Scientific Research Publishing

Communications and Network, 2020, 12, 199-219
https://www.scirp.org/journal/cn
ISSN Online: 1947-3826
ISSN Print: 1949-2421

Using Heuristics to the Controller Placement Problem in Software-Defined Multihop Wireless Networking

Afsane Zahmatkesh, Chung-Horng Lung

Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada
Email: afsane.zahmatkesh@carleton.ca, chlung@sce.carleton.ca

How to cite this paper: Zahmatkesh, A. and Lung, C.-H. (2020) Using Heuristics to the Controller Placement Problem in Software-Defined Multihop Wireless Networking. *Communications and Network*, **12**, 199-219.
https://doi.org/10.4236/cn.2020.124010

Received: October 9, 2020
Accepted: November 23, 2020
Published: November 26, 2020

Abstract

Solving the controller placement problem (CPP) in an SDN architecture with multiple controllers has a significant impact on control overhead in the network, especially in multihop wireless networks (MWNs). The generated control overhead consists of controller-device and inter-controller communications to discover the network topology, exchange configurations, and set up and modify flow tables in the control plane. However, due to the high complexity of the proposed optimization model to the CPP, heuristic algorithms have been reported to find near-optimal solutions faster for large-scale wired networks. In this paper, the objective is to extend those existing heuristic algorithms to solve a proposed optimization model to the CPP in software-defined multihop wireless networking (SDMWN). Our results demonstrate that using ranking degrees assigned to the possible controller placements, including the average distance to other devices as a degree or the connectivity degree of each placement, the extended heuristic algorithms are able to achieve the optimal solution in small-scale networks in terms of the generated control overhead and the number of controllers selected in the network. As a result, using extended heuristic algorithms, the average number of hops among devices and their assigned controllers as well as among controllers will be reduced. Moreover, these algorithms are able to lower the control overhead in large-scale networks and select fewer controllers compared to an extended algorithm that solves the CPP in SDMWN based on a randomly selected controller placement approach.

Keywords

Software-defined Multihop Wireless Networking (SDMWN), Controller Placement Problem (CPP), Control Overhead, Heuristic Algorithms

1. Introduction

Applying the software-defined networking (SDN) architecture to multihop wireless networks (MWNs), which are self-configuring and self-organizing, can be beneficial to overcome some of the existing challenges, including distributed network management, mobility of devices, energy consumption, and quality of service [1]. Using a distributed control plane in software-defined multihop wireless networking (SDMWN), *i.e.*, the MWN is managed by multiple SDN controllers, is a potential solution to address the challenges of using a single controller in the network in terms of scalability, reliability, energy depletion, etc. [2] [3] [4]. However, SDMWN raises some new challenges such as determining the number of controllers, controller placements and assignments, which could have a considerable impact on network performance in terms of delay, reliability and control overhead. The authors in [5] refer to these challenges as the controller placement problem (CPP), which is an NP-hard problem. Several studies have been reported so far to consider various metrics and multiple objectives such as minimizing latency among controllers and network devices [5], load-balancing among controllers [6], minimizing the cost of the control plane deployment [7] [8] and improving reliability [9] [10], to solve the CPP in wired networks. However, only a few studies consider solving the CPP in wireless networks introduced as the wireless CPP in [11], *i.e.*, communications among controllers and network devices are wireless. Therefore, it is important to address the aforementioned challenges in SDMWN using a distributed control plane based on its unique characteristics and constraints, e.g., control overhead, multihop communications, capacities of links, which is further explained in the remainder of this section.

In SDMWN, in addition to the shared and unreliable communications and the capacity limit of links, in some techniques, the data and control traffic share the same channel. Therefore, in such networks, solving the CPP while minimizing the generated control overhead plays an important role in reducing energy consumption, packet losses, and delay that in turn has a significant impact on the reliability of the control plane. The control overhead in the network consists of controller-device communications to set up flow tables and exchange network configurations. Moreover, in a multi-controller environment, a number of control packets are required to be exchanged periodically among controllers (inter-controller communications) to synchronize and integrate different network views and obtain a global view of the network [12]. In this paper, we use the terms control overhead and the network cost interchangeably, as the network cost is directly related to the control overhead.

In [13], an optimization model was proposed to find the number of controllers and assign the controllers to network devices in SDMWN with stationary devices while minimizing the control overhead (*control packets*/*second*) in the network. Moreover, the capacity of wireless links was considered to solve the problem. The proposed model for the CPP was formulated as a nonlinear pro-

gramming (NLP) problem. The results demonstrate that the proposed optimization model is able to solve the CPP problem, and minimize the control overhead and satisfy the multiple defined constraints. In addition, the results in [13] show the effect of different parameters, including the number of controllers, the arrival rate of new traffic flows, and the capacity of links on the generated control overhead in the network.

In the proposed model in [13], it is assumed that all devices can be candidates to place a controller. Therefore, to find the optimal solution, all possibilities of selecting N placements from V devices are searched to find a set of controllers to minimize the network cost and satisfy the defined constraints. The optimal solution considers multiple factors, including the capacity of links and assigning each device to exactly one controller in the network. Moreover, the proposed algorithm finds the minimum number of controllers required to minimize the network cost by solving the optimization model for different numbers of controllers from placing only one controller in the entire network to place one controller on each device in the network. As a result, although solving the optimization problem finds the optimal solution in the network, due to the high computational complexity for various combinations, it takes a long time to solve the problem and it is only able to investigate a small network [13], 6 devices, which is considered an impractical solution for SDMWN.

On the other hand, various heuristic algorithms, as described more in Section 2, have been proposed to solve the CPP and find near-optimal solutions in a reasonable time while considering different objectives in larger networks [14]. Hence, the objective of this paper is to find near-optimal solutions for the proposed optimization model in [13] in large-scale SDMWN. To achieve this goal, we adapt and extend the existing proposed heuristic algorithms in [15] [16] [17] [18], which are proposed for wired networks. We tailor them to minimize the generated control overhead by considering more SDMWN-specific characteristics. The extended heuristic algorithms can then be used to solve the proposed optimization problem in [13] in SDMWN efficiently for larger networks, up to 500 network devices. To evaluate the performance of the three proposed heuristic algorithms compared to the solution based on the optimization model in [13], we simplify the proposed optimization model such that all communications use the shortest paths without the consideration of the capacity of links as used in [13]. By simplifying the problem as stated, the complexity of the CPP is reduced from a nonlinear problem to a linear problem, which allows us to conduct experiments for large networks.

We have performed a number of experiments using large networks. The results demonstrate that the proposed heuristic algorithms that find the controller placements based on a ranking degree assigned to the possible placements, *i.e.*, the ranking of connectivity degree or the average distance degree to other devices, are able to find optimal and near-optimal solutions, including controller placements and assignments while minimizing the network cost. We also eva-

luate the performance of those adapted heuristic algorithms and the optimal solution with multiple metrics, including the network cost, the average number of hops among devices and their assigned controllers, the average number of hops among controllers and the computational time. To make the model easier before considering the mobility in the network, the proposed heuristic algorithms in the literature and in this paper consider static networks such that investigating the effect of mobility on heuristics should be conducted in the future.

The rest of the paper is organized as follows. Section 2 describes the related work in heuristic algorithms to solve the CPP while minimizing the control overhead. Section 3 presents the proposed optimization problem and Section 4 shows the proposed extension to the existing heuristic algorithms to solve the proposed optimization model. Section 5 evaluates the proposed heuristic algorithms and compare the obtained results with the optimal solution achieved from solving the optimization model in networks with different topologies and a different numbers of devices. Finally, we conclude the paper in Section 6.

2. Related Work

In this section, we review some existing heuristic approaches proposed in the literature to solve the CPP, while minimizing control overhead in the network.

SDN was originally proposed to have a central controller. Having a centralized controller has some drawbacks, such as a single point of failure and scalability issue [2] [3] [4]. Applying the traditional concept of SDN to MWNs, more factors need to be considered, including direct connections of network devices with the central controller and constrained resources. In SDMWN, network devices often need to communicate with the controller in a multihop manner using an unreliable and shared wireless medium for network updates. Consequently, network devices may face higher latency, especially for devices that are farther away from the controller [19]. Using a distributed control plane in SDMWN, *i.e.*, the network is divided into multiple domains; each domain is managed by a controller, is a potential solution to address the challenges of using a single controller.

However, there are other challenges for SDMWN. Among them include determination of the number of controllers and their locations, and assignment of controllers to network devices. Those challenges in SDMWN are referred to as the CPP that is a NP-hard problem [5]. The main objective of the CPP is to find the optimal number of controllers and their placements, and to assign controllers to network devices [14]. Further, integrating local views of each domain into a global view for the entire network is another challenge, which requires inter-controller communications via a multihop manner over limited link capacities and perhaps noisy channels.

There are only few studies investigating the CPP in wireless networks. This problem was introduced as the wireless CPP in [11], in which communications among controllers and network devices are wireless. In this environment, the characteristics of unreliable and shared wireless medium should be considered

for the solution. The authors in [11] presented an approach to find the optimal number of controllers and controller placements and the minimum total delay including the network access delay for the devices, transmission delay, propagation delay and the queuing delay at the controller. The authors formulated the problem as a chance-constrained stochastic program (CCSP). The results demonstrate that the proposed approach can reduce the number of selected controllers and delay in the network.

Dvir *et al.* [20] formulated the CPP as a multi-objective optimization problem to minimize propagation delay and link failure probability among controllers and wireless access points. The authors also introduced two heuristic algorithms to find the number of controllers. A two-layer model for controllers was proposed in [21] for VANET. Compared to random placement of controllers, the results show that the proposed approach improves delay and packet delivery ratio. Qin *et al.* [22] also proposed a solution based on a randomized greedy algorithm for CPP in wireless edge networks to minimize delay and control overhead (Mbps) in the network. In [22], the authors do not consider the controller-device communication control overhead for topology discovery and the characteristics of wireless medium to solve the problem.

A few heuristic algorithms for the CPP have been reported in the literature to minimize the control overhead in the network. However, those existing algorithms do not consider the characteristics of wireless networks. In [15] [16] [17], to minimize the control overhead, controller placements are selected using a ranking algorithm. In [15] [16], based on the connectivity degrees of the possible controller placements to other devices in the network, in each iteration, a possible placement with the highest degree (more neighbors) is selected to be added to the set of controllers. In [15], these iterations continue until adding a new controller increases the network cost. Therefore, the proposed algorithm in [15] also finds a near-optimal number of controllers. On the other hand, in [16], the number of controllers is estimated by dividing the total degrees of all devices to the capacity of the controllers that defines the maximum allowed number of flow requests to be assigned to a controller in the network. In [17], the ranking algorithm is based on the average delay of the possible placements to all other devices in the network as a degree. Then, the possible placement with the smallest delay degree is selected to place a controller. However, the proposed ranking algorithm does not find the near-optimal number of controllers in the network.

In the proposed algorithm in [18], to find the number of controllers in a wired network, in the first iteration, all possible controller placements are selected. After that, in each iteration, one of the possible placements is selected and reduced from the set of controllers. The devices assigned to the reduced controller will be assigned to the closest controllers while satisfying the delay constraint among controllers and their assigned devices. Finally, the iteration stops if reducing a new controller increases the network cost. In [18], the authors do not provide any details about how to select a controller in each iteration to be reduced from

the set of controllers.

Summary: In [16], the heuristic algorithm does not find the number of controllers and the network cost considered is only the controller-device communications to set up flow rules. On the other hand, although in [15] the proposed heuristic algorithm finds the number of controllers, the authors do not consider the control overhead generated by controller-device communications to discover the network topology. The generated control overhead in [15] consists of the control overhead injected to the network by inter-controller communications to synchronize different network views and the controller-device communications to set up flow rules. In addition, the algorithm presented in [17] only finds the controller placement of a centralized controller in the network.

In general, the existing heuristic algorithms [15] [16] [17] [18] presented in the literature do not consider the characteristics of SDMWN, including wireless communications in the control plane in one or multihop manner. Therefore, investigating the heuristic algorithms to find near-optimal solutions to the CPP in SDMWN with the objective of minimizing the generated control overhead is the main goal of this paper.

3. Proposed CPP to SDMWN

In this section, we present the simplified model of [13] with an aim to convert the CPP from a nonlinear problem to a linear problem. In the proposed linear problem, all communications among devices and controllers as well as among controllers are changed to use the shortest paths. In other words, the capacity of links is not considered in solving the CPP.

3.1. Notations

We use the following notations in this paper for the proposed models. All notations except $Nhop_{m,n}$ are also presented in the optimization model proposed in [13].

- $Cost_{TD}$ shows the total cost of topology discovery in the network (*control packets*/*second*) calculated using Equation (2).
- R_{TD} is the rate of running topology discovery by each controller in the network (1/*second*).
- R_{FlowRq} is the arrival rate of new flows in each network device, *i.e.*, the devicesends a flow request message toward its assigned controller (1/*second*).
- N shows the number of controllers.
- $neighbor[i]$ presents a set of neighbors of device i in the network.
- $neighbor_{i,j}$ demonstrates the j^{th} neighbor of device i in the network.
- $Nhop_{m,n}$ is the number of hops in the shortest path between device m and device n.

3.2. Model Outputs

In the simplified optimization model proposed in this paper, since controllers

and devices communicate using the shortest paths without considering the capacity of links, the outputs are defined as follows in the objective function.

- y_k : The value equals one if and only if device k hosts a controller.
- $x_{k,i}$: The value equals one if and only if the controller placed on device k is assigned to device i.

Therefore, the outputs of the proposed model are the optimal placements of N controllers, controller assignments to network devices and the optimal cost of N controller placements in a SDMWN.

3.3. Objective Function

This section presents the objective function and the cost function that we adapt the one used for optimization from [13] by removing the consideration of the link capacity for the purpose of heuristic algorithms. In the objective function presented in Equation (1), we define $Nhop_{m,n}$ as the number of hops in the shortest path between device m and device n.

$$Min\left(Cost_{TD} + \sum_{k=1}^{|V|}\sum_{i=1,i\neq k}^{|V|}\left(R_{Flow_{Rq}} x_{k,i}\left[Nhop_{k,i} + Nhop_{i,k}\right]\right)\right) \tag{1}$$

subject to: (3), (4), (5), (6).

The first part of Equation (1) ($Cost_{TD}$), which is calculated using Equation (2) shows the total cost of topology discovery in the network. The second part of Equation (1) shows the total cost of controller-device communications to set up flow rules and exchange configurations using the shortest path communications.

$$\begin{aligned} Cost_{TD} = {} & R_{TD}\left[\sum_{k=1}^{|V|}\sum_{i=1,i\neq k}^{|V|}\left(Nhop_{k,i} + \sum_{m=1}^{|V|}\sum_{j=1,j\neq k,m}^{|neighbor[i]|} Nhop_{neighbor_{i,j},m} x_{m,neighbor_{i,j}}\right) x_{k,i}\right] \\ & + R_{TD}\left[\sum_{k=1}^{|V|}\sum_{p=1,p\neq k}^{|V|}\left(y_k y_p Nhop_{k,p}\right)\right] \end{aligned} \tag{2}$$

3.4. Constraints

The objective function presented in Equation (1) is subject to the following defined constraints. The constraint defined in Equation (3) avoids assigning a device to a controller that is not placed in the network.

$$x_{k,i} \leq y_k, \forall i,k \in V \tag{3}$$

The constraint defined in Equation (4) ensures that each device is assigned to exactly one controller.

$$\sum_{k=1}^{|V|} x_{k,i} = 1, \ \forall i \in V \tag{4}$$

Equation (5) ensures that there is a given number of controllers in the network.

$$\sum_{k=1}^{|V|} y_k = N \tag{5}$$

Equation (6) presents the integrality constraints.

$$x_{k,i}, y_k \in \{0,1\}, \forall i,k \in V, \forall (u,v) \in E \tag{6}$$

3.5. Algorithms

Table 1 demonstrates the algorithm of finding the optimal solution using the simplified optimization model. In this algorithm, all combinations of selecting N placements from V devices are searched to find a set of controllers to minimize the network cost and satisfy the defined constraints. Further, Table 2 shows the cost function that calculates the total network cost of the possible solutions using Equation (1) that consists of the cost of topology discovery and the cost of exchanging configurations and setting up flow tables for a possible solution.

4. Proposed Heuristic Approaches to Solve the Proposed CPP

As described in the introduction, we adapt the existing heuristic algorithms for the modified optimization problem. This section presents the three extended algorithms in details. The objective of those heuristic algorithms is to minimize the network cost and satisfy the defined constraints. This network cost consists of the cost of controller-device and inter-controller communications for network topology discovery, and the cost of controller-device communications to set up flow tables and exchange configurations.

Table 1. Algorithm 1: Optimization algorithm.

1:	***Input:*** $G = (V, E)$, N (number of controllers), Constraints (e.g., capacity of links)
2:	$MinCost \leftarrow \infty$
3:	**For** each solution k in the set of possible solutions **Do**
4:	Assign the set of controllers in k to network devices
5:	$Cost_k = CostFunc(G, k)$
6:	**If** ($Cost_k < MinCost$) and ($Constraint\ Check$) **Then**
7:	$MinCost = Cost_k$
8:	**End If**
9:	**End For**
10:	**Return** N Controller Placements, Controller Assignments, $MinCost$

Table 2. Algorithm 2: Cost function.

1:	**Function** $CostFunc(G, k)$
2:	Calculate $Cost_{TD}^k$
3:	Calculate $Cost_{Setup}^k$
4:	$Cost_k = Cost_{TD}^k + Cost_{Setup}^k$
5:	**Return** $Cost_k$

Table 3 shows the algorithm 3 adapted from the existing heuristic algorithms presented in [15] and [16]. Both of them select controller placements based on the connectivity degrees of devices. The connectivity degree of each device shows the number of neighbors connected to the device directly. However, the heuristic algorithm in [16] does not find the number of controllers and the network cost considered is only the controller device communications to set up flow rules. On the other hand, the authors in [15] do not consider the control overhead generated by controller-device communications to discover the network topology.

Similar to [15] and [16], the input of Algorithm 3 presented in Table 3 is the network graph. As a part of this algorithm, $P[]$ is calculated which contains a list of devices based on their connectivity degrees in descending order. This paper also proposes to select the possible placement with the highest connectivity degree (most number of neighbors) in each iteration and add the placement to a set of possible controller placements k. Moreover, in each iteration, the algorithm finds the nearest controller for each device. Unlike [16], Algorithm 3 finds the number of controllers, *i.e.*, the iteration continues until adding a new controller increases the network cost. The outputs of the Algorithm 3 include the number of controllers, the placements of the controllers, controller assignments to network devices and the minimum achieved cost.

The authors in [17] propose the average delay as a degree that is proportional to the distance among devices. The main idea of the approach is to calculate the average distance among the possible placements to all other devices in the network

Table 3. Algorithm 3: Heuristic algorithm adapted from [15] and [16].

1:	***Input***: $G=(V,E)$, N (number of controllers), Constraints (e.g., number of controllers, number of controllers assigned to each device)
2:	$P[] \leftarrow$ Descending Sorted (V, Connectivity Degree (V))
3:	$MinCost \leftarrow \infty$
4:	$I \leftarrow 0$
5:	**Do**
6:	Add placement $P[i]$ to the set of possible placements k
7:	Assign the set of the controllers in k to network devices
8:	$Cost_k = CostFunc(G,k)$
9:	**If** ($Cost_k > MinCost$) **Then**
10:	Remove $P[i]$ from the set of possible placements k
11:	**End If**
12:	i++
13:	**While** ($Cost_k < MinCost$)
14:	**Return** N Controller Placements, Controller Assignments, $MinCost$

as the ranking degree. However, the algorithm presented in [17] does not find the number of controllers and it only finds the controller placements in the network.

Hence, Algorithm 4 presented in **Table 4** is a modification of [17] such that the input of the algorithm is the network graph. As shown in the algorithm, $P[]$ is calculated which is the list of devices based on their average distance degrees to other devices in ascending order. To find the number of controllers, in each iteration, a placement with the lowest distance degree is selected ($P[i]$) and added to the set of the controllers (k). Then, each device is assigned to the nearest controller and the network cost is evaluated. If adding the new controller decreases the network cost, another iteration continues; otherwise, the new controller is removed from the set of controllers and the iteration stops. The outputs of Algorithm 4 are controller placements and assignments, and the minimum cost of SDMWN is returned as result.

Moreover, Algorithm 5 presented in **Table 5** extends the heuristic algorithm proposed in [18] such that in each iteration, one of the placements is removed randomly from the set of controllers. All devices currently assigned to the removed controller placement are assigned to the closest controller in the set of controllers. If removing the new controller reduces the network cost, the iteration continues; otherwise, the controller is not removed from the set of the controllers and the algorithm stops. Algorithm 5 shows the extension of the proposed algorithm in [18] by removing the possible placements randomly from the set of controllers in each iteration.

Table 4. Algorithm 4: Heuristic algorithm adapted from [17].

1:	***Input***: $G = (V, E)$, N (number ofcontrollers), Constraints (e.g., number of controllers, number of controllers assigned to each device)
2:	$P[] \leftarrow$ Ascending Sorted (V, Avg Distance Degree (V))
3:	$MinCost \leftarrow \infty$
4:	$I \leftarrow 0$
5:	**Do**
6:	Add placement $P[i]$ to the set of possible placements k
7:	Assign the set of the controllers in k to network devices
8:	$Cost_k = CostFunc(G, k)$
9:	**If** ($Cost_k > MinCost$) **Then**
10:	Remove $P[i]$ from the set of possible placements k
11:	**End If**
12:	i++
13:	**While** ($Cost_k < MinCost$)
14:	**Return** N Controller Placements, Controller Assignments, $MinCost$

Table 5. Algorithm 5: Heuristic algorithm adapted from [18].

1:	***Input:*** $G=(V,E)$, N (number ofcontrollers), Constraints (e.g., number of controllers, number of controllers assigned to each device)
2:	$MinCost \leftarrow \infty$
3:	Add placements in V to the set of possible controller placements k
4:	Assign the set of the controllers in k to network devices
5:	**Do**
6:	Select a placement i randomly from k
7:	Remove i from the set of possible placement k
8:	Assign the set of the controllers in k to network devices
9:	$Cost_k = CostFunc(G,k)$
10:	**If** ($Cost_k > MinCost$)
11:	Add i to the set of possible placements k
12:	**End If**
13:	**While** ($Cost_k < MinCost$)
14:	**Return** N Controller Placements, Controller Assignments, $MinCost$

The computational complexity of the presented algorithms, *i.e.*, Algorithms 1, 2 and 3, in this section is ($O(|V|)$) which is lower than solving the optimization problem that needs to search all possible combinations of selecting N placements from V devices, which is a NP-hard problem [5]. However, the reviewed algorithms [15] [16] [17] [18] and their extensions, *i.e.*, Algorithms 3, 4 and 5 presented in **Table 3**, **Table 4** and **Table 5**, respectively, does not consider the capacity of links as a constraint and communications in the control plane use the shortest paths. In Section 5, we investigate the impact of the capacity of links on the obtained number of controllers in the network.

5. Experiments, Results and Analysis

As mentioned earlier, the proposed optimization model in [13] is a non-linear problem and due to the high computational complexity of the optimization model, only a small network is investigated in [13]. Therefore, in this paper, we evaluate the performance of the three proposed heuristic algorithms presented in Section 4 compared to the solution based on the proposed optimization model presented in Section 3.

A number of experiments have been performed for evaluation. We use randomly generated topologies with different numbers of devices and run the proposed algorithms on each topology 10 times and calculate the average network cost. Moreover, we use AMPL (a mathematical programming language) [23] and

the CPLEX solver [24] to implement the proposed optimization model. Because of the high computational complexity of the proposed optimization model, we use a more powerful NEOS server [25] [26] [27] [28] to run the optimization problem in AMPL. The solver is running on a system with the following configurations: an Intel Xeon E5-2698 @ 2.3GHz, 192GB RAM and 300G SAS drives setup in RAID5. In addition, we use an Intel Core i7 CPU (1.8 GHz) and 8.0 GB RAM to run the proposed heuristic algorithms.

To evaluate the optimization problem and the proposed heuristic algorithms, we assume that $R_{FlowRq} = 0.5$ $(1/second)$, *i.e.*, each network device receives a new flow every 2 seconds and $R_{TD} = 0.2$ $(1/second)$, *i.e.*, each controller runs the topology discovery process every 5 seconds, which is adopted from Open Daylight [29].

Figure 1 and Figure 2 demonstrate two randomly generated SDMWNs with 10 wireless network devices. In these figures, the devices with the red circle are the optimal controller placements with respect to network cost. Moreover, Table 6 and Table 7 show the network cost obtained from running the aforementioned linear optimization model and the three heuristic algorithms presented in Section 4. As shown in these tables, Algorithm 3 and Algorithm 4 presented in Table 3 and Table 4, respectively, are able to find the optimal number of controllers, which is 5 controllers for Topology 1 and the optimal controller placements, which are devices 1, 2, 3, 4 and 9, while minimizing the network cost. In addition, Algorithm 3 and Algorithm 4 are able to find the optimal number of controllers and controller placements in Topology 2, which is 6 controllers placed on devices 1, 2, 3, 4, 7 and 10, while minimizing the network cost. On the other hand, as shown in Table 6 and Table 7, Algorithm 5 presented in Table 5 produces a higher network cost for both topologies.

Table 8 demonstrates the average minimum cost achieved from each of the three extended algorithms and the linear optimization problem in randomly generated topologies with different numbers of devices. As shown in this table,

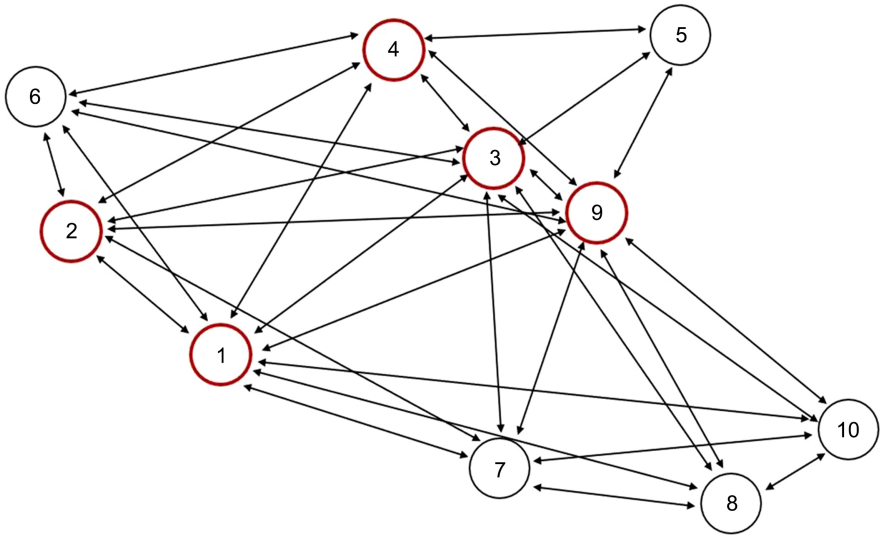

Figure 1. Topology 1—an SDMWN with 10 wireless network devices.

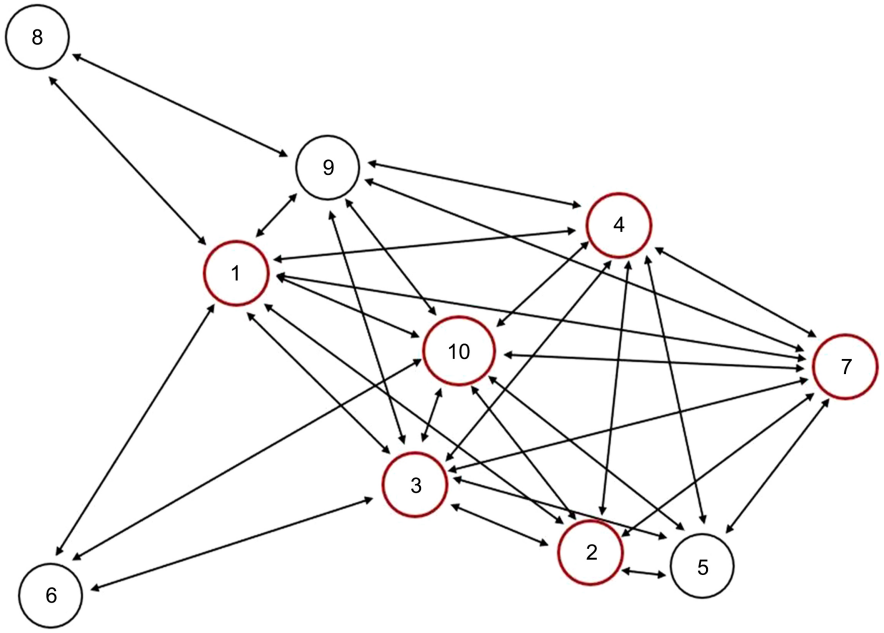

Figure 2. Topology 2—an SDMWN with 10 wireless network devices.

Table 6. Minimum cost obtained from the proposed algorithms and the optimal solution for Topology 1 shown in Figure 1 (*control packets/second*).

Optimal Solution	Algorithm 3	Algorithm 4	Algorithm 5
13.8	13.8	13.8	15.0

Table 7. Minimum cost obtained from the proposed algorithms and the optimal solution for Topology 2 shown in Figure 2 (*control packets/second*).

Optimal Solution	Algorithm 3	Algorithm 4	Algorithm 5
13.2	13.2	13.2	15.6

Table 8. Minimum cost obtained from the proposed algorithms and the proposed optimization problem (*control packets/second*).

Number of devices	Optimal Solution	Algorithm 3	Algorithm 4	Algorithm 5
10	13.8	13.8	13.8	15
20	42.6	42.6	42.6	54
30	79.2	79.6	80.4	88
40	126	149.8	131.4	143.8
50	186.4	199.4	194.8	223.8
60	219.8	274	252	286.6

for small-scale networks, Algorithm 3 based on the connectivity degree and Algorithm 4 based on the distance degree are able to find the same solution as the optimal solution obtained from solving the optimization problem for 10 or 20 nodes. The results generated from Algorithm 3 and Algorithm 4 are mostly close to that of the optimal solution, whereas Algorithm 5 generally has a higher cost. Moreover, Figure 3 shows the number of selected controllers in SDMWN in the optimal solution and the results of the proposed heuristic algorithms when increasing the number of devices in the network. As demonstrated in Table 8, when increasing the number of devices in the network, the network cost achieved from the optimal solution and the proposed heuristic algorithms are different that shows the impact of selecting the right number of controllers and controller placements on the network cost.

Table 9 shows the minimum cost achieved from the proposed heuristic algorithms in large-scale networks up to 500 network devices. Due to the high computational complexity of the optimal solution, the problem is solved for only 10 to 60 devices in the network. As depicted in Table 8 and Table 9, both Algorithm 3 and Algorithm 4 generate lower cost consistently compared to Algorithm 5. Algorithm 3 is based on the average connectivity degree. Intuitively, if a device has high connectivity degree, it has a higher chance to be selected as a controller. In this case, the selected device can directly communicate with more network devices without going over multiple hops, which is more effective in the network cost. Algorithm 4 is based on the average distance degree. A device with lower average distance degree is more likely to be selected as a controller. Similar to Algorithm 3, in Algorithm 4, the selected devices as controllers are likely to be able to directly communicate with more network devices due to the shorter distance; hence, multi-hop communications between a controller and network devices can be reduced as shown in Section 5.3. On the other hand, Algorithm 5 is based on random placement, which may not result in lower cost-effective placements.

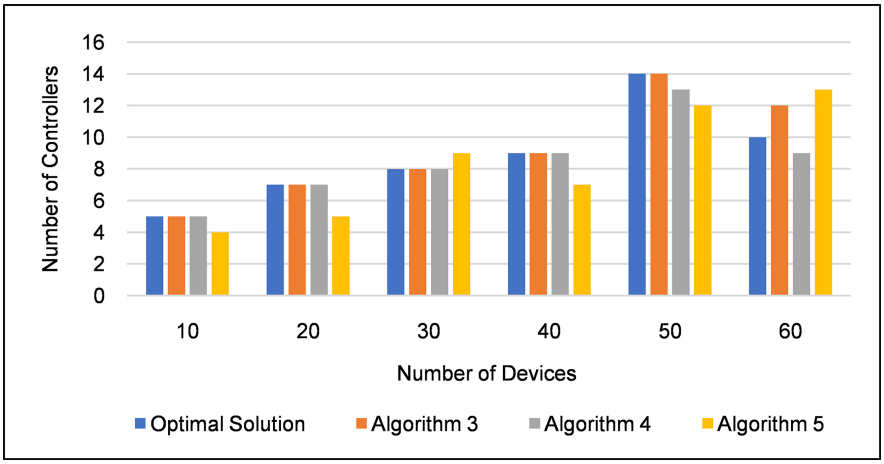

Figure 3. The number of devices versus the number of controllers selected in the presented algorithms and the optimization problem.

Table 9. Minimum cost obtained from the proposed heuristic algorithms in large-scale networks (control packets/second).

Number of devices	Algorithm 3	Algorithm 4	Algorithm 5
70	284.6	337	368.6
80	420	408.8	460.2
90	557	481.8	629.8
100	763.4	539	801.6
150	1091.4	1098.6	1604.7
200	1723.6	1930	2204.5
300	4035.6	4076.2	4303.4
400	6424.8	7111.4	7308.9
500	10,122.6	10,675	12,029.4

5.1. Capacity of Links

In the reviewed algorithms [15] [16] [17] [18] and their extensions, *i.e.*, Algorithms 3, 4 and 5, communications in the control plane use the shortest paths. However, the results from [13] show that using the proposed non-linear optimization models, the average control overhead flowing over links is almost around the network cost achieved by Algorithms 3, 4 and 5. For instance, in Topology 1 as shown in **Figure 1**, the average control overhead flowing over links is around 13.8 (*control packets/second*), which is identical tothe minimum cost obtained from running the optimization problem.

If we consider the capacity constraint of links and limit the capacity of all links to a certain value, using the network cost achieved from the proposed heuristic algorithms and the generated placement results, we are able to find the number of controllers that satisfies the capacity constraint. In this case, when placing a given number of controllers, if the heuristic algorithms or the proposed optimization problem result with higher cost than the limited capacity, the solution then is not able to satisfy the capacity constraint.

For example, when placing 5 controllers in Topology 1 as shown in **Figure 1**, the computed minimum cost is 13.8 (*control packets/second*). In this scenario, if we limit the capacity of all links to 12 (*control packets/second*) in the network, there is no feasible solution to place 5 controllers in this network while satisfying the capacity constraint.

5.2. Computational Time

In addition, **Table 10** demonstrates the computational time of solving the proposed linear optimization problem and finding the near-optimal solution using the proposed heuristic algorithms for 10 to 60 devices. As shown in **Table 10**, when the number of devices in the network increases, it takes a considerate

amount of time to solve the optimization problem and find the minimum number of controllers that aims to minimize the network cost, even if the solver is running on a faster and higher capacity machine, as described in the beginning of Section 5. On the other hand, Algorithms 3, 4 and 5 are able to find a near-optimal solution much faster than the optimization problem.

Table 11 depicts that when increasing the number of devices in the large-scale network (70 to 500 devices), the time saving for Algorithm 3 is significant compared to the other two algorithms. The main reason for this is that Algorithm 3 calculates the connectivity degree of the possible placements which can be completed faster than calculating the average distance degree values from the possible placements to other devices in the network used in Algorithm 4. Also, Algorithm 5 needs to evaluate a large number of possible solutions to stop the iterations. Table 11 shows the running time of the proposed heuristic algorithms when there are up to 500 devices in the network. This table demonstrates that, the algorithms are able to find near-optimal solutions in large-scale networks very efficiently.

Table 10. Execution time of the optimization model vs. running time of the proposed heuristic algorithms (*second*).

Number of devices	Optimal Solution	Algorithm 3	Algorithm 4	Algorithm 5
10	0.4484	0.0086	0.0120	0.0042
20	21.1262	0.0109	0.0163	0.0105
30	95.5204	0.0133	0.0220	0.0171
40	300.3674	0.0154	0.0272	0.0226
50	1062.1368	0.0163	0.0413	0.0348
60	1889.29	0.0172	0.0419	0.0382

Table 11. Execution time of the proposed heuristic algorithms in large-scale networks (*second*).

Number of devices	Algorithm 3	Algorithm 4	Algorithm 5
70	0.01754	0.04554	0.05268
80	0.01845	0.05377	0.07014
90	0.01954	0.06315	0.09617
100	0.02313	0.07222	0.11397
150	0.03531	0.18085	0.19192
200	0.06372	0.48571	0.39283
300	0.13912	0.94347	1.03180
400	0.17023	1.44209	1.64672
500	0.28923	3.39799	3.02576

5.3. Average Number of Hops

The objective of most studies in solving the CPP in both wired and wireless networks [14] is to minimize propagation delay between controllers and devices that is proportional to the distance among them. Minimizing the number of hops among controllers and devices as well as among controllers has an impact on the reliability of the control plane, especially in wireless networks with shared and unreliable communications. Moreover, Table 12 demonstrates the network cost, the average number of hops between devices and their assigned controllers (CD) and among controllers (CC) in SDMWN for the optimal solution and the proposed heuristic algorithms, respectively, in small-scale networks. As demonstrated in Table 12, although Algorithm 5 achieves the lower average number of hops among devices and their assigned controllers compared to other approaches, the number of selected controllers and the controller placements and assignments using Algorithm 5 result in higher average number of hops among controllers and network cost. In addition, the results show that in some cases, e.g., 50 network devices, although Algorithm 3 and Algorithm 4 achieve the lower average number of hops among controllers compared to the optimal solution, the controller placements and assignments in these two algorithms result in higher network cost and average number of hops among devices and their assigned controllers as demonstrated in Table 12.

Figure 4 shows the number of selected controllers in Algorithms 3, 4 and 5. In addition, as presented in Figure 4, in the randomly generated topologies with more than 150 devices, Algorithm 3 and 5 are able to achieve lower average number of hops between devices and their assigned controllers. Moreover, as shown in Table 9, when increasing the number of devices in the network, Algorithm 3 is able to achieve lower network cost compared to other two algorithms, while Algorithm 5 obtains higher network cost. Figure 5 and Figure 6 demonstrate the average number of hops between devices and their assigned controllers and the average number of hops among controllers, respectively when increasing

Table 12. Optimal solution vs. the proposed heuristic algorithms.

Number of devices	Optimal Solution			Algorithm 3			Algorithm 4			Algorithm 5		
	Cost	Avg. Hop Counts (CD)	Avg. Hop Counts (CC)	Cost	Avg. Hop Counts (CD)	Avg. Hop Counts (CC)	Cost	Avg. Hop Counts (CD)	Avg. Hop Counts (CC)	Cost	Avg. Hop Counts (CD)	Avg. Hop Counts (CC)
10	13.8	1.0	1.0	13.8	1.0	1.0	13.8	1.0	1.0	15.0	1.0	1.0
20	42.6	1.0	1.0	42.6	1.0	1.0	42.6	1.0	1.0	54.0	1.0	1.5
30	79.2	1.0	1.21	79.6	1.0	1.22	80.4	1.04	1.1	88.0	1.0	1.44
40	126.0	1.03	1.38	149.8	1.41	1.05	131.4	1.12	1.22	143.8	1.03	1.95
50	186.4	1.08	1.14	199.4	1.33	1.0	194.8	1.24	1.02	223.8	1.02	1.89
60	219.8	1.02	1.91	274.0	1.58	1.01	252.0	1.37	1.19	286.6	1.06	2.41

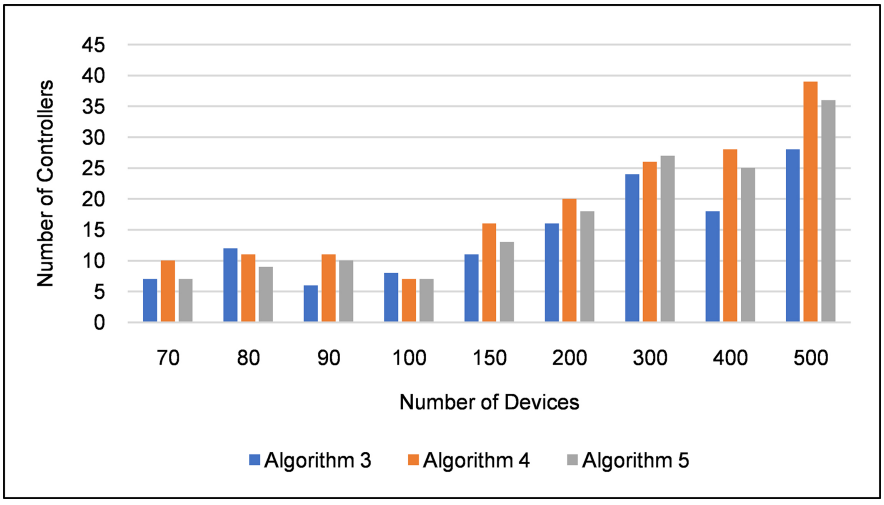

Figure 4. The number of devices versus the number of controllers selected in the presented algorithms in large-scale network.

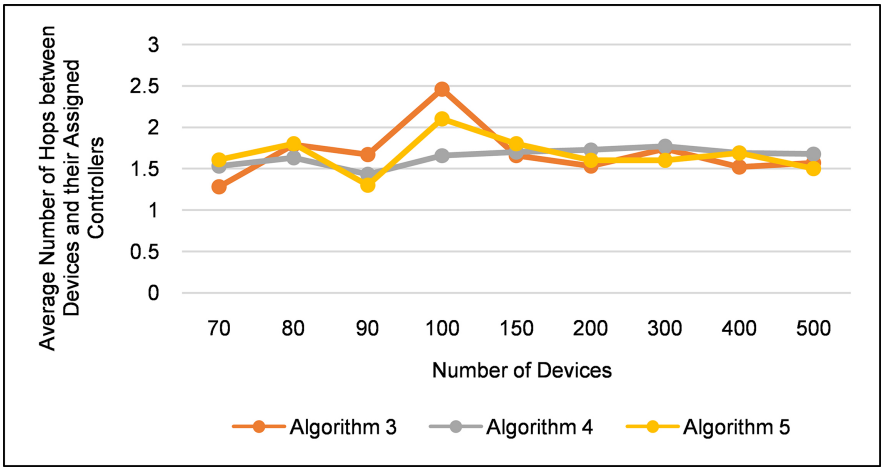

Figure 5. The number of devices versus the average number of hops between devices and their assigned controllers in the presented algorithms in large-scale network.

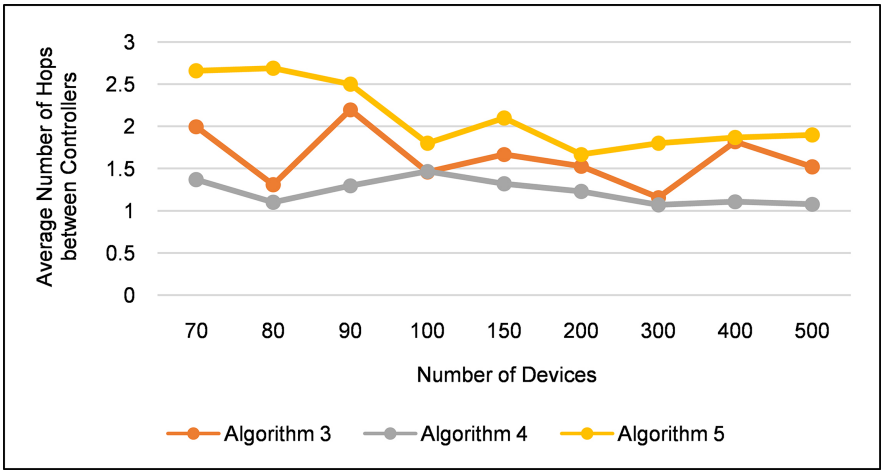

Figure 6. The number of devices versus the average number of hops between controllers in the presented algorithms in large-scale network.

the number of devices in the network. As shown in Figure 4, although in some cases, Algorithm 5 finds less number of controllers in SDMWN, the average number of hops among controllers is higher compared to Algorithms 3 and 4 as shown in Figure 6.

6. Conclusion and Future Research

The CPP is a potentially useful solution to SDMWN due to some issues in MWNs, such as reliability, scalability, and energy depletion. Control overhead is a crucial factor to consider for SDMWN. In this paper, we extended some of the existing heuristic algorithms proposed for the CPP in wired networks to SDMWN. The objective is to minimize the control overhead or the network cost for the proposed optimization problem in SDMWN. The extension of our algorithms considered the characteristics of SDMWN, e.g., wireless medium, multi-hop communications. We ran the algorithms in SDMWN with different numbers of devices, from 10 to 500.

The results obtained from the proposed three heuristic algorithms and the linear optimization model show that using a ranking algorithm based on the connectivity degree of the possible placements (Algorithm 3) or the degree of the average distance of the placements to other devices (Algorithm 4), we are able to find solutions that are identical to that of the optimization model in small-scale networks. Moreover, when increasing the number of devices, these two proposed heuristic algorithms are able to select a better number of controllers and controller placements, which results in lower network cost and the average number of hops among controllers compared to Algorithm 5 which selects the possible placements randomly.

Using the results obtained from the proposed heuristic algorithms, we were able to find the number of controllers that satisfy the capacity of links constraint defined in the proposed optimization model. Moreover, the results showed that the computational times for all three heuristic algorithms are significantly lower than that of the optimal solution, which makes them more practical. Among those three heuristic algorithms, Algorithm 3consistently requires less computational time which is based on calculating connective degree values of the possible placements.

Devices in SDMWN may be mobile. However, the algorithms presented in related works and in this paper are based on the model with stationary network devices. The motivation of this paper is to investigate the CPP and potential solutions, as the model is easier to understand before considering mobility. By doing so, we could also establish the baseline for further research to show the impact of solving the CPP in SDMWN with mobile devices on the network cost in the future.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Zahmatkesh, A. and Kunz, T. (2017) Software Defined Multihop Wireless Networks: Promises and Challenges. *Journal of Communications and Networks*, **19**, 546-554. https://doi.org/10.1109/JCN.2017.000094

[2] Oktian, Y.E., Lee, S., Lee, H. and Lam, J. (2017) Distributed SDN Controller System: A Survey on Design Choice. *Computer Networks*, **121**, 100-111. https://doi.org/10.1016/j.comnet.2017.04.038

[3] Hu, T., Guo, Z., Yi, P., Baker, T. and Lan, J. (2018) Multi-Controller Based Software-Defined Networking: A Survey. *IEEE Access*, **6**, 15980-15996. https://doi.org/10.1109/ACCESS.2018.2814738

[4] Zhang, Y., Cui, L., Wang, W. and Zhang, Y. (2018) A Survey on Software Defined Networking with Multiple Controllers. *Journal of Network and Computer Applications*, **103**, 101-118. https://doi.org/10.1016/j.jnca.2017.11.015

[5] Heller, B., Sherwood, R. and McKeown, N. (2012) The Controller Placement Problem. *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, Helsinki, 13 August 2012, 7-12. https://doi.org/10.1145/2342441.2342444

[6] Yao, G., Bi, J., Li, Y. and Guo, L. (2014) On the Capacitated Controller Placement Problem in Software Defined Networks. *IEEE Communications Letters*, **18**, 1339-1342. https://doi.org/10.1109/LCOMM.2014.2332341

[7] Sallahi, A. and St-Hilaire, M. (2015) Optimal Model for the Controller Placement Problem in Software Defined Networks. *IEEE Communications Letters*, **19**, 30-33. https://doi.org/10.1109/LCOMM.2014.2371014

[8] Sallahi, A. and St-Hilaire, M. (2017) Expansion Model for the Controller Placement Problem in Software Defined Networks. *IEEE Communications Letters*, **21**, 274-277. https://doi.org/10.1109/LCOMM.2016.2621746

[9] Hu, Y.N., Wang, W.D., Gong, X.Y., Que, X.R. and Cheng, S.D. (2012) On the Placement of Controllers in Software-Defined Networks. *Journal of China Universities of Posts and Telecommunications*, **19**, 92-97. https://doi.org/10.1016/S1005-8885(11)60438-X

[10] Lu, J., Zhang, Z., Hu, T., Yi, P. and Lan, J. (2019) A Survey of Controller Placement Problem in Software-Defined Networking. *IEEE Access*, **7**, 24290-24307. https://doi.org/10.1109/ACCESS.2019.2893283

[11] Abdel-Rahman, M.J., Mazied, E.A., Mackenzie, A., Midkiff, S., Rizk, M.R. and El-Nainay, M. (2017) On Stochastic Controller Placement in Software-Defined Wireless Networks. *Proceedings of IEEE Wireless Communications and Networking Conference*, *WCNC*, San Francisco, 19-22 March 2017, 1-6. https://doi.org/10.1109/WCNC.2017.7925942

[12] Zhang, T., Giaccone, P., Bianco, A. and De Domenico, S. (2017) The Role of the Inter-Controller Consensus in the Placement of Distributed SDN Controllers. *Computer Communications*, **113**, 1-13. https://doi.org/10.1016/j.comcom.2017.09.007

[13] Zahmatkesh, A., Kunz, T. and Lung, C.-H. (2020) Cost-Effective Controller Placement Problem for Software Defined Multihop Wireless Networks. *Proceedings of the Annual International Conference on Ad Hoc Networks* (*ADHOCNETS*), Virtual Conference, November 2020, 1-17.

[14] Das, T., Sridharan, V. and Gurusamy, M. (2020) A Survey on Controller Placement in SDN. *IEEE Communications Surveys Tutorials*, **22**, 472-503. https://doi.org/10.1109/COMST.2019.2935453

[15] Su, Z. and Hamdi, M. (2015) MDCP: Measurement-Aware Distributed Controller

Placement for Software Defined Networks. *Proceedings of the International Conference on Parallel and Distributed Systems*, Melbourne, 14-17 December 2015, 380-387.

[16] Yao, L., Hong, P., Zhang, W., Li, J. and Ni, D. (2015) Controller Placement and Flow Based Dynamic Management Problem towards SDN. *Proceedings of IEEE International Conference on Communication Workshop*, London, 8-12 June 2015, 363-368. https://doi.org/10.1109/ICCW.2015.7247206

[17] Ashraf, U. (2018) Placing Controllers in Software-Defined Wireless Mesh Networks. *Proceedings of International Conference on Computing, Mathematics and Engineering Technologies* (*iCoMET*), Sukkur, January 2018, 1-4. https://doi.org/10.1109/ICOMET.2018.8346386

[18] Obadia, M., Bouet, M., Rougier, J.L. and Iannone, L. (2015) A Greedy Approach for Minimizing SDN Control Overhead. *Proceedings of the* 2015 1*st IEEE Conference on Network Softwarization* (*NetSoft*), London, 13-17 April 2015, 1-5. https://doi.org/10.1109/NETSOFT.2015.7116135

[19] Ur Rahman, S., Kim, G., Cho, Y. and Khan, A. (2017) Deployment of an SDN-Based UAV Network: Controller Placement and Tradeoff between Control Overhead and Delay. *Proceedings of International Conference on Information and Communication Technology Convergence* (*ICTC*), Jeju Island, 18-20 October 2017, 1290-1292. https://doi.org/10.1109/ICTC.2017.8190924

[20] Dvir, A., Haddad, Y. and Zilberman, A. (2019) The Controller Placement Problem for Wireless SDN. *Wireless Networks*, **25**, 4963-4978. https://doi.org/10.1007/s11276-019-02077-5

[21] Kalupahana Liyanage, K.S., Ma, M. and Joo Chong, P.H. (2018) Controller Placement Optimization in Hierarchical Distributed Software Defined Vehicular Networks. *Computer Networks*, **135**, 226-239. https://doi.org/10.1016/j.comnet.2018.02.022

[22] Qin, Q., Poularakis, K., Iosidis, G., Kompella, S. and Tassiulas, L. (2018) SDN Controller Placement with Delay-Overhead Balancing in Wireless Edge Networks. *IEEE Transactions on Network and Service Management*, **15**, 1446-1459. https://doi.org/10.1109/TNSM.2018.2876064

[23] Fourer, R., Gay, D.M. and Kernighan, B.W. (2002) AMPL: A Modeling Language for Mathematical Programming. 2nd Edition, Duxbury Press, Scituate.

[24] CPLEX: Ibm's Linear Programming Solver. http://www.ilog.com/product/cplex

[25] Czyzyk, J., Mesnier, M.P. and Moré, J.J. (1998) The NEOS Server. *IEEE Journal on Computational Science and Engineering*, **5**, 68-75. https://doi.org/10.1109/99.714603

[26] Gropp, W. and More, J.J. (1997) Optimization Environments and the NEOS Server. In: Buhman, M.D. and Iserles, A., Eds., *Approximation Theory and Optimization*, Cambridge University Press, Cambridge, 167-182.

[27] Dolan, E.D. (2001) The NEOS Server 4.0 Administrative Quide. Technical Memorandum ANL/MCS-TM-250, Mathematics and Computer Science Division, Argonne National Laboratory, Lemont. https://doi.org/10.2172/822567

[28] The NEOS Server for CPLEX/AMPL. https://neos-server.org/neos/solvers/lp:CPLEX/AMPL.html

[29] OpenDaylight (ODL). https://www.opendaylight.org