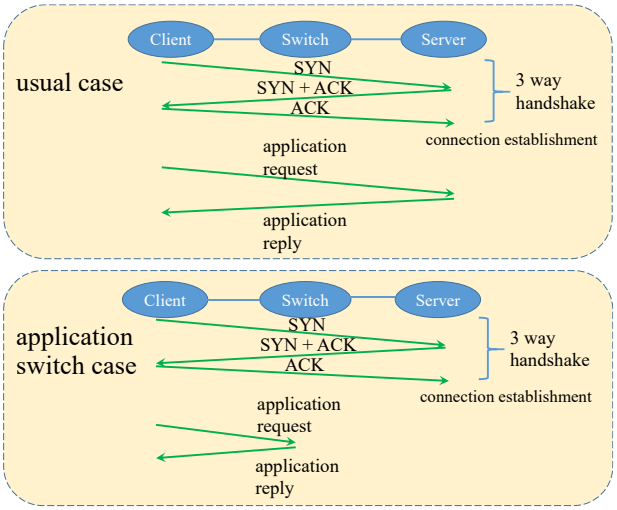

Fig. 4. Overview of application switch for TCP based application

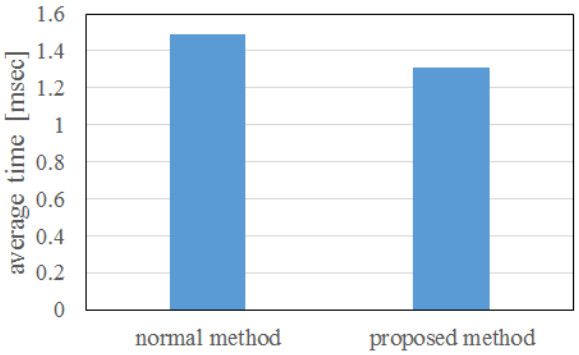

Fig. 5. Experimental result

processed in the switch, the application implementation interprets requests in a packet and processes it. In the case of our experiments, we have implemented http server function in the switch for small documents. The switch inspects a packet as follows. The software in the switch obtains byte array of a packet and analyzes IP and TCP header. If the packet is TCP packet with port 80, the switch analyzes its payload and interprets http protocol. If the switch can reply the request, it creates the reply without involving the processing in the server.

The specifications of the computer are as follows. CPU is AMD Athlon II X2 220 Processor, the memory size is 3.64[GB], the network device is Broadcom Tigon3 Ethernet driver which supports Gigabit Ethernet, and the operating system is Cent OS 6.5. The specification of the switch is as follows. CPU is Intel Celeron CPU 440 @ 2.00[GHz], the size of memory is 1.01[GB], the network device is Realtek RTL-8169 Gigabit Ethernet driver [Gbps] support, Gigabit Ethernet, and switch is implemented click modular router.2.0.1, <new-limit>

Figure. 5 shows the experimental results. The figure shows that the application switch provided the similar, exactly a little better, performance.

From the figure, we can expect that application switches can provide similar performance to a usual case with a network switch and a server computer.

There, putting the most important function such as caching the most frequently accessed data in the switch in the switch is effective for improving application performance.

V. Discussion

We present discussion on comparison between our application switch and application's reverse proxy. Putting reverse proxy may seem similar to our application switch.

A reverse proxy is a kind of server computers and this is visible in the third layer, i.e. the network layer. Thus, it cannot be installed and removed easily because application is required to use or net to use the proxy. On the other hand, application switch is transparent in the third layers. Its installation and removing does not require any modification of application.

In addition, reverse proxies cannot avoid being bottleneck of its application because all the requests and replies must reach the destination server computers via the proxy. On the contrary, application switches can be transparent. Therefore, it can avoid being bottleneck by only forwarding the requests to servers like usual network switches. In other words, application switch chose to be visible for improving application performance only when it can achieve it.

VI. Conclusion

In this paper, we introduced application switches which performs some of functions of data center application inside switches. We then proposed a method for applying the scheme to a TCP based application. The method monitors TCP connections which go through the switch. The switch sometimes is handovered a TCP connection and replies the request from a client instead of the server by succeeding a TCP connection. Our evaluation has demonstrated that an application switch can execute functions with similar performance, or a little better performance. This implies that helping application by switch can improve application performance.

In future work, we plan to apply our method to practical and more complex data center applications such as distributed database management systems.

Acknowledgment

This work was supported by CREST, JST.

This work was supported by JSPS KAKENHI Grant Numbers 24300034, 25280022, 26730040, 15H02696.

[1] Shinnosuke Nirasawa, Masaki Hara, Akihiro Nakao, Masato Oguchi, Shu Yamamoto, Saneyasu Yamaguchi, "Network Application Performance Improvement with Deeply Programmable Switch," International Workshop On Mobile Ubiquitous Systems, Infrastructures, Communications, And AppLications (MUSICAL 2016), 2017

[2] Akihiro, N. 2013. FLARE: Open Deeply Programmable Switch. GEC 16, USA.

[3] Akihiro NAKAO, "Software-Defined Data Plane Enhancing SDN and NFV", IEICE TRANS. COMMUN., VOL. E98-B, NO.1 JANUARY 2015

[4] Eddie K., Robert M., Benjie C., John J., and M. Frans Kaashoek. 2000. The click modular router. ACM Trans. Comput. Syst. 18, 3 (August 2000), 263-297. DOI=http://dx.doi.org/10.1145/354871.354874

[5] S. Nirasawa, M. Hara, S. Yamaguchi, M. Oguchi, A. Nakao and S. Yamamoto, "Application performance improvement with application aware DPN switches," *2016 18th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, Kanazawa, 2016, pp. 1-4. doi: 10.1109/APNOMS.2016.7737290

[6] Shinnosuke Nirasawa, Masaki Hara, Akihiro Nakao, Masato Oguchi, Shu Yamamoto, and Saneyasu Yamaguchi. "Network Application Performance Improvement with Deeply Programmable Switch," In *Adjunct Proceedings of the 13th International Conference on Mobile and Ubiquitous Systems: Computing Networking and Services* (MOBIQUITOUS 2016). ACM, New York, NY, USA, 2016, pp. 263-267. DOI: https://doi.org/10.1145/3004010.3004030

[7] Apache CASSANDRA http://cassandra.apache.org/

[8] Sailesh, K., Sarang, D., Fang, Y., Patrick, C., and Jonathan, T. 2006. Algorithms to accelerate multiple regular expressions matching for deep packet inspection. In Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM '06). ACM, (New York, NY, USA). 339-350. DOI= http://dx.doi.org/10.1145/1159913.1159952.

[9] Fang, Y., Zhifeng, C., Yanlei, Diao., T. V. Lakshman., and Randy, H, K. 2006. Fast and memory-efficient regular expression matching for deep packet inspection. In Proceedings of the 2006 ACM/IEEE symposium on Architecture for networking and communications systems (ANCS '06). ACM, (New York, NY, USA). 93-102. DOI= http://dx.doi.org/10.1145/1185347.1185360.

[10] Randy, S., Cristian, E., Somesh, J., and Shijin, K. 2008. Deflating the big bang: fast and scalable deep packet inspection with extended finite automata. In Proceedings of the ACM SIGCOMM 2008 conference on Data communication (SIGCOMM '08). ACM, (New York, NY, USA). 207-218. DOI= http://dx.doi.org/10.1145/1402958.1402983.

[11] Sailesh, K., Jonathan, T., and John, W. 2006. Advanced algorithms for fast and scalable deep packet inspection. In Proceedings of the 2006 ACM/IEEE symposium on Architecture for networking and communications systems (ANCS '06). ACM, (New York, NY, USA). 81-92. DOI= http://dx.doi.org/10.1145/1185347.1185359.

[12] Young, H, C., and William, H, Mangione-Smith. 2004. Deep Packet Filter with Dedicated Logic and Read Only Memories. In Proceedings of the 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM '04). IEEE Computer Society, (Washington, DC, USA). 125-134.

[13] S, Dharmapurikar., P, Krishnamurthy., T, Sproull., and J, Lockwood. 2003. Deep packet inspection using parallel Bloom filters. In Proceedings of the High Performance Interconnects, 2003. Proceedings. 11th Symposium. 44 - 51.

[14] Michela, B., Mark, F., and Patrick, C. 2008. A workload for evaluating deep packet inspection architectures. In Proceedings of the Workload Characterization, 2008. IISWC 2008. IEEE International Symposium (Seattle, WA). 79 - 89.

[15] N, Hua., H, Song., and T, V, Lakshman. 2009. Variable-Stride Multi-Pattern Matching For Scalable Deep Packet Inspection. In Proceedings of the INFOCOM 2009, IEEE (Rio de Janeiro). 415 - 423.