Optimal Controller Selection Scheme Using Artificial Bee Colony and Apriori Algorithms in SDN

Kyung Tae Kim(✉)

College of Computing and Informatics, Sungkyunkwan University, 2066, Seobu-ro, Jangan-gu, Suwon, Korea
kyungtaekim76@gmail.com

Abstract. Software Defined Networking (SDN) is one of the most recent Internet technology that manages the large scale network. SDN decouples the control plane from data plane, which simplifies the logic of network devices and reduces the cost of the network infrastructure. The control plane is the key component of a network which ensures smooth management and operation of the entire network. Distributed SDN controllers have been proposed to solve the scalability and a single point of failure problem. It is a critical issue for the switch to find the optimal controller among the distributed controllers. In this paper we propose a novel scheme for controller selection in distributed SDN environments. The proposed scheme decides optimal controller from distributed controllers by applying the Artificial Bee Colony (ABC) algorithm for meta-heuristic search and Apriori algorithm for effective association rule mining between switch and controller. Computer simulation reveals that the proposed scheme consistently outperforms the scheme employing only ABC and Apriori algorithms separately in terms of response time, arrival rate, number of messages, and accuracy.

Keywords: Software Defined Network · Distributed controllers · Artificial Bee Colony algorithm · Apriori algorithm · Selection of controller · Edge computing

1 Introduction

The explosive growth of the Internet of Things (IoT) and mobile devices leads to an explosion of new applications and services, increasing the burden of what today's Internet could carry [1]. Especially, since the data collected from a lot of devices can generate excessive traffic, several researchers have tried to solve this issue using Edge Computing [2, 3]. To cope with the numerous and diverse IoT devices, the edge computing infrastructure has to support a lot of connected devices and the processing of the massive data collected and complex applications. However, the edge server in edge computing has limited computational and processing resources compared to high-end servers in the cloud server [4]. Therefore, to support high scalability, ultra- low latency, high throughput, and reliable transmission of data, the SDN paradigm is regarded as one of the suitable solutions [5–7].

L. Uden and I-H. Ting (Eds.): KMO 2023, CCIS 1825, pp. 347–359, 2023.
https://doi.org/10.1007/978-3-031-34045-1_28

SDN is an innovative technology in the field of computer network that separates data transmission function and control function from each other, allowing the users the flexibility of using the functions of the network in their own devices [8, 9]. SDN has numerous advantages including direct programming, centralized management, fast delivery, and flexibility. In SDN a single controller may be used as a centralized controller. If excessive packets need to be processed, however, its performance is significantly degraded because of limited processing capacity and distance to the switching devices [2]. Meanwhile, the failure of a single controller may lead to the collapse or congestion of entire network. In order to effectively resolve these issues, the distributed controller architecture was proposed [10–12]. Distributed controller architectures with more than one controller could be used to address some of the challenges of a single SDN controller such as availability [13]. Furthermore, distributed controllers can reduce the latency or increase the scalability and fault tolerance of the SDN deployment. However, this architecture increases the lookup overhead of communication between switches and distributed controllers. Moreover, this approach is difficult to maintain the consistent state in the overall distributed system [14, 15].

In this paper we propose a novel scheme which allows a switch to select an optimal controller from distributed controllers in SDN for edge computing environments. The proposed scheme decides optimal controller based on the data and weight of the controllers using the improved ABC algorithm based on Apriori algorithm for effective association rule mining between switch and controller. It can achieve controller optimization while keeping the excellent performance of the improved ABC algorithm. In the proposed scheme, the priority of each controller was determined by considering the computing and communication capacity of the controllers using an improved ABC algorithm, which includes Apriori algorithm and FCFS (First-Come-First-Served) policy. The proposed scheme significantly reduces the communication latency between the switch to the controller by selecting an optimal controller compared to the existing schemes. Also, the proposed scheme can solve the consistency problem by employing the meta-heuristic association rule mining algorithm. Furthermore, through the uniformly distributed controller, the proposed scheme increase scalability, connectivity, and flexibility of the network, which increases the communication efficiency and reduces the propagation delay of the link. Computer simulation reveals that the proposed scheme consistently outperforms the scheme employing only ABC and Apriori algorithm separately in terms of response time, arrival rate, number of messages, and accuracy.

The rest of the paper is organized as follows. Section 2 introduces the related work for proposed scheme. Section 3 describes the proposed scheme in a detailed manner. The experimental results of the proposed scheme are explained in Sect. 4. Finally, Sect. 5 concludes the paper and describes future research directions.

2 Related Work

2.1 Software Defined Networking

SDN is an effective networking paradigm which makes it easier for the network manager to control the network [16, 18]. SDN is a conceptual architecture that decouples the control plane and data plane of the network and enables network partitioning [5]. SDN architecture is divided into three categories as shown in Fig. 1: the physical infrastructure layer, the controllable control layer, and the application layer. This lets SDN not only create complex paths that cannot be configured in existing networks but also effectively cope with changing traffic patterns and quickly configure the virtual networks required in cloud environments. The control plane in SDN is decoupled from the data plane by drawing the networking functions from the forwarding devices as shown in Fig. 1. The separation of the control plane and the data plane of the network has the advantage of being able to respond more quickly to a malfunction caused by a problem and increase the flexibility and availability of the network. The control functions are deployed to logically centralized controllers so that they can be implemented on a centralized software platform [17]. Using a single, centralized controller might be efficient since the overloaded switch can migrate to a new controller from the previously connected one.

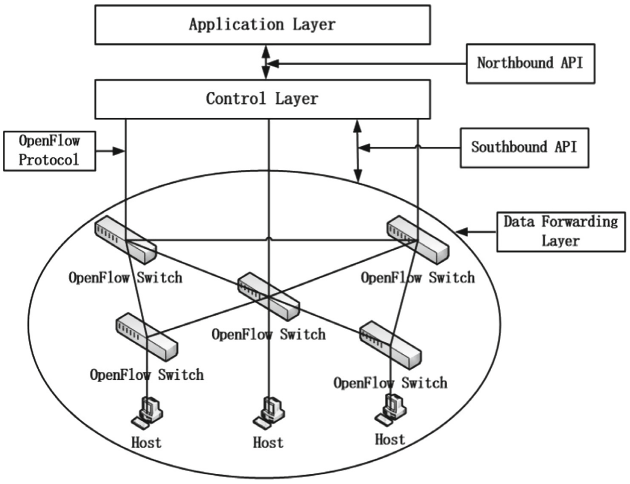

Fig. 1. The three-layer structure of SDN

2.2 OpenDaylight

OpenDaylight (ODL) is a modular open SDN platform for the networks of any size and scale [19, 20]. By sharing YANG data structure in the common data store and messaging infrastructure, ODL allows for fine-grained services to be created and combined together to solve more complex problems. In the ODL Model Driven Service Abstraction Layer (MD-SAL), any app or function can be bundled into a service that is loaded into the controller. The model-driven approach is being increasingly used in the networking domain to describe the functionality of network devices [21], services [22], policies [23, 24], and network APIs [25]. The protocols of choice are NETCONF and RESTCONF; the modeling language of choice is YANG. NETCONF [26] is an IETF network management protocol that defines configuration and operational conceptual data stores and a set of Create, Retrieve, Update, Delete (CRUD) operations that can be used to access these data stores. RESTCONF is a model that describes the mapping of YANG data to a REST-ful API [27, 28]. It is a REST-based protocol that runs over HTTP and is used to access YANG defined data, using Network Configuration Protocol (NETCONF) defined data stores. The YANG data modeling language is used to define the data sent over NETCONF [29]. It can model both the configuration data as well as the manipulated state data.

OpenDaylight SDN controller has several layers. The top layer consists of business and network logic applications. The middle layer is the framework layer, and the bottom layer consists of physical and virtual devices. The middle layer is the framework in which the SDN abstractions can manifest. This layer hosts north-bound and south-bound APIs. The controller exposes open north-bound APIs which are used by applications. OpenDaylight supports the OSGi framework and bidirectional REST for the northbound API. The business logic resides in the applications above the middle layer. The applications use the controller to gather network intelligence, run algorithms to perform the analytics, and then use the controller to orchestrate new rules, if any, throughout the network. ODL supports multi-controllers composing a cluster. If there is only a single ODL controller, it works individually. The multi-controller structure could avoid the consequence of single controller crash, and controllers directly communicate with each other rather than via data plane.

2.3 Artificial Bee Colony Algorithm

ABC algorithm is one of the more recent swarm intelligence based optimization algorithms for solving multidimensional optimization problems [30]. Figure 2 is the flowchart of ABC algorithm.

The intelligent behavior of honey bee colony which search new food sources around their hive was considered to compose the algorithm. In the algorithm, the colony of artificial bees consists of three groups of bees called employed bees, onlookers and scouts. While a half of the colony consists of the employed artificial bees, the other half includes the onlookers. There is only one employed bee for every food source. That is, the number of employed bees is equal to the number of food sources around the hive. The main steps of the algorithm are given below.

First, source initialization is the initial source to a random value.

$$X_{ij} = X_{minj} + rand(0, 1)(X_{maxj} - X_{minj}) \tag{1}$$

Second, the employed bee searches the neighbor source and estimates the amount of nectar of the source, and informs the onlooker bee of the source of higher fitness.

$$V_{ik} = X_{ik} + rand(-1, 1)(X_{ik} - X_{jk}) \tag{2}$$

Third, here a source is selected probabilistically by onlooker bee based on the source discovered by employed bee and the estimated amount of nectar. Onlooker bee selects the source by

$$P_i = \frac{f(X_i)}{\sum f(X_n)} \tag{3}$$

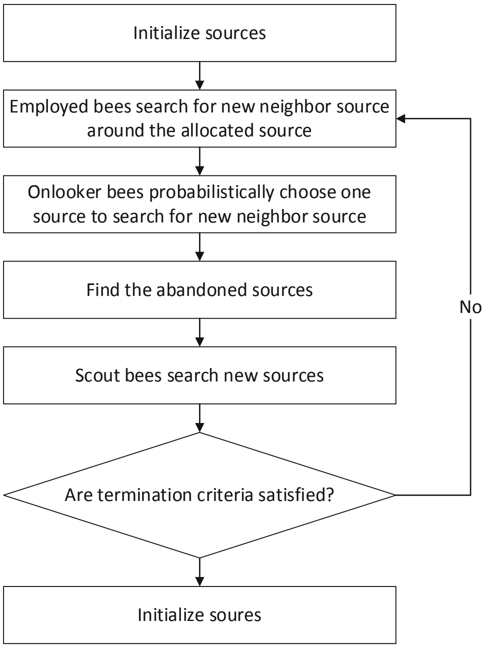
Initialize sources
Employed bees search for new neighbor source around the allocated source
Onlooker bees probabilistically choose one source to search for new neighbor source
Find the abandoned sources
Scout bees search new sources
Are termination criteria satisfied?
No
Initialize soures

Fig. 2. The Flowchart of ABC Algorithm

2.4 Apriori Algorithm

Data Mining is a way of obtaining undetected patterns or facts from massive amount of data in a database. Association rule mining is a major technique in the area of data mining. Association rule mining finds frequent itemsets from a set of transactional databases. Apriori algorithm is one of the earliest algorithms of association rule mining [31, 32]. Apriori employs an iterative approach known as level-wise search. In Apriori, $(k + 1)$ itemsets are generated from k-itemsets. First, scan the database for count of each candidate and compare candidate support count with minimum support count to generate set of frequent 1-itemsets. The set is denoted as $L1$. Then, $L1$ is used to find $L2$, set of frequent 2-itemsets, which is further used to find $L3$ and so on, until no more frequent k-itemsets can be found [33]. After finding set of frequent k-itemsets, it is easy to generate strong association rules. The process of finding each L_k requires the database to be scanned completely once. To improve the efficiency of the level-wise generation of frequent itemsets, an important property called the Apriori property, presented is used to reduce the search space. In Apriori property, all nonempty subsets of a frequent itemset must also be frequent. A two-step process is used to find the frequent itemsets: join and prune actions.

1) The join step: To find L_k a set of candidate k-itemsets is generated by joining L_k–1 with itself. This set of candidates is denoted C_k.
2) The prune step: The members of C_k may or may not be frequent, but all of the frequent k -itemsets are included in C_k. A scan of the database to determine the count

of each candidate in C_k would result in the determination of L_k (i.e., all candidates having a count no less than the minimum support count are frequent by definition, and therefore belong to L_k). To reduce the size of C_k, the Apriori property is used as follows. Any (k–1)-itemset that is not frequent cannot be a subset of a frequent k-itemset. Hence, if any (k–1)-subset of a candidate k-itemset is not in L_k–1, then the candidate cannot be frequent either and so can be removed from C_k.

3 The Proposed Scheme

3.1 Basic Operation

The proposed scheme gathers data from all the controllers to measure how often each controller is used, and then finds the association rules based on the items run frequently by the controllers. In the SDN distributed controller environment, the controllers are selected by Apriori algorithm according to the frequency of use. For this, the data of all the controllers are initialized to random values, and then the neighbor controllers of each controller are searched regarding the amount of nectar. Using the transaction support of Apriori algorithm, the controller's goodness of fit is estimated. In order to minimize the time for searching the association rules, the FCFS (First-Come-First-Served) policy is applied. If there exists a priority rule, the rule is selected first. Then the remaining rules are found.

In searching the source only the one of the highest value is selected, while the others are discarded. This is for minimizing the communication cost. Figure 3 is the flowchart of the proposed scheme for selecting an optimal controller based on the data and weight of the controllers.

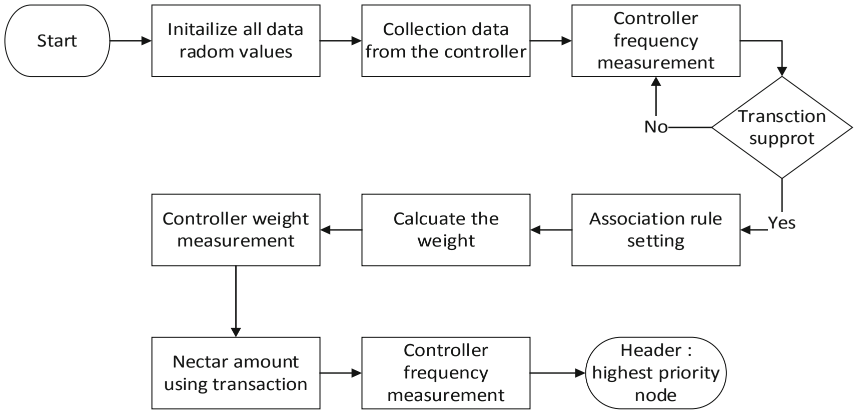
Start
Initailize all data radom values
Collection data from the controller
Controller frequency measurement
Transction supprot
No
Yes
Association rule setting
Calcuate the weight
Controller weight measurement
Nectar amount using transaction
Controller frequency measurement
Header : highest priority node

Fig. 3. The Flowchart of the proposed scheme

3.2 Priority and Weight of Controller

The priority of the controllers is decided using the ABC algorithm as follows. First, the data of the controllers are initialized to random value, and then the distance between a controller and its neighbor one is measured. The fitness of the controllers is then calculated using the transaction support of the Apriori algorithm and weight of them. The weight of each controller is defined to represent the throughput. The performance of a controller is affected by various factors such as distance to communicating controller, bandwidth, transmission delay, load, and packet loss probability, etc. Only the most frequently used controller identified by the proposed approach is selected, while the remaining controllers are excluded. By selecting the most frequently used controller, collision between the controllers and communication load can be reduced. Also, the detailed selection algorithm of controller is shown in Algorithm 1.

The following is to calculate the weight of controller_v, $w(v)$.

$$w(v) = \omega_1 P(v) + \omega_2 S(v) \tag{4}$$

where $\boldsymbol{\omega}_1$ and $\boldsymbol{\omega}_2$ are weight coefficients. $P(v)$ is the operational performance and $S(v)$ is amount nectar of controller_v, respectively.

$$\begin{aligned} W(v, s) =& \alpha \cdot dis(v, s) + \beta \cdot ban(v, s) \\ &+ \gamma \cdot del(v, s) + \delta \cdot load(v, s) \end{aligned} \tag{5}$$

In Eq. (5) $0 \leq \ \alpha, \beta, \gamma, \lambda \leq 1, \alpha + \beta + \gamma + \lambda = 1$. The parameter of distance, bandwidth, delay, and load should be normalized:

$$\mathrm{dis}(w_v, w_s) = \frac{dis(w_v, w_s)}{\sum_{i,j \in l(v,s) \& i \neq j} dis(w_i, w_j)}$$

$$\mathrm{ban}(w_v, w_s) = \frac{ban(w_v, w_s)}{\sum_{i,j \in l(v,s) \& i \neq j} ban\left(w_i, w_j\right)}$$

$$\mathrm{del}(w_v, w_s) = \frac{del(w_v, w_s)}{\sum_{i,j \in l(v,s) \& i \neq j} del\left(w_i, w_j\right)}$$

$$\mathrm{load}(w_v, w_s) = \frac{load(w_v, w_s)}{\sum_{i,j \in l(v,s) \& i \neq j} load\left(w_i, w_j\right)}$$

Then the weight of w_v to all other $(n - 1)$ controllers in the network is:

$$All - Weight(w_v) = \sum\nolimits_{\mathrm{s=1\&s \neq v}}^{n} weight(w_v, w_s) \tag{6}$$

The weight of w_v is calculated by

$$W(w_v) = \mu \cdot weight(w_v) + \sigma \cdot All - Weight(w_v) \tag{7}$$

ALGORITHM 1 Selection of Controller

1. Assume that the path of P_1 is (x_1, y_1, z_1) and that of P_2 is (x_2, y_2, z_2). The controller synchronization occurs with y_1 and y_2, and one of the new paths obtained is P_3: (x_1, y_2, y_1, z_1).
2. Deleting the duplicated switch, the new path P_3 is decided.
3. By the same way, another new path P_4 is obtained.
4. Applying the fitness function to P_1, P_2, P_3, P_4, an optimal path is selected.

3.3 Transaction

The neighbor source of an assigned source is checked, and the source of more amount of nectar is notified to the onlooker bee. They are also weighted, and the one of low nectar amount is discarded. Next, based on the source searched by employed bee and the estimated nectar amount, the onlooker bee selects the source to search. In this way, the onlooker bee selects the source of the largest amount of nectar among the ones the employed bee found.

Let S and C denote a switch and controller, respectively. Each food source, $\boldsymbol{X_C^S}$, in the population is represented as

$$X_C^S = \left\{X_C^S, X_C^S, \ldots, X_C^S\right\}, \forall s \in N \forall c \in P_s \tag{8}$$

where P_s the set of controllers and N is the number of switches. It estimates the amount of nectar a switch can have from the controllers of identical schedule number. Equation (9) is used for deciding the fitness of a switch connected to a controller, F_{cs}:

$$F_{cs} = C_{cs} + \theta_{cs}, \forall cs \in P_s \tag{9}$$

where C_{cs} indicates the sum of coverages for all schedule numbers with complete coverage, θ_{cs} shows the maximum incomplete coverage and F_{cs} represents the fitness value for the cs in the controller.

Next, each new cs in the controller for each switch is generated by only updating.

$$V_c^s = \begin{cases} s, \quad pri > 0.5 \ \forall c \in P_s, \forall c \in N_{nc} \\ X_c^s, \ otherwise \qquad s \in \varphi \end{cases} \tag{10}$$

Each switch will select a certain number s in the incomplete schedule vector φ as a priority.

3.4 Selection Manager

The controller of a higher weight is needed to have more networking operation. In the proposed scheme, the controller appropriate for leading the update process is elected according to the weight. The network manager has the privilege of commanding the entire network since it keeps the network view. The network manager has the privilege of commanding the entire network since it keeps the network view. The priority of controller_i, P_i, is defined as

$$P_i = \begin{cases} \lfloor S * W(c_i) \rfloor \ without\ manager \\ \quad \infty \qquad\quad with\ manager \end{cases}$$

where S is the scale of the network. The controllers broadcast their priority and receive the priority of other controller in a present time. They regard the controller with the highest priority as their header.

The header selection is done following three steps.

Step 1. P_i ($i = 1,2,3,\ldots,n$) are calculated.

Step 2. P_i is broadcast during T_b which is broadcast period. It also receives the priority of other controller.
Step 3. The controller saves the address and priority of the controller which priority is bigger than itself. And it considers the controller with the highest priority as its header. Once a controller receives the priority which is same with itself, its priority will be subtracted one to avoid the same priority.

4 Performance Evaluation

In this section, the performance of the proposed scheme is evaluated via computer simulation. The simulation is performed on a PC consisting of Intel i5-7500 CPU, Window OS, and 8GB memory, and the scheme was implemented with Python and MATLAB. Also, the performance of the proposed scheme is compared with two other schemes to verify its relative effectiveness, the ABC and Apriori algorithm. The test data set is from Stanford Network Analysis Project (SNAP), which is a general purpose network analysis and graph mining library. The controller appropriate to lead the update process is elected to handle massive network of hundreds of millions of nodes and billions of edges. It is efficient for manipulating large graphs, calculates structural properties, generates regular and random graphs, and supports the attributes of nodes and edges.

To investigate the effectiveness of the proposed scheme, we first evaluate the response time with various sizes of data. The comparison results with ABC, Apriori algorithm, and the proposed scheme is shown in Fig. 4. Figure 4 shows that the proposed scheme displays the smallest response time among the three schemes, while the ABC algorithm is the largest. As a result, the proposed scheme effectively reduces the communication overhead in searching the controllers than existing schemes.

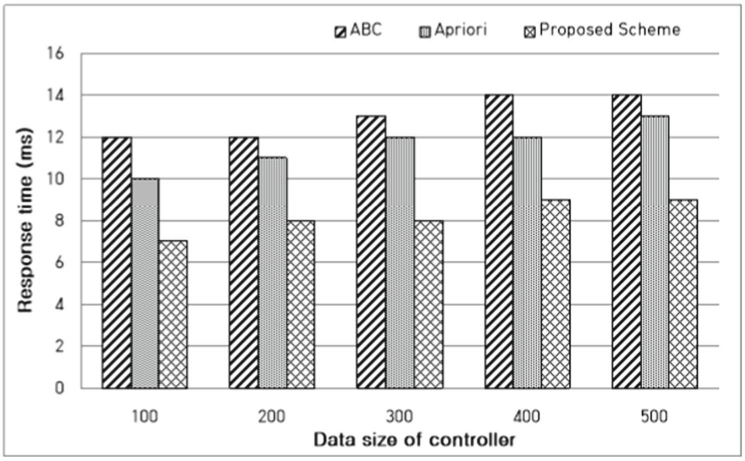

Fig. 4. The comparison of response time between proposed scheme and existing schemes

Arrival rates of the proposed scheme and existing schemes are compared in Fig. 5, which demonstrates that the proposed scheme consistently shows the lowest arrival rate with different sizes of controller data.

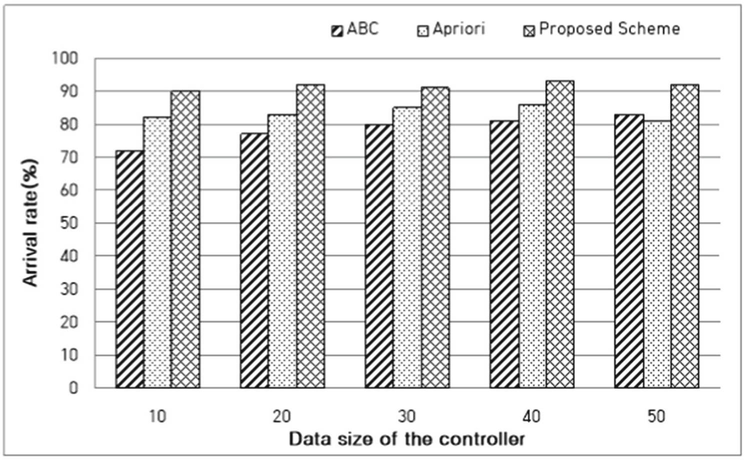

Fig. 5. The comparison of arrival rate between proposed scheme and existing schemes

Figure 6 shows the total amount of messages in ABC, Apriori algorithm, and the proposed scheme. Observe from the Fig. 6 that the proposed scheme always requires the smallest number of messages than ABC and Apriori algorithm. When the size of controller data is between 10 and 30, the ABC algorithm generates more messages than Apriori algorithm. However, Apriori algorithm needs slightly more messages than ABC algorithm after 40. This indicates that Apriori algorithm becomes overloaded when the data size grows beyond a certain level.

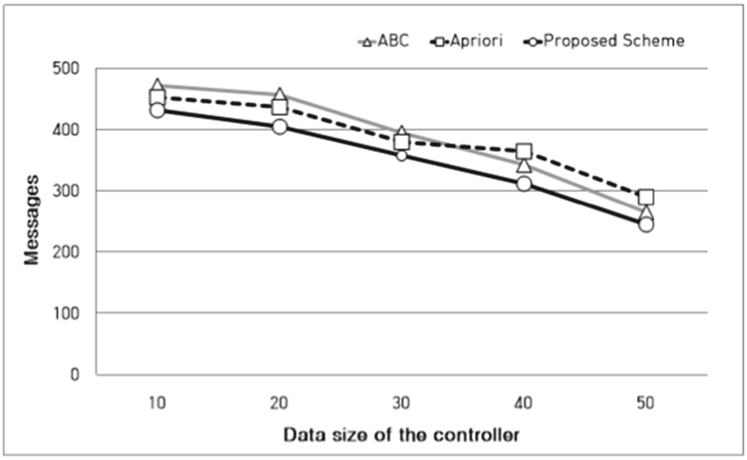

Fig. 6. The total amount of messages between proposed scheme and existing schemes

Figure 7 shows the comparison results of the accuracy in ABC algorithm, Apriori algorithm, and the proposed scheme. In Fig. 7, it demonstrates that the accuracy of the proposed scheme is about 90%, and it is the highest accuracy compared with the accuracies of the existing schemes. The accuracy of the ABC algorithm is the lowest, however, it continues getting higher accuracy as the controller data gets bigger, and it has the same accuracy as the Apriori scheme while the update data is 50 eventually.

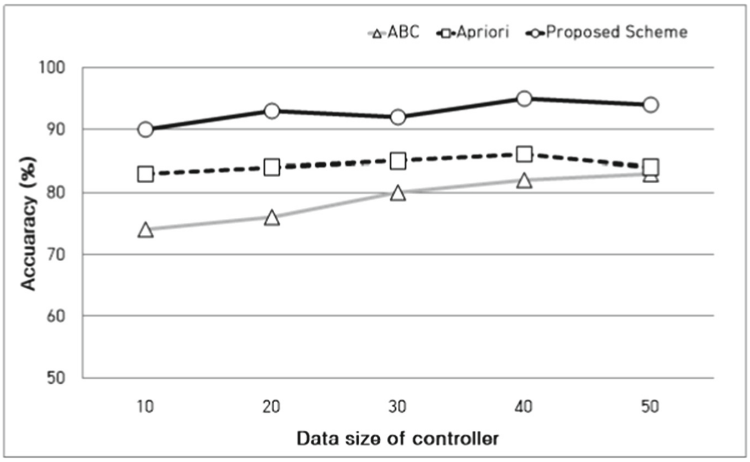

Fig. 7. The comparison of accuracy with the proposed scheme and existing schemes

5 Conclusion

The SDN paradigm shifts control to a centralized omniscient controller. The controller, however, creates a bottleneck due to the enormous amount of message exchanges between the switches and the controller in SDN for edge computing. As a result, inappropriate switch assignment to the controller reduces performance. In this paper, we have proposed a novel scheme which allows a switch to select an optimal controller from distributed controllers in order to reduce communication and propagation latency and improve throughput and reliability in SDN. The optimal controller is selected using the ABC and Apriori algorithms. Also, the priority of each controller in the proposed scheme was determined by considering the computing and communication capacity of the controllers. Moreover, the proposed scheme can solve the consistency problem by employing the meta-heuristic association rule mining algorithm. Computer simulation reveals that the proposed scheme consistently outperforms the ABC and Apriori algorithms in terms of response time, arrival rate, and number of messages exchanged. In the future, we will expand the proposed controller selection scheme by employing more sophisticated scheme such as Gaussian mixture model and artificial intelligence technique. Also, the proposed scheme will also be tested and expanded considering various environments and applications where the requirements on the energy and communication latency are diverse.

Acknowledgments. This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2022R1I1A1A01053800).

References

1. Wang, A., Zha, Z., Guo, Y., Chen, S.: Software defined networking (SDN) enhanced edge computing: a network centric survey. Proc. IEEE **107**(8), 1500–1519 (2019)
2. European Telecommunication Standards Institute, Mobile Edge Computing (MEC), Technical Requirements (ETSI GS MEC 002 V.1.1.1) (2016). https://www.etsi.org/deliver/etsi_gs/MEC/001_099/002/01.01.01_60/gs_MEC002v010101p.pdf. Accessed 10 Jan 2023
3. Mao, Y., You, C., Zhang, J., Huang, K., Letaief, K.B.: A survey on mobile edge computing: the communication perspective. IEEE Commun. Surv. Tutor. **19**(4), 2322–2358 (2017)
4. Lee, C.H., Park, J.S.: An SDN-based packet scheduling scheme for transmitting emergency data in mobile edge computing environments. Hum.-Cent. Comput. Inf. Sci. **11**(28), 2–15 (2021)
5. Open Networking Foundation, Software-Defined Networking (SDN) definition (2021). https://www.opennetworking.org/sdn-definition/. Accessed 10 Jan 2023
6. Shamsan, A.H., Faridi, A.R.: SDN-assisted IoT architecture: a review. In: Proceeding of the 4th International Conference on Computing Communication and Automation (ICCCA), pp. 1–7 (2018)
7. Lv, Z., Xiu, W.: Interaction of edge-cloud computing based on SDN and NFV for next generation IoT. IEEE Internet Things J. **7**(7), 5706–5712 (2019)
8. Kirkpatrick, K.: Software-defined networking. Commun. ACM **56**(9), 16–19 (2013)
9. Khan, S., et al.: Software-defined network forensics: motivation, potential locations, requirements, and challenges. IEEE Netw. **30**(6), 6–13 (2016)
10. Balakiruthiga, B., Deepalakshmi, P.A.: Distributed energy aware controller placement model for software-defined data centre network. Iran. J. Sci. Technol. Trans. Electr. Eng. **45**, 1083–1101 (2021)
11. Radam, N.S., Faraj, S.T., Jasim, K.S.: Multi-controllers placement optimization in SDN by the hybrid HSA-PSO algorithm. Computers **11**(7), 1–26 (2022)
12. Blial, O., Mamoun, M.B., Benaini, R.: An overview on SDN architectures with multiple controllers. J. Comput. Netw. Commun. **2016**(2), 1–8 (2016)
13. Hakiri, A., Gokhale, A., Berthou, P., Schmidt, D.C., Gayraud, T.: Software-defined networking: challenges and research opportunities for future internet. Comput. Netw. **75**(24), 453–471 (2014)
14. Xiao, L., Zhu, H., Xiang, S., Vinh, P.C.: Modeling and verifying SDN under Multi-controller architectures using CSP. Concurr. Comput. Pract. Exp. 1–17 (2019)
15. Sahoo, K.S., et al.: ESMLB: efficient switch migration-based load balancing for multicontroller SDN in IoT. IEEE Internet Things J. **7**(7), 5852–5860 (2020)
16. Xue, H., Kim, K.T., Youn, H.: Dynamic load balancing of software-defined networking based on genetic-ant colony optimization. Sensors **19**(2), 1–17 (2019)
17. Ahmad, S., Mir, A.H.: SDN Interfaces: protocols, taxonomy and challenges. Int. J. Wirel. Microwave Technol. **2**, 11–32 (2022)
18. Farhady, H., Lee, H., Nakao, A.: Software-defined networking: a survey. Comput. Netw. **81**, 79–95 (2015)
19. OpenDaylight Association, Opendaylight. https://www.opendaylight.org/. Accessed 12 Jan 2023
20. Eftimie, A., Borcoci, E.: SDN controller implementation using OpenDaylight: experiments. In: Proceedings of the 13th International Conference on communications, Bucharest, pp. 1–5 (2020)
21. Clemm, A.: Navigating device management and control interfaces in the age of SDN (2014). http://blogs.cisco.com/getyourbuildon/navigating-device-managementand-control-interfaces-in-the-age-of-sdn. Accessed 13 Jan 2023

22. Wallin, S., Wikstrom, C.: Automating network and service configuration using NETCONF and YANG. In: Proceedings of the 25th Large Installation System Administration (LISA), pp. 1–13 (2011)
23. Application centric infrastructure object-oriented data model: gain advanced network control and programmability. http://docplayer.net/15876333-Application-centric-infrastructure-object-oriented-data-model-gain-advanced-network-control-and-programmability.html. Accessed 13 Jan 2023
24. Cisco Systems, The Cisco Application Policy Infrastructure Controller. https://www.cisco.com/c/en/us/products/collateral/cloud-systems-management/aci-fabric-controller/at-a-glance-c45-730001.html. Accessed 12 Jan 2023
25. Alghamdi, A., Paul, D., Sadgrove, E.: Designing a RESTful northbound interface for incompatible software defined network controllers. SN Comput. Sci. **3**, 1–7 (2022)
26. Enns, R., Bjorklund, M., Schoenwaelder, J., Bierman, A.: Network configuration protocol (NETCONF) (2011). https://www.rfc-editor.org/rfc/rfc6241. Accessed 12 Jan 2023
27. Bierman, A., Bjorklund, M., Watsen, K., Fernando, R.: RESTCONF protocol, draft-bierman-netconf-restconf-04 (2014). https://datatracker.ietf.org/doc/draft-bierman-netconf-restconf/. Accessed 12 Jan 2023
28. Jethanandani, M.: YANG, NETCONF, RESTCONF: what is this all about and how is it used for multi-layer networks. In: Proceedings of the 2017 Optical Fiber Communications Conference and Exhibition (OFC), Los Angeles, CA, USA, pp. 1–65 (2017)
29. Bjorklund, M.: YANG - a data modeling language for the network configuration protocol (NETCONF), RFC 6020. https://www.rfc-editor.org/rfc/rfc6020. Accessed 12 Jan 2023
30. Karaboga, D.: Artificial bee colony algorithm. Scholarpedia (2010)
31. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: Proceedings of the 20th International Conference on Very Large Data Bases, pp. 487–499 (1994)
32. Zareian, M.M., Mesbahb, M., Moradic, S., Ghateec, M.I.: A combined Apriori algorithm and fuzzy controller for simultaneous ramp metering and variable speed limit determination in a freeway. AUT J. Math. Comput. **3**(2), 237–251 (2022)
33. Hu, X.G., Wang, D.X., Liu, X.P., Guo, J., Wang, H.: The analysis on model of association rules mining based on concept lattice and Apriori algorithm. In: Proceedings of 2004 International Conference on Machine Learning and Cybernetics, Shanghai, China, pp. 1620–1624 (2004)