Load Balancing Algorithm in a Software-Defined Network Environment with Round Robin and Least Connections

Chandra Wijaya[1,2], Rita Wiryasaputra[1,3], Chin-Yin Huang[1], Jodi Tanato[2], and Chao-Tung Yang[4,5(✉)]

[1] Department of Industrial Engineering and Enterprise Information, Tunghai University, Taichung 407224, Taiwan
huangcy@go.thu.edu.tw
[2] Department of Informatics, Parahyangan Catholic University, Bandung 40141, Indonesia
chandraw@unpar.ac.id
[3] Department of Informatics, Krida Wacana Christian University, Jakarta 11470, Indonesia
rita.wiryasaputra@ukrida.ac.id
[4] Department of Computer Science, Tunghai University, Taichung 407224, Taiwan
ctyang@thu.edu.tw
[5] Research Center for Smart Sustainable Circular Economy, Tunghai University, Taichung 407224, Taiwan

Abstract. In a traditional computer network, each device has its configuration. The Software Defined Network (SDN) architecture ensures that every device in the network will become a dummy device, which must connect to a controller before action can be taken on every packet that the device receives. Each device does not require manual configuration with the centralized configuration. The huge growth of internet traffic can affect the server's performance when handling requests of many clients. The algorithms, such as Round Robin and Least Connection, influence the load-balancing technology to divide the request to some servers. The server's performance increases after implementing load balancing with the algorithm in testing. The comparative result of the Round Robin and Least Connection algorithms shows that the Least Connection performs better than the Round Robin algorithm. By using load balancing in the SDN architecture, the increasing of the request capacity can be handled by the server.

Keywords: SDN · load balancing · least connection · round robin · Openflow

1 Introduction

The functionality of traditional intermediary network devices is the control and data planes. The control plane is an entity that controls network packets from

D.-J. Deng and J.-C. Chen (Eds.): SGIoT 2023, LNICST 557, pp. 148–157, 2024.
https://doi.org/10.1007/978-3-031-55976-1_15

a source device until it reaches the destination device. The data plane sends the packets using the path determined by the control plane. In a traditional network, each intermediary device controls its control and data planes. In a Software Defined Network (SDN), the control functions are located on a centralized controller. SDN refers to designing and managing network packets in which each device's control plane and data plane are separated [11]. The control plane will be handled by one or more controller(s). The intermediary devices will handle the data plane function that moves a packet from the inbound port to the outbound port, according to the rules given by the controller. Rapid technological developments support sustainable development goals. The Internet of Things (IoT) is a technology that is commonly used to capture data from sensors, store it and then analyze the data for specific needs. The IOT has numerous applications such as in agriculture and aquaculture management, air quality monitoring, water resources monitoring [7]. As more and more sensors are being used, bigger data is generated by the sensors hence requiring increasingly large storage resources over time. Big data is a technology for storing large amounts of data, up to the terabyte scale. Several important aspects of big data technology include scalability, reliability, and efficiency to support large amounts of data [1,9,10]. The growth of internet users forces the server to process client requests rapidly without dropping the requests, even if it is in large quantities. A technique to increase the capacity of the server is load balancing which works by dividing client requests and forwarding it into a group of the same functionality servers. After being processed by the server, the reply is sent back to the clients without the clients knowing which server is responding to their requests. Some algorithms for load balancing techniques include Round Robin, Weighted Round Robin, Least Connection, and Resource Based. Some examples of load balancing technology are Hadoop and Ceph which are distributed storage systems that provide scalable, reliable, and fault-tolerant data storage that supports high-performance data transactions [8]. This research compares the Round Robin and Least Connection algorithms regarding the load balancing implementation in the SDN Controller environment. The paper is structured as follows: the first section reviews the research background, Sect. 2 describes the previous research relevant to this research, Sect. 3 presents the research methodology, and the experiment and conclusions are outlined in Sects. 4 and Sect. 5, respectively.

2 Related Works

Nugroho [2] conducted the performance test on both traditional network infrastructure and a network with SDN infrastructure. Two scenarios were used for the test: the network with and without any generated traffic that loads from 20 Mbps to 100 Mbps. Using Wireshark as a packet capture application and analyzer, the researchers concluded that a network utilizing Software-Defined Networking (SDN) exhibits significantly improved performance compared to a traditional network in terms of reduced latency, decreased delay, increased throughput, and lower packet loss. Pramono [3] implemented HAProxy and Nginx for load balancing servers using the Least Connection approach. Each server's load balancing

will be tested with the same parameter values. Using the Apache JMeter and Apache Benchmark as stress test tools, the test shows a significant difference in throughput (requests/minute) when the leastconn algorithm is used instead of the Round Robin algorithm. Saumendu [5] explained that load balancing allocates the workloads evenly to all the nodes with the purpose of improving the entire system's performance. Tasks come from various clients and are received and distributed along the servers. Different load balancing algorithms measure various testing metric parameters such as response time, throughput, optimizing resource utilization, ability to handle faulty conditions, migration time, and scalability. Rana [4] discussed the challenges of SDN, namely reliability, scalability, low-level interface, performance, and security. Reliability is essential for SDN to validate network management and handle any system failures as it must perform automatic fault detection and then, reroute the traffic to achieve reliability. A feature of SDN is scalability which is the ability to handle the growth of a system, network, or process during daily events. A low-level interface means that the SDN translates a policy into a low-level configuration for the network switches. The SDN framework must be able to translate and coordinate the multiple asynchronous events at the switch. The performance and security of SDN networks may be compromised due to the open interface which can introduce new types of network attacks and degrade the overall performance of the SDN. POX is a Python-based open-source SDN controller used to increase the development and prototyping of network applications; a POX controller comes with mininet. It has several scripts that can be developed further such as DHCP service, layer2 and layer3 forwarding service, network discovery, and many more [6].

3 Methodology

Figure 1 illustrates the overall research stages. To create the network simulation, the infrastructure virtualization employed a Linux machine. Based on the framework, the mininet and the SDN controller were used in the framework. The next stage was the network topology where some nodes played roles as clients and others as servers. The controller and the switches interconnecting the nodes were created to manage all traffic in the network. After the configuration, the Round Robin algorithm and the Least Connection were evaluated with the httperf software that can measure the total requests handled (connection rate/second) and total packet loss of each algorithm. The packet loss and throughput parameters were used as the benchmark to determine the best approach.

4 Experiment

This section explains the implementation of the proposed model. The Hypervisor Oracle Virtual Box was used to create the virtual machine. The operating system was a precompiled Ubuntu Linux, including the SDN Controller and all the required software. The research topology employed the load balancing of

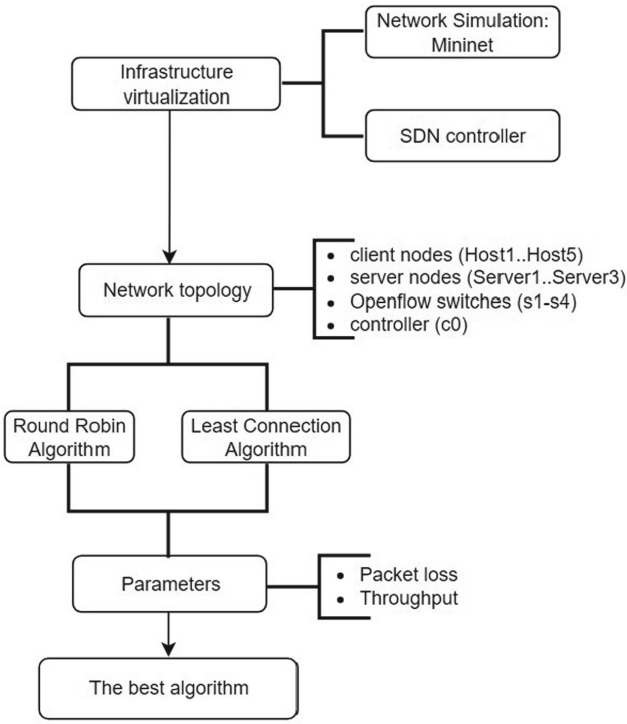


Fig. 1. Research Methodology

client requests to multiple servers, shown in Fig. 2. There were 5 nodes as the clients with the name Host1 until Host5, and 3 servers, namely: server1, server2, and server3, which handled the client requests. A set of switches (s1-s4) also established interconnections among these nodes. The node c0 was designated as the controller responsible for managing all network traffic. In SDN, the switch must support the Openflow protocol which is the protocol used to communicate between the switch and the controller. As such, the network management can be centralized in one controller. In this research, Openvswitch, an Openflow-enabled virtual switch, was used. A virtual switch refers to the capability of forwarding data packets between virtual machines. A virtual switch can forward data packets from a virtual machine to the physical network. Openvswitch can be programmed and controlled by Openflow and OpenvSwitch Database (OVSDB). The SDN Controller utilized in this study is POX, an open-source and openflow-based controller. POX runs in Command Line Interface (CLI), thus lacking any graphical interface. Figure 3 shows the directory structure of the POX SDN Controller. The directory comprises numerous Python scripts. Users might modify or add controller features by modifying or adding the scripts in this directory. These modified or added scripts can then be called when the user runs the POX Controller. In this study, there are two algorithms implemented, specifi-

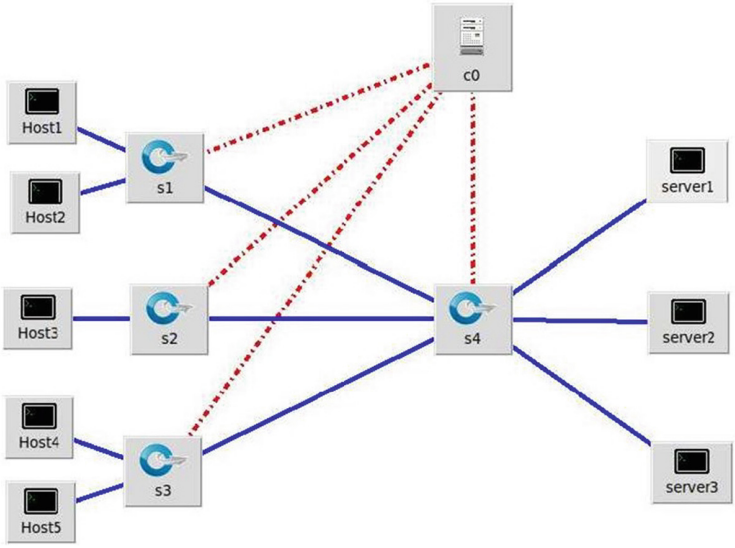
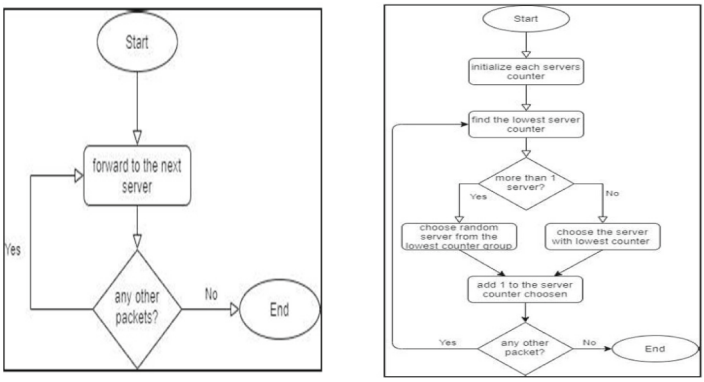


Fig. 2. Research Topology

Fig. 3. POX Directory Structure

cally Round Robin and Least Connection. The flowchart for each algorithm is shown in Fig. 4a, and Fig. 4b. To test the implementation of the load balancing algorithm, the virtual server is added to serve the client requests. When the controller receives the data packet, it will forward the packet to several servers with different IP addresses. The server which responds to the request is automatically chosen by the controller using the load balancing algorithm. Figure 5 depicts the ip of the virtual servers available and which server would respond to the data packet forwarded by the controller. Each server will respond to each one of the data packets sent to them. After testing the connectivity with the virtual server is succeeded, mininet can also emulate the webserver so the load balanc-



(a) Round Robin Algorithm (b) Least Connection Algorithm

Fig. 4. The Flowchart

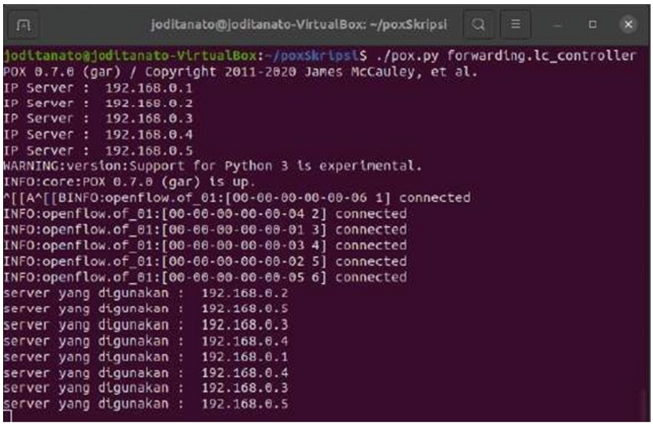
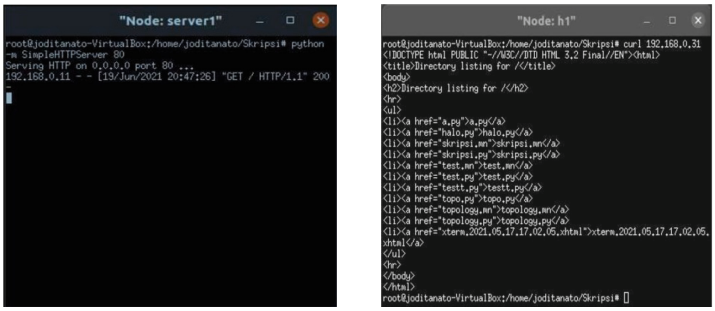


Fig. 5. Testing the Load Balancing Algorithm using Virtual Server

ing algorithm can be conducted. The command "python -m SimpleHTTPServer 80" emulates those webservers. The command instructs the host to run Python script, which runs simple webserver and bind the service to port 80 (http). In the Fig. 6a, shown the node server1, which represents webserver listened on port 80. Figure 6b shows Host1 trying to access the webserver from the Host1 interface by using CURL command. To benchmark the algorithm and the SDN controller, the httperf software was used to simulate an HTTP request and record the total requests, total responses, total errors, the total time needed to test the average/minimum/maximum time needed for the total request, requests sent per second, and replies sent per second. The httperf showed the total requests handled (connection rate/second) and total packet loss of each algorithm. As shown in Fig. 2, there are three servers assigned to handle the requests from the clients.



(a) Mininet host running simple webserver

(b) Host1 try to access the web server using CURL

Fig. 6. Simple Webserver Implementation and Testing

Using the httperf, the experiment sent 100 requests, 200 requests, 300 requests, 400 requests, and 500 requests from clients node Host1, Host2, Host3, Host4, Host5. All requests addressed to the main server will be distributed to the backline servers, according to the algorithm used by the controller. Figure 7 shows the experiment of sending 100 requests from Host1 clients resulting in a connection rate of 44.6 connection/s and the clients received 78 replies, thus, the packet loss rate is 22%. Figure 8 features the experiment of sending 500 requests from the Host5 client to the main server in which the connection rate decreased to only 29.1 connection/s and the client received 471 packets replied hence the packet loss rate is 5,8%. The testing results for the Round Robin Algorithm and the Least Connection algorithm are shown in Table 1. Figure 9 shows the comparison of throughput (in conn/s) between the Round Robin algorithm and the Least Connection algorithm. The performance of the Round Robin algorithm decreases in line with the increase in the requests. However, in the Least Connection algorithm, the performance is stable with a range of 32.52 and 34.00 although the requests are increasing. This means the Least Connection algorithm is better in terms of throughput. Based on the parameter packet loss in the Round Robin algorithm, its packet loss increases in line with the increase of the requests. Even though the Least Connection algorithm experienced a packet loss increase like the Round Robin algorithm, its packet loss has a lower percentage compared to the Round Robin algorithm. The comparison of packet loss from both algorithms is shown in Fig. 10. The algorithm's ideal performance is the algorithm with the lower percentage of packet loss.

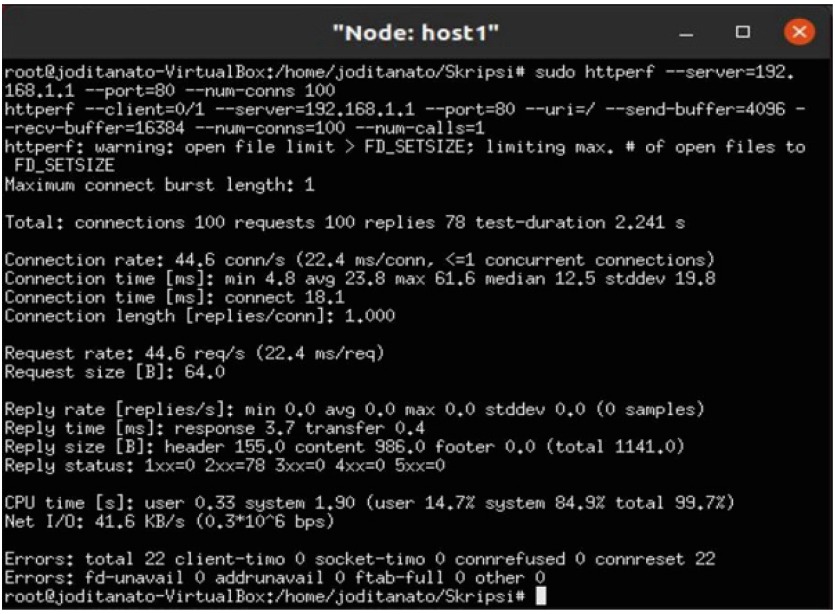


Fig. 7. Sending 100 requests from node Host1

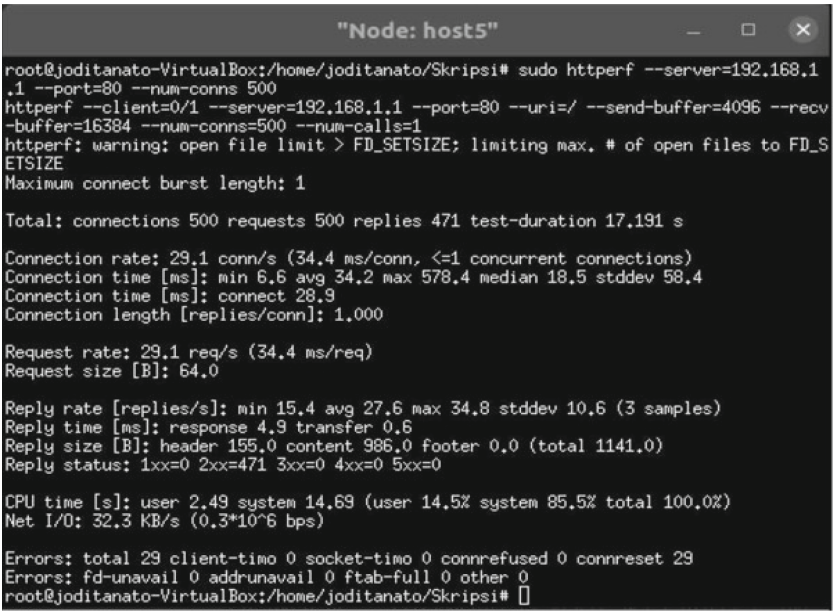


Fig. 8. Sending 100 requests from node Host5

Table 1. Test Result

Request	Round Robin		Least Connection	
	Avg. Conn/s	Avg. Packet Loss (%)	Avg. Conn/s	Avg. Packet Loss (%)
100	40.96	15.20	32.70	10.60
200	34.90	17.20	33.44	11.10
300	33.94	20.67	32.62	12.27
400	32.06	23.30	34.00	15.25
500	29.66	29.40	32.52	17.64

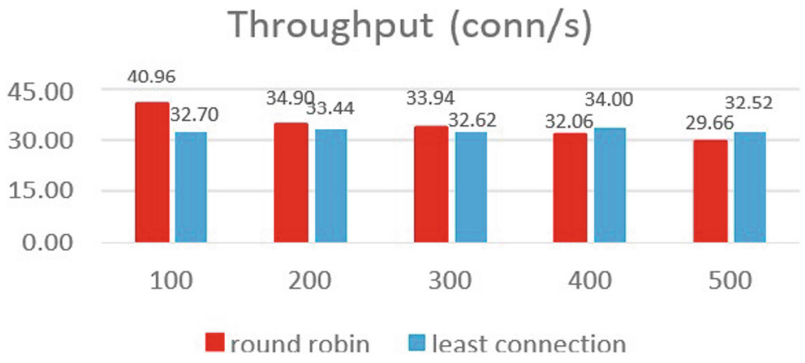


Fig. 9. Throughput Comparison

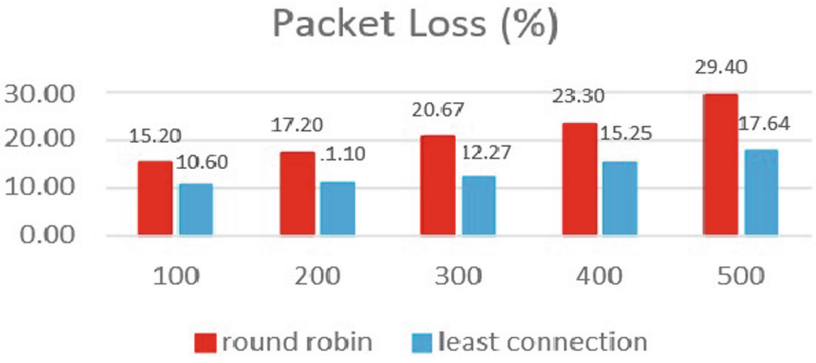


Fig. 10. Packet Loss Comparison

5 Conclusion

The implementation of SDN offers several advantages to processing network packets. Tracking each packet and which route is chosen to forward the packet is possible. The benefits of using an SDN controller are that network manage-

ment can be centralized compared to traditional network management. In terms of increasing network server performance, the load balancing algorithm can be used to improve the network performance, which forwards the requests coming to the server pool. Load balancing distributes the load into several servers, thus, improving the service's reliability since the average throughput can be higher, and the packet loss can be minimized. With the implementation of a load balancing algorithm in the SDN controller, the comparison shows that the least connection algorithm is better than the Round Robin algorithm in terms of throughput (conn/s) and average packet loss.

Acknowledgement. This research was supported in part by the National Science and Technology Council (NSTC), Taiwan R.O.C. grants numbers 112-2622-E-029-003,112-2621-M-029-004, and 110-2221-E-029-020-MY3.

References

1. Chang, C.-H., Jiang, F.-C., Yang, C.-T., Chou, S.-C.: On construction of a big data warehouse accessing platform for campus power usages. J. Parallel Distrib. Comput. **133**, 40–50 (2019)
2. Nugroho, H., Irfan, M., Faruq, A.: Software defined networks: a comparative study and quality of services evaluation. Sci. J. Inform. **6**(2), 181–192 (2019)
3. Pramono, L., Cokro, R., Waskito, Y.: Round-robin algorithm in HAProxy and Nginx load balancing performance evaluation: a review, pp. 367–372 (2018)
4. Rana, D.S., Dhondiyal, S.A., Chamoli, S.K.: Software defined networking (SDN) challenges, issues and solution. Int. J. Comput. Sci. Eng. **7**(1), 884–889 (2019)
5. Roy, S., Hossain, D.M.A., Sen, S.K., Hossain, N., Al Asif, M.R.: Measuring the performance on load balancing algorithms. Global J. Comput. Sci. Technol. **19**, 41–49 (2019)
6. Tok, M.S., Demirci, M.: Security analysis of SDN controller-based DHCP services and attack mitigation with DHCPguard. Comput. Secur. **109**, 102394 (2021)
7. Wiryasaputra, R., Huang, C.Y., Kristiani, E., Liu, P.Y., Yeh, T.K., Yang, C.T.: Review of an intelligent indoor environment monitoring and management system for covid-19 risk mitigation. Front. Public Health **10**, 1022055 (2023)
8. Yang, C.-T., Chen, C.-J., Tsan, Y.-T., Liu, P.-Y., Chan, Y.-W., Chan, W.-C.: An implementation of real-time air quality and influenza-like illness data storage and processing platform. Comput. Hum. Behav. **100**, 266–274 (2019)
9. Yang, C.-T., Chen, H.-W., Chang, E.-J., Kristiani, E., Nguyen, K.L.P., Chang, J.-S.: Current advances and future challenges of AIoT applications in particulate matters (PM) monitoring and control. J. Hazard. Mater. **419**, 126442 (2021)
10. Yang, C.-T., Chen, S.-T., Den, W., Wang, Y.-T., Kristiani, E.: Implementation of an intelligent indoor environmental monitoring and management system in cloud. Futur. Gener. Comput. Syst. **96**, 731–749 (2019)
11. Yang, C.-T., Chen, S.-T., Liu, J.-C., Yang, Y.-Y., Mitra, K., Ranjan, R.: Implementation of a real-time network traffic monitoring service with network functions virtualization. Futur. Gener. Comput. Syst. **93**, 687–701 (2019)