Low-cost Network Control System Based on Software-defined Networking over World Wide Web Using Sigle-board Computer

International Journal of Innovative
Computing, Information and Control
Volume **18**, Number **3**, June **2022**

ICIC International ©2022 ISSN 1349-4198
pp. 755–767

LOW-COST NETWORK CONTROL SYSTEM BASED ON SOFTWARE-DEFINED NETWORKING OVER WORLD WIDE WEB USING SINGLE-BOARD COMPUTER

Akihiro Imae, Osanori Koyama*, Ippei Tomo, Minoru Yamaguchi
Kanami Ikeda and Makoto Yamada

Graduate School of Engineering
Osaka Metropolitan University
Gakuen-cho 1-1, Naka-ku, Sakai, Osaka 599-8531, Japan
{ sab01019; dd104005 }@edu.osakafu-u.ac.jp; { nf101345; kanami; myamada }@eis.osakafu-u.ac.jp
*Corresponding author: koyama@eis.osakafu-u.ac.jp

Received December 2021; revised March 2022

Abstract. *Recently, software-defined networking (SDN) is one of the network virtualization technologies that has attracted significant attention. SDN allows network devices to be controlled centrally and flexibly, allowing for changes to configuration parameters, packet-exchange settings, and quality of service in routing, among other things. Furthermore, SDN reduces network administrators' network management workloads. SDN-enabled network devices, on the other hand, are generally more expensive than legacy network devices, and the total cost is a significant barrier when SDN is deployed in a datacenter, enterprise, and small-scale networks. In this paper, from the viewpoint of the cost and network management workload, we propose a network management and control system based on SDN for Internet protocol (IP) over the world wide web (WWW) using low-cost single-board computers. To validate the work of the proposal system, we designed and constructed a small-scale experimental system that mimicked a part of the proposal system and conducted experiments by using two functions. The experiment result showed that the proposed system worked properly. As a result, using the proposed system can significantly reduce total construction costs when compared to replacing all legacy network devices with SDN-enabled ones. To be more specifically, the total cost of SDN-enabled network devices multiplied by the number of devices can be reduced to the cost of a single Raspberry Pi multiplied by the number of nodes. Furthermore, our proposal system also contributes to reducing the management workload for network administrators and guaranteeing a certain level of security for network devices.*

Keywords: Software-defined networking, IP networks, Single-board computer, Raspberry Pi, OpenFlow, Ryu

1. **Introduction.** Recently, the amount of data flowing into data centers and corporate networks has been increasing due to the rise of new services such as social network services and cloud computing. Therefore, resource management and optimization for such networks have been important [1-3]. The number of network devices in the network is growing in response to the increase in the data flow, and so is the workload on network administrators. Furthermore, the future will necessitate the design and operation of complex networks to realize previously unimaginable services. In this context, one of the network virtualization technologies, software-defined networking (SDN), has gained traction in recent years [4-7]. In SDN, the data plane and control plane, which are integrated into legacy network devices, are separated, and the control plane is integrated into a software called the controller. With this separation, network administrators can operate

DOI: 10.24507/ijicic.18.03.755

the controller to change the configuration parameters of each SDN-enabled network device, thus reducing the management workload on network administrators. Furthermore, by connecting the controller to external applications, various functions such as routing and quality of service can be intricately controlled. With these external applications, it is also possible to build complex, flexible, and dynamic networks that were previously unimaginable [8-10]. As a result, SDN has been actively deployed in data centers and large enterprise networks to take advantage of its significant benefits. To fully reap the benefits of SDN, all existing network devices must be replaced with SDN-enabled ones. SDN-enabled network devices, on the other hand, are generally more expensive than legacy networks devices. Therefore, it is difficult to replace all network devices at once from the point of view of the cost, which is a major barrier to SDN introduction [11,12].

Several pieces of research have been done to solve the problem of total cost caused by replacing all devices with SDN-enabled devices at once. One solution is to adopt the idea of hybrid SDN [13-15], where legacy network devices and SDN-enabled devices coexist in the network. Hybrid SDN is based on the idea of gradually replacing network devices with SDN-enabled ones, rather than replacing all network devices at once. The gradual replacement approach solves the cost problem, but it does not allow you to fully benefit from SDN's reduced management workload. As a result, to replace all network devices at once with SDN-enabled ones while lowering costs, a study using inexpensive single-board computers as SDN-enabled Ethernet switches was conducted [16-19]. Due to the use of low-cost single-board computers as switches, switching performance was low and limited to use in low-speed networks. Therefore, the solution cannot be used in a high-speed network such as the gigabit-speed network. On the other hand, research has been conducted on how to update legacy network devices to SDN-enabled network equipment to use high-performance server equipment [20,21] and to use high-performance virtual switches [22]. The data plane in SDN was newly implemented in a high-performance server or virtual switch in these studies, and the legacy network device was updated by linking with them. Aside from the development costs, such as program implementation, it is difficult to apply them to a gigabit-speed network because the server or virtual switch is responsible for data transfer. If used, it necessitates the use of a very high-performance server or virtual switch, which raises the overall system construction cost. From this background, we have researched how to introduce a cost-effective SDN system that utilized legacy network devices by newly adding inexpensive single-board computers into the existing network [23-27]. In our previous research, we have proposed a system that has an inexpensive single-board computer for a legacy network device in order to make it SDN-enabled. In addition, we have proposed applications that work together with a controller and a new function that notifies the controller of network information using packet-in message. In the research, important points are only adding a single-board computer for a legacy network device and translating SDN control commands into vendor-specific control commands in the single-board computers.

In this paper, we propose a system that is more cost effective than our previous proposed system to realize SDN deployment at a low cost. Surprisingly, the translation of control commands by a single-board computer at each node has the advantage of preventing unencrypted packets from flowing over the IP network. Furthermore, by introducing a web server to make the controller interface (IF) web-based, we reduce the management workload on the network administrator in our proposal system. Furthermore, by incorporating a database, we have realized a function to visualize network device configuration and parameters on the controller interface. The proposal system not only significantly reduces the cost of implementing SDN, but it also allows the use of the network class

used before the SDN update. The high functionality of the system allows network administrators to change the configuration parameters of network devices and refer to the information of the network device from anywhere in the network via the world wide web (WWW), independent of where the controller is implemented.

This paper is organized as follows. Section 2 describes the proposed system. In Section 3, we describe the results of verifying the operation of the proposed system on an experimental network. Finally, the conclusions of this paper are given.

2. Cost-Effective Network Control System.

2.1. Proposal method. Figure 1 shows the methods used in the proposal system in this paper. In Figure 1, the desired control commands are delivered from the SDN controller to the single-board computer using OpenFlow [28-31] (OF) which is SDN protocol. In other words, the control commands sent by the OpenFlow controller are received as OpenFlow messages by the OpenFlow switch in the single-board computer, and after reading the information in the OpenFlow message, they are translated into vendor-specific control commands and sent to the single-board computer via vendor-specific protocols such as Telnet and secure shell (SSH). The legacy network device is configured in response to the control commands that are sent to it. Furthermore, it is critical to note that a single-board computer can translate control commands for multiple legacy network devices. In other words, in Figure 1, there are many OpenFlow connections between the OpenFlow controller and the OpenFlow switch. The number of connections equals the number of legacy network devices, and each connection can be used to identify the desired legacy network device. As a result, we can control many network devices with a single-board computer, which is less expensive than the proposed system in our previous study [23-27].

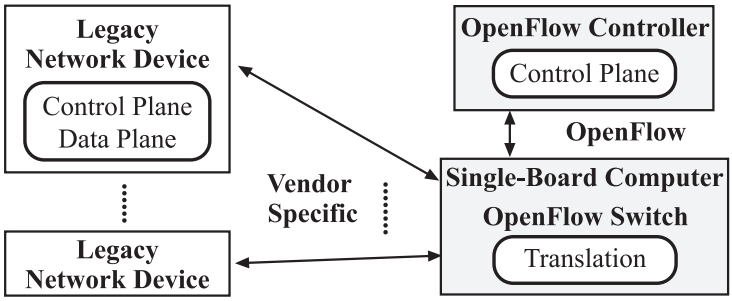


FIGURE 1. Method of translating control commands

The translation mechanism in which SDN control commands are translated into vendor-specific control commands is very simple. Setting values in legacy network device such as port number, static routing table, and virtual local area network (VLAN) number are set to parameters in OpenFlow-Mod message. The parameters are included in the fields of header, match, and action of the OpenFlow-Mod message. The OpenFlow message is received by OpenFlow switch in the single-board computer. The single-board computer has a dictionary that includes one-to-one correspondences between the parameter in OpenFlow-Mod message and vendor-specific control command. We have developed the dictionary in previous researches [23-27]. If the type of the received message is OpenFlow-Mod, the parameters in OpenFlow-Mod message are changed to vendor-specific control commands corresponding one-to-one with them using the dictionary. Finally, the vendor-specific control commands are sent by vendor-specific communication protocol.

2.2. **Overview of the proposal system.** Figure 2 shows an overview of the proposal system. The proposal system can be divided into three layers for each main function. In Figure 2, we adopt the method described in Figure 1. Furthermore, the controller establishes SDN connections with SDN-enabled network devices as well as single-board computers. Furthermore, various applications can be linked to the controller, allowing for flexible network construction and operation. In the proposal system, for example, web servers and databases are used as applications. As a result, the network administrator's user interface (UI) is the web pages provided by the web server rather than the controller. In addition to storing the settings information of network devices in the database and visualizing them, the UI is improved by implementing a language switching function (Japanese or English) on the web page to reduce the management workload of network administrators.

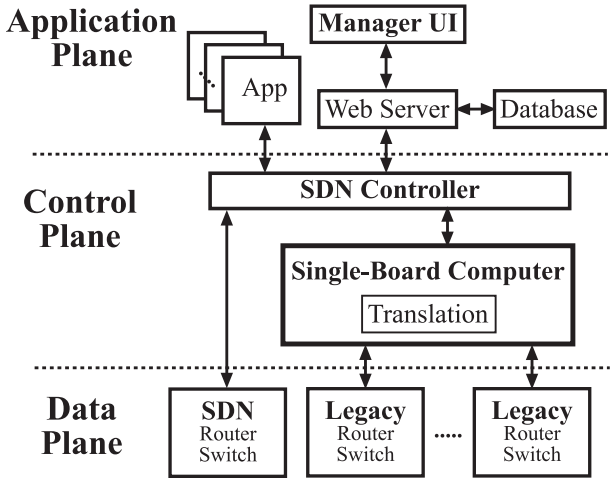


FIGURE 2. Overview of the proposal system

2.3. **Configuration of the proposal system.** Figure 3 shows a configuration of the proposed system in this paper. In the proposal system, there are Node-1 to Node-N, and each node is connected by Internet protocol (IP). In Node-1, there is a control server and a Raspberry Pi [32], one of the single-board computers, and the number of SDN connections established is equal to the number of legacy network devises in the node. In addition, the method in Figure 1 is used to update the legacy network devices in Node-1 to support SDN. Only the Raspberry Pi is present from Node-2 to Node-N, and it has established connections with the controller in Node-1. Similarly, the Raspberry Pi in each node updates the legacy network devices to support SDN. In Figure 3, the manager PC is also in Node-1, but it could be in any other node. In other words, the controller of Node-1 can be used from any other node via the web, which greatly contributes to reducing the management workload on network administrators.

2.4. **Functional components and processing flow.** Figure 4 shows the configuration of functional components and the operation flow at Node-1 in the proposal system shown in Figure 3. The network administrator uses the manager PC running a web browser to change the settings of the legacy network devices from Layer-1 to Layer-3 in the open systems interconnection reference model and to refer to the current configuration information of the network devices. First, the network administrator selects the desired device from the legacy network devices to be managed, then selects the desired configuration items, which are the functions on the device from Layer-1 to Layer-3, enters the parameters, and

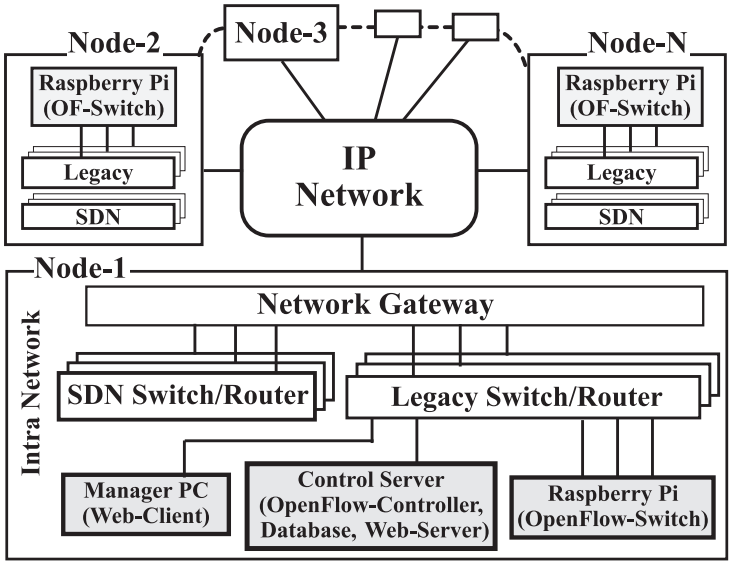


FIGURE 3. Proposal system

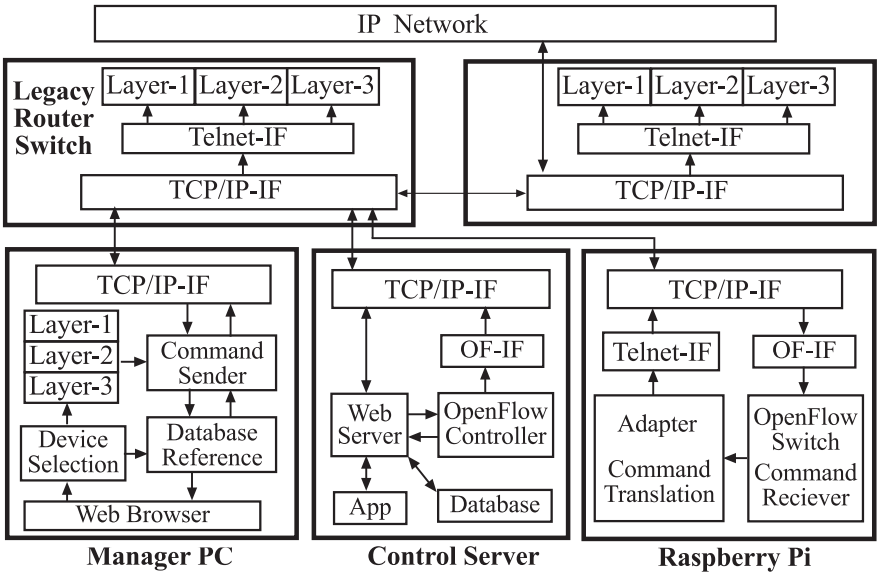


FIGURE 4. Functional components and processing flow

sends them. Alternatively, once a device has been selected by the network administrator, a control command referring to the device's configuration information can be sent. Next, the sent control command is received by the web server in the control server through the transmission control protocol/Internet protocol (TCP/IP)-IF, and if it is a control command to change the device settings, it is passed to the OpenFlow controller after updating the database, and if it is the control command to refer the device configuration information, the information is retrieved from the database and is sent back. Following that, the OpenFlow controller uses the received control commands to identify the OpenFlow connection to be used and sends the control commands as an OpenFlow message. The OpenFlow message sent is received by the OpenFlow switch in the Raspberry Pi that has established the OpenFlow connection, and the control commands are translated through the adapter and sent to the device using the vendor-specific protocol. In this paper, we adopt Telnet which has long been used as the vendor-specific protocol. Telnet is

a communication protocol used in IP networks to control remote servers and routers and switches from a terminal generally. Because all communication in Telnet is unencrypted, sending packets over the IP network is risky; however, in our proposed system, packets flowing over the IP network are limited to encrypted packets such as OpenFlow packets. Because unencrypted packets, such as Telnet packets, can only travel within each node, the network can be run relatively safely.

3. **Experimental Demonstration.** To verify the work of the proposal system, we built a small-scale experimental system that mimics Node-1 in Figure 3. In the experimental system, the two functions were executed three times each on two legacy layer-3 switches (L3SW) and the work was verified by observing the receiving throughput of the router tester. Furthermore, Wireshark [33], one of the packet capture tools, was used in the experimental system to capture packets before and after the execution of the two functions. As a result, the proposed system's correct operation was confirmed further by capturing major packets such as OpenFlow packets.

3.1. **Experimental setup.** Figure 5 shows the experimental setup. Two L3SWs, a router tester, a manager PC, a control server, a Raspberry Pi, and an analyzer for packet capture were located as shown in Figures 3 and 4. The router tester kept 0.9 Gbps traffic flowing through the two L3SWs. The traffic was received on three ports, Rx1 through Rx3. In other words, three IP addresses (192.168.23.2, 192.168.24.2, and 192.168.30.2) were assigned to the receiving ports of the router tester, and 0.3 Gbps traffic destined for each IP address was continuously flowing. As a default setting, L3SW1 was not configured with a static routing table for Rx1, Rx2, and Rx3, but L3SW2 was configured to route correctly, including the IP interface settings. The IP interface settings referred to the IP address assigned to the network device's virtual local area network (VLAN), and the VLAN was configured with the ports that participate in that VLAN. In other words, L3SW2 had three VLANs and their respective IP interfaces configured on a single port that was used as the sending port for routing packets to Rx1, Rx2, and Rx3. As the result of initial settings, the receiving throughput of the router tester by default was 0 Gbps.

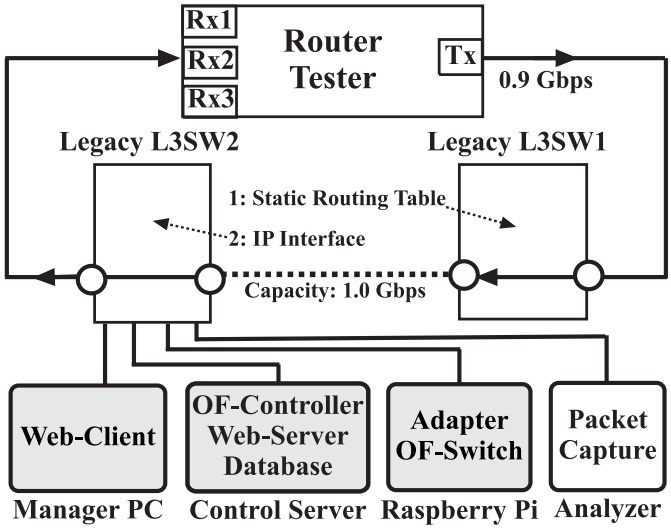
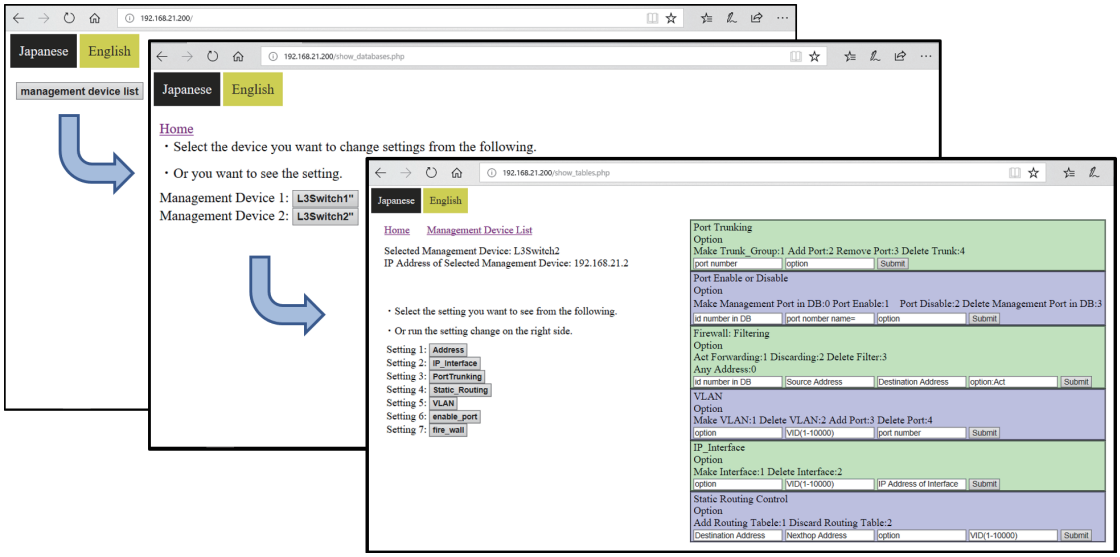
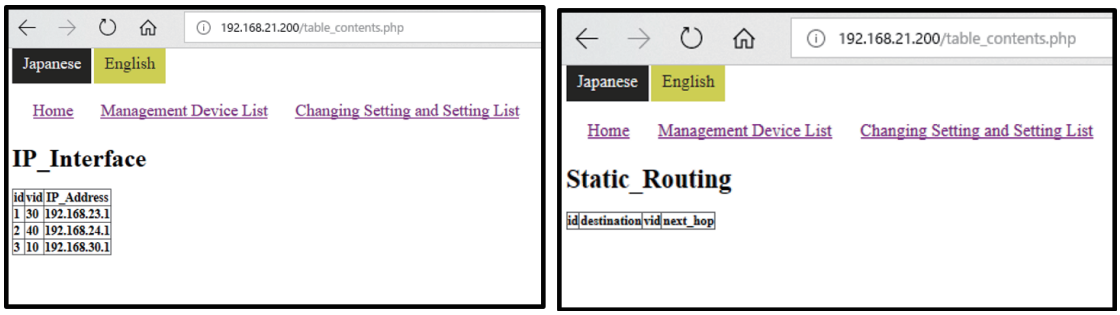


FIGURE 5. Experimental setup

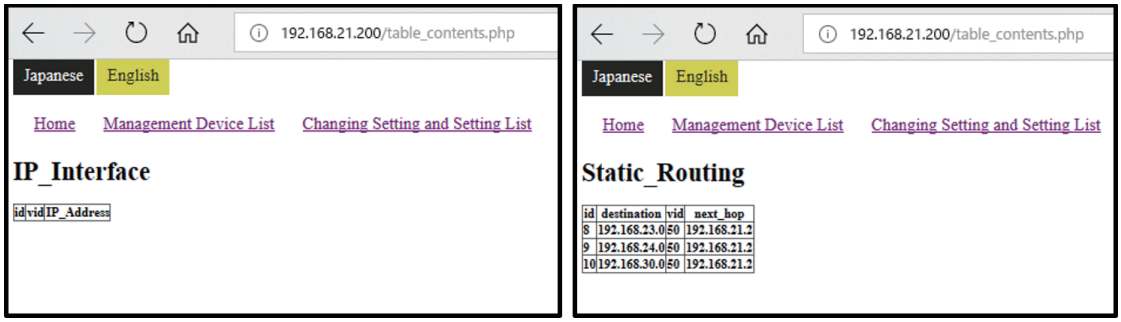
3.2. **User interface.** Figure 6 shows the UI used by the network administrator. Figure 6(a) shows the transition of the UI when accessing the web server from a web browser and selecting L3SW2 from the list of managed devices. The left side of the final UI by the transition of the web page shows the setting list referring to the database. On the right side, there are boxes for entering parameters for executing some functions from Layer-1 to Layer-3, and after entering parameters in each function and clicking the send button, control commands can be sent to the web server and the database can be updated. In Figures 6(b) and 6(c), the left side shows the UI of referring to the database of IP Interface in L3SW2 of the initial configuration and after the experiment, respectively, and the right side shows the UI of referring to the database of static routing table in L3SW1.



(a) User interface on web browser in manager PC



(b) Database reference in initial settings



(c) Database reference after executing functions

FIGURE 6. User interface

3.3. **Experimental procedure.**

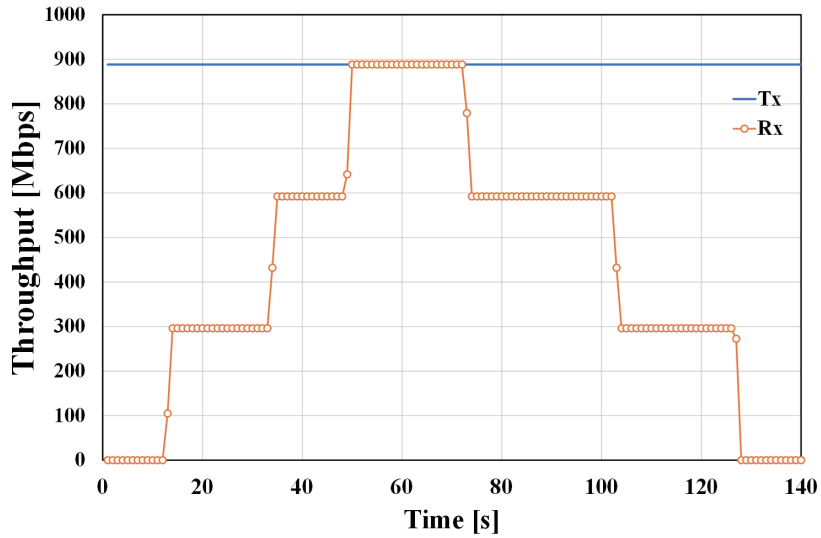
3.3.1. *System verification with increasing IP-throughputs.* As shown in Figure 5, static entries were added to the routing table in L3SW1. Router tester had three destinations, and the three destinations were configured one by one so that traffic could flow to each destination. In other words, after the initial settings, the manager PC sent the control command to add the static routing table three times. When the control command was correctly sent to L3SW1 according to the processing flow depicted in Figure 4, the router tester's receiving throughput increased by about 0.3 Gbps for each sending control command. Therefore, it was possible to verify the work of the proposal system by observing the receiving throughput of the router tester and seeing the change.

3.3.2. *System verification with decreasing IP-throughputs.* After completing the addition static entries of the routing table in 3.3.1, the IP interfaces of L3SW2 were removed as shown in Figure 5. In other words, the three IP interfaces initially set in L3SW2 were removed in order by operating the manager PC. When the control command is correctly sent to L3SW2 according to the processing flow shown in Figure 4, the router tester's receiving throughput drops by about 0.3 Gbps for each sending control command until it reaches 0 Gbps. Therefore, it was possible to verify the work of the proposal system from the change of the throughput as well as 3.3.1.

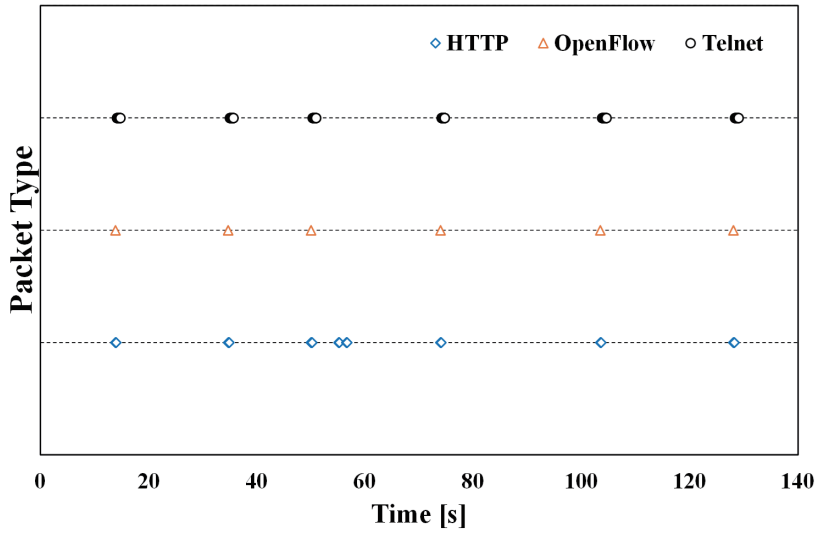
3.3.3. *System verification with packet analysis.* As shown in Figure 5, packet capture was performed using LAN analyzer. We set the port mirroring function for the ports connected to the management PC, control server, and Raspberry Pi in layer-3 switch 2, and all packets are mirrored from each device to the layer-3 switch on the analyzer. Wireshark, one of the packet capture tools, was used to capture the major packets in the experiment. In other words, a total of six OpenFlow packets were observed in the experiments of 3.3.1 and 3.3.2 and the capture of these packets confirmed the correct work of the proposed system.

3.4. **Results of experimental demonstration.** Figure 7(a) shows the observed changes in the receiving throughput in the router tester, and Figure 7(b) shows the results of packet capture. In Figure 7(a), the receiving throughput increases three times at approximately 300 Mbps, and then decreases three times at approximately 300 Mbps before reaching 0 bps. This result indicates that the addition of the static routing table and deletion of the IP interface described in Sections 3.3.1 and 3.3.2 were carried out correctly. Furthermore, in Figure 7(b), OpenFlow packets are correctly captured six times, described in Section 3.3.3, and changes in receiving throughput begin with the observation of OpenFlow packets within a few seconds of each other. Therefore, two results confirm that the proposed system was working properly.

Next, we explain how much the proposed system can reduce the cost compared to replacing all the legacy network devices with SDN-enabled network devices. Table 1 shows the price of the SDN-enabled white-box switch in 2020 and the prices of the Raspberry Pi and control server used in the proposal system and software. As shown in Table 1, depending on performance, SDN-enabled network device costs about \$2,900-\$3,900 [21] and replacing all the legacy network devices with SDN-enabled ones will require a huge cost. The price of a Raspberry Pi, on the other hand, is around \$33.9 [34] and Raspberry Pi in the proposed method only requires the number of nodes in the network, so the network construction cost is low. Furthermore, operating system (OS) for the Raspberry Pi and OpenFlow switch are both free software. The control server is also a general-purpose, low-cost one, and both the operating system and the Ryu [35] used as the SDN controller



(a) Receiving throughput change of router tester



(b) Packet analysis

FIGURE 7. Results of experiments

TABLE 1. Comparison of cost

	Reference [21]		Proposal System		
Hardware, Software/OS	SDN-enabled white-box switch		Raspberry Pi 3 Model B+	Control server (HP Elite Desk 705 G3 SFF)	• Ryu (OF Controller) • Open vSwitch (OF Switch)
	24 ports 10 Gbps	40 ports 10 Gbps	Raspbian 10 (Free)	CentOS 7.9 (Free)	
Cost	$2,900	$3,900	$33.9	$590	Free

are free software. For these reasons, the proposed system can realize the deployment of SDN at a low cost.

Finally, we explain how much the proposed system in this research can reduce the cost compared to our previous system [23-27]. In the previous system, the number of Raspberry Pi required is the same as the number of legacy network devices. However, in the proposed

system, the number of Raspberry Pi is the same as the number of network nodes. In general, dozens or hundreds of the legacy network devices are included in a network node. It is possible to reduce the number of Raspberry Pi as single-board computer. Therefore, the proposed system can significantly reduce the system construction cost compared to our previous system.

4. **Conclusions.** In this paper, we proposed a system that added new single-board computers as a low-cost method to achieve centralized management by SDN. When compared to replacing all legacy network devices with SDN-enabled ones, the cost could be significantly reduced. The central idea is to use single-board computers to translate SDN control commands into vendor-specific control commands. Furthermore, this translation function can provide some level of security for network devices that only support non-encrypted communication protocols like Telnet. In addition, linking the SDN controller with external applications can achieve high functionality and contribute to reducing the management workload for network administrators. In this study, we adopted a web server and a database for advanced functionality to remove the management restriction based on the implemented location of the SDN controller, and by storing the configuration information of network devices in the database, the administrator could refer to it at any time. We also created an experimental network that mimicked a portion of the proposal system and tested its functionality. As a result of the receiving throughput change of the router tester and packet capture, the proposal system was found to be operationally sound. However, the number of network devices that a single Raspberry Pi can manage has yet to be validated and the controller and single-board computer are the single point of failure in each other. In the future, the number of network devices that can be managed by one single-board computer will be investigated and the single point of failure in the proposed system will be overcome.

Acknowledgment. This work was supported by JSPS KAKENHI Grant Number 16K06-306.

REFERENCES

[1] W. Li, D. Guo, K. Li, H. Qi and J. Zhang, iDaaS: Inter-datacenter network as a service, *IEEE Transactions on Parallel and Distributed Systems*, vol.29, no.7, pp.1515-1529, 2018.

[2] N. C. Luong, P. Wang, D. Niyato, Y. Wen and Z. Han, Resource management in cloud networking using economic analysis and pricing models: A survey, *IEEE Communications Surveys & Tutorials*, vol.19, no.2, pp.954-1001, 2017.

[3] M. Noormohammadpour and C. S. Raghavendra, Datacenter traffic control: Understanding techniques and tradeoffs, *IEEE Communications Surveys & Tutorials*, vol.20, no.2, pp.1492-1525, 2018.

[4] I. Farris, T. Taleb, Y. Khettab and J. Song, A survey on emerging SDN and NFV security mechanisms for IoT systems, *IEEE Communications Surveys & Tutorials*, vol.21, no.1, pp.812-837, 2019.

[5] A. H. Mohammed, R. M. Khaleefah, M. K. Hussein and I. A. Abdulateef, A review software defined networking for Internet of Things, *Proc. of 2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, Ankara, Turkey, pp.1-8, 2020.

[6] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky and S. Uhlig, Software-defined networking: A comprehensive survey, *Proceedings of the IEEE*, vol.103, no.1, pp.14-76, 2015.

[7] R. Mijumbi, J. Serrat, J. Gorricho, N. Bouten, F. De Turck and R. Boutaba, Network function virtualization: State-of-the-art and research challenges, *IEEE Communications Surveys & Tutorials*, vol.18, no.1, pp.236-262, 2016.

[8] W. Zhou, L. Li, M. Luo and W. Chou, REST API design patterns for SDN northbound API, *Proc. of the 28th International Conference on Advanced Information Networking and Applications Workshops*, Victoria, BC, Canada, pp.358-365, 2014.

[9] C. Banse and S. Rangarajan, A secure northbound interface for SDN applications, *Proc. of 2015 IEEE Trustcom/BigDataSE/ISPA*, Helsinki, Finland, pp.834-839, 2015.

[10] D. Adami, G. Antichi, R. G. Garroppo, S. Giordano and A. W. Moore, Towards an SDN network control application for differentiated traffic routing, *Proc. of 2015 IEEE International Conference on Communications (ICC)*, London, UK, pp.5827-5832, 2015.

[11] B. Kar, E. H. Wu and Y. Lin, The budgeted maximum coverage problem in partially deployed software defined networks, *IEEE Transactions on Network and Service Management*, vol.13, no.3, pp.394-406, 2016.

[12] D. Levin, M. Canini, S. Schmid, F. Schaffert and A. Feldmann, Panopticon: Reaping the benefits of incremental SDN deployment in enterprise networks, *Proc. of 2014 USENIX Annual Technical Conference*, Philadelphia, PA, pp.333-345, 2014.

[13] A. H. Fakhteh, V. Sattari-Naeini and H. R. Naji, Increasing the network control ability and flexibility in incremental switch deployment for hybrid software-defined networks, *Proc. of 2019 9th International Conference on Computer and Knowledge Engineering (ICCKE2019)*, Mashhad, Iran, pp.263-268, 2019.

[14] D. F. T. Pontes, M. F. Caetano, G. P. R. Filho, L. Z. Granville and M. A. Marotta, On the transition of legacy networks to SDN – An analysis on the impact of deployment time, number, and location of controllers, *Proc. of the IM 2021 – 2021 IFIP/IEEE International Symposium on Integrated Network Management*, Bordeaux, France, pp.367-375, 2021.

[15] R. Amin, M. Reisslein and N. Shah, Hybrid SDN networks: A survey of existing approaches, *IEEE Communications Surveys & Tutorials*, vol.20, no.4, pp.3259-3306, 2018.

[16] H. Kim, J. Kim and Y. Ko, Developing a cost-effective OpenFlow testbed for small-scale software defined networking, *Proc. of the 16th International Conference on Advanced Communication Technology: Content Centric Network Innovation*, Pyeongchang, Korea, pp.758-761, 2014.

[17] A. N. Toosi, J. Son and R. Buyya, CLOUDS-Pi: A low-cost Raspberry-Pi based micro data center for software-defined cloud computing, *IEEE Cloud Computing*, vol.5, no.5, pp.81-91, 2018.

[18] V. Gupta, K. Kaur and S. Kaur, Developing small size low-cost software-defined networking switch using Raspberry Pi, *Advances in Intelligent Systems and Computing*, vol.638, pp.147-152, 2018.

[19] M. Ariman, G. Seçinti, M. Erel and B. Canberk, Software defined wireless network testbed using Raspberry Pi of switches with routing add-on, *Proc. of 2015 IEEE Conference on Network Function Virtualization and Software Defined Network*, San Francisco, CA, USA, pp.20-21, 2015.

[20] L. Csikor, L. Toka, M. Szalay, G. Pongrácz, D. P. Pezaros and G. Rétvári, HARMLESS: Cost-effective transitioning to SDN for small enterprises, *Proc. of 2018 IFIP Networking Conference (IFIP Networking) and Workshops*, Zurich, Switzerland, pp.208-216, 2018.

[21] L. Csikor, M. Szalay, G. Rétvári, G. Pongrácz, D. P. Pezaros and L. Toka, Transition to SDN is HARMLESS: Hybrid architecture for migrating legacy ethernet switches to SDN, *IEEE/ACM Transactions on Networking*, vol.28, no.1, pp.275-288, 2020.

[22] S. S. W. Lee, K. Li and M. Wu, Design and implementation of a GPON-based virtual OpenFlow-enabled SDN switch, *Journal of Lightwave Technology*, vol.34, no.10, pp.2552-2561, 2016.

[23] S. Aso, Y. Tomioka, O. Koyama, T. Niihara, Y. Ogura and M. Yamada, Web-based remote management system for optical switch in AWG-STAR with loopback function, *Proc. of the 23rd Opto-Electronics and Communications Conference*, Jeju, Korea, P1-03, 2018.

[24] K. Minou, S. Aso, O. Koyama, M. Yamaguchi, Y. Tomioka, Y. Ogura, K. Ikeda and M. Yamada, OpenFlow-based remote control of optical switch employing IoT device in AWG-STAR with loopback function, *Proc. of the 24th OptoElectronics and Communications Conference*, Fukuoka, Japan, WP4-G1, 2019.

[25] A. Imae, K. Mino, O. Koyama, K. Oyama, M. Yamaguchi, K. Ikeda and M. Yamada, Router control function using IoT device supported OpenFlow switch in IP over AWG-STAR network, *Proc. of the 25th OptoElectronics and Communications Conference*, Taipei, Taiwan, VP74, 2020.

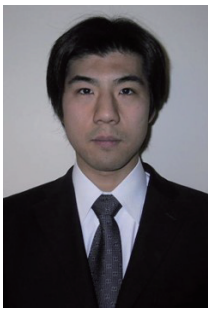
[26] A. Imae, O. Koyama, K. Mino, I. Tomo, M. Yamaguchi, K. Oyama, K. Ikeda and M. Yamada, Cost-effective router/switch control system based on software-defined networking over world wide web, *International Journal of Innovative Computing, Information and Control*, vol.17, no.5, pp.1617-1627, 2021.

[27] K. Mino, A. Imae, O. Koyama, I. Tomo, M. Yamaguchi, K. Oyama, K. Ikeda and M. Yamada, Event-driven remote configuration function in cost-effective router/switch control systems based on software-defined networking using IoT devices, *ICIC Express Letters, Part B: Applications*, vol.13, no.2, pp.145-153, 2022.

[28] F. Hu, Q. Hao and K. Bao, A survey on software-defined network and OpenFlow: From concept to implementation, *IEEE Communications Surveys & Tutorials*, vol.16, no.4, pp.2181-2206, 2014.
[29] W. Braun and M. Menth, Software-defined networking using OpenFlow: Protocols applications and architectural design choices, *Future Internet*, vol.6, no.2, pp.302-336, 2014.
[30] M. Alsaeedi, M. M. Mohamad and A. A. Al-Roubaiey, Toward adaptive and scalable OpenFlow-SDN flow control: A survey, *IEEE Access*, vol.7, pp.107346-107379, 2019.
[31] *Open Network Foundation*, https://www.opennetworking.org/, Accessed in December 2021.
[32] *Raspberry Pi*, https://www.raspberrypi.org/, Accessed in December 2021.
[33] *Wireshark*, https://www.wireshark.org/, Accessed in December 2021.
[34] *Th Pi Hut*, https://thepihut.com/, Accessed in December 2021.
[35] *Ryu SDN Framework*, https://ryu-sdn.org/, Accessed in December 2021.

Author Biography

Akihiro Imae received B.E. degree from Osaka Prefecture University, Japan, in 2020. Currently he is a master's student of engineering at Osaka Prefecture University. His main research interests include Web-based network control and management based on software defined networking using single-board computer.



Osanori Koyama received B.E., M.E. and Ph.D. degrees from Osaka Prefecture University, Japan, in 1999, 2001 and 2013, respectively. Currently he works at Photonic Innovative Systems Research Group as associate professor in Osaka Metropolitan University. His research interests include design and control issues related to optical IP networks based on software defined networking, and optical fiber sensing system over IP network.

Ippei Tomo received B.E. degree from Osaka Prefecture University, Japan, in 2021. Currently he is a research member of Photonic Innovative Systems Research Group in Osaka Metropolitan University. His interests include software development for wavelength path management in IP over AWG-STAR network.

Minoru Yamaguchi received B.E. and M.E. degrees from Osaka Prefecture University, Japan, in 2014 and 2016, respectively. He is currently enrolled in Ph.D. program in engineering at Osaka Prefecture University. His research interests include computation and control methods for wavelength path management in IP over AWG-STAR network.

Kanami Ikeda is assistant professor of the Department of Electrical & Electronic Systems, Osaka Metropolitan University, Japan. She received Ph.D. degree in engineering from the University of Electro-Communications, Japan, in 2018. Her research interests include optical communications and networking, optical information processing, and optical sensing. She is a member of Optica.

Makoto Yamada is professor of the Department of Electrical & Electronic Systems, Osaka Metropolitan University, Japan. He received B.E. and M.E. degrees in electrical engineering from the Technical University of Nagaoka, Niigata, Japan in 1983 and 1985, respectively. He joined NTT Laboratories in 1985, where he was engaged in research on planar lightwave circuits. Since 1989, he has been engaged in research on optical fiber amplifiers. In 1999, he received D.E. degree in the area of optical amplifiers. He joined Osaka Prefecture University in 2008, and has been the professor since 2013. His research interests include design & control for optical amplifiers and other components in optical networks.

Prof. Yamada is a Senior Member of the Institute of Electrical and Electronics Engineers (IEEE), a Senior Member of the Institute of Electronics, Information and Communication Engineers of Japan (IEICE), a Member of Japan Society of Applied Physics, and a Fellow of the Optical Society (OSA).

Prof. Yamada received the Paper Award (1994) from the IEICE, the Electronics Letters Premium (1997) from the Institution of Electrical Engineers, and the Sakurai Prize (1998) and Meritorious Award of the 40th foundation anniversary (2021) from the Optoelectronics Industry and Technology Development Association (OITDA).