Iğdır Üniversitesi Fen Bilimleri Enstitüsü Dergisi, 11(4): 2456-2466, 2021
Journal of the Institute of Science and Technology, 11(4): 2456-2466, 2021
ISSN: 2146-0574, eISSN: 2536-4618
Bilgisayar Mühendisliği / *Computer Engineering* **DOI: 10.21597/jist.932575**
Araştırma Makalesi / *Research Article*
Geliş tarihi / Received: 04.05.2021 **Kabul tarihi / Accepted: 15.06.2021**

Atıf İçin: Babayiğit B, Ulu B, 2021. Yüksek kullanılabilirliğe sahip SDN çoklu-denetleyici yapısı ve optimize edilmiş k-means algoritması ile SDN'nin çoklu-denetleyicilerinin yerleştirilmesi. Iğdır Üniversitesi Fen Bilimleri Enstitüsü Dergisi, 11(4): 2456-2466.

To Cite: Babayigit B, Ulu B, 2021. A high available multi-controller structure for SDN and placement of multi-controllers of SDN with optimized k-means algorithm. Journal of the Institute of Science and Technology, 11(4): 2456-2466.

A High Available Multi-Controller Structure for SDN and Placement of Multi-Controllers of SDN with Optimized K-means Algorithm

Bilal BABAYİĞİT[1], Banu ULU[1*]

ABSTRACT: Facilitating the management of the traditional networks, Software Defined Networking (SDN), separates data plane and control plane, so providing advantages such as programmability, flexibility, and cost-effective configuration. But, SDN has some problems such as security, infrastructure, single-point-of-failure, and controller placement. A single-point-of-failure problem can be solved with a multi-controller; however, it needs to be improved. The most critical issue in solving the multi-controller placement problem is minimizing latency between controllers and their associated switches. In this paper, an SDN-based multi-controller system using Docker-swarm mode is presented to solve the single-point-of-failure problem, and using the presented system, the multi-controller placement problem is solved with optimized k-means (Opk-means) in order to reduce the end-to-end latency. The experimental results show that the proposed testbed provides a high availability control plane for multi-controller, and the Opk-means algorithm significantly reduces the latency when compared to the standard k-means in the testbed.

Keywords: Software defined network, docker swarm mode, multi-controller, controller placement, optimized k-means

[1]Bilal BABAYİĞİT (**Orcid ID:**0000-0002-2923-5263), Banu ULU (**Orcid ID:**0000-0002-3593-0756), Erciyes Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, Kayseri, Türkiye

***Sorumlu Yazar/Corresponding Author:** Banu ULU, e-mail: banudaglioglu@hotmail.com

* Bu çalışma BU'nun Doktora tezinden üretilmiştir.

INTRODUCTION

Big data, cloud services, and the Internet of Things (IoT) are new computing and information technologies. These technologies fundamentally change the way we store, obtain, and transfer information and data. However, these changes are increasing the requirement for a new network because traditional network architectures have several limitations such as complicated management, seller dependent network, predefined policies of communication devices (such as layer three switches and the router), and also they are improper for the dynamic computing, communication technologies, and source capabilities. Software defined networking (SDN) (McKeown, 2009; Nunes et al., 2014) is a promising network technology that eliminates the limitations of the traditional network. The main idea of SDN is to achieve a programmable network and to improve the network performance by decoupling the control plane and data plane. The planes have different responsibilities; while the overall behavior of the network is managed by the control plane, the process of sending and receiving data to other devices in the network is performed with the data plane. So, the data plane is kept the same as in the traditional network, but the control plane is isolated and moved to a central location within the network named controller. The network manager can determine and produce the rules and policies of the routing devices with the assistance of the SDN controller(s), according to the requirements. It is accepted that SDN, network has a single controller. However, with the enlargement of the SDN network, a single controller cannot meet the comprehensive management requirements (Lu et al., 2019). To develop the reliability and scalability of the network and to avoid the single-point-of-failure, the multi-controller network architecture is proposed. In a multi-controller system, new controllers are created on different machine containers to improve SDN control plane availability and to enable continuously processing controller. The machines can be real computer systems or container-based virtualization systems. Today, one of the commonly used container-based virtualization machines is Docker (Bella et al., 2018).

Docker is an open-source project that automates software deployment using Linux containers (Nguyen and Bein, 2017). Docker runs on macOS, Linux distributions, Windows, and various cloud service providers. In contrast to virtual machines, Docker containers are much more functional and do not require a real secondary operating system owing to the advanced level of abstraction that virtualizes only operating system. Managing several containers is a difficult task in creating a single virtual system. On multiple machines, managing a cluster of Docker Engines is solved by Docker-swarm mode. Swarm mode combines the orchestration ability of Docker-swarm into Docker Engine, which is the layer between the operating system and the container images. The Swarm is controlled through a swarm manager to orchestrate and schedule the containers. Also, Swarm has scheduling capabilities to provide enough resources for spanned containers. Swarm dedicates containers to underlying nodes and optimizes the resources by automatically scheduling the container.

Deployed multi-controllers typically must rapidly establish new flows and synchronize global network views to be supported several network applications. This creates a Controller Placement Problem (CPP). CPP mentions how to place controllers in an SDN-enabled network to reduce the overall latency and how many numbers of SDN controllers required for a reliable and flexible network processing and how to distribute associated switches. The overall latency is important in SDN because the control process of the network is decoupled from simplified switches, and all functions of the network are realized through message exchanging between switches and controllers. Overall latency contains the queuing latency and end-to-end latency, which includes packet transmission latency, switch processing latency, and propagation latency (Wang et al., 2018). Recent studies on the overall latency problem have been focused on the reducing of packet propagation latency by using artificial intelligence techniques. In these studies, artificial intelligence techniques such as genetic algorithm (Hu et al., 2017; Sanner et al., 2017; Ahmadi and Khorramizadeh, 2018; Huang et al., 2019), particle swarm algorithm (Sahoo et al., 2017; Liao and Leung, 2017), POCO (Lange et al., 2015) and k-means (Wang et al., 2016) are used.

The k-means algorithm MacQueen (MacQueen, 1967) is used to solve the clustering problems and is one of the basic unsupervised learning algorithms. K-means has a simple procedure to classify a given data set through a determined number of clusters (assume k clusters). It chooses randomly initial centers and calculates the distance with Euclidean distance for clustering. So, this algorithm cannot be directly implemented to partition network topologies. In recent studies (Wang et al., 2016; Wang et al., 2018) k-means algorithms are used to divide the network into different sub-networks for CPP. In these studies, the smallest maximum latency times of the algorithms are compared.

In this paper, we implement an SDN-based multi-controller system using Docker-swarm mode, and multi-controller placement with optimized k-means (Opk-means) is proposed to reduce the end-to-end latency. If the controller connection has failed, high uptime cannot be achieved in the multi-controller structure. In the proposed system, a new controller is automatically created by Docker swarm mode whether the controller is disconnected from the Docker swarm. The high uptime problem is solved by Docker-swarm mode. However, in Docker-swarm mode, the controllers are randomly selected. That is, if one of the controllers shuts down, the controller that will replace it is selected randomly. This prevents the stable configuration of the network structure. To solve this problem, standard k-means (Stdk-means) and Opk-means algorithms are proposed. The Opk-means clustering algorithm is applied to the Ulaknet dataset of Turkey in order to optimally solve for CPP in a multi-controller structure with Docker-swarm mode. In contrast to the previous studies (Wang et al., 2016; Wang et al., 2018), in the paper, the Opk-means algorithm used to determine the number of controllers according to the minimum latency and reduce the end-to-end latency which directly affects the overall latency. With an Opk-means algorithm, eliminating the disadvantages of the Stdk-means algorithm, the end-to-end latency time of the CPP is reduced to an optimum level.

MATERIALS AND METHODS

Optimized k-means (Opk-means)

The Stdk-means algorithm (MacQueen, 1967) involves four main steps:

I. Initialize k clusters and separate one center to each cluster using random sampling.
II. Separate nodes into one of the clusters based on Euclidean distance.
III. Calculate again centroid for each cluster.
IV. Repeat steps 2 and 3 until there is no change in each cluster.

The Stdk-means algorithm cannot be applied directly to network partitioning topologies because, k-means algorithm randomly chooses initial centers using Euclid distances. However, the centroid place, where the controller is placed, is not guaranteed. In order to eliminate, the k -means algorithm should be optimized.

In the first step in the Opk-means algorithm, the central node of the network is randomly selected. Thus, the algorithm will learn better the main center of the network. In the second step, the shortest path from the centroid to all nodes is selected. Each node that has infinite distance at the beginning is assigned to the cluster closest to it. The Dijkstra algorithm is used to select the shortest path. In the third step, the centroid is updated to minimize the sum of the shortest distance from all points. The procedure is ongoing until the network is eventually divided into k sub-networks, which is selected according to the minimum latency time parameter determined. During each partition, the algorithm can reduce the maximum end-to-end latency between controllers and their related nodes. So, the maximum latency is importantly reduced compared with the standard clustering algorithms such as k-means.

Methods

This study is organized into two parts. First, a multiple controller structure is designed and implemented by using Docker to solve the problems caused by a single controller. Docker swarm mode is used to create a new controller when one of the controllers cannot be reached. Second, an Opk-means algorithm is applied to solve the CPP caused by these controllers.

The experimental setup is as follows: First, it is performed on Linux 16.04 operating system. The computer used for the study has an i7 processor with 8 GB Ram capacity. The SDN topology is created in a mininet (Anonymous, 2013) simulation environment. Floodlight (Anonymous, 2012) is used as a controller. Docker (Anonymous, 2013) containers are installed on the Odroid mc1. The jupyter notebook is used to distribute the controllers according to the Opk-means algorithm.

The block diagram of the proposed structure created with Docker swarm mode is shown in Figure 1. The controllers must be created in floodlight. For this purpose, IP, port numbers, and the node ID of each controller should be given in the setting files. Also, a leader is selected among the active controllers. The necessary topology is created in the mininet emulator to see that the active controllers run efficiently.

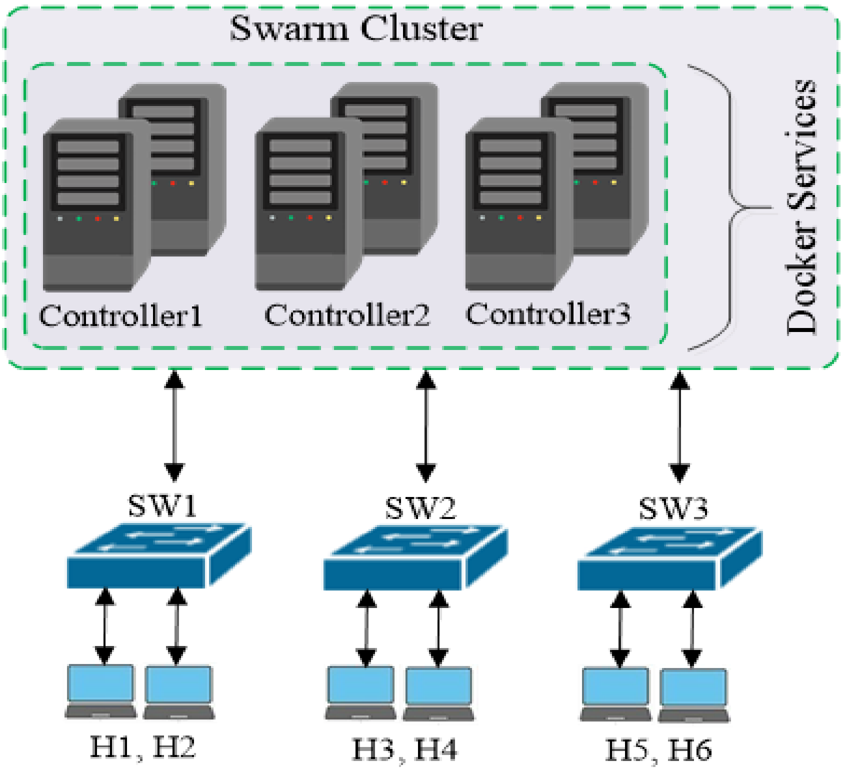


Figure 1. The diagram of the structure created with docker- swarm mode

Odroid mc1 devices are used in order to build the multiple-controller structure in the test environment. The Docker system is installed on the Odroid mc1. The Docker system is used to resume the system running in case of failure to one of the controllers. The controllers are not installed directly on a server; instead of this, the controller is started as a service in the Docker system. Thus, the controller is always accessible from different devices. Docker files are used to create the Docker image. The controllers are turned into an image file. This image file is started as a service in Docker-swarm mode. The controller's 8080 port is transposed as 8090 and initialized with one replica. The purpose of one replica is to ensure that one controller is active in the system. The structure created with Docker-swarm mode implemented is shown in Figure 2.

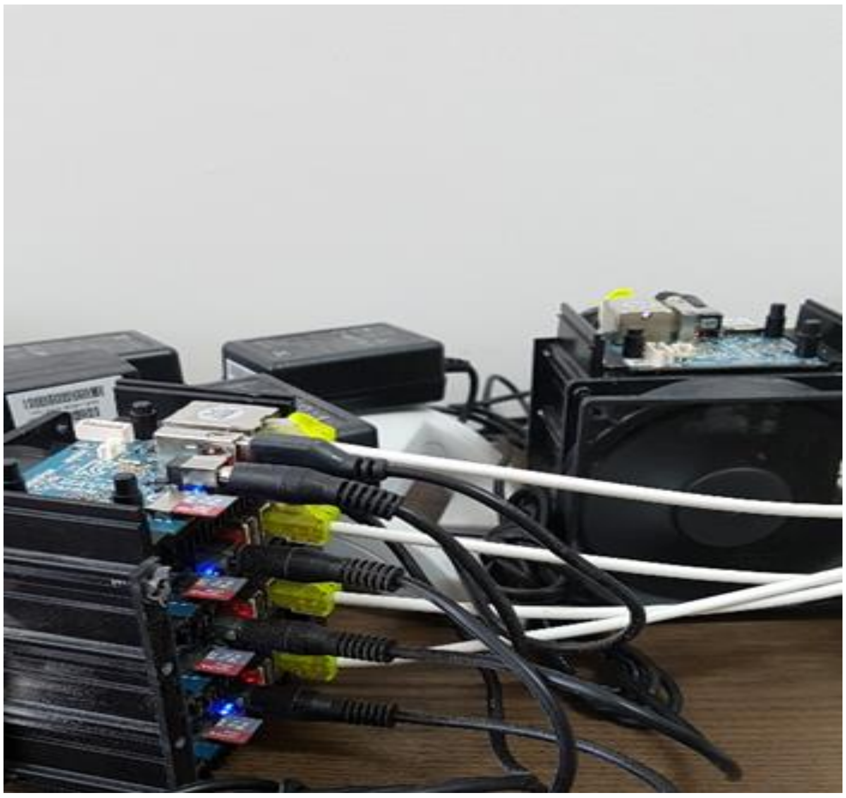
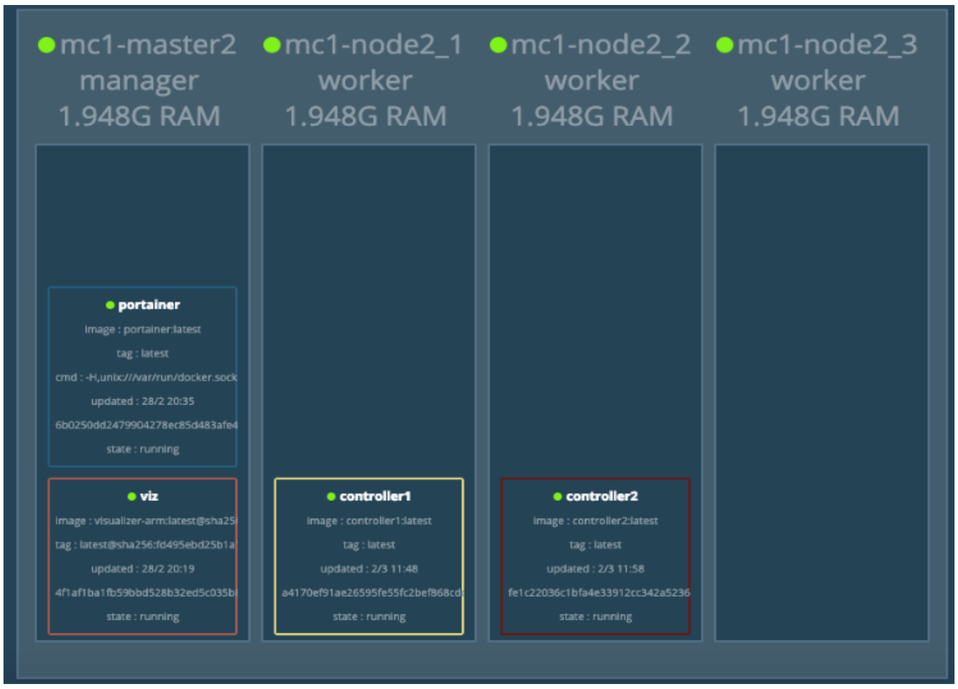


Figure 2. The structure created with docker-swarm mode

Also, visualizer has installed to display active nodes. Which nodes operate the created services can be actively monitored. This system is shown in Figure 3.



```
● mc1-master2        ● mc1-node2_1        ● mc1-node2_2        ● mc1-node2_3
  manager              worker               worker               worker
  1.948G RAM           1.948G RAM           1.948G RAM           1.948G RAM

● portainer
image : portainer:latest
tag : latest
cmd : -H,unix:///var/run/docker.sock
updated : 28/2 20:35
6b0250dd2479904278ec85d483afe4
state : running

● viz                                  ● controller1                      ● controller2
image : visualizer-arm:latest@sha25    image : controller1:latest         image : controller2:latest
tag : latest@sha256:fd495ebd25b1a      tag : latest                       tag : latest
updated : 28/2 20:19                   updated : 2/3 11:48                updated : 2/3 11:58
4f1af1ba1fb59bbd528b32ed5c035b         a4170ef91ae26595fe55fc2bef868cd    fe1c22036c1bfa4e33912cc342a5236
state : running                        state : running                    state : running
```

Figure 3. Controller monitoring system

The services in the Figure 3 are running on node one and node two. However, in the event of any disconnection, another node becomes active and can run with other controllers. There is no latency when selecting the controller. As the controller information is automatically copied to the selected controller, the controller accesses the network information from that copy. Thus, the latency time to scan the network is eliminated.

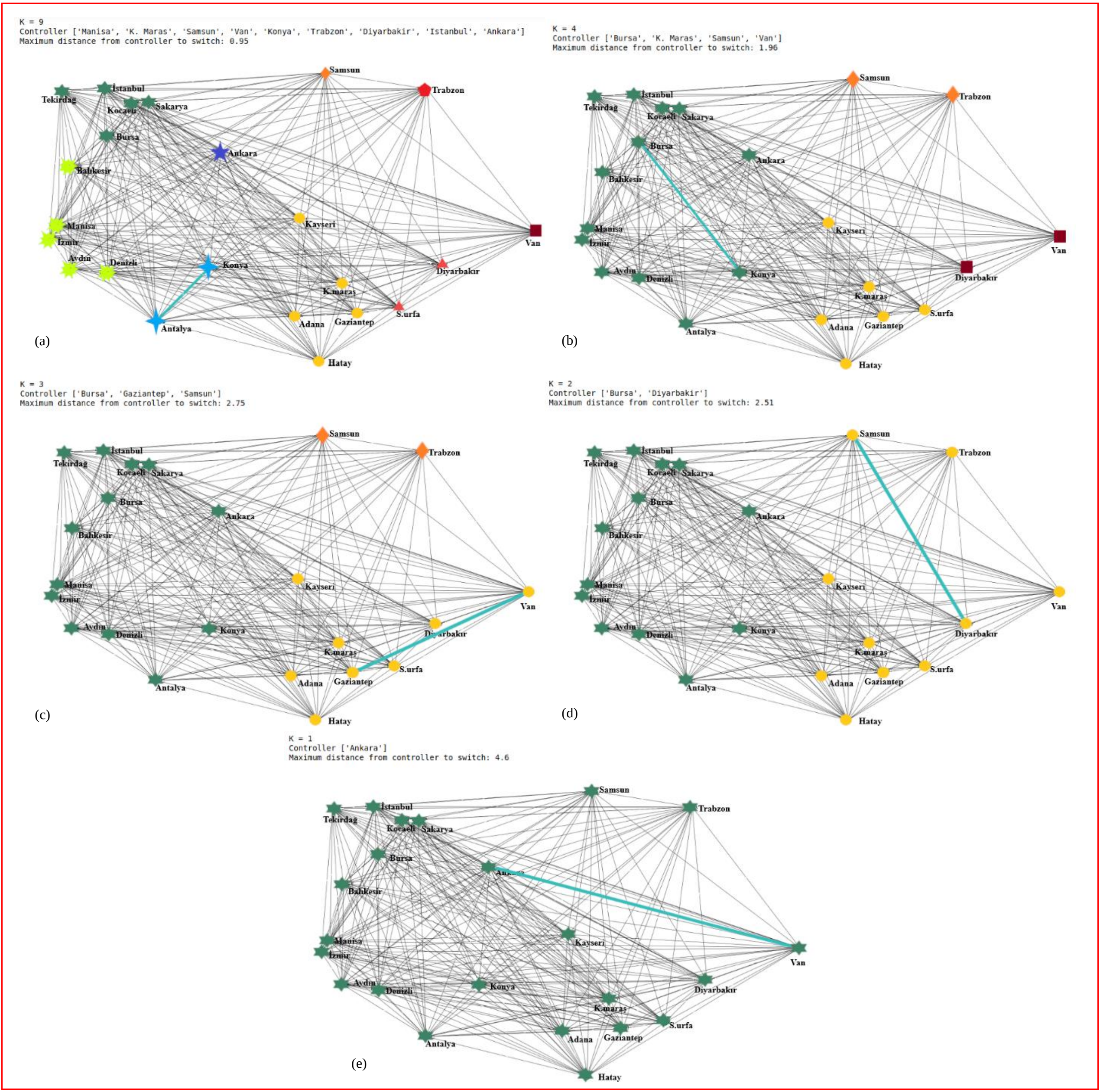
We carry out experiments to observe and investigate the performance of the Opk-means algorithm under the proposed implemented multi-controller scheme in the SDN-based network, as mentioned above. Also, to evaluate the Opk-means algorithm, we compare it with the Stdk-means algorithm.

CPP is to determine which controller should be placed and which switches are under the control of which controller. Opk-means algorithm is to solve CPP of the controllers operating on Dockers, and the Stdk-means algorithm is used in terms of the Opk-means algorithm, the number, and location of the controllers are determined and is intended to minimizing the end-to-end delay between the controllers. Unlike Stdk-means, in the proposed Opk-means algorithm, k parameter varies according to the determined minimum milliseconds value of the end-to-end latency. The Turkey-UlakNet dataset (Anonymous, 2010) from the Topology Zoo website is considered to analyze the proposed k-means, and Stdk-means algorithm in order to precisely determine the controller location is used. First, the Ulaknet data set for the controller and switch points to be placed are obtained from the Topology Zoo database. The Ulaknet data set contains latitude and longitude information of cities. The haversine formula is applied to the cities with latitude and longitude information so that the distance between the cities is measured. The obtained distance values are kept in an array. Besides, two separate functions are created to find the distance of the controllers to the controllers and the controllers to the switches. These functions measure the shortest distances with the Dijkstra shortest path finding algorithm. Opk-means algorithm and the Stdk-means algorithm are run with these functions. The latency is determined as the limiting factor to k in Opk-means. Opk-means algorithm and Stdk-means algorithm steps are applied, and the system is started.

RESULTS AND DISCUSSION

The data obtained as a result of the study are mapped by the graph method. Figure 4 shows the results obtained with Opk-means, and Figure 5 shows the results obtained with the Stdk-means.

In the figures, the controllers are listed in the first. Also, the area which the controller is has, is given in a different color and shape to distinguish it from other controllers.



```
K = 9
Controller ['Manisa', 'K. Maras', 'Samsun', 'Van', 'Konya', 'Trabzon', 'Diyarbakir', 'Istanbul', 'Ankara']
Maximum distance from controller to switch: 0.95
```

```
K = 4
Controller ['Bursa', 'K. Maras', 'Samsun', 'Van']
Maximum distance from controller to switch: 1.96
```

```
K = 3
Controller ['Bursa', 'Gaziantep', 'Samsun']
Maximum distance from controller to switch: 2.75
```

```
K = 2
Controller ['Bursa', 'Diyarbakir']
Maximum distance from controller to switch: 2.51
```

```
K = 1
Controller ['Ankara']
Maximum distance from controller to switch: 4.6
```

Figure 4. Opk-means algorithm for determined latency times (millisecond: ms). (a) one ms (b) two ms (c) three ms (d) four ms (e) five ms

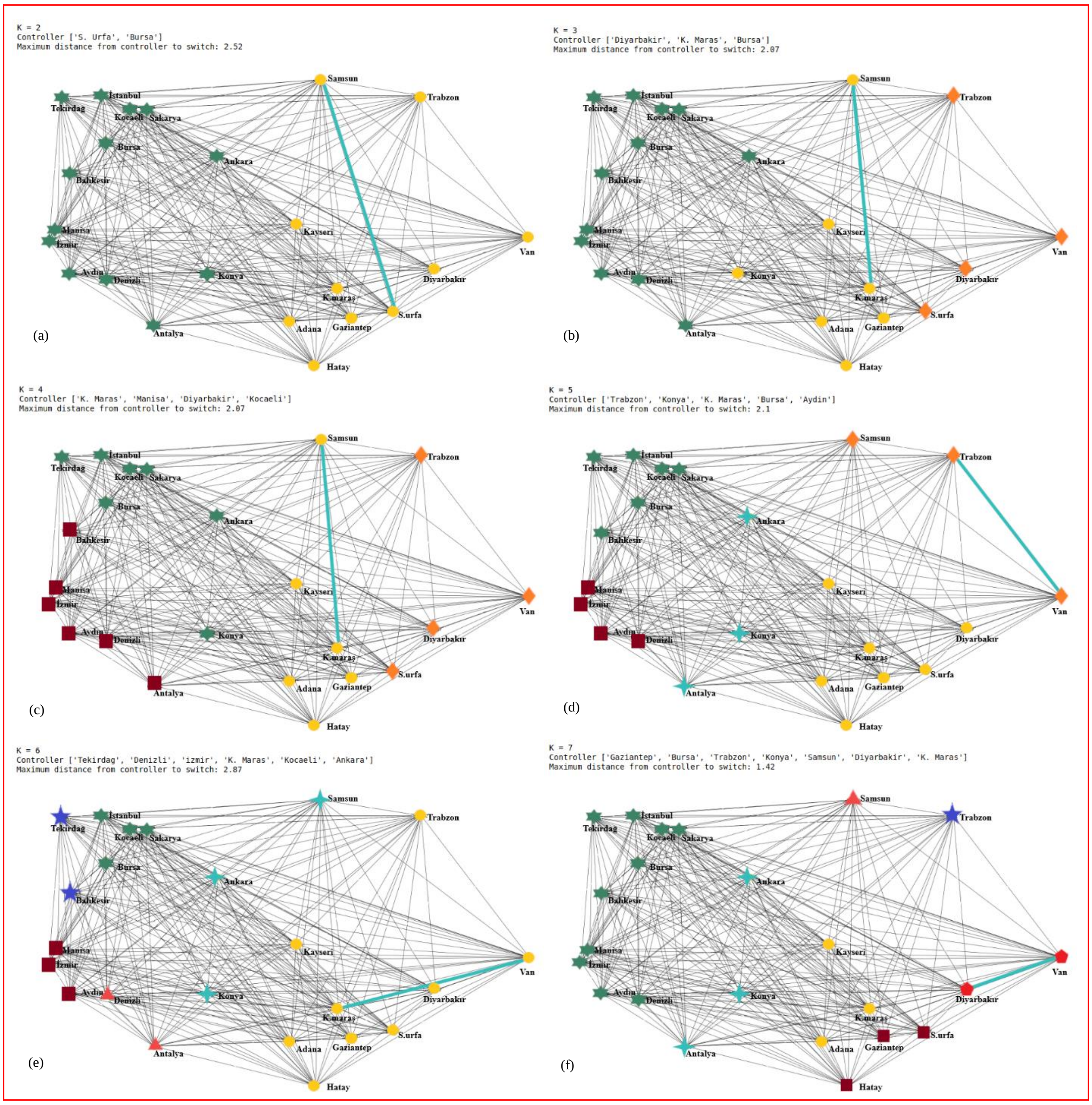

```
K = 2
Controller ['S. Urfa', 'Bursa']
Maximum distance from controller to switch: 2.52
```
```
K = 3
Controller ['Diyarbakir', 'K. Maras', 'Bursa']
Maximum distance from controller to switch: 2.07
```
```
K = 4
Controller ['K. Maras', 'Manisa', 'Diyarbakir', 'Kocaeli']
Maximum distance from controller to switch: 2.07
```
```
K = 5
Controller ['Trabzon', 'Konya', 'K. Maras', 'Bursa', 'Aydin']
Maximum distance from controller to switch: 2.1
```
```
K = 6
Controller ['Tekirdag', 'Denizli', 'izmir', 'K. Maras', 'Kocaeli', 'Ankara']
Maximum distance from controller to switch: 2.87
```
```
K = 7
Controller ['Gaziantep', 'Bursa', 'Trabzon', 'Konya', 'Samsun', 'Diyarbakir', 'K. Maras']
Maximum distance from controller to switch: 1.42
```

Figure 5. Stdk-means algorithm for determined numbers of controllers. (a) two controllers (b) three controllers (c) four controllers (d) five controllers (e) six controllers (f) seven controllers

In the Stdk-means algorithm, the number of controllers is determined externally by us. Moreover, in this algorithm, control plane is randomly selected, so the latency between the controllers does not change by stable, as shown in Figure 6.

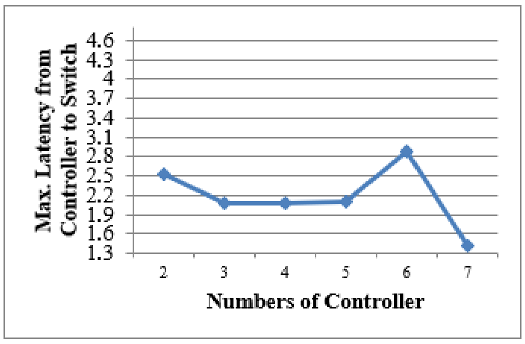


Figure 6. Relationship between maximum latency with controller number in Stdk-means algorithm

In the Opk-means algorithm, the number of controllers is determined by the maximum latency time given. As shown in Figure 7 the higher the number of controllers, the lower the end-to-end latency between controllers to switches and between controllers to controllers. The k-means algorithm is applied after finding the shortest paths on the whole map, and the controllers are placed with the shortest distance, so the location of the controllers is stable.

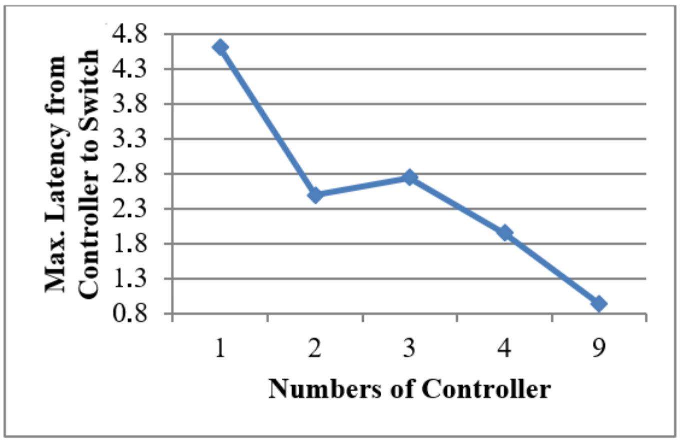


Figure 7. Relationship between maximum latency with controller number in Opk-means algorithm

The last studies (Wang et al., 2016; Wang et al., 2018) are focused on comparing the maximum latency times of different k-means algorithms for CPP. In these studies, the network is divided according to the number of subnetworks specified. However, in CPP, how many controllers to choose is one of the major issues. In this study, the Opk-means algorithm is proposed as a new algorithm that reduces both end-to-end latency and determines the number of controllers according to the minimum latency. In order to compare our algorithm with the results obtained in previous studies, the k-means algorithm is used to divide the network into subnetworks with the determined network segmentation parameter. This Stdk-means algorithm is compared with the improved Opk-means algorithm, for network segmentation and end-to-end latency methods. These algorithms' performances are shown in Figure 6 and Figure 7.

As a result, the Opk-means algorithm is more stable than the Stdk-means algorithm. Since the initial selection and subsequent selections are randomized in the Stdk-means algorithm, the output of each run result is different. However, the Opk-means algorithm proceeds, finding the shortest path, so that the output remains the same in all algorithms run. Also, in the Stdk-means algorithm, the number of controllers must be given as external user input. The number of controllers is a parameter that must be solved for the CPP. So, the number of controllers in terms of the latency time is determined by the Opk-means algorithm. In the Opk-means algorithm proposed in the study, it is enough to determine the latency time. Thus, with the Opk-means algorithm, a more feasible solution is obtained, and end-to-end latency is optimally determined for the CPP.

CONCLUSION

In this paper, a highly available SDN architecture is presented to overcome the single-point-of-failure in the multi-controllers. Also, an optimized algorithm is proposed for the optimal placement of the controllers. The proposed architecture uses Docker-Swarm Mode to allow one controller to work as a service in case of the failure of the first controller. Thus, this architecture enables continuous controller operation. To provides the stable configuration of the network structure Opk-means algorithm is proposed. The Opk-means algorithm is used to determine the number of controllers according to the minimum latency and reduce the end-to-end latency which directly affects the overall latency. The Opk-means algorithm is more stable than the Stdk-means algorithm so eliminating the disadvantages of the Stdk-means algorithm. In the Opk-means algorithm, is reduced the end-to-end latency time of the CPP to the optimum latency.

ACKNOWLEDGEMENTS

We would like to thank the Erciyes University Scientific Research Projects Coordination Unit (ERU/BAP) and TUBITAK. This work is supported by the Erciyes University Scientific Research Projects Coordination Unit (ERU/BAP) under project codes FDK-2018- 8073 and TUBITAK 2211- C program under project code 1649B031802979.

Conflict of Interest

The article authors declare that there is no conflict of interest between them.

Author's Contributions

The authors declare that they have contributed equally to the article.

REFERENCES

Ahmadi V, Khorramizadeh M, 2018. An adaptive heuristic for multiobjective controller placement in software-defined networks. Computers and Electrical Engineering, 66: 204-228.

Anonymous, 2010. TopologyZoo-Ulaknet Dataset. http://www.topology-zoo.org/files/Ulaknet.gml. (Date of access: 10 March 2021).

Anonymous, 2012. SDN Controller, Floodlight. http://www.projectfloodlight.org/floodlight/. (Date of access: 10 March 2021).

Anonymous, 2013. An Instant Virtual Network on your Laptop, Mininet. http://mininet.org/. (Date of access: 10 March 2021).

Anonymous, 2013. Empowering App Development for Developers, Docker. https://www.docker.com/. (Date of access: 10 March 2021).

Bella MRM, Data M, Yahya W, 2018. Web Server Load Balancing Based On Memory Utilization Using Docker Swarm. International Conference on Sustainable Information Engineering and Technology (SIET) IEEE, 2018, pp:220-223.

Hu Y, Luo T, Beaulieu NC, Deng C, 2017. The energy-aware controller placement problem in software defined networks. IEEE Communication Letter, 21: 741-744.

Huang V, Chen G, Fu Q, Wen E, 2019. Optimizing Controller Placement for Software-Defined Networks. Symposium on Integrated Network and Service Management (IM) IEEE, 2019, pp:224-232.

Lange S, Gebert S, Zinner T, Tran-Gia P, Hock D, Jarschel M, Hoffmann M, 2015. Heuristic Approaches to the Controller Placement Problem in Large Scale SDN Networks. IEEE Transactions on Network and Service Management, 12: 4-17.

Liao L, Leung VC, 2017. Genetic algorithms with particle swarm optimization-based mutation for distributed controller placement in SDNs. Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN) IEEE, 2017, pp:1-6.

Lu J, Zhang Z, Hu T, Yi P, Lan J, 2019. A Survey of Controller Placement Problem in Software-Defined Networking. IEEE Access, 7: 24290-24307.

MacQueen JB, 1967. Some Methods for classification and Analysis of Multivariate Observations. 5-th Berkeley Symposium on Mathematical Statistics and Probability, University of California Press, 1, 1967, pp:281-297.

McKeown N, 2009. Software-defined networking. INFOCOM Keynote Talk, 17:30-32.

Nguyen N, Bein D, 2017. Distributed MPI cluster with Docker Swarm mode. 7th Annual Computing and Communication Workshop and Conference (CCWC) IEEE, 2017, pp:1-7.

Nunes BAA, Mendonca M, Nguyen X, Obraczka K, Turletti T, 2014. A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks. IEEE Communications Surveys Tutorials, 16: 1617-1634.

Sahoo KS, Sahoo S, Sarkar A, Sahoo B, Dash R, 2017. On the placement of controllers for designing a wide area software defined networks. Region 10 Conference IEEE, 2017, pp:3123-3128.

Sanner J, Hadjadj-Aoul Y, Ouzzif M, Rubino G, 2017. An evolutionary controllers' placement algorithm for reliable SDN networks. 13th International Conference on Network and Service Management (CNSM) IEEE, 2017, pp:1-6.

Wang G, Zhao Y, Huang J, Duan Q, Li J, 2016. A k-means-based network partition algorithm for controller placement in software defined network. Communications Software, Services and Multimedia Applications Symposium (ICC) IEEE, 2016, pp:1-6.

Wang G, Zhao Y, Huang J, Wu Y, 2018. An effective approach to controller placement in software defined wide area networks. IEEE Transactions Network and Service Management, 15: 344-355.