International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Controller Placement Problem Optimization in Software Defined Networks Using Intelligent Methods

Sif K. Ebis[1], Behrouz Tousi[2]

Submitted: 06/05/2024 **Revised**: 17/06/2024 **Accepted**: 25/06/2024

Abstract: Software-Defined Networks are new paradigm of networks that separate control plane from data plane, hence transforming the network into a logically centralized network. In this type of networks, the control plane is assumed to have a global view of the network, therefore providing a more efficient management of the network. Control plane performs the management by means of "*Controllers*", which are the master-mind of the network. Controllers can operate along with a number of other controllers, while all being logically centralized. Choosing proper number of controllers and the best locations for them, are among the important challenges of SDN which influence network "*performance"*. An additional important challenge of SDN networks is scalability. In this paper we introduce a linear optimization model with the following goals: increasing scalability, reducing delay, reducing cost, reducing the number of control messages, and providing load balancing in the network. We solved the model and analyzed the problem. The proposed model assumes the network links to be in-band in order to provide scalability and to reduce costs. It means not all of the switches will directly connect to the controllers; instead, some of the switch-to-controller connections will be indirect – through other switches. The assumption of intermediate switches, and also the severally number of objectives in optimization will furtherly complicate the model, causing the model to become non-linear. In this paper, we propose a linear model that has a global unique solution in small scale. Then we suggest two algorithms for solving this model in shorter time: Using Genetic algorithm and Tabu Search. Next, we compare cost and run time for the two suggested algorithms, and also compare the results of the algorithms with each other.

Keywords: Controller - Software Defined Networks – Problem Optimization

1- Introduction

With the increasing number of computer network users, especially the Internet, the need for the development of these networks is growing every day. Computer networks are formed by a large number of devices such as switches, routers, and numerous middle box, which work with the help of a large number of complex protocols. These devices, which are vertically aggregated black boxes, create many management challenges. Network administrators are responsible for configuring policies on these devices to manage significant changes occurring within the network. They must translate high-level language (policy language) into low-level language that is understandable to the devices, often with limited tools available for this task. Optimal network management and increased efficiency represent one of the major challenges of the current network. These challenges exist at every scale of the network.

Another problem of current networks is the rapid growth of Internet users. The rapid growth of various applications and services complicates the Internet every day, creating new problems and challenges.

Another problem is the emergence of a conceptual term called "Cloud ". The use of cloud and its commercialization, along with the management of it through machines and virtual servers, has created numerous managerial issues. The migration of virtual machines and the discussion of Virtual Lan (VLAN) traffic are some of these problems [2].

The idea of "Programmable Networks" was introduced as a way to facilitate network assessment and management. Software-Defined Networking is a new definition of a network in which the hardware responsible for packet transmission in the network is separated from the control decisions, which are software-based."

The main idea is to allow software developers to rely on network resources similar to computational and storage resources. In SDN, network intelligence is logically centralized in software-based controllers (control layer), and all network devices will be transformed into simple packet forwarder devices programmable through open interfaces like OpenFlow or FORCES [1].

Decisions that were previously made in a distributed manner with the help of routers will now be centrally (logically) managed by the network's central controller.

[1,2] *Department of electrical and computer engineering, Urmia University, Urmia, Iran*
Saifkather@gmail.com b.tousi@urmia.ac.ir
Corresponding author: ***Behrouz Tousi,*** *b.tousi@urmia.ir*

This central decision-making poses new challenges to the network, including scalability.

Certainly, a single controller will not suffice for all networks, and the appropriate number of controllers for different networks will vary depending on geographical scope, network traffic, and other factors. Determining the suitable number of controllers and their placement is an important issue, considering that network transmission delay must be minimized.

1-2- The necessity of conducting research

In recent years, the global Internet network has experienced significant growth and is in the process of evolving into a network of interconnected things. The current complex IP-based network is inadequate to accommodate this growth, and its management is quite challenging. One of the issues with the current network is the inability to configure the network with predetermined policies or to reconfigure the network based on occurring errors, network load, or topology changes in mobile networks. In the current network, the network control and data forwarding functions have been intertwined, making network management more difficult. Software-defined networks have separated these two functions and simplified network management using a central controller. These networks address the limitations of current networks by vertically integrating the control and data forwarding plane, creating a network with separate control and data forwarding components. As a result, network switches and routers will be transformed into simple data forwarding devices, and network management and control will be handled by a centralized logical network controller that can also operate in a distributed manner.

1-3- Definition of Software-Defined Networks

Software-Defined Networks are defined by the following four characteristics. You can see the layers of software-defined networks and the structure of these networks in Figs. 1-1 and 1-2.

1) The network control plane is separated from the data forwarding plane, and the control functions of devices are removed and transformed into data forwarding devices.

2) Data forwarding decisions will be based on flow rather than the current destination-based approach. A flow is defined as a set of packet field values that act as a filter and a set of operations.

3) Control is transferred to an external entity called the SDN controller or network operating system (NOS). NOS is a software platform that runs on a server.

4) The network is programmable with the help of software applications running on NOS [4].

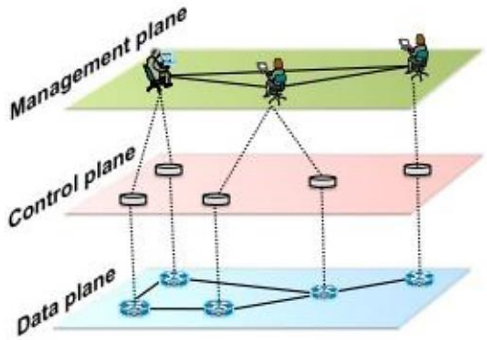

Fig. 1-1: Layers of Software-Defined Networks" in academic English.

Given this definition, it can be expected that management issues, network optimization, robustness, security, and data center problems that were not fully soluble in the current network, could be improved or resolved with the help of SDN.

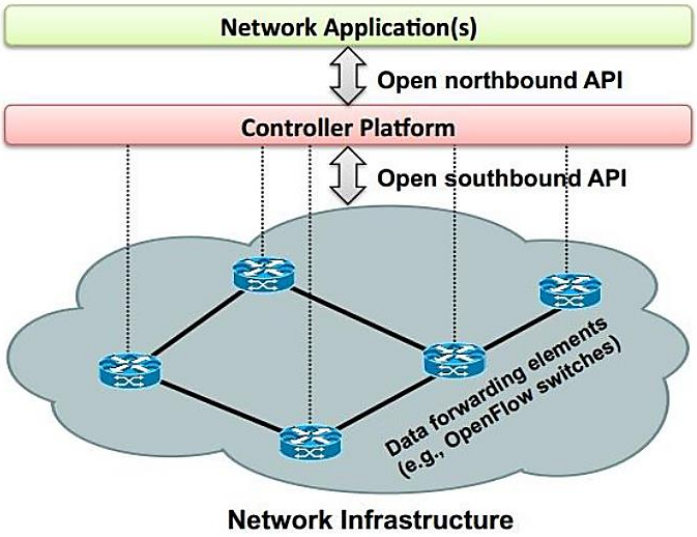

Fig. 1-2: Network Structure [4].

Given these concepts, an SDN is defined by three main abstractions: 1) forwarding 2) distribution 3) orchestration. Abstractions are the primary tools of research in computer science and information technology, which are currently among the key features of many computer and system architectures. The deep coupling between the data and control layers in traditional networks has made it difficult to add new features and programmability to these networks.

The integration of new features often requires the purchase of expensive equipment with complex configurations and settings, such as load balancers for servers, intrusion detection systems, and firewalls. These intermediary devices, which are essential in a network, make network configuration and modification more challenging. For example, an intrusion detection system may require a copy of all network traffic from all network switch devices in a specific logical or physical path.

Creating new features in SDN with the help of a new software program that runs on the control layer or the mastermind of the Network Operating System (NOS) network is much easier than traditional networks. This approach has several advantages:

- With the help of abstraction provided by Platform or a high-level programming language that can be shared, the planning of controller modules becomes easier.

- All programs can benefit from network information in a similar manner (network-wide view) and be guided towards Consistent coordination and more effective policy decisions of the network, while also reusing software modules of the control layer. These programs can exhibit reactive behavior from any part of the network, for example, reconfiguring transmitter devices. In fact, there is no need to explicitly define a strategy regarding the location of new features.

- It is easier to integrate different programs. For example, load balancing and routing programs can be sequentially combined, meaning that load balancing decisions can have higher priority than routing policies and be executed prior to them. Fig. 1-3 provides a comparative illustration between software-defined network structures and current networks.

1-4- Components of Software-Defined Networks

Each SDN consists of various components, which will be discussed further below.

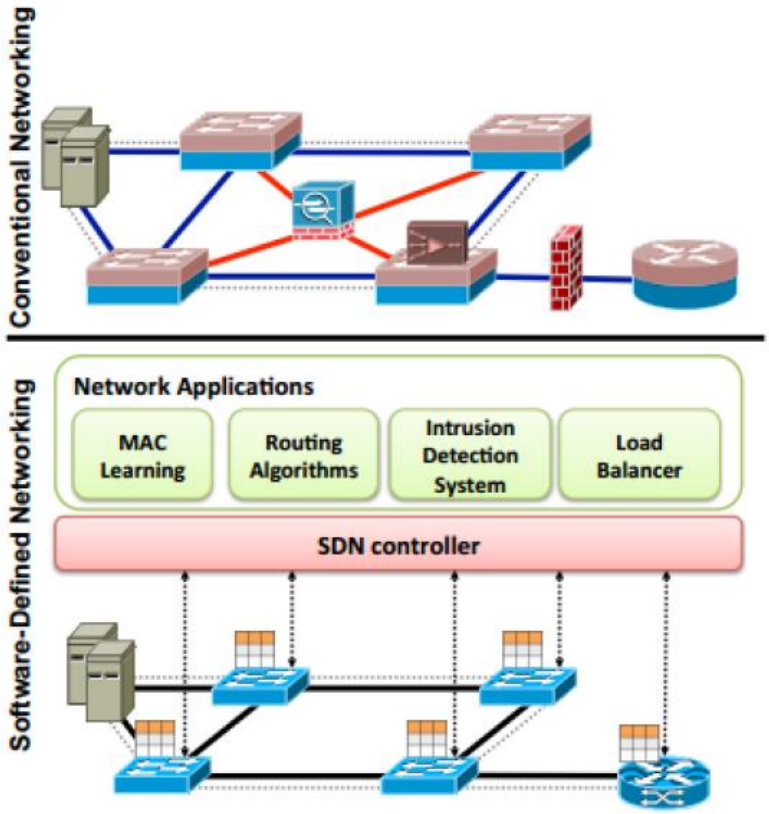

Fig. 1-3: A Comparison between the Structure of Software-Defined Networks and Conventional Networks [4].

2- Problem statement

Selecting suitable controller placements is not a new challenge in SDN networks. Given the scale of modern networks, it is a natural concern [7]. Previous works have utilized different approaches to formulate or readdress this issue [9] and [10]. Some of these studies are based on mathematical models, while others are based on graphical models [9].

2-2- Past Works:

Numerous works have been conducted on this subject from 2012 to the present. The main article in this field is the Sherwood article [3], which raises two primary questions: "How many controllers are required?" and "Where should the controllers be located in the topology?" They state that the answers to these two questions have an impact on important aspects of SDN networks, including fault tolerance and network performance metrics. The answers to these two questions also affect availability and convergence time in geographically large-scale networks. They did not address the problem of finding a location for minimizing delay, but rather delved deep into examining this problem by testing different metrics on network efficiency based on the number and location of controllers. They defined two metrics: Average-case latency and Worst-case latency. They placed one to five controllers in different locations and measured various cases using brute force. An important result obtained in this paper was that, beyond a certain stage, adding more controllers from 1 to k does not reduce the latency by a factor of 1/k. Increasing the number of controllers does not necessarily decrease the latency, but it becomes close to 1/k.

The metrics they introduced in this paper, L_{avg} (Average-case latency) and L_{wc} (Worst-case latency), have been widely used in other papers, which will be further discussed. They used the OS3E Internet2 project topologies for their evaluations [11]. They also mentioned that a trade-off needs to be made between these two metrics since both cannot be improved simultaneously. Additionally, for further analysis, they examined the Zoo Internet topologies, which are a collection of interconnected graphs gathered from public network traces [12]. They showed that the latency reduction from random placement differs significantly from optimal placement.

Schmid and his colleagues examined the distributed SDN control plane. In their article, they discussed the methods of using distributed local algorithms to solve this problem and investigated each of them [13]. In this paper, they provide various reasons for using a distributed control plane: 1) network management, 2) geographical issues, 3) scalability, 4) reducing latency from a switch to the nearest controller, 5) fault tolerance, 6) load balancing, 7) efficiency, and 8) it may also be suitable to use multiple controllers for task offloading

within a domain. Two models of the control plane are presented and discussed in this article:

1- Vertical structure, an example of which is Kandoo [14], consists of a root controller and controllers connected to it.

2- Flat structure, in which the network is divided into domains of different controllers, each of which contains a set of disconnected switches. This structure has advantages. For example, whenever economic actors manage various parts of the network independently, or when we want to reduce the delay between controllers and switches and perform traffic control locally. After appropriately concluding the superiority of the second structure for SDN, locality, they examined localness and distributed algorithms.

In terms of demonstrating the feasibility of using SDN in large networks and employing multiple controllers, several articles have been presented.

Google has implemented its own data centers with SDN and has briefly explained the implementation and interconnection of different data centers and their architectures in article [15]. The authors mention in this article that they have used multiple controllers for this purpose, but they have not discussed the details of their connection, placement, and quantity. The only important point is that the path between the controllers and switches is dedicated and out-of-band, which does not impose significant costs on network administrators in data centers as they can easily add additional links.

Aduit Dixit has proposed a distributed architecture of flexible controllers. In this method, a pool of controllers is created and the controllers are added or removed as needed. Minimum and maximum CPU load values are also considered for load balancing [16].

Soheil Hassas Yeganeh and colleagues presented a framework for outsourcing SDN control programs in their paper "Kandoo," where they utilized distributed controllers. However, they did not discuss the location and quantity of these controllers [14].

Teemu Koponen and colleagues also implemented the Onix control platform for large-scale networks in a distributed manner in their paper "Onix" [17]. They also did not address the number and location of controllers. Both this paper and the previous one emphasize the importance of a distributed control layer and the use of multiple controllers.

In article [18], a distributed controller framework and a collaborative framework called Volken Yarici are presented, which ensure scalability and reliability. In this framework, clustering is used to handle this value. The framework supports dynamic addition and removal of controllers and works seamlessly with current OpenFlow-based controllers without any modifications.

Benson et al. in their paper [19] state that data center traffic volume is approximately 150 million flows per second. Modern controllers based on OpenFlow can manage 6 million flows per second. They enumerate advantages of network clustering, including:

1) Scalability: By easily adding or removing servers to/from a cluster, it can handle increased or decreased load.

2) Reliability: Implementing a controller on a single hardware increases the likelihood of a single point of failure. Therefore, clustering is preferable, with the implementation spread across multiple servers. They implemented the servers in each cluster as Master and Slave and provided an algorithm for master selection [19].

Yury has attempted, in his article, to form the network tree in such a way with algorithms that increase resilience and quick return to normal mode after network errors occur. He also mentions in his articles that the communication delay between controllers is an important factor. Yury and his colleagues have presented a criterion for evaluating the robustness of the SDN control layer and have provided an algorithm for its calculation. They have also presented the K-Critical algorithm in another article for selecting suitable controllers and forming the network tree in a way that maintains resilience [8] and [20].

In the field of examining various criteria such as scalability and reversibility, some works have been done which are briefly described. Soheil Hassas Yeganeh and his colleagues have addressed the issues of scalability in an SDN network in their study [21]. They have stated that the controller may become a bottleneck, and depending on the network type, overhead may occur during the signaling phase. Other potential problems that may arise in this area include flow setup overhead and reversibility after error. They have also mentioned that many of these issues exist in regular networks as well. Important findings they have stated in this paper include:

- Concerns about scalability are not limited to SDN networks and also exist in virtual networks.
- Most of these problems can be solved without eliminating the advantages of SDN.

Jie Hu and colleagues have proposed a criterion for evaluating scalability and after examining different control layer architectures, they have stated that the most scalable state is the vertical architecture. In this type of article, they compare four types of data layer architectures: centralized, distributed with local view,

distributed with global view, and hierarchical. They have concluded that when the number of network hosts is much larger than the controllers, distributed architectures with local view and hierarchical structures have the highest scalability (Figs. 2-1, 3-1, and 4-1) [11].

In a study by Smiriti [22], she has also concluded by reviewing the work done in this regard that vertical distributed architecture has the highest scalability. They examined centralized, vertically distributed, and horizontally distributed architectures and found that vertically distributed and locally distributed architectures have the highest scalability.

Tootoonchian in the article [23] has examined the performance of controllers in SDN networks. The important results obtained in this article include:

- An average cluster of 1500 servers has an input rate of 100,000 flows per second.
- A network with 100 switches can generate up to 10 million flows per second in the worst-case scenario.
- A delay of 10 ms for setting up a controller adds a 10% delay to most flows.
- The NOX controller can manage 30,000 flow creation requests per second with a 10 ms delay [23].

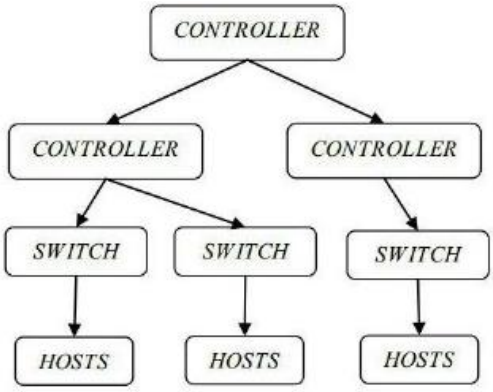

Fig. 2-1: The vertical distributed control layer.

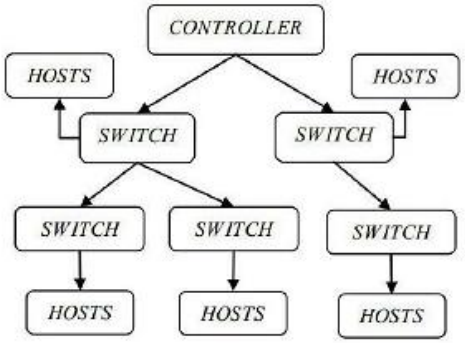

Fig. 2-2: A vertically distributed control layer at larger scales [22].

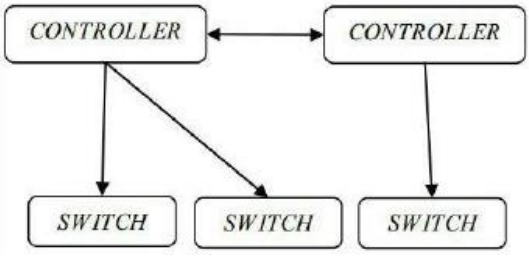

Fig. 2-3: The distributed horizontal control layer [22].

In their study [9], Xiao et al. addressed this issue in Wide Area Network (WAN) networks. They proposed a method for segmenting WAN networks into SDN domains, where each domain has its own dedicated controller. They assumed that the communication between controllers has been fully resolved. Their criterion was the reduction of propagation delay [2] and the improvement of domain efficiency by minimizing the number of nodes. They used the min-max cut function for segmentation and solved the controller placement problem in each domain by employing an algorithm based on the distance between switches to reduce propagation delay.

Yunnan and colleagues in their article [24] conducted the localization of controllers based on a criterion they provided for evaluating network reliability. They expressed the percentage of expected loss of control path as a measure of network reliability. Furthermore, they formulated a numerical optimization model to minimize this measure and addressed its solution. The formulated model is NP-hard, thus they proposed an algorithm to solve it.

In article [16], Gue introduces a new criterion for evaluating network reversibility based on cascade failures analysis and presents an algorithm to enhance network reversibility.

Zhang and his colleagues in article [25] also delve into the investigation of reversibility in these networks. They state that connecting all switches to the controller is favorable in terms of reversibility, but it is not cost-effective, and it is not scalable when the number of switches is large.

They cluster the network in such a way that nodes within a cluster are similar while nodes outside the cluster are different, and they select a central switch as the controller's location in each cluster.

Yan-nan tried to increase the reliability capability in their paper [7]. They presented a numerical programming model based on the reliability metric described in reference 13 of the same paper. They considered the number of controllers as input and only specified their locations. Then, they provided greedy algorithms to solve the model in less time and compared the algorithm results with the model results. The reliability metric they considered was the percentage of valid control path expectation.

Hock et al. attempted to consider multiple aspects of a network in their paper [26]. They stated that when various metrics of network efficiency and reversibility are considered together, there is usually no specific optimal location solution. However, trade-offs among the metrics can be made. In this regard, they used Pareto-optimal. Pareto-optimal is a type of game in which a player tries to improve their own situation without causing others to worsen or make changes. They considered delay, reversibility, and failure probability (error) metrics and presented models in these aspects. They dealt with solving the models with brute-force and analyzed the obtained results.

The authors in article [27] enhanced the previous proposed model by incorporating a more comprehensive load balancing mechanism and provided a heuristic algorithm aligned with the stated models. They compared the response time of the heuristic algorithm with the solved model.

Zang et al. aimed to minimize the cost of switch-to-controller assignment in their article [28]. They attempted to find the number of controllers needed. They presented a non-linear programming model MINLP for this purpose and later linearized the model MILP using specific techniques. They also provided heuristic algorithms to solve this model and ultimately compared the results obtained from the model in CPLEX and the heuristic algorithms. The overall objective of the article is to reduce the connection delays between switches and controllers and improve their communication.

In article [29], Obadia et al. presented a linear programming model 3 to reduce the overhead caused by the distributed nature of the control layer. They found the optimal number of controllers based on this criterion. They further introduced heuristic algorithms to solve the model in a shorter time and compared the results of these algorithms with the linear model.

Bari and colleagues presented a mathematical programming model, in article [30], for dynamically selecting the optimal number and location of controllers in order to minimize the time of flow establishment and communication overhead. The proposed model is non-linear, but they linearized it using certain techniques. Furthermore, they provided two heuristic algorithms to solve this model and ultimately compared the results obtained from the model and the two proposed algorithms.

In article [31], Rivera and colleagues introduced a planning model for reducing energy consumption in SDN networks. Their model focuses on optimally allocating controllers to switches, selecting the optimal number of controllers, and turning off unnecessary links. Finding the solution for the proposed model in large-scale problems is infeasible. Therefore, they presented a heuristic algorithm to solve the model and compared the results of the model and the heuristic approach. They also introduced the percentage of turned-off links as an improved metric.

Al Yusuf et al. utilized complex network analysis to evaluate their work. They employed two measures: 1) Betweenness centrality to calculate the number of shortest paths passing through each node, and 2) Closeness centrality to assess and manage propagation delays as their foundation. They endeavored to find suitable locations for controllers. Ultimately, they assessed their work using the Global Environment for network innovation (GENI) tool and achieved results consistent with the findings of Heller and Sherwood's paper.

Sallahi and colleagues presented a linear programming model in their paper [33] to minimize the cost of installing controllers and connecting them to switches.

They proposed a network design model and, similar to previous studies, considered out-band connections. The resulting model was a linear model that they solved using CPLEX software. The costs taken into account are the financial costs of the connections.

Rath and colleagues attempted to solve this problem using game theory in their paper [34] by defining the problem as a non-zero-sum game. Their output determines whether a controller should currently be added or removed or not. Their final framework contributes to maintaining QoS (Quality of Service) in the network. After formulating the problem with the goal of minimizing the cost of connecting the new controller, they provided three algorithms for solving the problem, one of which is based on a non-zero-sum game. Finally, they used MATLAB software for implementation and comparison of the algorithms, and conducted simulations on a scenario.

In the paper [35], Lange et al. proposed a Pareto-based (game-theoretical) planning model with the objective of reducing delays and balancing the load among controllers. They also presented two algorithms to solve it within an acceptable time frame and compared the results obtained from the model and algorithms.

In the paper [36], Muller et al. introduced a linear programming planning model to increase the connectivity between controllers and switches, manage requests, and provide failover mechanisms. They further proposed a heuristic algorithm to solve the model in less time and examined the results of the model and algorithm. They introduced a new criterion for connectivity and examined the failure mechanism with different failure probabilities.

In their paper [37], Schiff et al. presented an in-band network control layer based on a network overlay tree to enhance connectivity between controllers and switches, cover controllers, identify other switches, and cover switches recently connected to the network [37].

3- Proposed method

This passage describes a technical process. It repeats 100 times, and the best solution at each stage is selected as the optimal answer. Since the termination condition is based on the number of repetitions, it is possible that the optimal solution may not be found. However, in practice and based on the obtained results, acceptable results have been achieved in most cases. In the following, each stage will be described separately.

3-3-1- First phase - Initial Population Generation

First, based on the adjacency matrix of the examined network graph, a random spanning tree of the network is generated. Then, control nodes are placed in appropriate locations based on the maximum acceptable network delay. The type of control nodes is randomly selected.

The output matrix will be a gene. In such a way that the adjacency matrix of this tree will be modified in a way that the element on the main diagonal of each node (switch) represents the control node number to which this node is connected, and if this node itself is a control node, this number will be the sum of the control node number and the total number of switches in the graph. For example, if there are 3 nodes and the third node is a control node and control node type 2 is assigned to it, the corresponding gene will be as follows:

3-3-2- Second Phase - Fitness Function

The generated genes from the previous phase are examined by the fitness function. If they satisfy the model constraints, the final cost is calculated for each gene according to the defined cost in the model, and it is used as the fitness measure.

3-3-3- Third Phase - Combination

In each phase, a number of the best genes directly proceed to the next phase, and a number of genes are created based on combination and mutation operators. Combination operates on the structure of trees, meaning that the adjacency matrices of two trees are randomly combined. Then, it is ensured that the created tree is a network spanning tree.

3-3-4- Fourth Phase - Mutation

Mutation operates on a single gene, unlike combination which operates on two genes. The resulting matrices from the previous phase (combination) undergo mutations in order to decrease the cost of the selected controller type. The mutation operator is only defined for the controller types.

3-3-5- Fifth Phase - Evaluation of New Generation and Selection of the Best Solution

In this phase, the fitness of the new generation is calculated, and the best solution of this phase along with the average of the solutions is maintained. After repeating the algorithm for a predetermined number of times, the best solution from all phases is selected as the optimal solution. The pseudocode of this algorithm is summarized in Fig. 3-8.

The input of the algorithm includes the specifications of controllers, the delay cost matrix between each pair of switches using the shortest path, the matrix of total requests on these found paths, and the maximum acceptable total delay in the network. The output of the algorithm is the minimum network span tree, where the locations of the controllers along with their selected types are also determined.

After inputting the data (line 3), the initial population is created according to the explanations in section 1-4-3, and the fitness of each gene is determined (line 4). Then, until the number of iterations is completed, a new generation is created using the combination and mutation operators (line 6), the best and average solutions of each iteration are determined and stored (line 7). Finally, after completing the iterations, the best solution is selected from the obtained set of solutions as the final solution (line 9).

3-4-The Tabu search algorithm for controller localization

Due to the simultaneous need to determine both the number of controllers and the optimal placement of these controllers in this problem, generating initial solutions can be somewhat challenging. If both of these factors are considered as variables simultaneously, the number of possible initial solutions and the search space will be relatively large. To constrain the search space while finding the optimal solution, the controller locations are treated as the second variable under consideration, while the number of controllers is treated as the first variable.

The algorithm begins by estimating the required number of controllers based on a delay criterion, and starts examining candidate locations with the minimum number. Each time a location is examined and does not satisfy the constraints, it is added to the Tabu list. After examining the candidate locations and not satisfying the constraints, one controller is added to the number of controllers and the Tabu list is cleared. This process continues until an optimal solution is found. The pseudocode for the controller placement algorithm using the Tabu search algorithm is described in Fig. 3-10.

Algorithm 1: Controller placement Genetic Algorithm

1. input : (1) Controllers properties, (2) matrix of Shortest path delay between nodes and (3) matrix of requests which should traverse in shortest path between nodes, (4) Maximum acceptable delay in network
2. output : Matrix of final tree of the network, The place of controllers and the type of selected controllers
3. Insert Data
4. Create Initial Population
5. **For** i = 1 to maxiter **do**
6. Next Generation
7. Find Mean and Best Answer
8. **End**
9. Find Best (Bests)

Fig. 3-8: The Genetic Algorithm for solving the controller placement problem.

The algorithm takes as input the specifications of the controllers, the delay cost matrix between each pair of switches using the shortest path found, the accumulated requests matrix for these discovered paths, and the maximum acceptable total delay in the network. The output of the algorithm is the minimum network tree, where the locations of the controllers and their selected types are also specified.

After the arrival of the data, the number of controllers is determined in the first stage based on the maximum acceptable network delay between nodes (line 3). In each stage, an initial population of solutions is created for a predetermined number of iterations (line 5). In each initial solution, candidate controllers are randomly placed and other switches are randomly connected to these controllers. The type of controller is greedily selected based on the minimum possible installation cost in that location. The created solutions are evaluated based on the constraints stated in the optimization model, and the final cost is calculated for each. If an acceptable solution is found among the solutions in this iteration, the minimum cost discovered will be selected as the result (lines 6 and 7). Otherwise, the number of candidate controllers will be increased by one, and the algorithm will be repeated from line 4.

Algorithm 2: Controller Placement Tabu Search

```
1. input: (1) Controllers properties, (2) matrix of Shortest path delay between nodes and
(3) matrix of requests which should traverse in shortest path between nodes, (4) Maximum
acceptable delay in network
2. output: Matrix of final tree of the network, The place of controllers and the type of selected controllers
3. Find Minimum Number of Controllers
4. While True do
5.     Create Initial Solution With Minimum Number of Controllers
6.     If any Solution Found do
7.         Find all Minimum Cost of Results
       break;
8.     Else
9.         Minimum Number of Controllers++
10.    End
11. End
```

Fig. 3-9: Tabu Search Algorithm for Controller Problem Solution

In the next section, the results obtained from solving the model will be compared with the results obtained from the algorithms. Additionally, the results of the algorithms and their execution time will be compared to each other using real-world data.

4- Results and Analysis

In this section, the values of the parameters, the results obtained from the model and the presented algorithms, and the comparison of these results will be examined.

The reason why this work is not quantitatively compared with previous works and only qualitatively compared is that mathematical models can only be compared qualitatively when the cost function is equal. In this case, if the cost function is equal, no new work has been done, and only the execution time of the corresponding model or algorithm has improved.

When a new objective function is proposed, new challenges of the problem are considered, which have not been considered so far due to problem simplification or lack of importance. It is obvious that considering new parameters makes the optimization model more complex and solving it more difficult. It is even possible that the new model is not linear and can only be approached by providing algorithms to approximate the solution. All of these depend on the researcher's goal.

In some cases, it may not be necessary to consider all these parameters and create additional complexity. For example, in a home network, there is no need to consider load balancing; receiving responses in a timely manner is sufficient. In some cases, where network traffic is low and the home or office network has a small number of switches, having a controller is enough. However, what really matters are the provision of a model that is closer to reality and takes into account all aspects and is linear (meaning that it can find the best solution and the most certain answer with optimization software), has great value, and can be applied to small networks as well. It is necessary for large networks to have these models.

4-2- Qualitative Comparison of the Proposed Model with Previous Models

Most of the proposed works in this regard have considered network connections in an out-band manner. Only in article [37] is this structure and its necessity briefly mentioned. The model presented in this paper considers the network connections in an in-band manner. Additionally, an attempt has been made to increase the network scalability according to the structure proposed in article [22]. Scalability is one of the issues addressed in the investigation of SDN network problems and challenges. The proposed model is linear and solvable with software in a small scale (few switches), but in a large scale with the CPLEX software on a system with Core i7 specifications and 8 GB of RAM, it will not yield a global optimal solution and will be trapped in local optimization (more than 10 switches), especially when solved using the cracked version of CPLEX software, which itself is a hindrance to the work.

When the presented model is linear, it means that the model's solution is deterministic and globally optimal. The model is not trapped in a local solution. In this case, there is no need for simulation, and if an algorithm is provided to solve it in less time, only the algorithm's results are compared to the model's results to validate the algorithm's accuracy.

Therefore, in the following section, the values of the constant parameters are first stated, and the model's results are examined on a small scale, and then the algorithm's results are presented and compared with the

same given values. Then, the algorithms are analyzed for larger graphs, and their execution time and the number of controllers are compared on a larger scale. Additionally, the amount of control messages exchanged by the algorithms is also compared. The algorithms are written in MATLAB software, and the MATLAB execution time is used for comparison.

4-3- Problem Data 4-3-1 Analyzed Graphs

To analyze and execute algorithms, the collection of topology graphs called Zoo is utilized. The graphs are available in two formats, .gml and .graphml, which contain data on the number of switches, their interconnections, and geographical positions that can be extracted. The smallest size of these graphs consists of 9 switches, while the largest size comprises 71 switches.

To examine the model's outcome and compare algorithm results, a hypothetical graph in Fig.4-1, consisting of 5 switches, is utilized.

The request rate of each controller is randomly generated between the range of 1000 to 10000. For more accuracy, each scenario is executed 30 times and their average results are taken. All graphs are connected. The model assumption is that the input graph is also connected.

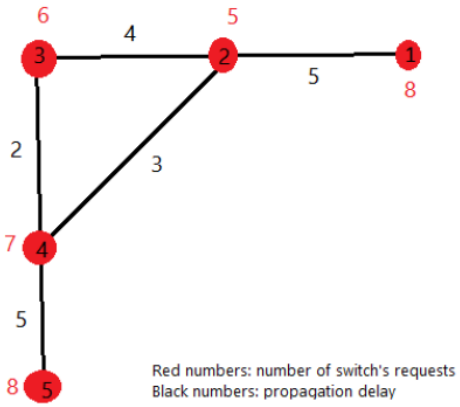

Fig. 4-1: Graph under investigation

4-4-Constant Data

4-4-1- Types of Controllers

Except for the scenario involving 5 hypothetical switches, it is assumed that there are three types of controllers with the specifications in Table 4-1:

Table 4-1: Controller Specifications

Specification-type	port	Time to process a request	The number of requests that can be supported at a time	cost
1	52	1	1000000	200000
2	34	1	500000	150000
3	44	1	700000	170000

The acceptable maximum network delay has been considered to be 100 milliseconds, and this delay is calculated as follows:

$$\text{Total Delay} = 2 * \text{Propagation Delay} + 2 * \text{Transmission Delay} + \text{Processing Delay} \quad (1\text{-}4)$$

The propagation delay is calculated based on the distance between two nodes and the propagation speed in the transmission medium using the following formula:

$$\text{Propagation Delay} = \text{Distance} / \text{Propagation Speed} \quad (2\text{-}4)$$

The propagation speed for copper cable ranges between 0.59*C to 0.79*C, where a value of 0.59*C is used in this paper (C represents the speed of light).

Transmission delay is the time required to place data on the wire. This delay includes the time spent in each layer of the TCP/IP stack up to the data layer (sending data in bits). The transmission delay is calculated as follows:

$$\text{Transmission Delay} = \text{Packet Size (bytes)} / \text{Link Speed (bytes/sec)} \quad (3\text{-}4)$$

The links used in the Zoo and Internet2 projects have speeds of 100 Gbits/s and 10 Gbits/s, respectively. Most of the links in the Zoo project are optical fibers with a minimum speed of 10 Gbits/s. Some maps have links with speeds of 45 Mbit/s, which are relatively old and

not used in this paper. Additionally, the control packets are very small in size. Therefore, due to their significantly low delay, they are ignored.

Processing delay is also the time required to process a request in the controller. The time taken for processing tasks that a controller needs to perform within a given time unit is added to the overall delay.

Furthermore, queue delay should be considered, but since control traffic has a higher priority and is sent earlier than other types of traffic, and a separate VLAN is defined for this type of traffic in SDN, this delay is also neglected.

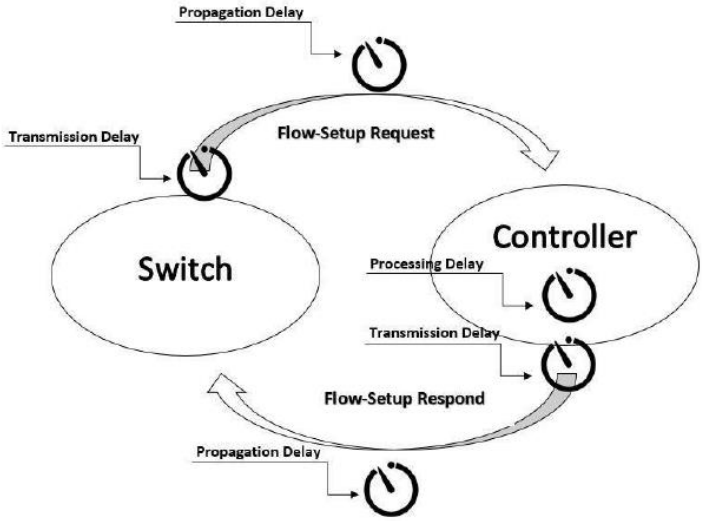

Fig. 4-2: Network delays for transmitting a control request from a switch to a controller and receiving a response from it.

4-5- Scenarios

4-5-1 -First Scenario - Network consisting of 5 switches

For this scenario, a hypothetical graph of 1-4 is used. To solve the problem of the number and location of graphs for this issue, an optimization model was written and solved using the CPLEX software. Only for this example, three hypothetical controllers were used to assess the power of the model.

Table 4-2: Controller Specifications for Small Graphs

Specification-type	port	Time to process a request	The number of requests that can be supported at a time	cost
1	3	2	90	100000
2	4	1	100	15000
3	4	1	105	17000

The placement of controller type 1 at the first location and controller type 2 at the fourth location. If the model is solved manually, the correct solution will be the same values. Switches 1 and 3 will be connected to the controller installed at the second location, and switch 5 will be connected to the controller installed at the fourth location.

As observed in Fig. 4-3, the load levels on each controller have been minimized, resulting in the minimum number of control network packets. The solution time for this scenario in CPLEX is 79.0 seconds, and the cost set with normalized coefficients is 17526.

However, it is obvious that this solution will vary for scenarios with the same number of switches, depending on how the switches are connected. In the following, the results of two algorithms for this scenario will be examined:

The Tabu Search Algorithm also selects switches 1 and 4 for controller positions and obtains a cost equal to the cost calculated by the model. The execution time of this algorithm for this scenario in MATLAB is 2 seconds, and the assigned switches to these controllers and the selected controller types are the same as the model.

Table 4-3: comparison between the results of CPLEX and the algorithms.

Solver	Cost
COLEX	17526

Tabu Search	17526
Genetic	17526

The genetic algorithm also selects two controllers for this problem and assigns them to switches 1 and 4. However, it has a longer execution time compared to the taboo search algorithm. The execution time for the genetic algorithm in this scenario is 3.2 seconds.

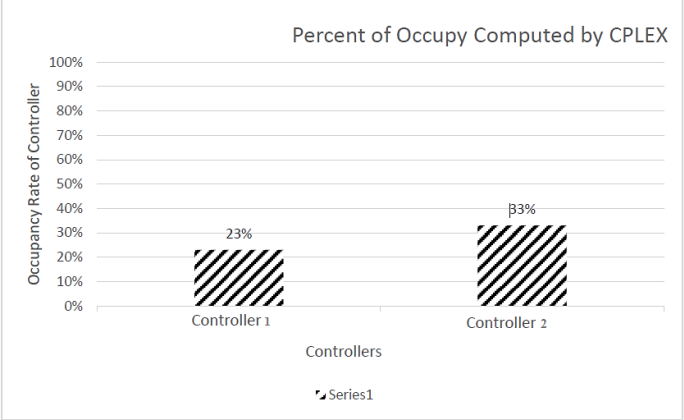

Fig. 4-3: The graph of the percentage of the capacity of the controllers according to the load

4-6- Evaluating Larger Scenarios

For this section, Zoo topology graphs are utilized. After extracting data from the graphs, each scenario is repeated 30 times with a uniformly random control request rate. The following different scenarios are examined:

4-6-1- Comparing Execution Time of Two Algorithms Based on Different Switches

To compare the execution time of the two algorithms, their execution time in MATLAB software is employed. The comparison graph of these two algorithms is presented in Fig. 4-4. It can be observed that the execution time of the Tabu search algorithm is significantly lower than the execution time of the genetic algorithm. However, the execution time of the genetic algorithm is dependent on the initial population size and the number of algorithm iterations. The initial population size for each scenario is four times the number of selected graph switches, and the number of iterations is set to 50 for all scenarios. The graph of the execution time for these two algorithms is presented in Fig. 4-4 for different scenarios. It should be noted that the solutions and execution time vary depending on the different adjacency matrices of the graph. Similar graphs are used for comparison with an equal number of switches. The minimum number of switches is 9, and the maximum is 71.

4-6-2- Comparison of the Number of Calculated Controllers Based on Different Switch Counts

To compare the number of calculated controllers based on different switch counts and establish comparable conditions, random graphs were used. Therefore, for different scenarios consisting of various switch counts, including delay and uniform random request rates within a selected interval across all cases, 1000 random graphs were created using the Erdos-Renyi algorithm. The median number of controllers was selected as the examinable results. As observed in Figure 6-4, increasing the number of switches in random graphs generally leads to an average increase in the number of controllers. However, it cannot be specifically stated that the number of controllers necessarily increases with the increase in the number of switches. The required number of controllers depends on various factors, such as the graph's shape, the number of edges, the degree of each node, minimum and maximum delay between nodes, and so on.

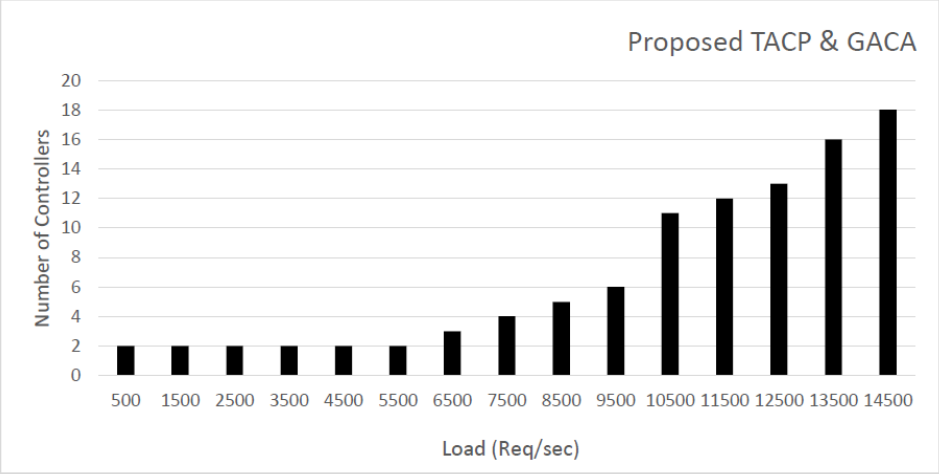

Fig. 4-4: The runtime of algorithms based on the number of switches.

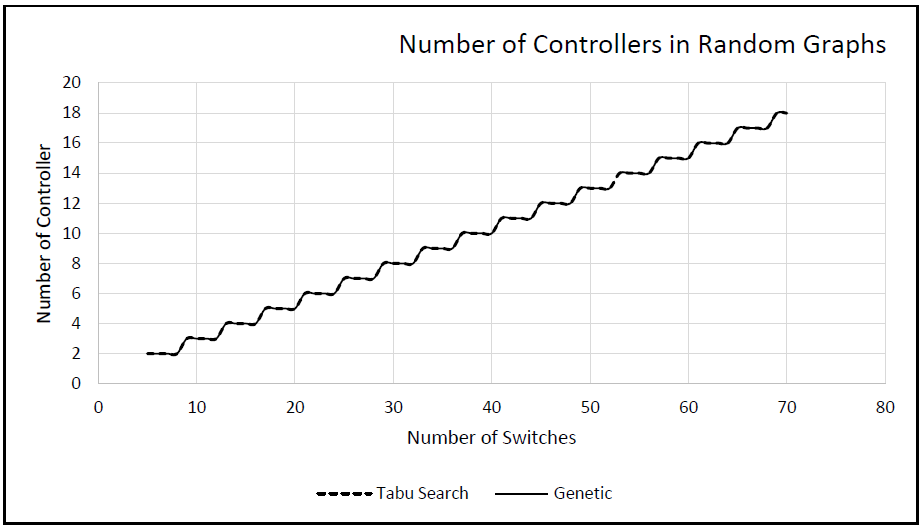

Fig.4-5: Required Number of Controllers Based on the Number of Switches in Random Graphs.

4-6-3- Comparison of the calculated number of controllers with respect to the maximum network delay

This scenario and subsequent scenarios have been tested on a sample graph of Bell South with 50 nodes. As expected, when the maximum acceptable network delay increases, the number of required controllers decreases. This decrease is steep, reaching 60 seconds, indicating significant changes in the function in this region. This indicates that the cost of maintaining network QoS at low delays is high. As expected, after a certain point, increasing the number of controllers does not decrease the maximum acceptable network delay.

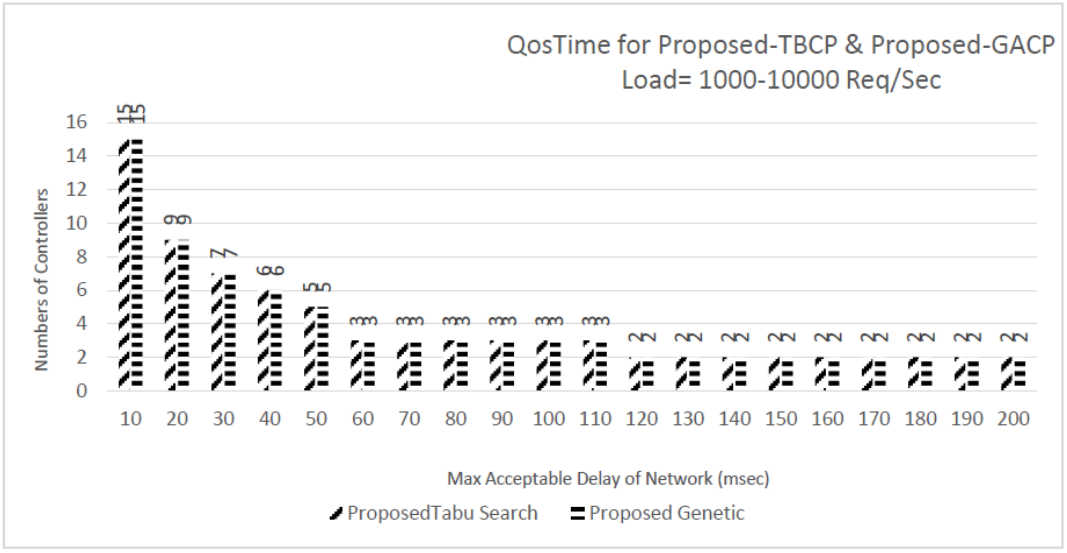

Fig. 4-6: Number of control nodes based on the maximum acceptable network delay.

4-6-4- Comparison of the number of computed controllers based on the average request load per switch in a scenario.

As observed, the number of required controllers increases with the increase in network load. However, this number increases only when the request volume on a controller, which manages the maximum network load, exceeds a certain threshold (in the chart, this value is 6000). When the request volume surpasses this threshold, the number of controllers increases gradually. As long as not all controllers are saturated, this trend continues with a gentle slope. Once all controllers become saturated, the increase occurs at a steeper slope. Again, until the complete saturation of these controllers, the increase in the number of controllers will have a steeper slope with their saturation.

The execution time and cost in relation to the average request load can also be observed in Figs. 4-9 and 4-10, respectively.

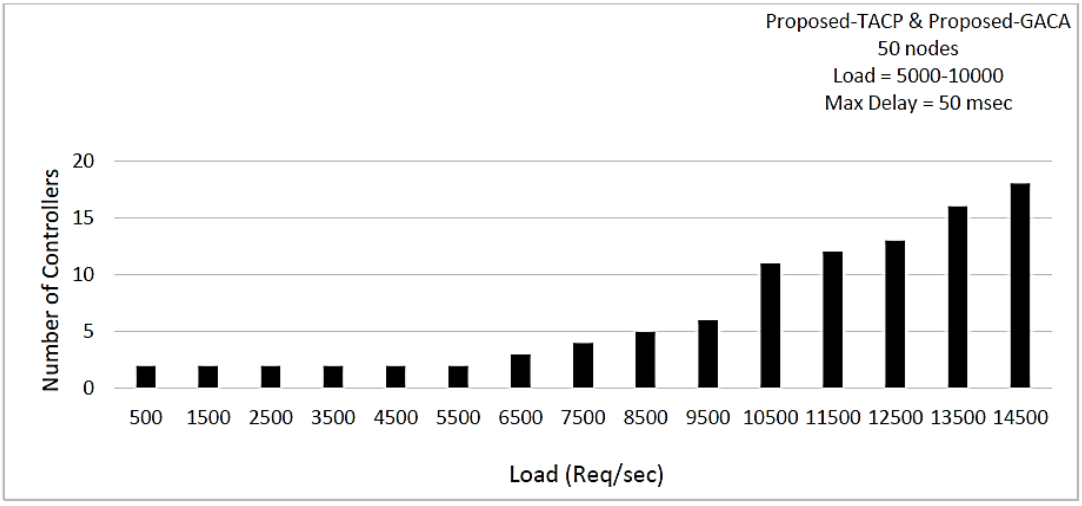

Fig. 4-7: The number of controllers based on the average request per switch.

It can be observed that in the genetic algorithm, when the average switch requests exceed 6500 and the initial saturation of controllers occurs, the algorithm execution time increases significantly. This increase in execution time occurs in the Tabu search algorithm, on average, when there are 12500 switch requests, and the number of controllers increases from 13 to 16 within this range, resulting in a significant cost increase.

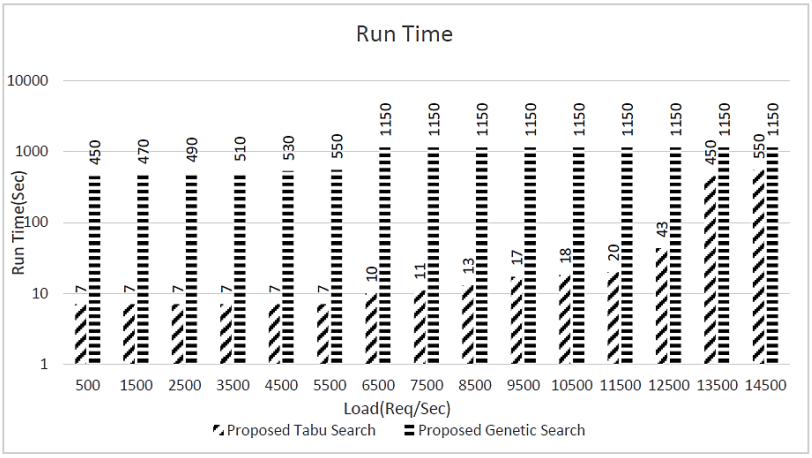

Fig. 4-8: The execution time in terms of the average request per switch.

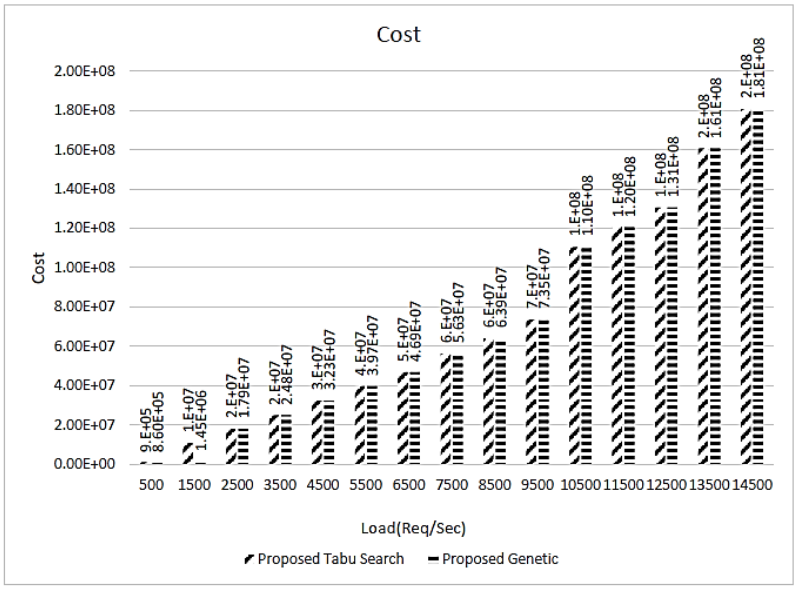

Fig. 4-9: Cost in terms of average request per switch.

4-6-5- Load balancing analysis in a scenario.

Continuing, load balancing between the selected controllers is examined. As seen in Fig. 4-10, the request distribution among the controllers is almost equal.

The occupancy rate of the controller is calculated using the following formula [45]:

Occupancy Rate of Controller
= All Requests to the Controller
/ Capacity of the Controller

The imbalance percentage of the distributed load among the controllers is calculated using the following equation:

In this case, since each controller may have different capacity levels, instead of using the maximum load and average load, the occupied maximum capacity and occupied average capacity are used. For this scenario, the imbalance percentage is 28.13%.

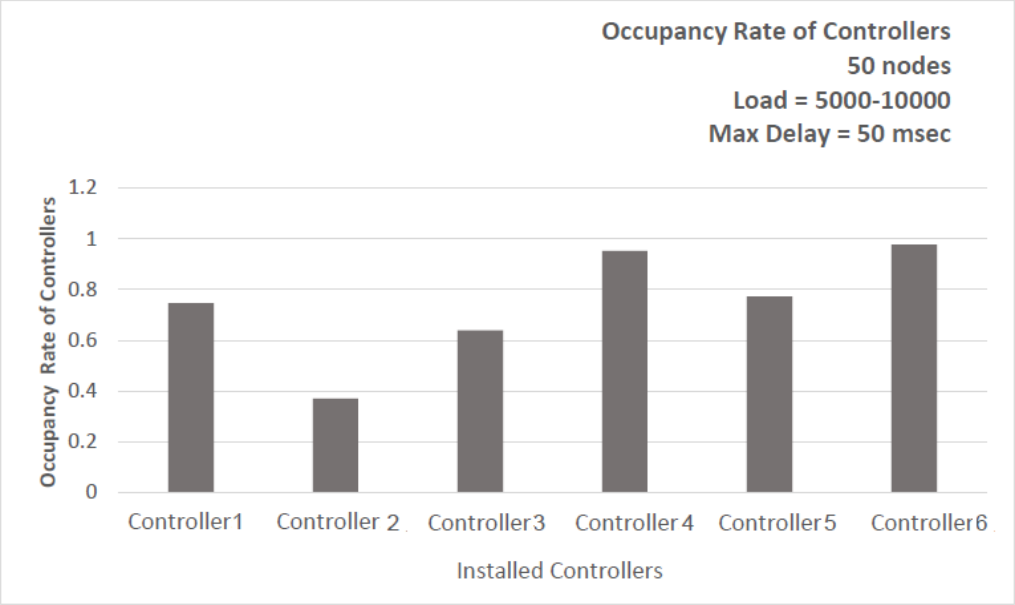

Fig. 4-10: The level of occupied capacity for each controller

5- Conclusion

This paper investigates the problem of controller placement in software-defined networks. To solve this problem, an optimization model is proposed with a focus on scalability (both geographic and scalability resulting from the growth of new requests in the network), reducing the delay caused by control message transmission between switches and controllers in the network, the financial cost of controller installation, and the amount of control messages exchanged in the network while considering the in-band links between controllers and switches. Scalability is one of the main challenges in SDN networks.

Existing models have considered out-band links. By considering in-band links and using certain switches as intermediaries for transmitting control traffic from other switches to the controller, the complexity of the problem significantly increases, and the resulting model becomes nonlinear. In this paper, a linear model is proposed for this purpose. The proposed model also aims to achieve load balancing among installed controllers.

The model presented has a global optimum solution at a small scale (maximum 10 switches) and a local optimum solution at a larger scale (more than switches).

Furthermore, two metaheuristic algorithms, namely Genetic Algorithm and Tabu Search, are proposed to solve the model in shorter time. So far, the Tabu Search algorithm has not been introduced to solve this problem. The Genetic Algorithm, which has been previously proposed for solving this problem, considers the number of controllers as an input to the algorithm. However, the Genetic Algorithm presented in this paper considers the number of controllers as unknown. The fitness level of each algorithm is also calculated based on the criteria of the presented model.

Moreover, by examining the two algorithms, it was observed that the Tabu Search algorithm has a shorter execution time compared to the Genetic Algorithm and performs better in this problem. Additionally, the probability of getting trapped in a local optimal solution is also lower for the Tabu Search algorithm.

For future tasks, it is recommended that this issue be addressed using Pareto-Optimal game and as a min-max problem with the approach of reducing the number of controllers, reducing energy consumption, and increasing the level of managed requests by each controller. It is also possible to consider queuing delay at switches as future tasks. Another solution could be to dynamically turn off some controllers at certain times by providing a prediction model for incoming requests and finding the peak time of requests in order to reduce energy consumption. This paper assumes that the communication problem between controllers has been solved. The communication protocol between them can be examined, and an optimization model can be proposed for their synchronization.

References

[1] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, “A survey of software-defined networking: Past, present, and future of programmable networks,” *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1617-1634, 2014.

[2] W .Stallings, *Foundations of Modern Networking: SDN, NFV, QoE, IoT, and Cloud*: Addison-Wesley Professional, 2015.

[3] Brandon Heller, Rob Sherwood, and Nick McKeown. 2012. The controller placement problem. In *Proceedings of the first workshop on Hot topics in software defined networks* (HotSDN '12). ACM, New York, NY, USA, 7-12. DOI=http://dx.doi.org/10.1145/2342441.2342444.

[4] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14-76, 2015.

[5] Thomas A. Limoncelli. 2012. OpenFlow: A Radical New Idea in Networking. *Queue* 10, 6, Pages 40 (June 2012), 7 pages. DOI: http://dx.doi.org/10.1145/2246036.2305856.

[6] E. Ng, Z. Cai, and A. Cox, "Maestro: A system for scalable openflow control," *Rice University, Houston, TX, USA, TSEN Maestro-Techn. Rep, TR10-08*, 2010.

[7] Y.-n. HU, W.-d. WANG, X.-y. GONG, X.-r. QUE, and S.-d. CHENG, "On the placement of controllers in software-defined networks," *The Journal of China Universities of Posts and Telecommunications*, vol. 19, pp. 92171-97, 2012.

[8] Y. Jiménez, C. Cervelló-Pastor and A. J. García, "On the controller placement for designing a distributed SDN control layer," *2014 IFIP Networking Conference*, Trondheim, 2014, pp. 1-9. doi: 10.1109/IFIPNetworking.2014.6857117.

[9] P. Xiao, W. Qu, H. Qi, Z. Li and Y. Xu, "The SDN controller placement problem for WAN," *2014 IEEE/CIC International Conference on Communications in China (ICCC)*, Shanghai, 2014, pp. 220-224. doi: 10.1109/ICCChina.2014.7008275.

[10] M. Guo, and P. Bhattacharya, "Controller placement for improving resilience of software-defined networks." pp. 23-27.

[11] A .T. Campbell, I. Katzela, K. Miki, and J. Vicente, "Open signaling for ATM, internet and mobile networks (OPENSIG'98)," *ACM SIGCOMM Computer Communication Review*, vol. 29, no. 1, pp. 97-108, 1999.

[12] A. Greenberg, G. Hjalmtysson, D. A. Maltz, A. Myers, J .Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang, "A clean slate 4D approach to network control and management," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 5, pp. 41-54, 2005.

[13] Stefan Schmid and Jukka Suomela. 2013. Exploiting locality in distributed SDN control. In *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking* (HotSDN '13). ACM, New York, NY, USA, 121-126. DOI=http://dx.doi.org/10.1145/2491185.2491198.

[14] Soheil Hassas Yeganeh and Yashar Ganjali. 2012. Kandoo: a framework for efficient and scalable offloading of control applications. In *Proceedings of the first workshop on Hot topics in software defined networks* (HotSDN '12). ACM, New York, NY, USA, 19-24. DOI=http://dx.doi.org/10.1145/2342441.2342446.

[15] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, and M. Zhu, "B4: Experience with a globally-deployed software defined WAN," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 3-14, 2013.

[16] Advait Dixit, Fang Hao, Sarit Mukherjee, T.V. Lakshman, and Ramana Kompella. 2013. Towards an elastic distributed SDN controller. In *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking* (HotSDN '13). ACM, New York, NY, USA, 7-12. DOI: http://dx.doi.org/10.1145/2491185.2491193.

[17] Teemu Koponen, Martin Casado, Natasha Gude, Jeremy Stribling, Leon Poutievski, Min Zhu, Rajiv Ramanathan, Yuichiro Iwata, Hiroaki Inoue, Takayuki Hama, and Scott Shenker. 2010. Onix: a distributed control platform for large-scale production networks. In *Proceedings of the 9th USENIX conference on Operating systems design and implementation* (OSDI'10). USENIX Association, Berkeley, CA, USA, 351-364.

[18] V. Yazici, M. O. Sunay and A. O. Ercan**,** "Controlling an SDN Network via Distributed Controllers", in Proceedings of NEM Summit 2012, Istanbul, Turkey, October 2012.

[19] Theophilus Benson, Aditya Akella, and David A. Maltz. 2010. Network traffic characteristics of data centers in the wild. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement* (IMC '10). ACM, New York, NY, USA, 267-280. DOI=http://dx.doi.org/10.1145/1879141.1879175.

[20] Y. Jimenez, J. A. Cordero and C. Cervelló-Pastor, "Measuring robustness of SDN control layers," *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, Ottawa, ON, 2015, pp. 774-777. doi: 10.1109/INM.2015.7140373.

[21] S. H. Yeganeh, A. Tootoonchian, and Y. Ganjali, "On scalability of software-defined networking," *IEEE Communications Magazine*, vol ,51 .no. 2, pp. 136-141, 2013.

[22] S. Bhandarkar, G. Behera, and K. A. Khan, "Scalability Issues in Software Defined Network

(SDN): A Survey," *Advances in Computer Science and Information Technology (ACSIT)*, pp. 81, 2014.

[23] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood, "On Controller Performance in Software-Defined Networks," *Hot-ICE*, vol. 12, pp. 1-6, 2012.

[24] Y. Hu, W. Wang, X. Gong, X. Que, and S. Cheng, "On reliability-optimized controller placement for software-defined networks," *China Communications*, vol. 11, no. 2, pp. 38-54, 2014.

[25] Y. Zhang, N. Beheshti and M. Tatipamula, "On Resilience of Split-Architecture Networks," *2011 IEEE Global Telecommunications Conference - GLOBECOM 2011*, Houston, TX, USA, 2011, pp. 1-6. doi: 10.1109/GLOCOM.2011.6134496.

[26] D. Hock, M. Hartmann, S. Gebert, M. Jarschel, T. Zinner and P. Tran-Gia, "Pareto-optimal resilient controller placement in SDN-based core networks," *Proceedings of the 2013 25th International Teletraffic Congress (ITC)*, Shanghai, 2013, pp. 1-9. doi: 10.1109/ITC.2013.6662939.

[27] S. Lange, S. Gebert, T. Zinner, P. Tran-Gia, D. Hock, M. Jarschel, and M. Hoffmann, "Heuristic approaches to the controller placement problem in large scale SDN networks ",*IEEE Transactions on Network and Service Management*, vol. 12, no. 1, pp. 4-17, 2015.

[28] Deze Zeng, Chao Teng, Lin Gu, Hong Yao and Qingzhong Liang, "Flow setup time aware minimum cost switch-controller association in Software-Defined Networks," *2015 11th International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness (QSHINE)*, Taipei, 2015, pp. 259-264.

[29] M. Obadia, M. Bouet, J. L. Rougier and L. Iannone, "A greedy approach for minimizing SDN control overhead," *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, London, 2015, pp. 1-5. doi: 10.1109/NETSOFT.2015.7116135.

[30] M. F. Bari *et al.*, "Dynamic Controller Provisioning in Software Defined Networks," *Proceedings of the 9th International Conference on Network and Service Management (CNSM 2013)*, Zurich, 2013, pp. 18-25. doi: 10.1109/CNSM.2013.6727805.

[31] A. Ruiz-Rivera, K.-W. Chin, and S. Soh, "Greco: an energy aware controller association algorithm for software defined networks," *IEEE Communications Letters*, vol. 19, no. 4, pp. 541-544, 2015.

[32] H. Alyusuf, "Solving the Software Defined Network Controller Placement Problem Using Complex Network Analysis," 2115.

[33] A. Sallahi, and M. St-Hilaire, "Optimal model for the controller placement problem in software defined networks," *IEEE Communications Letters*, vol. 19, no ,1 .pp. 30-33, 2015.

[34] H. K. Rath, V. Revoori, S. M. Nadaf and A. Simha, "Optimal controller placement in Software Defined Networks (SDN) using a non-zero-sum game," *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*, Sydney, NSW, 2014, pp. 1-6. doi: 10.1109/WoWMoM.2014.6918987.

[35] S. Lange *et al.*, "Specialized Heuristics for the Controller Placement Problem in Large Scale SDN Networks," *2015 27th International Teletraffic Congress*, Ghent, 2015, pp. 210-218. doi: 10.1109/ITC.2015.32.

[36] L. F. Müller, R. R. Oliveira, M. C. Luizelli, L. P. Gaspary and M. P. Barcellos, "Survivor: An enhanced controller placement strategy for improving SDN survivability," *2014 IEEE Global Communications Conference*, Austin, TX, 2014, pp. 1909-1915. doi: 10.1109/GLOCOM.2014.7037087.

[37] L. Schiff, S. Schmid, and M. Canini, "Medieval: Towards A Self-Stabilizing, Plug & Play, In-Band SDN Control Network," *SOSR (Demo)*, 2015.

[38] http://www.topology-zoo.org/.

[39] J. Holland, "Holland, Adaptation in natural and artificial systems," MIT Press, Cambridge, MA, 1992.

[40] M. Srinivas, and L. M. Patnaik, "Genetic algorithms: A survey," *computer*, vol. 27, no. 6, pp. 17-26, 1994.

[41] S. Jayaswal, "A comparative study of tabu search and simulated annealing for traveling salesman problem," *Proj. Rep. Appl. Opt. Univ. Water., Canada. MSCI-703*, 2008.

[42] F. Glover, "Tabu search—part I," *ORSA Journal on computing*, vol. 1, no. 3, pp. 190-206, 1989.

[43] F. Glover " ,Tabu search—part II," *ORSA Journal on computing*, vol. 2, no. 1, pp. 4-32, 1990.

[44] Colin R. Reeves (Ed.). 1993. *Modern Heuristic Techniques for Combinatorial Problems*. John Wiley & Sons, Inc., New York, NY, USA.

[45]] Olga Pearce, Todd Gamblin, Bronis R. de Supinski, Martin Schulz, and Nancy M. Amato. 2012. Quantifying the effectiveness of load balance algorithms. In *Proceedings of the 26th ACM international conference on Supercomputing* (ICS '12). ACM, New York, NY, USA, 185-194.