computers

Article

Multi-Controllers Placement Optimization in SDN by the Hybrid HSA-PSO Algorithm

Neamah S. Radam *, **Sufyan T. Faraj Al-Janabi** and **Khalid Sh. Jasim**

College of Computer Science and Information Technology, University of Anbar, Ramadi 31001, Iraq; sufyan.aljanabi@uoanbar.edu.iq (S.T.F.A.-J.); khalidalhity@gmail.com (K.S.J.)
* Correspondence: nea19c1010@uoanbar.edu.iq

Abstract: Software-Defined Networking (SDN) is a developing architecture that provides scalability, flexibility, and efficient network management. However, optimal controller placement faces many problems, which affect the performance of the overall network. To resolve the Multi-controller SDN (MC-SDN) that is deployed in the SDN environment, we propose an approach that uses a hybrid metaheuristic algorithm that improves network performance. Initially, the proposed SDN network is constructed based on graph theory, which improves the connectivity and flexibility between switches and controllers. After that, the controller selection is performed by selecting an optimal controller from multiple controllers based on controller features using the firefly optimization algorithm (FA), which improves the network performance. Finally, multi-controller placement is performed to reduce the communication latency between the switch to controllers. Here, multiple controllers are placed by considering location and distance using a hybrid metaheuristic algorithm, which includes a harmonic search algorithm and particle swarm optimization algorithm (HSA-PSO), in which the PSO algorithm is proposed to automatically update the harmonic search parameters. The simulation of multi-controller placement is carried out by the CloudsimSDN network simulator, and the simulation results demonstrate the proposed advantages in terms of propagation latency, Round Trip Time (RTT), matrix of Time Session (TS), delay, reliability, and throughput.

Keywords: software defined networking (SDN); multi-controller; controller placement problem; controller selection; network construction; hybrid optimization

check for updates

Citation: Radam, N.S.; Al-Janabi, S.T.F.; Jasim, K.S. Multi-Controllers Placement Optimization in SDN by the Hybrid HSA-PSO Algorithm. *Computers* **2022**, *11*, 111. https://doi.org/10.3390/computers11070111

Academic Editor: Seyedali Mirjalili

Received: 30 May 2022
Accepted: 28 June 2022
Published: 4 July 2022

1. Introduction

Currently, the large use of mobile devices expands the scale of the networks. Traditional networks are likely not suitable for the changing scenario, which faces network management issues, and network flexibility issues [1]. To alleviate the issues in traditional networking, an efficient solution was driven by Software Defined Networking (SDN). The SDN is an emerging paradigm that supports network programmability by unravelling the network equipment with the control layer [2]. Typically, the SDN consists of three planes, namely the data plane, control plane, and application plane. The data plane consists of multiple switches that collect the data as a packet from the devices and perform forwarding to the control plane by following SDN protocols (i.e., OpenFlow) [3]. The control plane consists of a controller, which is the heart of the SDN, that manages the whole network. The application plane is managed by the application administrator, whose policies are also managed by the controller [4]. A controller has the power to remove or add the rules and policies in SDN networks. Therefore, it improves the network programmability and supports network management [5].

Initially, the whole SDN network is managed by a single SDN controller. These single SDN controllers are mostly suitable for small coverage networks (i.e., local area networks) [6]. Some of the prior works exploit single SDN controllers for small coverage networks; however, they face single-point failure and scalability issues [7]. The use of a single SDN controller for large area network limits with high latency, congestion, and link failure in the network also affects the network performance [8]. To address the issues in the single SDN controllers in terms of scalability and latency, the multi-controller concept is introduced, which alleviates the problem of single-point failure and provides a distributed latency-free intelligent solution to SDN networks [9]. Some of the prior work utilizes the multi-controller SDN concept and mitigates the traditional drawback to a certain level. The SDN multi-controller utilization is collectively called the controller placement problem [10], which raises the question of how many controllers are needed for providing a flexible service to the SDN network [11], and which controller is selected to be deployed in the network [12].

The aforementioned question is also linked with the optimal placement of controllers, satisfying switch requirements, and selecting a highly fault-tolerant controller. The selection and placement of controllers in an optimal location decreases the latency between multi-controllers and switches [13]. The existing works employ the heuristics approaches for controller placement problems; however, they are still limited in considering some of the important constraints (e.g., not considering controller fault tolerance, and/or not considering switch requirements) [14]. The existing heuristics algorithms parameters are faced with convergence issues, which also affect the controller placement, thus affecting network performance [15]. Overall, the controller placement problem in a multi-controller environment is still an open issue that needs to be investigated deeply to enhance the network performance of SDN environments. Table 1 represents the comparison of single controller and multi-controllers.

Table 1. Characteristics of single vs. multi-controllers.

Single SDN Controller	Multi SDN Controller
Acquires the network's global view	Lack of network global view
Single point of failure	No single point of failure
Poor scalability	Highly scalable
Easy to deploy in legacy environments	Supports both legacy and SDN environments
Subjected to high latency	Less latency
Not suitable for mobile scenarios	Suitable for mobile scenarios
Prone to a high link failure rate	No link failure
High network congestion	No network congestion

Designing an efficient methodology for optimizing the multi controllers' placement in an adaptable manner is a very important and challenging issue in such a complex environment, especially due to the various related requirements. Thus, optimizing the deployment of multiple controllers within SDN is a complex issue due to the variability exhibited by many parameters and current operating conditions. An effective hybrid metaheuristic-based solution is provided for controller placement problems in a SDN environment. This research is aiming to minimize the latency between controllers, minimize delay between switches, and maximize the controller fault tolerance rate.

This research addresses the major problems of latency between controllers to controllers, and the selection of controllers. The existing works consider the latency between controllers to switch while some work considers the latency between controller to controller. However, they are limited with high latency as the controller was not placed in the precise location. The existing heuristics algorithm also lacks convergence issues, which also affects the controller placement. The selection of controllers before deployment is a major concern, however, most works are not considering controller selection, which also leads to decreased network performance in the SDN.

Motivated by the above problems in the existing works, this research is proposing to provide an efficient solution for the problem. The main objective of this research is to alleviate the issues in controller placement in a multi-controller environment. To meet this objective, some of the sub-objectives are:

- To improve the network scalability by constructing the network as a graph structure;
- To enhance the network performance by selecting the high fault-tolerant optimal controller using an optimization algorithm;
- To minimize the propagation delay between inter-controllers and switches by using hybrid metaheuristics algorithms.

The proposed MC-SDN (Multi-Controller based SDN) approach addresses Controller Placement Problem (CPP) to improve the network performance in an SDN environment. The network is constructed in a graph manner to increase scalability, connectivity, and flexibility of the network, which increases the communication efficiency and reduces the propagation delay of the link. Thus, the main contributions of this research are defined as follows:

- First, the optimal controller is selected from multiple controllers using the Firefly optimization algorithm (FA), which improves network performance by selecting the optimal controller to manage the network;
- Secondly, the multi-controller placement is performed by using hybrid harmony search algorithm and particle swarm optimization algorithm (HSA-PSO), which reduces the communication latency between the switch to the controller by selecting an optimal location to place the controller.

The performance of the proposed work is evaluated based on several performance metrics such as propagation latency, Round Trip Time (RTT), matrix of Time Session (TS), delay, reliability, throughput, cost, and fitness value.

The remainder of this paper is organized as follows. Section 2 represents the preliminaries of the proposed MC-SDN method. Section 3 illustrates the literature survey, which is followed by the problem statement in Section 4. Section 5 describes the proposed MC-SDN method in a detailed manner. The experimental results of the proposed MC-SDN method are explained in Section 6. Finally, Section 7 concludes the proposed MC-SDN method and the future scope of this research.

2. Preliminaries

This section introduces the preliminary knowledge about the hybrid meta-heuristic algorithm called Harmony Search Algorithm (HSA) and Particle Swarm Optimization algorithm (PSO) to provide a better understanding of the proposed work.

2.1. Harmony Search Algorithm (HSA)

Harmony Search Algorithm (HSA) is a metaheuristics algorithm that is inspired by musicians, whose aim is to improve the music by adjusting the musical instrument parameters for a better harmony state. The HSA supports a wide variety of optimization problems that perform better than the prior optimization algorithms in terms of mathematical analysis [16]. The HSA algorithm solves a wide range of engineering optimization problems in various fields such as Puzzle Solving, Routing problems and Distributed network problems. The typical HSA consists of three steps, namely initialization, improvisation, and updating, which can be formulated as

$$H_i^{New} = \begin{cases} H_i(l) \in \{H_i(1), H_i(2), \ldots, H_i(l)\} & R_1 > hmcr \\ H_i(l) \in \{H_i^1, H_i^2, \ldots H_i^{HMS}\} & R_1 \leq hmcr \\ H_i(l) + R_3 * BW & R_2 \leq par \end{cases}, \tag{1}$$

where H_i^{New} is a new harmony, $R_1, R_2,$ and R_3 are the random numbers between [0, 1], hmcr denotes the harmony memory consideration rate, par denotes the pitch adjustment rate, BW denotes the bandwidth, and H_i^{HMS} denotes the harmony memory solution of harmonic.

The HSA algorithms are mostly limited to a high convergence rate, which affects the accuracy. Hence, some of the modifications are introduced in the HSA algorithm, which are:

- Dynamic adaption of HSA parameters to enhance the convergence rate;
- Improved operations in terms of accuracy by combining with other optimization algorithms (i.e., hybrid solutions);
- Set of pre-defined rules for new harmony (i.e., Hybrid harmony).

2.2. Particle Swarm Algorithm (PSO)

The particle swarm algorithm is the stochastic optimization algorithm that is inspired by the population (i.e., behavior) of several animals such as birds' flocks, fish schools, etc. [17]. Various PSO algorithms are mostly correlated to two main research techniques, such as evolutionary algorithms and artificial algorithms. This algorithm provides better optimization results when compared with genetic optimization algorithms. It is performed based on the swarm size of the position vector of every particle at a finite dimension with the velocity vector. Optimal positions of every individual are experienced by the swarm and update the optimal position of individuals from the initial state. We update the optimal position of every individual using swarm by considering position and velocity. The position update formula is expressed as follows

$$P_{ij}(R+1) = P_{ij}(R) + v_{ij}(R+1), \tag{2}$$

where $P_{ij}(R)$ represent position of the particles, $v_{ij}(R+1)$ denotes the velocity of the particles, i denotes particle and R denotes iteration. In addition, the velocity updating is performed by considering the inertia weight to the update formula of velocity, which is expressed as follows

$$v_{ij}(R+1) = W(R)v_{ij}(R) + A_1R_1\left(P\_Best_{ij} - P_{ij}(R)\right) + A_2R_2\left(G_{Best} - P_{ij}(R)\right), \tag{3}$$

where $v_{ij}(R)$ represent the particle velocity of iteration, $W(R)$ represent the weight value, $A_1,\ A_2$ represent the acceleration constants, $R1,\ R2$ denotes the random values between $[0,\ 1]$, and P_Best and G_Best represent the local and global position of the particles.

It overcomes the limitations of the HSA algorithm. This algorithm provides high computational efficiency with efficient control parameters when compared with several heuristic optimization algorithms.

3. Literature Survey

Reference [18] introduced a novel solution for the problem of controller placement in software-defined networks. This work adopts nature-inspired algorithms such as manta ray foraging optimization and salp swarm algorithm for problems in controller placement. To improvise the performance of the individual algorithms, the discretization is performed by triple operators. The discretized algorithms are utilized in a hybrid manner for controller placement. This work solved the problem of controller placement by ensuring the latency in the network. However, the other constraints (i.e., switch requirement, fault tolerance, etc.) are considered for optimal controller placement.

Reference [19] introduced an effective method for the problem of controller placement using heuristic algorithms. Initially, the delay among the switches and controllers is analyzed and provides a delay-aware model. The delay-aware model is assessed by three optimization algorithms, namely bat optimization algorithm, firefly algorithm, and VARNA-based optimization algorithm. All three optimization algorithm parameters are optimized by particle swarm optimization for improved performance. This work only utilized limited network indicators which affected its optimality in solving the controller placement problem.

Reference [20] introduced an optimization algorithm for controller placement in software-defined networks. This work utilizes two algorithms for controller placement, namely the genetic algorithm and PSO algorithm. The particle swarm optimization algorithm initially selects the best controller based on the fitness values by considering its delay constraints. The genetic algorithm is utilized to update the position and velocity of the controllers. This work utilizes genetic algorithm and particle swarm optimization algorithm for controller placement; however, the genetic algorithm is limited with high computation time consumption. Reference [21] introduced an optimized approach for the CPP. This work adopted VARNA-based optimization algorithm, which diminishes the overall time consumption during placement of the controller. This work outperforms the Teacher Learning Based Optimization algorithm and Jaya algorithm in terms of time consumption. This work considers only latency as a constraint of the placement of the controller problem. However, for a better placement, other constraints should also be considered.

Reference [22] proposed controller placement using an optimization algorithm in the SDN network. Here, a controller was placed for each cluster to reduce the latency between the switch to the controller. The proposed method calculates the maximum distance to reduce the latency between the switch to the controller. The proposed method achieved less latency for various types and counts of controller placement. The experimental results demonstrate that the proposed work achieved better performance in fault tolerance compared to existing works. A novel controller placement framework was proposed using multi-criteria-based clustering method [23]. The main aim of this research was to reduce communication latency and end-to-end delay. The proposed SDN topology is constructed in a graph manner. Here, the controller and switches are located using the proposed clustering approach. After that, the path between controller and switch is determined. For that purpose, moth flame optimization was proposed. Finally, the simulation results demonstrate that the proposed work achieved better performance compared to some other existing approaches.

Controller placement problem was solved by VARNA optimization algorithm in the SDN environment [24]. Initially, populations are initialized and classified into two types of varnas based on particle superiority. Varna optimization calculates the fitness values of the particles for two classes. Based on the fitness values, controller placement was performed. The experimental results show that the proposed work achieved better performance compared to other optimization algorithms. Reference [25] proposed an approach to perform controller placement in SDN for reducing link failures. Initially, link investigation was performed to analyze and regularize the rate of link failures. Improved NSGA-II based heuristic algorithm was developed to reduce the controllers count to provide an efficient solution for CPP-MLF multi-link failures. Non-dominated sorting genetic algorithm-II (NSGA-II) is improved specifically in terms of crowding distance and the non-dominated set using an even distribution-based operator and adaptive competition technique to attain optimal Pareto solutions for solving the CPP-MLF. Finally, the load variance of the controllers was evaluated to achieve efficient decisions for placing the controllers efficiently. Experimental analysis was performed by comparing the developed heuristic approach to several previous algorithms. However, this approach is not suitable for large-scale networks because it leads to network load increase.

Reference [26] proposed an approach to perform efficient multi-controller placement in SDN networks. Initially, Steiner tree was implemented for computing the inter-controller optimally by joining the failed links' endpoints through the shortest path by considering controller count, controller capacity, and mapping of the switch controller. Redirecting the flows by a specific feature is named as fast failover in the group tables of OpenFlow. Finally, joint-controller placement is optimally performed by considering the weight parameter of the pre- and post-failure Steiner tree to reduce the link failure. Performance evaluation of this method was performed in terms of network synchronization cost and reconfiguration of failure network, etc. Reference [27] proposed an approach to perform optimal controller placement with optimal selection of controllers using dynamic chaotic-SALP swarm optimization algorithm (SSOA) in an SDN network. Initially, the SSOA algorithm was introduced with chaotic maps to improve the performance of the optimizer. Evaluation of optimal controllers count and connections between controllers and switches were performed in which the optimal allocations and controllers count were performed by chaotic SSA-based algorithms to reduce the cost during deployment and network latency.

Extraction of random parameters by Gaussian distributed implementation of this method is performed using Internet topology zoo and evaluation of this work by considering several parameters.

Reference [28] proposed an approach to perform optimal placement of controllers in a SDN environment. Initially, formalization of a comprehensive mathematical approach for CPP selects the controllers' location, reduces the controllers' count, and node assignment for each controller by considering latency during propagation, capacity of controllers, and load balancing using a heuristic approach as NP-hard. In the heuristic approach, cluster formation was performed between nodes to manage the network and assign controllers for each cluster by selecting an optimal node based on the trade-off method. Finally, the shortest path was selected by considering the network diameter. Experimental analysis was compared with several state-of-the-art works to prove the efficient performance of this work. Reference [29] proposed an approach to perform placement of multi-controllers in SDN by affinity propagation. Initially, the network was partitioned by implementing a modified-affinity propagation-based algorithm, which is a clustering algorithm that computes the clusters count automatically and identification of candidate exemplars to place the SDN controllers by considering the similarity of several parameters such as link bandwidth and Euclidean distance. Finally, simulation was performed by constructing the network topology using internet zoo topology to evaluate the proposed work in terms of latency between inter-controller, imbalance factors, worst case, and average case. Table 2 summarizes the advantages and disadvantages of the related works considered in this survey.

Table 2. Advantages and disadvantages of the related work survey.

Study Ref/No	Advantages	Disadvantages
[18]	Solved CPP by hybrid manner method. Ensuring the latency in the network.	Limitations (e.g., switching requirements, fault tolerance) for optimal controller placement.
[19]	Provides a delay-aware model by bat optimization, firefly algorithm, and VARNA.	Uses only utilized limited network indicators.
[20]	Genetic and PSO algorithm were used to select the best controller based on fitness values by considering their delay constraints.	The genetic algorithm limits with high computation time consumption
[21]	It uses a VARNA-based optimization VBO algorithm that reduces the total time consumption during console mode. Better than the teacher-learning-based optimization algorithm, and the Jaya algorithm in terms of time consumption.	This latency-only work is considered a limitation of CPP and other limitations of a better situation.
[22]	A controller was set up for each group to reduce the latency between S2Cs by the "$" method. The proposed method achieved lower latency for various types of controller mode counts. The results show that the proposed work performed better in fault tolerance.	There is only one failure of the network controller number 5 in terms of path reliability.

Table 2. *Cont.*

Study Ref/No	Advantages	Disadvantages
[23]	They used an uncontrolled type of flame-controlled locus optimizer for three metrics, hop count, propagation latency, and link utilization to assign S2C. They have the reliability of the path and the best positioning of the consoles.	This focused on the clustering method, and there are other limitations The "k" of a few controllers and did not specify the number of failures in the path to the proposed framework.
[24]	They used the VBO algorithm in SDN environment to solve CPP and to reduce the average SDN latency so VBO provides the best dynamical performance.	It is based on clustering.
[25]	An improved heuristic algorithm based on NSGA-II has been developed to improve CPP performance in SDN and reduce link failures for the number of controllers. Crowding distance and non-dominant group adaptive competition were determined to arrive at the optimal Pareto solutions for the CPP-MLF Multi-Link Failures solution.	This approach is not suitable for wide area networks, which leads to increased network load.
[26]	Steiner tree was implemented to calculate the optimum internal controller in terms of the weight parameter to reach the failed links through the shortest path. The number and capacity of the controllers are considered and mapped into the switch. Quick failover in OpenFlow group tables.	There is a cost in case of network synchronization and network reconfiguration of the failure.
[27]	SDN's dynamic salp swarm optimization algorithm (SSOA) was used to get the best performance for optimal controller placement with chaotic mapping. The optimum number of controllers and the connection between controllers and switches have been evaluated in terms of making the optimum number of controller assignments in order to reduce cost during deployment and network latency.	It is based on extracting random parameters.
[28]	Use a comprehensive mathematical approach to a CPP solution, which locates and minimizes controllers by selecting an optimal node based on the swap method and assigning the node to each controller by considering propagation latency, controllers' capacity, and load balancing using a heuristic approach such as NP-hard. The shortest path was chosen by considering the diameter of the grid.	Propagation latency between nodes to controllers is within finite limits in case of distribution between controllers.

Table 2. *Cont.*

Study Ref/No	Advantages	Disadvantages
[29]	A partitioning-based algorithm was used for the clustering feature, where it automatically counts the number of clusters and selects candidate placement of multi-controllers in SDN by considering the similarity of several parameters such as correlation bandwidth and Euclidean distance and thus calculating the latency between the controller and imbalance factors.	Not all controllers are specified for the purpose of partitioning and network capacity.

4. Problem Statement

The controller placement problem is one of the vital problems in SDN networks. The placement of the controller in the precise location would enhance the performance of the network. There are several challenges employed during controller placement, such as high latency during inter-controller communication, propagation delay between switches and controllers, controller fault tolerance, and satisfying the switch requirements.

The latency between controllers is a challenging problem during controller placement. Let the controllers (C_1) and (C_2), C∈C_1 and C_2 want to communicate with each other. The unprecise location between these controllers during controller placement would increase the inter-controller latency, which can be formulated as

$$Lat_{C_1 \leftrightarrow C_2} = \frac{1}{C} \sum_{C} \max_{C \in C_1, C_2} Lat, \tag{4}$$

where $Lat_{C_1 \leftrightarrow C_2}$ denotes the latency between controller C_1 and C_2 and $\max\limits_{C \in C_1, C_2} Lat$ denotes the high latency during communication due to the imprecise placement of controllers.

The propagation delay between switches and controllers would affect the communication reliability between them. Let the switches in the network be $s \in S$, and controllers be $c \in C$, then the propagation delay (PD) between them can be formulated as,

$$PD_{S \leftrightarrow C} = \sum_{N} \max_{S \leftrightarrow C} PD, \tag{5}$$

where $\max\limits_{S \leftrightarrow C} PD$ denotes the increased propagation between controllers and switches and N denotes the SDN network. Selecting a highly fault-tolerant controller is also a major concern to be considered during controller placement. For the aforementioned problems during controller placement, the proposed work aims to provide a precise solution for the CPP with the following objectives. The objectives are

$$Minimize \rightarrow Lat_{C_1 \leftrightarrow C_2}, \tag{6}$$

$$Minimize \rightarrow PD_{S \leftrightarrow C}, \tag{7}$$

$$Maximize \rightarrow Controller\ fault\ tolerence. \tag{8}$$

The existing works attempt to solve the issues in the controller placement. However, an effective solution that addresses the above problems is not yet given. Latency-aware optimal controller placement in precise location by considering link failures was introduced by Ref. [30]. The optimized controller placement solution for the CPP by considering the capacity of controllers was introduced by Ref. [31]. The load balancing method for mitigating the issues during controller placement by considering delay and controller stability was introduced by Ref. [32]. A heuristic approach for mitigating the controller placement problem by considering computation time was introduced by Ref. [33]. The overall common issues employed in the above-listed papers are:

- The existing works are considering only the delay between controllers and switches. However, the controller-to-controller communication after placement needs to be considered for achieving a precise solution for the CPP, which increases the communication latency between controllers, and affects the SDN network performance;
- The selection of constraints during controller placement are latency between switches and controllers, propagation delay, and link failure. However, the controller fault tolerance rate was not considered, which also leads to degradation during controller placement;
- The existing works utilized heuristics algorithms for controller placement problems. However, they achieve less performance in terms of latency and propagation delay, as the parameters of the existing algorithms was not tuned effectively. In addition, the consideration of single optimization algorithms limits them with imprecise controller placement.

To overcome the shortcomings faced by the existing works, the proposed work initially constructs the network as a graph structure to improve the scalability of the SDN network. After that, controller selection is performed to improve the performance of controller placement. The controller selection is performed by utilizing the Firefly optimization algorithm (FA), which considers controller features. The use of this firefly optimization algorithm is to select the optimal controller with high fault tolerance and capacity. After the selection of optimal controllers, controller placement is done in an optimal location with awareness of distance. The selection of optimal location for controller placement is utilized by a hybrid algorithm called Harmony Search Algorithm (HSA) and Particle Swarm Optimization (PSO). The PSO is utilized for HSA parameter initialization, which also improves the controller placement performance. The controller placement problem minimizes the latency between controllers and the propagation delay between switches and controllers. The proposed work achieves the improved throughput and network performance by using the above processes. The proposed MC-SDN model can be considered to be applicable to generic SDN network characteristics and can be used in an efficient, flexible, scalable, and reliable manner.

5. The Proposed Work

This section denotes the research methodologies of the proposed MC-SDN approach. The proposed work provides scalability, reliability, and enhances the overall network performance by three processes, which are explained as follows. Figure 1 represents the overall system model of the proposed MC-SDN architecture with distribution mechanism and selection of controllers by FA algorithm.

5.1. Network Construction

The network considers multiple controllers, switches, and devices. The network topology is constructed based on an undirected graph structure, which is defined as

$$G = (V, E, U), \tag{9}$$

where G represents the graph, U represents the number of controllers, E represents the edges and V represents the connection between the switches and controllers. This topology reduces the latency between switches and controllers. Here, the controllers and nodes are the forwarding elements, therefore we assume the controller location as the node location in the SDN environment. Hence, we need to calculate the value of k, which represents the count of controllers to find the relation of $U \rightarrow V$ mapping, which from through it calculate the objective function. Let us assume $C = (C_1, C_2, \ldots C_n)$ is the number of controllers deployed in the network. $S = (S_1, S_2, \ldots S_n)$ represent the number of switches so that $V = C \cup S$. $n = V$ represents the number of nodes and $k = U$ represents the count of controllers. $P_c = \{p_{c1}, p_{c2}, \ldots, p_{cm}\}$ represents the possibilities of controller placement.

Where m represents the n elements variations that are taken from the group. The calculation of m is defined as follows

$$m = \frac{n!}{k!(n-k)!}. \tag{10}$$

In our network, the controller is selected based on the shortest distance d (s,c) between switch and controller from nodes s ∈ V and c ∈ V.

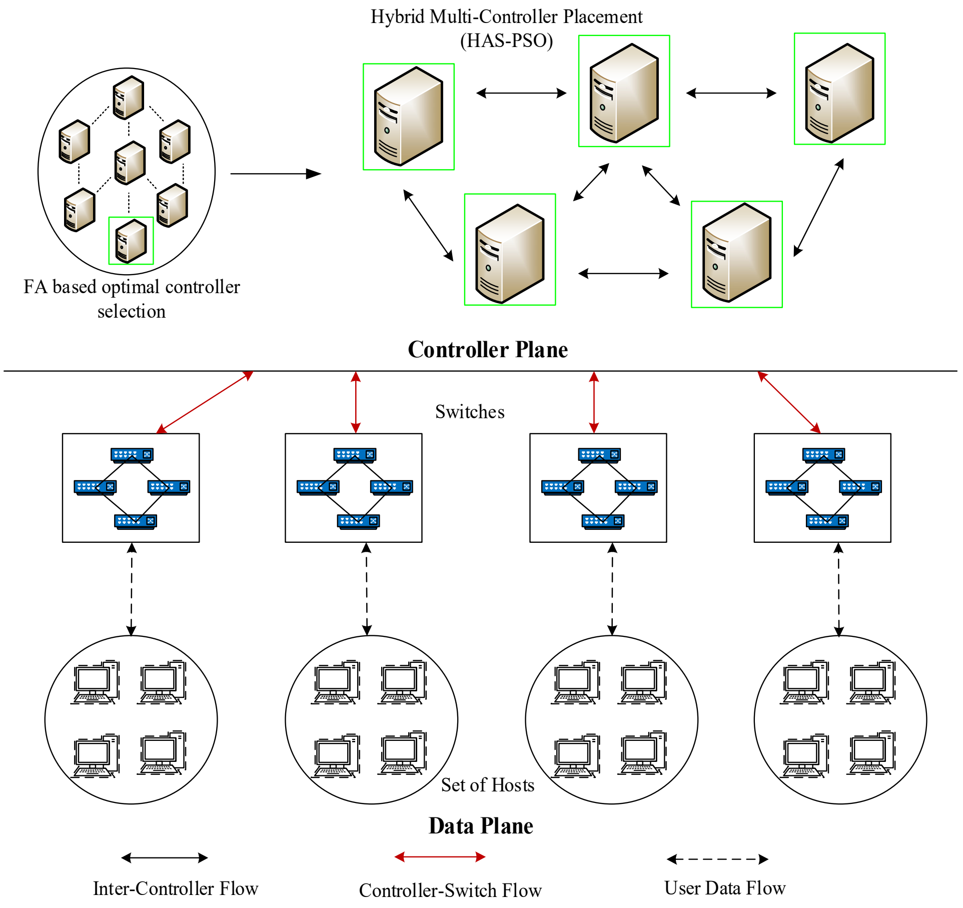

Figure 1. Proposed MC-SDN architecture.

5.2. Optimal Controller Selection

The controller is a core component in the SDN network, which manages the overall network. Therefore, an optimal selection of controllers must be selected to manage the overall network. The optimal selection of controllers is based on the controller features. The types of controllers and features are listed in Table 3, which is provided below.

Table 3. Controller features of SDN.

	Features							
Controllers	**GUI**	**APIs**	**Platform**	**of Version**	**Modularity**	**Language**	**Category**	**Legacy Network**
ONOS	Excellent	NETCONF, OVSDB, OFREST	MAC Windows, Linux	Between 1.0 to 1.3	Excellent	Java	Disseminated	Yes
ODL	Excellent	OVSDB, OFREST, BGPSNMP	MAC windows, Linux	Between 1.0 to 1.3	Excellent	Java	Disseminated	Yes

Table 3. *Cont.*

				Features				
Controllers	**GUI**	**APIs**	**Platform**	**of Version**	**Modularity**	**Language**	**Category**	**Legacy Network**
Flood Light	Fair	OVSDB, OFREST	MAC windows, Linux	Between 1.0 to 1.3	Fair	Java	Single	No
Beacon	Bad	OVSDB, OFREST	MAC windows, Linux	1	Fair	Java	Single	No
Ryu	Fair	OVSDB, OFREST, NET-CONFG	MAC windows, Linux	Between 1.0 to 1.5	Fair	Python	Single	No
POX	Bad	OVSDB, OFREST	MAC windows, Linux	1	Bad	Python	Single	No
NOX	Bad	OVSDB, OFREST	Linux	1	Bad	C++	Single	No

The proposed controller selection method involves two steps, namely qualitative and quantitative approaches by the optimization algorithm. The proposed work utilizes the FA algorithm. FA is the nature-inspired algorithm, which is inspired by the beetles that produce attractive lights from their abdomen part. The working of FA-based controller selection is given. Initially, the controllers feature in the network are initialized, which can be formulated as

$$F_{Cn} = \{F_{C1}, F_{C2}, F_{C3}, F_{C4}, F_{C5}, F_{C6}, F_{C7}, \ldots, F_{Cn}\}. \tag{11}$$

The features that are highly responsible for controller selection are computed first. For each controller in the network, their corresponding effective features are calculated by computing the distance between them based on the attractive behavior of a firefly. Highly effective features (i.e., controller features with the behavior of high fault tolerance) are given much importance. The distance between the features is formulated as

$$FT(D) = \frac{F_C}{D_C}, \tag{12}$$

where $FT(D)$ is a high fault-tolerant feature based on the distance between controllers D_c and F_C denotes the controller features. The selection of highly fault-tolerant features is computed to place the controller in the optimal position. The objective function of the highly fault-tolerant feature can be formulated as

$$FT_{F_C} = f(F_C). \tag{13}$$

From the above FT_{F_C}, we select the controller features that are highly effective (i.e., highly fault-tolerant), which is based on the controller attraction coefficient being the relatively high light absorption coefficient of a firefly, which can be formulated as

$$\aleph(D) = \aleph_0 e^{-\gamma D^2}, \tag{14}$$

where $\aleph(D)$ is the controller absorption coefficient used to attract the best controllers based on the FA algorithm, $\aleph_0$ = best high light intensity for firefly, and γ is the absorption coefficient for absorbing a highly fault-tolerant controller. Based on the controller attractive value $\aleph$, the optimal controller is a selection that can be formulated as

$$\aleph(FT_{F_C}) = \{\aleph(FT_{F_{C1}}), \aleph(FT_{F_{C2}}), \ldots, \aleph(FT_{F_{C7}})\}. \tag{15}$$

The pseudocode of the FA-based controller selection is given by comparing only two controllers as an example. Based on the example, the FA optimally selects the best controller among seven controllers and Figure 2 represents optimal controller selection using FA, depending on the best firefly with high light intensity, which appears from their abdomen (Algorithm 1).

Algorithm 1 Pseudocode of the FA for Optimal Controller Selection

```
Initialize $F_C$
Formulate objective function using (13)
Formulate controller absorption coefficient using (14)
Initialize absorption coefficient using (15)
**While** all $F_C$ **do**
Compute controller feature distance using (12)
**For** $F_{C1} = 1$ to n (all n controllers)
**For** $F_{C2} = 1$ to n (all n controllers)
**If** $(D_{F_{C2}} > D_{F_{C1}})$, select $F_{C2}$ over $F_{C1}$
**End if**
Update the $\aleph(D)$
**End for** $F_{C2}$
**End for** $F_{C1}$
Rank the controller and find the current best controller
**End while**
```

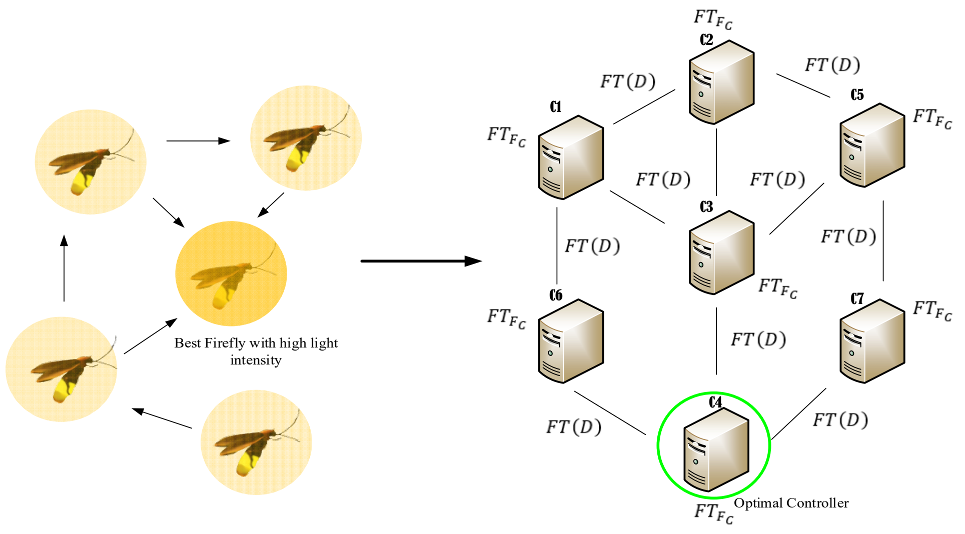

Figure 2. FA-based optimal controller selection.

5.3. Multi-Controller Placement

After completed controller selection, multi-controller placement is initiated. In a SDN environment, multiple controllers are placed to reduce communication latency between the controllers and switches. An efficient controller placement increases the performance of the network. For optimal controller placement, we have proposed a hybrid metaheuristic algorithm that includes the Harmony Search Algorithm (HSA) and Particle Swarm Optimization (PSO). The HSA algorithm easily falls into local optima, hence we need to periodically update the HS parameters using the PSO algorithm. The proposed HSA includes three phases; initialization phase, improvisation phase, and updating phase. Here, improvising is updated for three processes such as memory consideration, adjustment of pitch, and random selection. The HSA algorithm consists of memory storage, namely harmony memory (HM), which includes harmony vectors that store the objective function of HSA. First, we initialize the HM and size of the swarm for evaluating the fitness function, which is defined as follows,

$$\begin{bmatrix} h_1^1 & h_2^1 & \cdots & h_d^1 \\ h_1^2 & h_2^2 & \cdots & h_d^2 \\ \vdots & \vdots & \vdots & \vdots \\ h_1^n & h_2^n & \cdots & h_d^n \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix} = \begin{bmatrix} F_1 \\ F_2 \\ \vdots \\ F_n \end{bmatrix} \text{ where Fitness Function} = \frac{1}{\sum_{d=1}^{B} f(h)_d}. \tag{16}$$

Here, the fitness function is evaluated based on location and distance B and d parameters to provide an efficient optimal solution as follows. After completed initialization, harmony improvisation is performed for generating a new harmony vector $[h'_1, h'_1 \ldots, h'_t]$. Every new harmony vector component h'_j is generated using the following formula

$$h'_j \leftarrow \begin{cases} h'_j \in P_{HM} \; with \; Pb \; of \; HMCR \\ h'_j \in h_j \; with \; Pb \; of \; (1 - HMCR)' \end{cases} \tag{17}$$

where HMCR represents the harmonic memory consideration rate that defines the component selecting probability. The calculation of pitch adjustment for the selected h'_j is defined as follows

$$h'_j \leftarrow \begin{cases} h_j^n \in P_{HM} \; with \; Pb \; of \; PAR \\ \quad h'_j \; with \; Pb \; (1 - PAR) \end{cases}, \tag{18}$$

where PAR represents the pitch adjustment rate and P_{HM} represent the proposed hybrid HSA and PSO algorithm, and h'_j represent the optimal location for controller placement.

The new hybrid harmony vector is calculated based on the value of objective function at every P_Best. If the new harmony vector objective value is better than the objective value of the worst harmony, the new harmony vector is considered in P_{HM}. The worst harmony vector value is rejected from P_{HM}. The optimal position for controller placement is determined based on the best particles in the swarm, which is defined as P_Best. It generates n number of iterations, and the best particle among all is considered as G_Best. The position and velocity of all the particles are updated for each velocity. The current velocity and position are calculated by used Equations (2) and (3) above.

The calculation of weight value is defined as follows

$$W = W_u - (W_u - W_l)\left(\frac{i}{I_{max}}\right), \tag{19}$$

where I_{max} represent the total count of iterations and i represent the current iteration, and W_u and W_l represent the upper and lower limit of the weight values. These processes are continued until the termination criteria are met. Table 4 describes the parameters of the HSA algorithm. Figure 3 illustrates the flow of multi-controller placement.

Table 4. (HSA) parameters.

Parameter Names	Parameter Symbols	Value
Harmony Memory Consideration Rate	HMCR	0.8
pitch Adjusting Rate	PAR	0.2
Random	Rand	0.1
HS iteration	H(I)	100
Harmony Memory	HM	5
Minimum bandwidth	BW	0.1 mbps
Maximum bandwidth	MBW	0.4 mbps

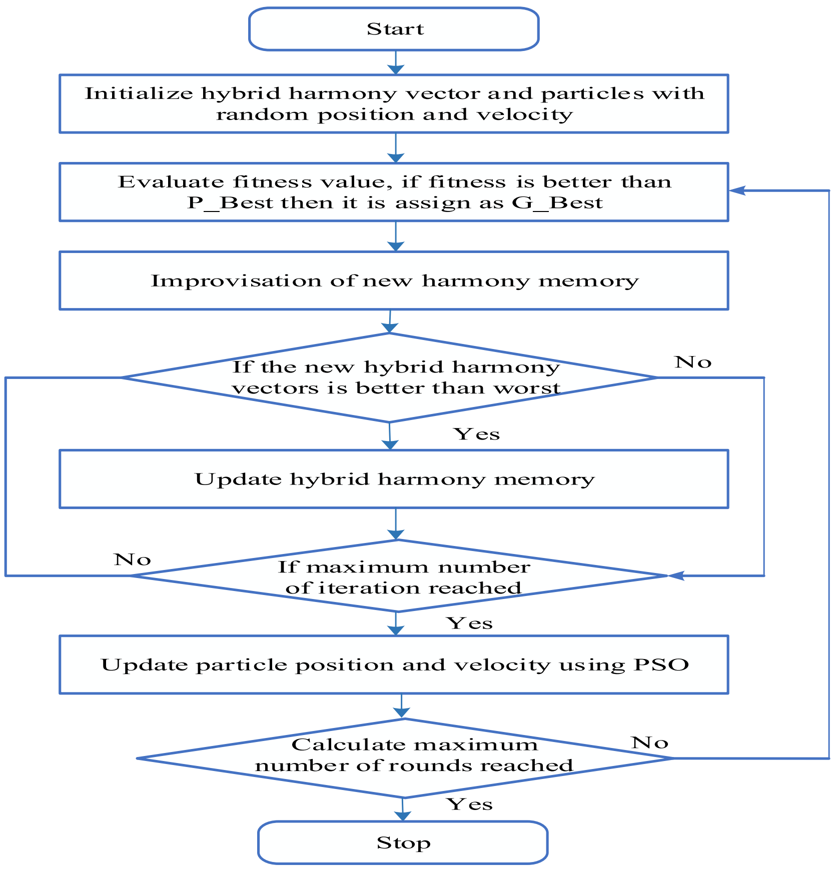
Start
Initialize hybrid harmony vector and particles with random position and velocity
Evaluate fitness value, if fitness is better than P_Best then it is assign as G_Best
Improvisation of new harmony memory
If the new hybrid harmony vectors is better than worst
No
Yes
Update hybrid harmony memory
No
If maximum number of iteration reached
Yes
Update particle position and velocity using PSO
Calculate maximum number of rounds reached
No
Yes
Stop

Figure 3. Hybrid optimization-based multi-controller placement.

6. Experimental Results

The experimental analysis of the proposed MC-SDN method is described in this section, which was carried out for evaluating the performance. This section also consists of four subsections, namely simulation setup, use case, comparative analysis, and research summary.

6.1. Simulation Setup

Implementation of proposed MC-SDN method by CloudSimSDN simulation tool with NetBeans 12.5 integrated development environment (IDE) kit to perform simulation with several entities such as devices, SDN switches, hosts, SDN controller, and cloud. This simulation tool is suitable for performing modeling and simulation of cloud entities and services including its services, in which this simulator is an open-source simulator established by the cloud computing and distribution systems laboratory at Melbourne University. Initially, the CloudSim environment is created with switches of SDN controllers and hosts. Table 5 illustrates the system configurations and Table 6 shows the parameters configuration of the proposed MC-SDN method.

The simulation environment of this MC-SDN method is illustrated in Figure 4, which consists of hosts, SDN switches, and multi-controllers. Hosts send the data to the controller via switches. The optimal controller is selected from the number of SDN controllers based on numerous packet features. The optimal controller is cloned into seven controllers to perform multi-controller placement. Location and distance are considered to place the multi-controller efficiently to reduce the controller placement problem. Optimal controller selection and multi-controller placement increase the communication efficiency between the switches to controllers as well as controllers to controllers.

Table 5. System configurations.

Software Specifications	Operating System	Windows 10 Pro (64 bits)
	IDE	NetBeans 12.5
	Development Kit	JDK 1.8
	Network Simulator	CloudSimSDN
	Language	Java
	Topology Type	Undirected Graph
	No. of Topology	3
Hardware Specifications	Hard Disk	1 T
	CPU	Intel (R) Core (TM) i7-4590S @ 3.00 GHZ
	RAM	8 GB

Table 6. Parameter configurations.

Cloud Server	Operating System	Windows 10 Pro (64 bits)
	Bandwidth	100,000 downlinks and uplinks
	RAM	64 GB
	Delay	900 ms
	MIPS	44,800
Devices	Delay	1 ms
	MIPS	1500
	RAM	4 GB
SDN	Number of controllers	7
	Number of switches	28
	Switch Delay	5 µs
	Bandwidth	Variable

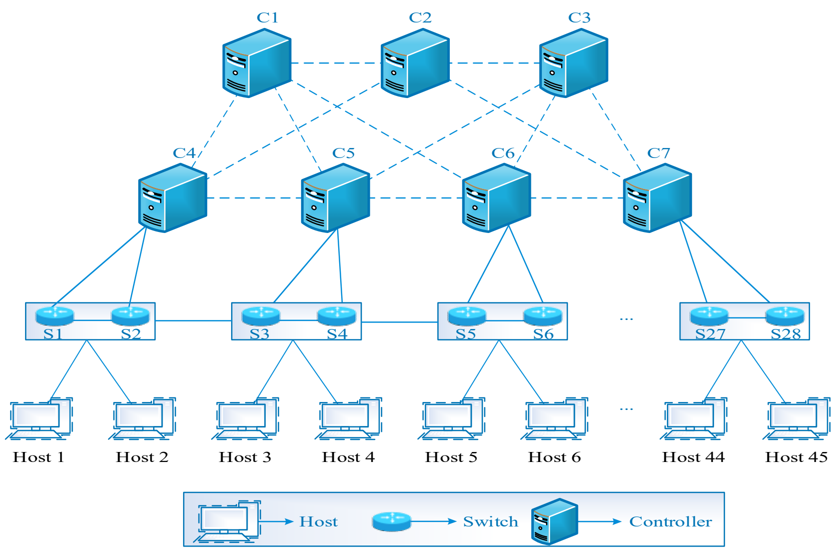

Figure 4. MC-SDN simulation environment.

6.2. Use Case: VANET

In recent times, the SDN has given its applications to Vehicular Ad networks (VANET). The mobile nature of the VANET environment has faced severe challenges in terms of network management. The VANET along with SDN called Software-Defined Vehicular Network (SDVNs) provides network management capability with the help of controllers.

The SDVN environment needs multi controllers, which should be placed in an optimal location to reduce the end-to-end processing delay in the SDVN environment. The proposed MC-SDN approach is easily adaptable to VANET environment in terms of mobility and reliability. Figure 5 denotes the diagrammatic representation of the SDVN environment, which consists of a large number of mobile vehicles that enable a various number of operations such as emergency messages, traffic rules, etc. Initially, the network graph is constructed to manage the network and provides high scalability. The appropriate high fault-tolerant controller is selected based on controller features by using the FA algorithm to provide effective services in the SDVN environment. Finally, the selected controller is placed in multiple optimal locations by using hybrid optimization algorithms. The optimal placement of controllers reduces the latency between controllers and controllers to switches to provide seamless communication between them.

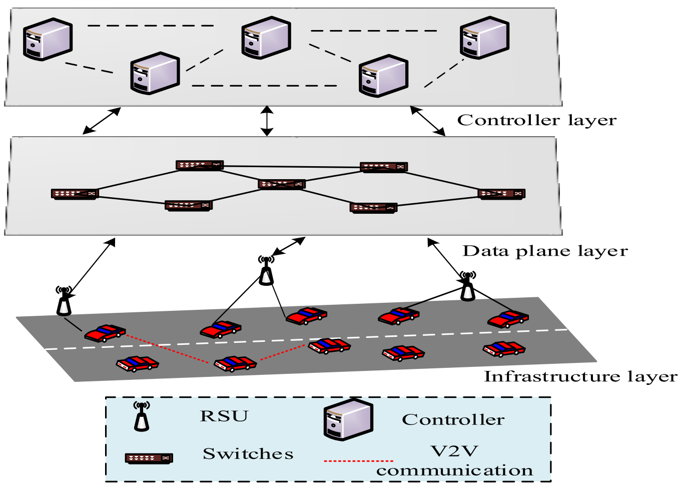

Figure 5. Application scenario of the MC-SDN method.

6.3. Comparative Analysis

Comparison of the proposed MC-SDN method with several existing methods such as Simulated Annealing Failure Foresight Capacitated Controller Placement Problem (SA-FFCCPP) [31] and Garter Snake Optimization Capacitated Controller Placement Problem (GSOCCPP) [33] is performed in this subsection to evaluate their performance. The evaluation of these methods is performed by considering several performance metrics such as propagation delay, average round-trip time (RTT), matrix of time Session (TS), average delay, reliability, and throughput, respectively.

6.3.1. Impact of Propagation Delay

Propagation delay is one of the important metrics that is used to evaluate the delay between switches and controllers during propagation. Propagation delay ($\text{Ƥ}_{\text{đ}}$) is defined as the ratio between the distance (ƛ) (m) and the propagation speed (ɋ) (m/s) in which the formulation of propagation delay is represented as follows

$$\text{Ƥ}_{\text{đ}} = \frac{\text{ƛ}}{\text{ɋ}}. \tag{20}$$

Figure 6 illustrates the comparison of the propagation delay of the proposed MC-SDN method with several existing approaches such as GSOCCPP and SA-FFCCP methods in terms of the number of iterations. An SDN network with low propagation delay attains efficient communication between switches and controllers. The propagation delay decreases when increasing the number of iterations. In the previous methods, controller placement was performed by considering only the distance between switches and controllers. In addition, poor tuning of algorithms leads to poor controller placement, which increases the propagation delay. In the proposed MC-SDN method, controller placement is performed by considering the distance using a hybrid optimization algorithm that increases the performance of controller placement, which reduces the propagation delay. The graphical results show that the proposed MC-SDN method achieves low propagation latency (11 to 20 ms) when compared with SA-FFCCPP (22 to 40 ms) and GSOCCPP (27 to 45 ms) methods. Table 7 describes the variation of propagation delay between the proposed MC-SDN method and previous approaches.

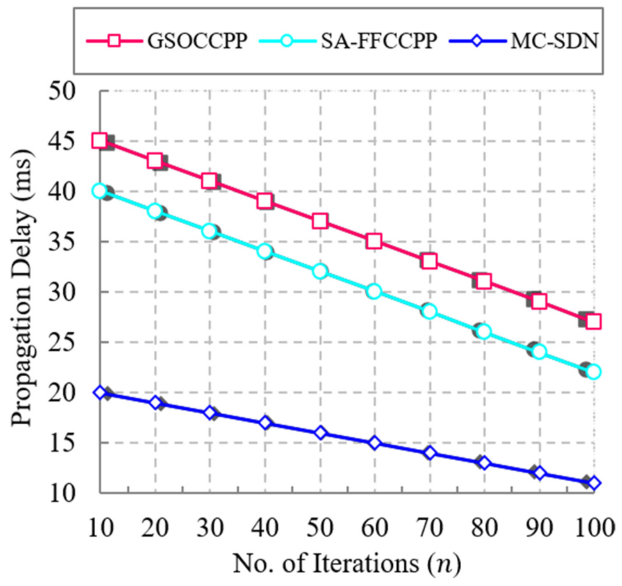

Figure 6. Comparison of propagation delay.

Table 7. Numerical analysis of propagation delay (ms).

Methods	Number of Iterations
GSOCCPP	36.2 ± 0.5
SA-FFCCPP	31.3 ± 0.3
MC-SDN	15.5 ± 0.1

6.3.2. Impact of Average Round-Trip Time (RTT)

This metric is used to measure the time taken between the devices to transmit and receive the packets from source to destination. It is measured by the difference between the packet return time (ζ) to the packet sending time (δ) and it is evaluated in ms, in which the formulation of RTT (η) is represented as follows

$$\eta = \zeta - \delta \tag{21}$$

Figure 7 represents the comparison of average RTT between the proposed MC-SDN method with several state-of-the-art works concerning the number of iterations. Low RTT in a network achieves high throughput. Average RTT increases by increasing the number of iterations. Controller placement was considered in the previous works, however, lack of considering the efficient controller leads to high RTT. In the proposed work, optimal controller is selected by using the FA algorithm by considering numerous controller features, which reduce the RTT when compared with the state-of-the-art works. The comparative results prove that the proposed MC-SDN method achieves low RTT when compared with other existing approaches. The proposed MC-SDN method achieves an average RTT of about 11 ms, which is 4 ms lower than the SA-FFCCPP method and 8 ms lower than the GSOCCPP method. The changes in average RTT of the proposed MC-SDN method and several existing methods are described in Table 8.

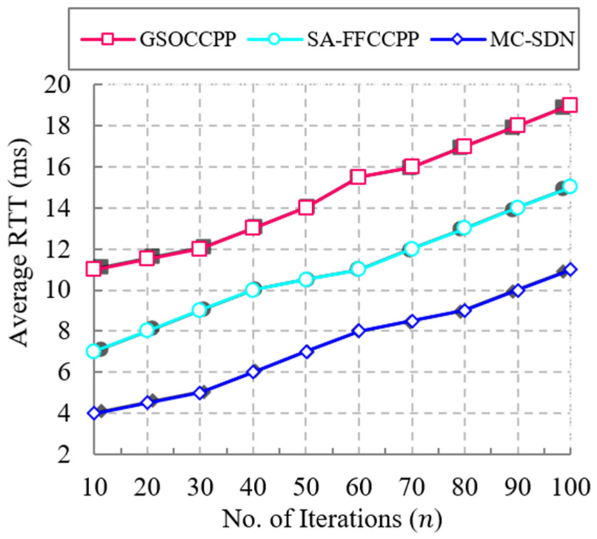

Figure 7. Comparison of average RTT.

Table 8. Numerical analysis of average RTT (ms).

Methods	Number of Iterations
GSOCCPP	14.7 ± 0.4
SA-FFCCPP	10.95 ± 0.2
MC-SDN	7.3 ± 0.1

6.3.3. Impact of Matrix of Time Session (TS)

The matrix of TS (ψ) is used to calculate the amount of time taken for performing efficient communication between switches to controllers. It is measured by the number of switches and the position of servers in the ratio between the amount of time taken for a specific Session (ς) to the overall Session count (6), which is formulated as follows

$$\psi = \frac{\varsigma}{6}. \tag{22}$$

Figure 8 shows the comparative analysis of the matrix of time session between the proposed MC-SDN approach to several existing works for the number of iterations. A network with low matrix of time session achieves better transmission efficiency. In the previous works, controller placement was performed by considering the shortest distance between the switches and controllers. However, the lack of considering the controllers' optimal increases the time for every session. In addition, a single controller also leads to high matrix of time session. To overcome these issues, optimal controller is selected in the network based on packet features using the FA algorithm, which reduces the matrix of time session efficiently when compared with existing works. From the figure, it proves that the proposed MC-SDN method achieves low matrix of time session in the range of about 300 for 100 number of iterations, which is 60 greater than SA-FFCCPP and 120 greater than GSOCCPP methods. The graphical variations of matrix of TS between the proposed MC-SDN method and the state-of-the-art works are illustrated in Table 9.

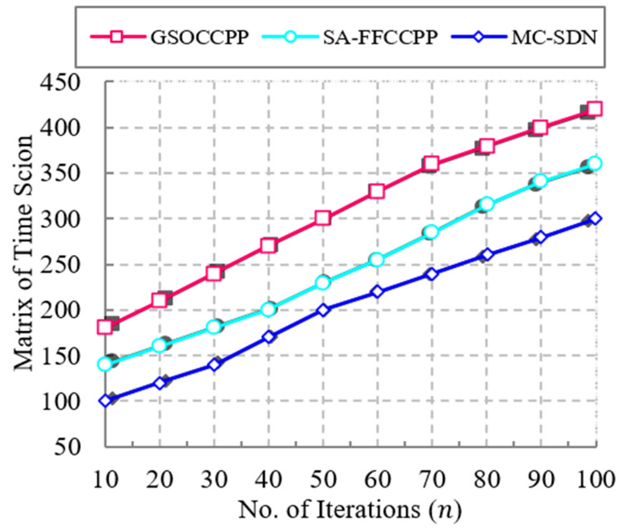

Figure 8. Comparison of the matrix of time session (TS).

Table 9. Numerical analysis of the matrix of time session (TS).

Methods	Number of Iterations
GSOCCPP	309.2 ± 0.5
SA-FFCCPP	246.5 ± 0.4
MC-SDN	203.5 ± 0.1

6.3.4. Impact of Average Delay

Average delay ($\underline{d}$) is defined as the amount of additional time taken for packet delivery from data to the control plane in the proposed MC-SDN method. Average delay is measured by the ratio of actual packet size ($\mathfrak{H}$) to the amount of bandwidth available ($\tilde{\ddot{\upsilon}}$), which is formulated below.

$$\underline{d} = \frac{\mathfrak{H}}{\tilde{\ddot{\upsilon}}}. \tag{23}$$

The comparison of the proposed MC-SDN method with several state-of-the-art works in terms of average delay with respect to the number of iterations is shown in Figure 9. A network with low average delay achieves a high packet delivery ratio. Average delay increases with respect to the increase of iterations. In the proposed MC-SDN method, low average delay is achieved by using the hybrid optimization algorithm, which consists of the HSA and PSO algorithms, and provides better results for placement of controllers, which reduces the average delay, whereas the previous methods perform controller placement by using a single optimization algorithm, which reduces the precision of controller placement and reduces the average delay. The results show that the proposed MC-SDN method achieves low average delay when compared with previous works. The proposed MC-SDN method has an average delay of about 100 ms, which is 30 ms faster than the SA-FFCCPP method and 60 ms faster than the GSOCCPP method. The average delay graphical changes of the proposed MC-SDN and other approaches are described in Table 10.

Table 10. Numerical analysis of average delay (ms).

Methods	Number of Iterations
GSOCCPP	108.3 ± 0.4
SA-FFCCPP	95.3 ± 0.3
MC-SDN	78.8 ± 0.1

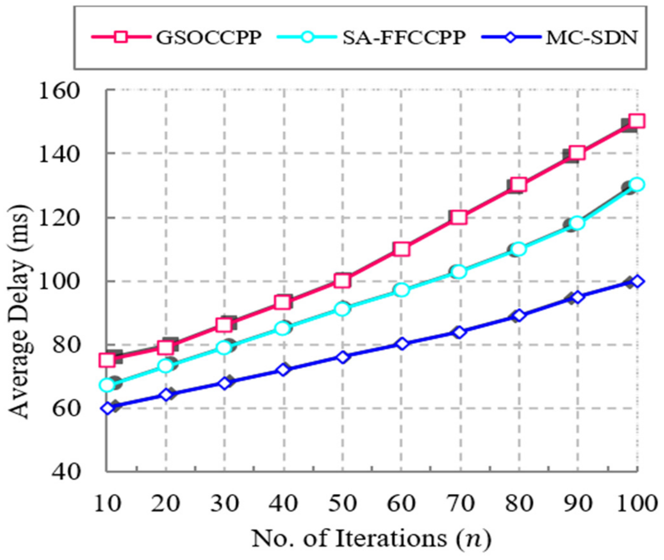

Figure 9. Impact of average delay.

6.3.5. Impact of Reliability

This metric is used to evaluate the effectiveness of the proposed MC-SDN method in terms of efficient optimal placement of controllers. A network with high reliability provides efficient communication with high accuracy.

$$\mathrm{RL} = \max \sum_{v \in V} \sum_{s \in S} p(v, s). \tag{24}$$

RL represents the reliability to minimize the control path between the MC-S and $p(v, s)$ indicates the available probability of the control path.

Figure 10 illustrates the comparison of the proposed MC-SDN method and several previous works in terms of reliability to the number of iterations. Increasing the number of iterations increases the reliability. In the existing methods, controller placement was performed with poorly tuned optimization algorithms and inefficient communication between controller to controller in the SDN network, which reduces the reliability of the network. In the proposed MC-SDN method, these problems are addressed, and performed efficiently tuned hybrid algorithms for placing the controllers and optimal selection of controllers to perform efficient communication between controller to controller, which increases the reliability when compared with state-of-the-art approaches. From the figure, it is clearly shown that the proposed MC-SDN method achieves high reliability when compared with other previous works. The proposed MC-SDN method attains a reliability of about 0.95 for 100 iterations, which are 0.07 greater than the SA-FFCCPP method and 0.11 greater than the GSOCCPP method for the same number of iterations. Table 11 describes the reliability variations of the proposed MC-SDN method and several existing works.

Table 11. Numerical analysis of reliability.

Methods	Number of Iterations
GSOCCPP	0.75 ± 0.5
SA-FFCCPP	0.79 ± 0.4
MC-SDN	0.88 ± 0.1

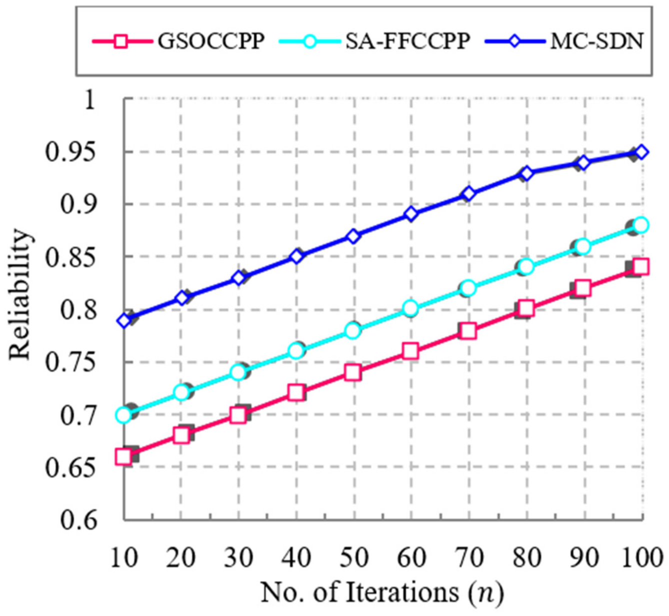

Figure 10. Comparison of reliability.

6.3.6. Impact of Throughput

Throughput (ɯ) of the network is defined as the amount of data delivered to the receiver from the transmitter (ɥ) within a specific amount of time (ξ). The throughput calculation is formulated below.

$$ɯ = \frac{ɥ}{\xi}. \tag{25}$$

Figure 11 represents the comparison of throughput between several previous methods and the proposed MC-SDN method in terms of number of switches in the network. A network with high throughput attains a high packet delivery rate. The figure shows that the throughput increased by increasing the number of switches. The comparative results show that the proposed MC-SDN method achieves high throughput when compared with previous approaches. In the existing methods, controller placement was performed by considering some constraints like communication latency, propagation delay, etc. However, lack of considering the fault-tolerant rate reduces the throughput of the network. In the proposed MC-SDN method, optimal controller selection is performed using the FA algorithm by considering numerous packet features that provide a high fault tolerance rate, which increases the throughput when compared with state-of-the-art works. From the figure, it is proved that the proposed MC-SDN method achieves high throughput of about 300 k response/s, which is 30 k response/s higher than the SA-FFCCPP approach and 60 k response/s higher than the GSOCCPP approach. The throughput variations between the proposed MC-SDN method and other existing methods are described in Table 12.

Table 12. Numerical analysis of throughput (response/s) (k).

Methods	Number of Switches
GSOCCPP	150.2 ± 0.3
SA-FFCCPP	165.7 ± 0.2
MC-SDN	197.1 ± 0.1

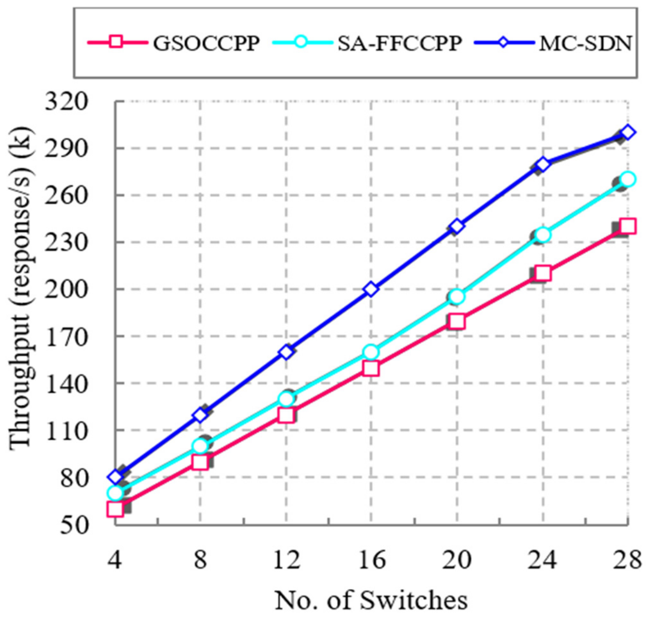

Figure 11. Comparison of throughput.

6.3.7. Impact of Cost

This metric is used to determine the cost of the entire network based on time. A network with low cost that achieves all the necessities provides efficient Quality of Service (QoS). Figure 12 illustrates the comparison of cost with respect to time for the proposed MC-SDN method and several previous approaches. The graph shows that the cost increases with increasing time. In the previous methods, controller-to-controller communication was not efficiently performed, which is more time consuming and increases the cost.

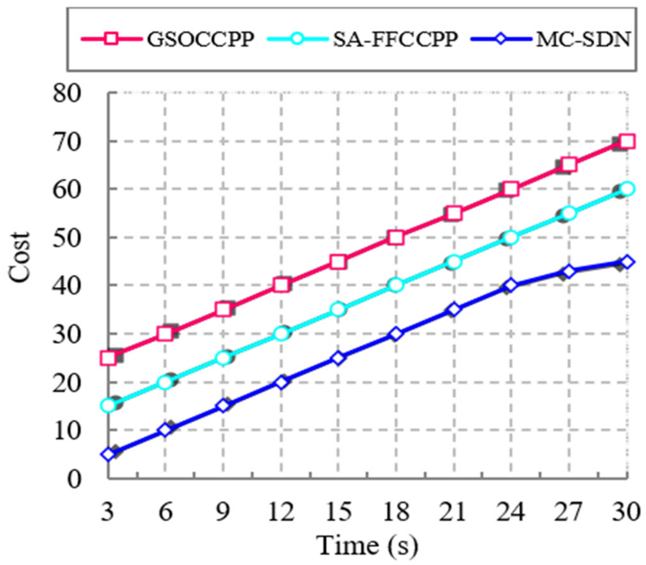

Figure 12. Comparison of cost.

In the proposed MC-SDN method, the communications between switches to controllers and controllers to controllers are efficiently performed by selecting an optimal controller and placing multi-controllers using FA for selecting optimal controllers by considering location and distance, and multi-controller placement is performed using hybrid optimization algorithms by considering packet features to overcome these issues, which provide efficient communication that results in low cost when compared with previous approaches. The comparative results show that the proposed MC-SDN method achieves a low cost of about 45, which is 15 lower than the SA-FFCCPP method and 25 lower than the GSOCCPP method. Table 13 describes the cost variation of the proposed MC-SDN method and other previous methods.

Table 13. Numerical analysis of cost.

Methods	Number of Switches
GSOCCPP	47.5 ± 0.4
SA-FFCCPP	37.5 ± 0.2
MC-SDN	26.8 ± 0.1

6.3.8. Impact of Fitness Value

This metric is one of the important metrics used to analyze the fitness value of the proposed MC-SDN method. Fitness value is calculated by evaluating the optimal solution for the controller placement problem for analyzing the solutions' goodness (i.e., fitness) to the problem. An optimization with a high fitness value provides an efficient optimal solution. Figure 13 illustrates the comparison of fitness value between the previous approaches and the proposed MC-SDN method in terms of number of iterations. The graph clearly shows that the fitness value decreases when the number of iterations increases. In the SA-FFCCPP method, the authors used a simulated annealing optimization algorithm and in the GSOCCPP method the authors implemented the garter snake optimization algorithm for providing solutions to controller placement. However, the fitness value of these algorithms does not fit the controller placement problem efficiently due to poor adaptation mechanisms and accuracy.

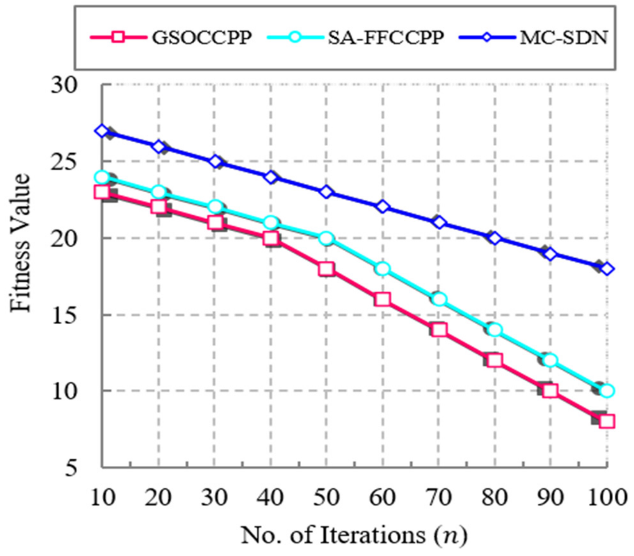

Figure 13. Comparison of fitness values.

To overcome these problems, we use hybrid optimization algorithms (i.e., HSA and PSO), which are optimization algorithms that increase the fitness value when compared with other existing works by dynamic adaptation to increase the convergence rate, which increases the controller placement accuracy. The graphical results show that the proposed MC-SDN method achieves a high fitness value when compared with several existing works. The proposed MC-SDN attains a high fitness value with an average of about 22.5, which is 4.5 greater than the SA-FFCCPP method and 6 greater than the GSOCCPP method. The variation of fitness values for the proposed MC-SDN method and other previous methods are described in Table 14.

Table 14. Numerical analysis of fitness values.

Methods	Number of Switches
GSOCCPP	16.4 ± 0.5
SA-FFCCPP	18.2 ± 0.4
MC-SDN	22.5 ± 0.1

6.4. Research Summary

This section emphasizes the discussion on the performance of the proposed MC-SDN approach. The overall method is proposed to alleviate the existing problems in SDN controller placement. The construction of network as graphical structure supports the network scalability and improves the throughput rate. The overall network performance is enhanced by selecting an optimal controller considering various controller features, which utilize an optimization algorithm. The selected optimal controllers are distributed and placed in an optimal location by using a hybrid metaheuristics algorithm, which considers the latency between controllers and the delay between controllers and switches, respectively.

7. Conclusions and Future Work

A hybrid metaheuristic algorithm is proposed in this research to deploy multiple controllers effectively, in order to reduce communication and propagation latency and improve throughput and reliability. Initially, network construction is performed to improve the scalability and connectivity between switches and controllers. After that, the optimal controller is selected based on controller features using a optimization algorithm, which improves the network performance in a SDN environment. Finally, multiple controllers are placed based on the selected controller using a hybrid metaheuristic algorithm, which increases the convergence rate that reduces deployment latency and communication latency in the environment. The simulation is performed by CloudsimSDN simulation tool, and the simulation result shows that the proposed MC-SDN approach achieved superior performance in multi-controller placement compared to other state-of-the-art works. In the future, we plan to increase the security during multi-controller placement in a SDN environment. In addition, the issue of in-band or out-of-band control approaches will be considered more thoroughly in order to better adjust the proposed methodology in the SDN environments.

Author Contributions: Conceptualization, N.S.R., S.T.F.A.-J. and K.S.J.; methodology, N.S.R., S.T.F.A.-J. and K.S.J.; software, N.S.R., S.T.F.A.-J. and K.S.J.; validation, N.S.R., S.T.F.A.-J. and K.S.J.; formal analysis, N.S.R., S.T.F.A.-J. and K.S.J.; investigation, N.S.R., S.T.F.A.-J. and K.S.J.; resources, N.S.R., S.T.F.A.-J. and K.S.J.; data curation, N.S.R., S.T.F.A.-J. and K.S.J.; writing and original draft preparation, N.S.R., S.T.F.A.-J. and K.S.J.; writing and review editing, N.S.R., S.T.F.A.-J. and K.S.J.; visualization, N.S.R., S.T.F.A.-J. and K.S.J.; supervision, N.S.R., S.T.F.A.-J. and K.S.J.; project administration, N.S.R., S.T.F.A.-J. and K.S.J.; funding acquisition, N.S.R., S.T.F.A.-J. and K.S.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All data were presented in the main text.

Acknowledgments: We would like to thank all who gave us support to complete this paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Qi, Y.; Wang, D.; Yao, W.; Li, H.; Cao, Y. Towards multi-controller placement for SDN based on density peaks clustering. In Proceedings of the 2019 IEEE International Conference on Communications (ICC), Shanghai, China, 20–24 May 2019; pp. 1–6.
2. Mohanty, S.; Priyadarshini, P.; Sahoo, B.; Sethi, S. A Reliable Capacitated Controller Placement in Software Defined Networks. In Proceedings of the 3rd International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 27–29 March 2019; pp. 822–827.
3. Ashrafi, M.; Faroq, A.-T.; Correia, N. Placement of Controllers in Software Defined Networking under Multiple Controller Mapping. *KnE Eng.* **2020**, *5*, 394–404. [CrossRef]
4. Moradi, A.; Abdi Seyedkolaei, A.; Hosseini, S.A. Controller placement in software defined network using iterated local search. *J. Artif. Intell. Data Min.* **2020**, *8*, 55–65.
5. Zhang, X.; Li, L.; Yan, C.-B. Robust controller placement based on load balancing in software defined networks. In Proceedings of the IEEE International Conference on Networking, Sensing and Control (ICNSC), Nanjing, China, 30 October 2020; pp. 1–6.
6. Fan, Y.; Wang, L.; Yuan, X. Controller placements for latency minimization of both primary and backup paths in SDNs. *Comput. Commun.* **2020**, *163*, 35–50. [CrossRef]
7. Lin, N.; Zhao, Q.; Zhao, L.; Hawbani, A.; Liu, L.; Min, G. A novel cost-effective controller placement scheme for software-defined vehicular networks. *IEEE Internet Things J.* **2021**, *8*, 14080–14093. [CrossRef]
8. Balakiruthiga, B.; Deepalakshmi, P. A Distributed Energy Aware Controller Placement Model for Software-Defined Data Centre Network. *Iran. J. Sci. Technol. Trans. Electr. Eng.* **2021**, *45*, 1083–1101. [CrossRef]
9. Killi, B.R.; Rao, S.V. Poly-stable matching based scalable controller placement with balancing constraints in SDN. *Comput. Commun.* **2020**, *154*, 82–91. [CrossRef]
10. Santos, D.; Gomes, T.; Tipper, D. SDN controller placement with availability upgrade under delay and geodiversity constraints. *IEEE Trans. Netw. Serv. Manag.* **2021**, *18*, 301–314. [CrossRef]
11. Alowa, A.; Fevens, T. Towards minimum inter-controller delay time in software defined networking. *Procedia Comput. Sci.* **2020**, *175*, 395–402. [CrossRef]
12. Amiri, E.; Alizadeh, E.; Rezvani, M.H. Controller selection in software defined networks using best-worst multi-criteria decision-making. *Bull. Electr. Eng. Inform.* **2020**, *9*, 1506–1517. [CrossRef]
13. Dvir, A.; Haddad, Y.; Zilberman, A. The controller placement problem for wireless SDN. *Wirel. Netw.* **2019**, *25*, 4963–4978. [CrossRef]
14. Calle, E.; Martínez, D.; Mycek, M.; Pióro, M. Resilient backup controller placement in distributed SDN under critical targeted attacks. *Int. J. Crit. Infrastruct. Prot.* **2021**, *33*, 100422. [CrossRef]
15. Santos, D.; Gomes, T. Joint optimization of primary and backup controller placement and availability link upgrade in SDN networks. *Opt. Switch. Netw.* **2021**, *42*, 100634. [CrossRef]
16. Rosle, M.S.; Mohamad, M.S.; Choon, Y.W.; Ibrahim, Z.; González-Briones, A.; Chamoso, P.; Corchado, J.M. A Hybrid of Particle Swarm Optimization and Harmony Search to Estimate Kinetic Parameters in Arabidopsis thaliana. *Processes* **2020**, *8*, 921. [CrossRef]
17. Bala, N.M.; bin Safei, S. A Hybrid Harmony Search and Particle Swarm Optimization Algorithm (HSPSO) for Testing Non-functional Properties in Software System. *Stat. Optim. Inf. Comput. (SOIC)* **2021**, *10*, 3.
18. Firouz, N.; Masdari, M.; Sangar, A.B.; Majidzadeh, K. A novel controller placement algorithm based on network portioning concept and a hybrid discrete optimization algorithm for multi-controller software-defined networks. *Clust. Comput.* **2021**, *24*, 2511–2544. [CrossRef]
19. Li, Y.; Guan, S.; Zhang, C.; Sun, W. Parameter Optimization Model of Heuristic Algorithms for Controller Placement Problem in Large-Scale SDN. *IEEE Access* **2020**, *8*, 151668–151680. [CrossRef]
20. Liao, L.; Leung, V.; Li, Z.; Chao, H.-C. Genetic Algorithms with Variant Particle Swarm Optimization Based Mutation for Generic Controller Placement in Software-Defined Networks. *Symmetry* **2021**, *13*, 1133. [CrossRef]
21. Singh, A.K.; Maurya, S.; Kumar, N.; Srivastava, S. Heuristic approaches for the reliable SDN controller placement problem. *Emerg. Telecommun. Technol.* **2020**, *31*, e3761. [CrossRef]
22. Dhar, M.; Bhattacharyya, B.K.; Kanti Debbarma, M.; Debbarma, S. A new optimization technique to solve the latency aware controller placement problem in software defined networks. *Emerg. Telecommun. Technol.* **2021**, *32*, e4316. [CrossRef]
23. Jalili, A.; Keshtgari, M.; Akbari, R. A new framework for reliable control placement in software-defined networks based on multi-criteria clustering approach. *Soft Comput.* **2020**, *24*, 2897–2916. [CrossRef]
24. Singh, A.K.; Maurya, S.; Srivastava, S. Varna-based optimization: A novel method for capacitated controller placement problem in SDN. *Front. Comput. Sci.* **2020**, *14*, 143402. [CrossRef]
25. Hu, T.; Ren, Q.; Yi, P.; Li, Z.; Lan, J.; Hu, Y.; Li, Q. An efficient approach to robust controller placement for link failures in Software-Defined Networks. *Future Gener. Comput. Syst.* **2021**, *124*, 187–205. [CrossRef]
26. Das, T.; Gurusamy, M. Controller placement for resilient network state synchronization in multi-controller SDN. *IEEE Commun. Lett.* **2020**, *24*, 1299–1303. [CrossRef]
27. Ateya, A.A.; Muthanna, A.; Vybornova, A.; Algarni, A.D.; Abuarqoub, A.; Koucheryavy, Y.; Koucheryavy, A. Chaotic salp swarm algorithm for SDN multi-controller networks. *Eng. Sci. Technol. Int. J.* **2019**, *22*, 1001–1012. [CrossRef]

28. Schütz, G.; Martins, J.A. A comprehensive approach for optimizing controller placement in Software-Defined Networks. *Comput. Commun.* **2020**, *159*, 198–205. [CrossRef]
29. Sminesh, C.; Kanaga, E.G.M.; Sreejish, A. A multi-controller placement strategy in software defined networks using affinity propagation. *Int. J. Internet Technol. Secur. Trans.* **2020**, *10*, 229–253. [CrossRef]
30. Ramya, G.; Manoharan, R. Enhanced optimal placements of multi-controllers in SDN. *J. Ambient. Intell. Humaniz. Comput.* **2021**, *12*, 8187–8204. [CrossRef]
31. Aravind, P.; Varma, G.S.; Reddy, P.P. Simulated annealing based optimal controller placement in software defined networks with capacity constraint and failure awareness. *J. King Saud Univ. Comput. Inf. Sci.* **2021**. [CrossRef]
32. Li, G.; Wang, X.; Zhang, Z. SDN-Based Load Balancing Scheme for Multi-Controller Deployment. *IEEE Access* **2019**, *7*, 39612–39622. [CrossRef]
33. Torkamani-Azar, S.; Jahanshahi, M. A new GSO based method for SDN controller placement. *Comput. Commun.* **2020**, *163*, 91–108. [CrossRef]