Journal of Information Systems Engineering and Management
2025, 10(1s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Load Balancing Based Intelligent Software Defined Networking (SDN) Controller

Liqaa Saadi Mezher[1,2], Noor Sabah[1,3], Dheyaa Jasim Kadhim[1]

[1]*University of Baghdad, College of Engineering, Electrical Engineering Department, Baghdad, Iraq*

[2]*Mustansiriyah University, College of Engineering, Computer Engineering Department, Baghdad, Iraq*

[3]*University of Wasit, College of Engineering, Electrical Engineering Department, Wasit, Iraq*

mail: leqaa.saady2302p@coeng.uobaghdad.edu.iq, iqa35@uomustansiriyah.edu.iq, noors@uowasit.edu.iq, ayammohsen@uomustansiriyah.edu.iq, coeng.uobaghdad.edu.iq

ARTICLE INFO

Received: 01 Oct 2024
Revised: 24 Nov 2024
Accepted: 10 Dec 2024

ABSTRACT

The data plane and the control plane are divided by the Software Defined Networking (SDN) paradigm, where an SDN controller governs the operation of the switches it controls while also receiving demand from associated switches. Switches are dynamically controlled by their respective reassigned to be able to distribute the volume of work among controls for SDN. The majority of creative solutions for assignments require a crucial component of collect data regarding switches that need to be reassigned in order to execute load balancing. An individual super controller is utilized data from all switches and gathers information for all controllers, there is a scalability issue that arises as the number of controller's increases. When assigning switches to controllers in a big network, distances between them can occasionally be a problem.

To solve the above problem, this study proposed an intelligent clustering network based on k-means clustering. Under the OpenFlow protocol, Forwarding Table Management Standard Flow Table, and Flow Management, the advantages demonstrate the intelligent network of control concentration. Therefore, the goals were to balance the load by adjusting the cluster node load method and comparing the performance with the traditional controller cluster and single controller. The experimental results reveal that the k-means clustering method proposed in this study can effectively reduce the packet loss rate or delay. And theoretical research shows that our approach delivers a close to ideal result. According to simulation results, our dynamic grouping enhances stable clustering by scaling factor of five, which is significant.

Keywords: Software-Defined Network; SDN Controller; Controller Load Balancing; K-Means.

INTRODUCTION

Software-Defined Networking (SDN) is an emerging solution that separates the control plane from the data plane ☐[1][1]. SDN moves the intelligence from the traditional distributed networking model and puts it into a centralized software application called the SDN controller. The SDN controller stores all the network state information in a global view and controls all the data forwarding inside switches. Due to the central system in the SDN controller, the network can be adapted to handle more complicated applications and policies, ranging from traffic load balancing and quality of service to security. However, the traditional centralized controller may become a bottleneck and face scalability issues in the SDN network. While the SDN supports high-performance advanced network policies and features [2], the most crucial challenges related to the wide range of network conditions and traffic require an efficient and smart controller, which can decide where the required control action must be taken accordingly.

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

To address the challenges in the previously proposed concepts, our previous work has experience utilizing the centralized processing power to develop a novel cost-effective solution called the SDN controller to partition the network services instead of replicating the original single centralized software controller [3]. The partitioned controllers will assist the original centralized controller in providing fundamental software-defined networking control services, where these services can be processed in a distributed large-scale system. However, in our previous design, it distributed the switch-oriented network control services from the single centralized controller instead of a smart controller that can process the complicated applications for the management of cloud data center infrastructures.

1.1. Work Objectives

- Accomplish controller load balance. To achieve this, considerations including **network scalability**, **technique flexibility**, **minimal complexity**, **improved optimization**, and **overhead reduction** are taken into account [4].
- **The benefits of Speed, Flexibility and Reach:** Programmers do not need to deal with the tedious task of manually encoding numerous hardware devices. They can now simply install a software-based controller and direct any traffic on the network instead. It also enables managers to control the network, change settings, provide resources, and increase the capacity without the need for additional hardware – all via one interface. Therefore, network managers have less restriction on what networking equipment to use since they can use one protocol to interact with as many hardware devices as possible through a main controller.
- **Network architecture:** the network design that enables alteration on the way the networks are used as a result of availing a user interface that allows for the configuration of network resources and services from one central point. This gives the network managers an avenue to up-rate the applications with more availability requirements and even optimize the data across the network.
- **Strong Security:** A software-defined network allows for a centralized network management perspective and therefore improves security risks assessment. With the ever-increasing number of devices that can be connected to the internet, SDN presents several merits above traditional networking [5]. In order to prevent the infected devices from spreading to the rest of the network, the managers can perform two options: turn off the infected devices or separate the infected devices in a compartment that has different devices that need a different security level.

1.2. Significance of the Paper

In the paper, it did take into account time-sensitive networks, therefore we changed our strategy to reduce network noise. In this new strategy, controllers stay constant, and switches are only reassigned during active TS.

- Since each controller has a certain amount of processing power, large networks require numerous controllers to handle all of the requests. This is how we deal with large networks at the internet size. Utilizing several identical instances of a single controller that each do essentially the same tasks as the others and are linked to separate SDN controllers is one way to accomplish this goal [6]. Due to the dynamic nature of the network, these questions should be regularly taken into account in addition to being pertinent during network setup based on static facts.
- Some controllers may experience a bottleneck as a result of the switches' dynamic request rate because each controller has a finite amount of processing power. As a result, if there are too many switches requested, the controller will have to process the requests in a queue, which will result in slow switch response times. To circumvent this issue, switches are dynamically allocated to controllers based on their request rates.
- Depending on the dynamic traffic, the importance of controllers and where they are changed, and they are alternately turned on and off during each cycle. Other methods deal with additional goals including minimal setup time and maximum utilization in addition to load balancing, which indirectly aids in balancing loads amongst controllers.
- The "Balance flow" load balancing technique emphasizes controller load balancing so that:
    1) To ***ensure speedy*** response, the ***flow- requirements*** are dynamically allocated across the controllers.

2) To maximize controller usage, the load on a ***high-loaded*** controller is immediately moved to suitable ***low-loaded*** controllers.

- The controllers can be ***set up*** as follows:

1) ***Hierarchically,*** in which case various controllers have a separate network section that specifies the streams it can handle.

2) ***Flatly***, with each controller able to handle all incoming request types.

- The least load discrepancies between clusters are established as objectives, and the shortest distances between controllers in each cluster are taken into consideration while defining the "Clustering" problem for the top level of the load balancing method.
- Due to the reliance between the SC operation and other controller actions, "***Hybrid Flow***" has a long run time. "***Hybrid Flow***" in which controllers are ***fixedly clustered***. The SC is just utilized to collect load information and communicate it to/from the controllers so as to lessen the strain on the SC. Instead, the procedure for reassignment is carried out by each cluster's controllers.
- ***DCC*** Problem Formulation, the challenge of assigning controllers to clusters is discussed [7]. Here, this issue is viewed as a conundrum involving minimization with restrictions by (Key notations “Re Clustering”, Clusters’ Load Differences, a Cluster's Controllers' Separation Distances, Optimization Challenge “Dynamic Controllers’ Clustering” to choose the most suitable clustering assignment), as shown if figure (1).

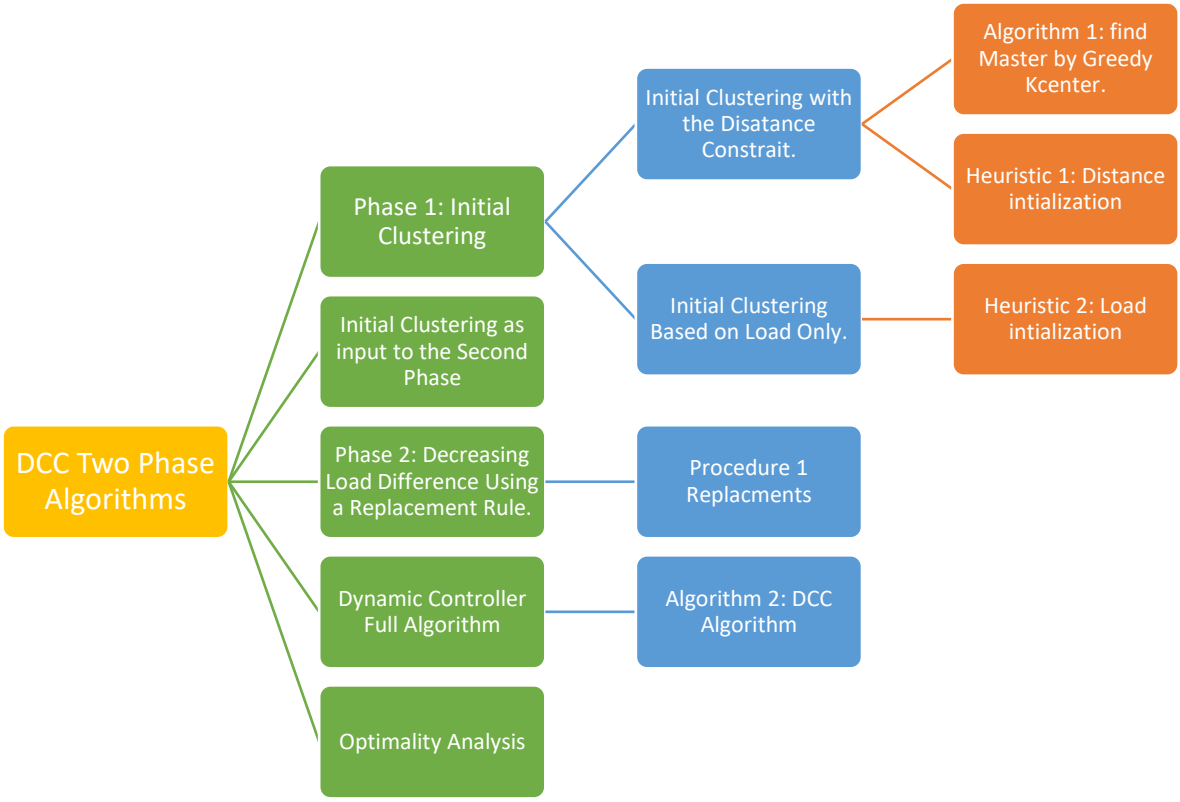


Figure (1): Dynamic Controller Clustering Two Phase Algorithms

SOFTWARE DEFINED NETWORKING (SDN)

In traditional networking, the network's data plane and control plane are both implemented in the devices in the network. Each network device's control logic makes its matrix forwarding decision based on the protocols executed therein. Different network devices with different control logics together form the network's control infrastructure. SDN is a new emerging networking paradigm that decouples the network's control and data planes, where the network's control logic is physically centralized and network devices, switches, and routers, among others, only have the data forwarding function. The network data forwarding function and control forwarding functions, policies, and rules are implemented independently in the network's data plane and the centralized control logic, respectively. This

new networking paradigm introduces a new network device, the SDN controller, that makes network device decisions on the basis of the abstracted network view collected from network devices [8]. The abstracted network view collects the states and configurations of network devices and forwards them to the SDN controller. In addition, the response decisions from the SDN controller are used to update network devices or are used directly as the forwarding decisions to network devices. Benefiting from the centralized controller advantage, the SDN architecture significantly reduces the complexity of the network and simplifies network device design [9].

There are three levels in the architecture of software defined networking, as shown figure (2).

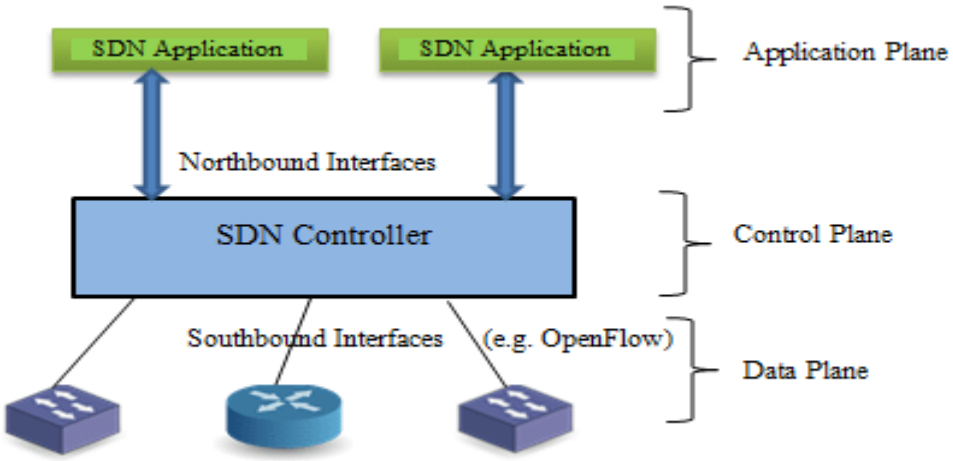


Figure (2) Architecture of SDN Layers

A. The Application Layer: This layer is made up of applications that directly and programmatically inform the SDN Controller of the intended network behavior and needs [6].

B. An Infrastructure Layer: It encompasses the networking technology and determines the volume of data that the network can transmit or steer. It is the equipment that operates on the data 'packets' based on how the controlling device has programmed it [10]. Control layer physical structure of networks works to gather information relating to their operational status, their traffic patterns, and network topology or load, etc. and releases it.

C. SDN Control Layer: The control layer corresponds to the section of the software-defined network where the software controller works. This controller sits on a server and is in charge of managing the flow and rules of traffic across the entire network. The SDN Controller is the middle layer that connects the application layer and the infrastructure layer [11]. The southbound interfaces are the connections with the infrastructure layer, and the northbound interfaces are the connections with the applications. The networking components are reached by this layer after processing the commands and specifications given by the application layer (through southbound interface), as illustrated in figure (3) [12].

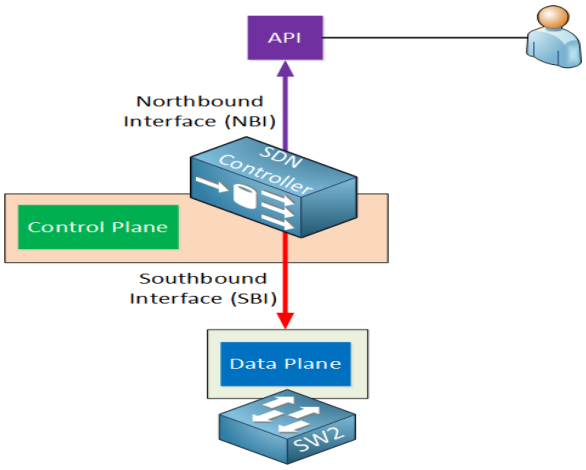


Figure (3) Control Layer SDN

Additionally, it transmits to the program the data it needs to function properly, information that has been collected from networking devices.

MECHANISM FOR BALANCING LOADS:

Load Balancing Mechanism the majority of the time, when a controller creates an overload, this is due to a significant number of switches being unable to locate a matching flow entry for the newly incoming packet [13], which causes them to send a packet in message to the default controller. A system that balances the load on the control plane. The control plane's controllers all have access to the same network view, and the switch's dynamic migration allows for load balancing [14]. The dynamic load balancing diagram is depicted in Figure 4. When a controller in the control plane (Controller C1) exhibits an overload trend, it enters the switch migration phase, and portion of the overload controller's switches (Switch S5 and Switch S9) migrate to the comparatively idle. Because of how complicated the current network environment is, the controller's real-time load ratio will alter as it does. The SDN control platform's load balancing technique shouldn't only modify the load value number of every controller [15].

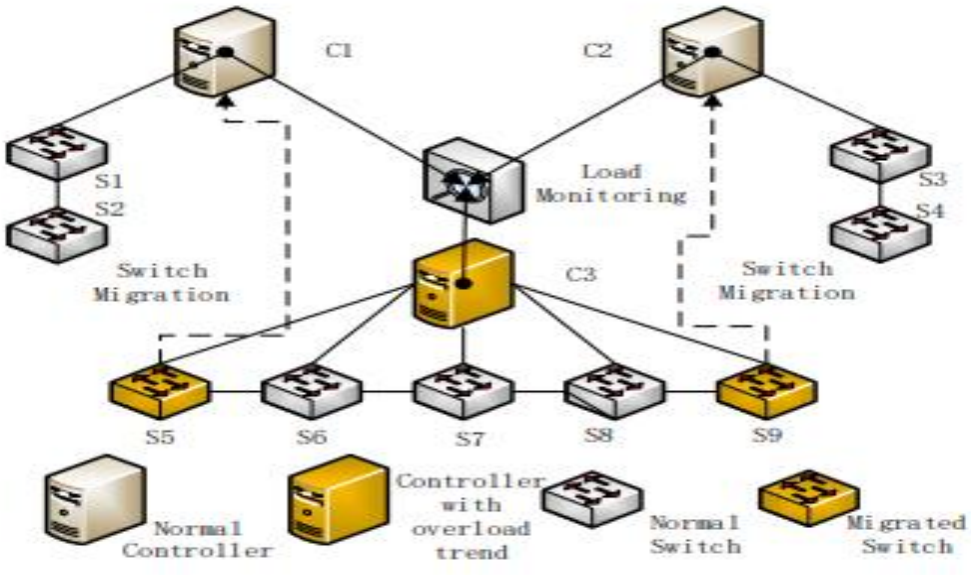


Figure (4) Controller Load Diagram

A load balancing architecture with two tiers is described (Super Controller, Master Controller), as shown in figure (5).

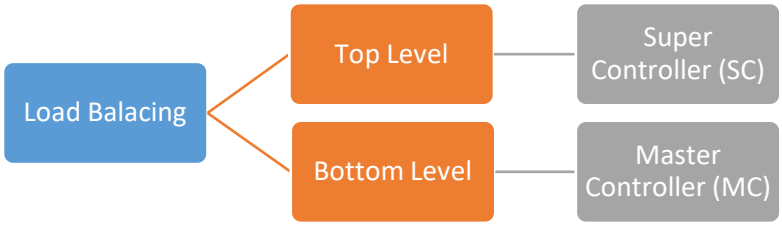


Figure (5) Architecture of Load Balance

In the "***Balance flow***" load balancing technique concentrates on controller load balancing in order to maximize controller usage:

- Flow needs are constantly allocated controllers for among achieve speedy reply.
- To get the most out of each controller, the weight of a loaded one is immediately switched to a suitable, controller with little load.

SDN FOR LOAD BALANCING

Software Defined Networking (SDN) technology is the most promising architecture for networks that provides programmability, simplicity, and abstraction [17]. In SDN, the controller acts as the brain of the network, executing the control plane functionality. Every communication from a switch goes through the controller. Therefore, the

communication between the controller and the switch can increase the load on the controller. This can cause some limitations, such as network latency and control channel overload. In order to overcome these limitations and achieve higher performance in SDN, load balancing should be considered [18].

Since the admin cannot have complete control over the balancing settings, balancing is very high installation cost and does not provide for flexibility. In order to address this issue, this paper suggests using an SDN application to serve as just a load balancer, which lowers costs and gives the administrator more freedom compared to the predefined paradigm established by balancer vendors. The administrator may choose the algorithm as well as the load distribution for every server.

I. **Controller Load Balancing:** Occasionally it is necessary to make the SDN controller decentralized to be able to expand the capability of the control plane [19]. Normally, the decentralization leads to the command plane having multiple controllers. The next section discusses load balancing when the control plane is made up of various SDN controllers [20].

II. **Server Balance of loads:** In the event that numerous servers make up a network, we now take server-based load balancing into consideration. When using a load balancing algorithm, client requests are split amongst several servers [21]. Based on their influence on the overall network performance, the algorithm makes sure that the following request is forwarded to the server that is least busy.

III. **Load Balancing in Wireless Links:** Due to the lack of an automobile load balancer in IEEE 802.11, Distributing the load among users in the current wireless LANs may be an issue of concern [22]. The algorithms embedded into the SDN controller may be used to ensure a dynamic balance of APs loads by choosing the least laden access points (APs) in the vicinity for the clients' connection.

IV. **Choosing a Communication Path Balanced load:** In the design of large and fast networks, it is common for network architects to face the problem of managing the traffic load efficiently on multiple links. To enhance the performance of the network in a more fluid and more scalable manner, SDN allows for the deployment of the methods which can easily disperse the traffic over similar routes. Existing and proposed load balancing strategies in data center networks with multipath routing should guarantee effective traffic distribution where no single path is excessively idle or over used. Because of the dynamic nature of networks like data centers, traffic congestion can occur at any time [23].

V. **Artificial Intelligence Based Balanced load:** ANNs are characterized as adaptable arithmetic frameworks that are able to identify intricate nonlinear relationships between input and output data points [24]. Due to the fact that ANNs have no restrictions on input vectors and that network traffic flow is unknown, a significant volume of information can be gathered to assemble a collection of information. Since there is a good chance that the created complicated information set on traffic flow will likely not follow a known probability distribution, ANNs are best suited to handle this ambiguous network traffic data. Figure illustrates the ANNs' structure (6) [25].

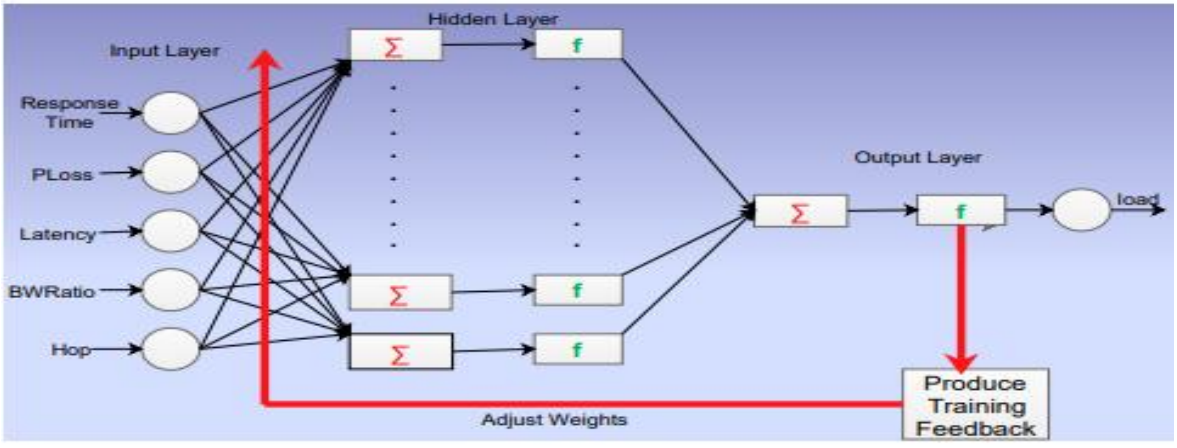


Figure (6) Load Balancing Neural Network Structure.

METHODOLOGY DYNAMIC CLUSTER CONTROLLER ALGORITHMS

- The Dynamic Cluster Flow (DCF) design has already been upgraded to allow the use of current the techniques in the "***Reassignment***" each cluster's procedure, in figure (7) [26].

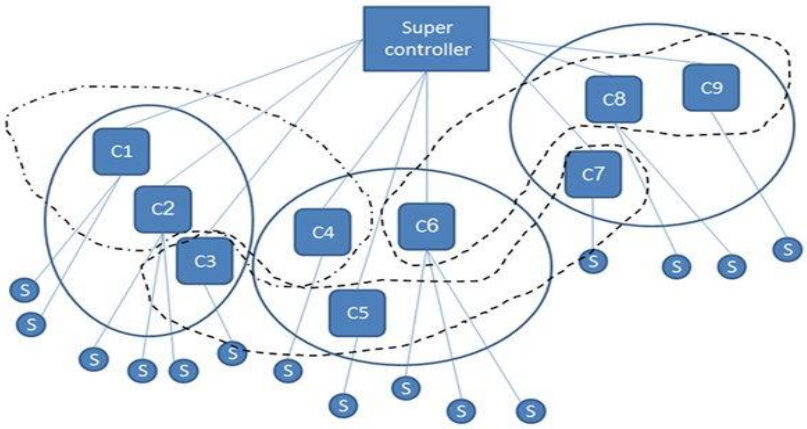


Figure (7): The Dynamic Cluster Flow Architecture

- Finding the ideal clustering is our aim.
- DCC is a technique that creates clusters of comparable things.
- The ultimate result is a group of clusters, each different from each other and including things which are largely similar to each other.
- Initial clustering's goal is to give the second phase the greatest possible start and the best outcome possible. For the initialization, there are two options. The first option is to concentrate on ***distance***, i.e., find first order grouping that fulfills the length requirement; the following option will be to concentrate on reducing the variation in load across clusters.
- The interaction of the controllers with their MC is what generates the majority of control messages pertaining to the cluster load balancing activity. So, to discover the nearest MC, we employ the K-Center problem solution, as shown figure (8).

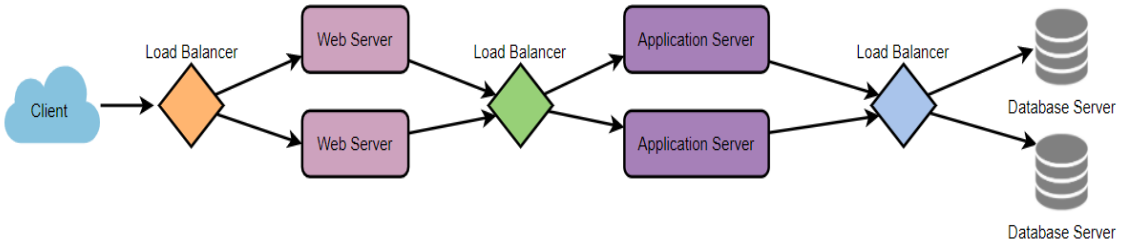


Figure (8): Loads Balance at three places

METHODOLOGY OF DCC (K-MEANS CLUSTERING ALGORITHM)

The k-means algorithm is well known for its clustering ability. It is the most widely used partitional clustering algorithm. K-means divides n objects into k clusters. The input to the algorithm is a set of n d-dimensional vectors and a parameter k indicating the number of clusters to be formed. The k-means algorithm has a number of applications in clustering analysis. For example, the k-means algorithm is a non-hierarchical method that is used for cluster analysis or segmentation. In brief, k-means is designed to partition n observations without any further knowledge about their cluster or distribution. In the k-means algorithm, each cluster is represented by a best prototype, also known as the mean, and an object is assigned to the cluster with the closest mean, as shown figure (9).

- As the initial master, the algorithm selects a random controller depended of number of clusters, as shown in algorithm 1.
- Then, it repeatedly executes the following two steps, as shown in Heuristic 1:

- Find the two groups that are most adjacent by measured the separation between points.
- Combine the two groups that are most similar.

➢ This repeated procedure continues until every cluster is combined together (became all cluster simile when repeat them).

➢ The issue has been conceptualized as an optimization problem with limits on the distance between controllers, with the goal of reducing the disparity between cluster loads. Finding the best clustering assignment is our aim.

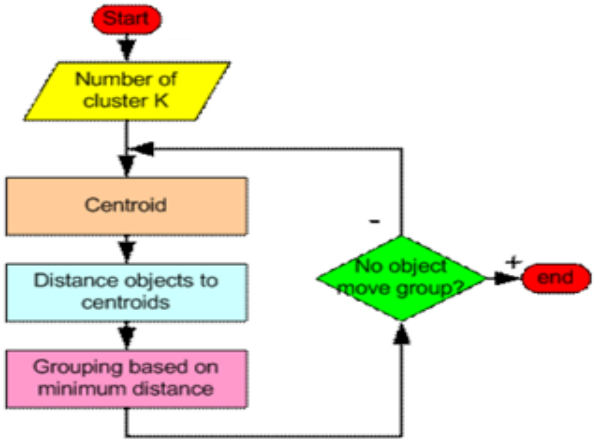


Figure (9): Flowchart of find Master by 2- Approximation Greedy k-Center clustering algorithm

Algorithm 1: Find Masters by Two Approximation Greedy K – Center Solution

Input: $P = \{p_1, p_2, \ldots\ldots, p_M\}$ controllers set, controller to controller matrix distances.

Output: $C = \{c_1, c_2, \ldots\ldots, c_K\}$ masters set.

Procedure:

1. $\emptyset \rightarrow C$
2. $p_i \rightarrow c_1$ // an arbitrary point P_i from.
3. $C \cup c_1 \rightarrow C$
4. For i = 1 : K do
5. For all $p \epsilon P$ do
6. $min_{c \epsilon C}\ d(p, c) \rightarrow d_i[p]$
7. *end for*
8. $max_{p \epsilon P}\ d_i[p] \rightarrow c_i$
9. $C \cup c_i \rightarrow C$
10. End for
11. Return C // The master set.

Heuristic 1: Distance initialization

Input: $C = \{c_1, c_2, \ldots\ldots, c_M\}$ controller list.

$M = \{m_1, m_2, \ldots\ldots, m_K\}$ masters set.

controller to controller matrix distances.

Output: $CL = \{cl_1, cl_2, \ldots\ldots, cl_k\}$ *clusters* list, $CL_i = \{cl_{i1}, cl_{i2}, \ldots\ldots, cl_{i(\frac{m}{k})}\}$

Max Distance = maxmume distance between controllers in cluster.

Procedure:

1. $C \rightarrow S$

2. $M - S \rightarrow S$
3. For i = 1 : K do
4. $M_i \rightarrow CL_i$
5. $end\ for$
6. $CL \rightarrow Candidates$
7. While $S \neq \emptyset$ do
8. The next controller in S → C_{next}
9. Find the nearst master from candidates list → CL_{near}
10. $C_{next} \cup CL_{near} \rightarrow CL_{near}$
11. $C_{next} - S \rightarrow S$
12. If $|CL_{near}| = M/K$ then
13. $CL_{near} - candidates \rightarrow Candidates$
14. End if
15. End while
16. For all $CL_i \in CL$ do
17. maxmume distance between two controllers in CL_i → Max Distance CL_i
18. End for
19. Maximum of all Max Distance CL_i → Max Distance
20. Return CL, Max Distance.

INTEGRATING K-MEANS CLUSTERING IN SDN CONTROLLER

In paper, proposed in the case of the K-means clustering algorithm, the K-centers of clusters represent the mean of those particular clusters. These centers are defined by the equation as follows: Let those K points be with as an element of the cluster. The purpose of this clustering algorithm is to divide the input data into K arbitrary groups so that the is minimized. Every point is allocated into a group with respect to K. This K-means clustering algorithm is used to compute the general descriptive feature of each occurrence of every different kind of group of associated workloads. In our proposed model, we plan to introduce the K-means clustering algorithm as an enhancement in the proposed software-defined network controller model. The proposed approach focuses on providing a smart way of intelligent balancing in the SDN controller with the help of the K-means clustering algorithm.

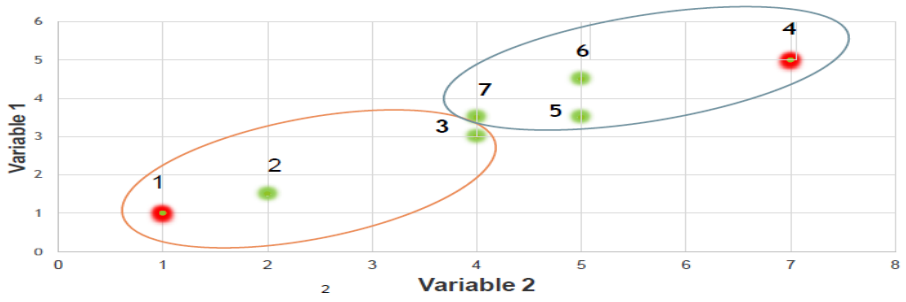
REQUIRED TESTED ENVIRONMENT

The simulation's goal is to demonstrate the effectiveness of our DCC method. Additionally, the number of substitutions is limited while still producing superior outcomes than the constant clustering algorithm. The simulator we utilized was created in the. Net framework using the Visual Studio environment. The value of controllers, the space between each one, and the loads placed on each controller are all selectable in this simulator. The simulator enables doing usage of the controllers at random and allocating an unpredictable load to each controller in order to take into account a global topology.

CASE STUDY AND RESULTS

A Simple case study of k-means clusters (utilizing K=2)

individual	Variable One	Variable Two
1	1	1
2	1.5	2
3	3	4
4	5	7
5	3.5	5
6	4.5	5
7	3.5	4.5



First Step:

Initializing: to select two centroids for two clusters, we randomly select the following two (k=2). In this instance the two centroids are: x1 = (1, 1) and x2 = (5, 7)

	Point	Mean Vector
Team 1	**1**	**(1,1)**
Team 2	**4**	**(5,7)**

Second Step:

	Centroid 1	Centroid 2
1	$f(1-1)^2+(1-1)^2 = 0$	$f(5-1)^2+(7-1)^2 = 7.21$
2	$f(1-1.5)^2+(1-2)^2 = 1.12$	$f(5-1.5)^2+(7-2)^2 = 6.10$
3	$f(1-3)^2+(1-4)^2 = 3.61$	$f(5-3)^2+(7-4)^2 = 3.61$
4	$f(1-5)^2+(1-7)^2 = 7.21$	$f(5-5)^2+(7-7)^2 = 0$
5	$f(1-3.5)^2+(1-5)^2 = 4.72$	$f(5-3.5)^2+(7-5)^2 = 2.5$
6	$f(1-4.5)^2+(1-5)^2 = 5.31$	$f(5-4.5)^2+(7-5)^2 = 2.06$
7	$f(1-3.5)^2+(1-4.5)^2 = 4.30$	$f(5-3.5)^2+(7-4.5)^2 = 2.92$

- As a result, we get two clusters that include the numbers 1, 2, 3, and 4, 5, 6, 7.
- They have new centroids, which are:

$$\text{Team 1} = \left(\frac{1+1.5+3}{3}, \frac{1+2+4}{3}\right) = (1.83, 2.33)$$

$$\text{Team 2} = \left(\frac{5+3.5+4.5+3.5}{4}, \frac{7+5+5+4.5}{4}\right) = (4.12, 5.38)$$

Third Step:

	Centroid 1	Centroid 2
1	$f(1.83-1)^2+(2.33-1)^2 = 1.57$	$f(4.12-1)^2+(5.38-1)^2 = 5.38$
2	$f(1.83-1.5)^2+(2.33-2)^2 = 0.47$	$f(4.12-1.5)^2+(5.38-2)^2 = 4.29$
3	$f(1.83-3)^2+(2.33-4)^2 = 2.04$	$f(4.12-3)^2+(5.38-4)^2 = 1.78$
4	$f(1.83-5)^2+(2.33-7)^2 = 5.64$	$f(4.12-5)^2+(5.38-7)^2 = 1.84$
5	$f(1.83-3.5)^2+(2.33-5)^2 = 3.15$	$f(4.12-3.5)^2+(5.38-5)^2 = 0.73$
6	$f(1.83-4.5)^2+(2.33-5)^2 = 3.78$	$f(4.12-4.5)^2+(5.38-5)^2 = 0.54$
7	$f(1.83-3.5)^2+(2.33-4.5)^2 = 2.74$	$f(4.12-3.5)^2+(5.38-4.5)^2 = 1.08$

- Consequently, the new clusters are 1, 2, and 3, 4, 5, 6, and 7.

$$\text{Team 1= } \left(\frac{1+1.5}{2}, \frac{1+2}{2}\right) = (1.25,1.5)$$

$$\text{Team 2= } \left(\frac{3+5+3.5+4.5+3.5}{5}, \frac{4+7+5+5+4.5}{5}\right) = (3.9,5.1)$$

Fourth Step:

	Centroid 1	Centroid 2
1	$f(1.25-1)^2+(1.5-1)^2 = 0.58$	$f(3.9-1)^2+(5.1-1)^2 = 5.02$
2	$f(1.25-1.5)^2+(1.5-2)^2 = 0.56$	$f(3.9-1.5)^2+(5.1-2)^2 = 3.92$
3	$f(1.25-3)^2+(1.5-4)^2 = 3.05$	$f(3.92-3)^2+(5.1-4)^2 = 1.42$
4	$f(1.25-5)^2+(1.5-7)^2 = 6.66$	$f(3.9-5)^2+(5.1-7)^2 = 2.20$
5	$f(1.25-3.5)^2+(1.5-5)^2 = 4.16$	$f(3.9-3.5)^2+(5.1-5)^2 = 0.41$
6	$f(1.25-4.5)^2+(1.5-5)^2 = 4.78$	$f(3.9-4.5)^2+(5.1-5)^2 = 0.61$
7	$f(1.25-3.5)^2+(1.5-4.5)^2 = 3.75$	$f(3.9-3.5)^2+(5.1-4.5)^2 = 0.72$

- As a consequence, there has been no change to the cluster.
- And the algorithm stops here, producing the ultimate outcome, which includes of the clusters [1, 2], and [3, 4, 5, 6, 7].

CONCLUSION AND CONTRIBUTIONS

On this paper, a better method for lowering the load balancing's temporal complexity in the SDN control and data planes is proposed. The requests (from of the switches to the controllers) are to be distributed among them so as to prevent overload on certain of them. Our architecture for a three-tiered control plane, which includes an overlord controller with subordinate controllers capable of performing balance of loads actions individually, is used for this purpose. As a solution, the complexity of the time period may be reduced by running the full load balancing process simultaneously. To assign clusters' controllers with optimization and the shortest possible the separation of two controllers in the same cluster, we present a system (made up of several methods). We demonstrate that employing dynamic clusters yields superior outcomes over fixed clustering. Future study will examine the ideal cluster size and support a range of cluster sizes. Overlapping clusters is an intriguing direction. A different approach is to look at the necessary ratio between the load balancing method's runtime and the duration of the timeline unit. The best location for the supreme controllers within each group is another crucial outstanding problem.

ACKNOWLEDGMENTS

The authors would like to thank Mustansiriyah University (www.uomustansiriyah.edu.iq) Baghdad-Iraq for its support in the present work.

REFERENCES

[1]. Abdi, Abdinasir Hirsi, et al. "Security Control and Data Planes of SDN: A Comprehensive Review of Traditional, AI and MTD Approaches to Security Solutions." *IEEE Access* (2024).

[2]. McCann, Ryan, Arash Rezaee, and Vinod M. Vokkarane. "SDONSim: An Advanced Simulation Tool for Software-Defined Elastic Optical Networks." *arXiv preprint arXiv:2410.13999* (2024).

[3]. Xu, Hui, et al. "An IoT-based low-cost architecture for smart libraries using SDN." *Scientific Reports* 14.1 (2024): 7022.

[4]. Patel, Nimeshkumar. "SECURE ACCESS SERVICE EDGE (SASE): EVALUATING THE IMPACT OF CONVEREGED NETWORK SECURITY ARCHITECTURES IN CLOUD COMPUTING." *Journal of Emerging Technologies and Innovative Research* 11.3 (2024): 12.

[5]. Zoraida, Berty Smitha Evelin. "A Comparative Study on Software-Defined Network with Traditional Networks." (2024).

[6]. Bhuiyan, Zaheed Ahmed, et al. "On the (in) security of the control plane of SDN architecture: A survey." *IEEE Access* (2023).

[7]. Asamoah, Emmanuel, Isaac Ampratwum, and Amiya Nayak. "On Using Genetic Algorithm for Optimal Controller Placement in Software-Defined Networks." *2024 International Conference on Information Networking (ICOIN)*. IEEE, 2024.
[8]. Abderrahmane, Amel, Hamza Drid, and Amel Behaz. "A Survey of Controller Placement Problem in SDN-IoT Network." *International Journal of Networked and Distributed Computing* (2024): 1-15.
[9]. Krentsel, Alexander, et al. "A Decentralized SDN Architecture for the WAN." *Proceedings of the ACM SIGCOMM 2024 Conference*. 2024.
[10]. Khalid, Wajahat, et al. "Open-Source Internet of Things-Based Supervisory Control and Data Acquisition System for Photovoltaic Monitoring and Control Using HTTP and TCP/IP Protocols." *Energies* 17.16 (2024): 4083.
[11]. Sarabia-Jácome, David, et al. "Progressive Adoption of RINA in IoT Networks: Enhancing Scalability and Network Management via SDN Integration." *Applied Sciences* 14.6 (2024): 2300.
[12]. Kumar, Puneet, and Behnam Dezfouli. "quicSDN: Transitioning from TCP to QUIC for southbound communication in software-defined networks." *Journal of Network and Computer Applications* 222 (2024): 103780.
[13]. Singh, Neelam, et al. "Load balancing and service discovery using Docker Swarm for microservice based big data applications." *Journal of Cloud Computing* 12.1 (2023): 4.
[14]. Liu, Yong, et al. "Load-aware switch migration for controller load balancing in edge–cloud architectures." *Future Generation Computer Systems* 162 (2025): 107489.
[15]. Sapkota, Binod, Babu R. Dawadi, and Shashidhar R. Joshi. "Controller placement problem during SDN deployment in the ISP/Telco networks: A survey." *Engineering Reports* 6.2 (2024): e12801.
[16]. Sharma, Aakanksha, Venki Balasubramanian, and Joarder Kamruzzaman. "A temporal deep Q learning for optimal load balancing in software-defined networks." *Sensors* 24.4 (2024): 1216.
[17]. Rahim, Johari Abdul, Rosdiadee Nordin, and Oluwatosin Ahmed Amodu. "Open-Source Software Defined Networking Controllers: State-of-the-Art, Challenges and Solutions for Future Network Providers." *Computers, Materials & Continua* 80.1 (2024).
[18]. Aguilar, Leonardo de Carvalho Freitas Padilha, and Daniel Macêdo Batista. "Effectiveness of implementing load balancing via SDN." *Anais* (2019).
[19]. Krentsel, Alexander, et al. "A Decentralized SDN Architecture for the WAN." *Proceedings of the ACM SIGCOMM 2024 Conference*. 2024.
[20]. Alhilali, Ahmed Hazim, and Ahmadreza Montazerolghaem. "Artificial intelligence-based load balancing in SDN: A comprehensive survey." *Internet of Things* 22 (2023): 100814.
[21]. Zhou, Jincheng, et al. "Comparative analysis of metaheuristic load balancing algorithms for efficient load balancing in cloud computing." *Journal of cloud computing* 12.1 (2023): 85.
[22]. Hadar S. and Yoram H., “DCF: Dynamic Cluster Flow Architecture for SDN Control Plane”, Conference: 2017 IEEE International Conference on Consumer Electronics (ICCE), 11 April 2018.
[23]. Wang, Jin, et al. "Load balancing for heterogeneous traffic in datacenter networks." *Journal of Network and Computer Applications* 217 (2023): 103692.
[24]. Belgaum, Mohammad Riyaz, et al. "Artificial intelligence based reliable load balancing framework in software-defined networks." *CMC—Comput. Mater. Contin* 70 (2022): 251-266.
[25]. Kazmi, Syed Hussain Ali, et al. "Survey on joint paradigm of 5G and SDN emerging mobile technologies: Architecture, security, challenges and research directions." *Wireless Personal Communications* 130.4 (2023): 2753-2800.
[26]. Sufiev, Hadar, and Yoram Haddad. "DCF: Dynamic cluster flow architecture for SDN control plane." *2017 IEEE International Conference on Consumer Electronics (ICCE)*. IEEE, 2017.