Optimization of open flow controller placement in software defined networks

Raghda Salam Al mahdawi, Huda M. Salih
Department of Computer Engineering, Diyala University, Iraq

Article Info

Article history:

Received Oct 2, 2019
Revised Dec 20, 2020
Accepted Jan 13, 2021

Keywords:

Computer networks
Network controller
Open flow controller
Software defined network

ABSTRACT

The world is entering into the era of big data where computer networks are an essential part. However, the current network architecture is not very convenient to configure such leap. Software defined network (SDN) is a new network architecture which argues the separation of control and data planes of the network devices by centralizing the former in high level, centralised devices and efficient supervisors, called controllers. This paper proposes a mathematical model that helps optimizing the locations of the controllers within the network while minimizing the overall cost under realistic constrains. Our method includes finding the minimum cost of placing the controllers; these costs are the network latency, controller processing power and link bandwidth. Different types of network topologies have been adopted to consider the data profile of the controllers, links of controllers and locations of switches. The results showed that as the size of input data increased, the time to find the optimal solution also increased in a non-polynomial time. In addition, the cost of solution is increased linearly with the input size. Furthermore, when increasing allocating possible locations of the controllers, for the same number of switches, the cost was found to be less.

Corresponding Author:

Raghda Salam Al mahdawi
Department of Computer Engineering
Diyala University
Diyala, Baqubah, 32001 Iraq
Email: raghdasalam@uodiyala.edu.iq, raghdasalam@ymail.com

1. INTRODUCTION

Recently, the number of users has rised exponentially, this urged the network operators and vendors to seek new network designs and advanced annaovations [1]. Today, computer networks are used extensively to enable communication throughout the globe. Wi-Fi and cellular networks shall be very adaptable within their network infrastructure. A resolution to this is moving the current networks towards more flexible and easier to manipulate architecture, called software defined network (SDN) [2].

SDN is an emerging network design that is dynamic, manageable and cost-effective. SDN can boost the network up for the bandwidth-hungry applications, which is the nature of today's requirements [3]. SDN architecture decouples the network's control and forwarding functions and permits the underlying infrastructure to be abstracted for network services via abstracting the network equipment and their operating system. It enables the control plane to become directly programmable and moved into a central location within the network, called "controller" [4].

The two main parities in this architecture are the controllers and switches, where the former makes decisions where information goes, and the latter is responsible for moving information hop-by-hop [5]. In between the controller and the switch, comes the channel, also called open flow protocol that is responsible for the necessary communications [6]. The controller uses this channel to install open flow tables into the switches, this table contains flow entries, such as IP and MAC addresses, and holds the necessary routing information that are required to direct the packets to their destinations [7].

Clearly, the placement of these controllers is very vital for the network to reduce the latency of the network that is the prerequisite of 5G and beyond systems. The optimal location offers reduced latency to all the switches within the network and therefore, reduced latency to the network subscribers [8]. Accordingly, such optimization decreases the transmitted power required to send the packets to their destinations. Once the transmitted power is reduced, the power consumption of the networks is reduced too [9]. At the end, the network efficiency shall be enhanced. Hence, this work sheds light upon the controller's placement under realistic constrains to minimize the cost while determining the number and the types of controllers to be optimised [10]. Different constraints and network metrics, such as controller capacity, controller and link types, link bandwidth and connectivity of the network devices have been considered. To achieve this, the following contributions have been made:

– Given the many changes related to the controller, we presented a model that decides to supply the ideal number, area and type of controller at the same time. The purpose of the model is to reduce network costs by taking into account requirements such as the controller's range, contact area and power management distance.
– Evaluation of the offer of the corresponding model and create typologies in different sizes.
– Develop typologies areas that determine how the number of potential areas may affect utilities.
– To evaluate organizational performance by expanding the size of the network in order to estimate the cost.

Assessment of the controlling body and limitation of the number of controllers as indicated by the dynamic understanding of traffic and the improvement of the use of resources by the controller's network to change links. We have provided for a direct ability to monitor the performance of the controllers based on continuous feedback from the current monitoring body in accordance with mandatory regulations. We have also proposed a reclassification algorithm to speed up the adjustment of the burden between controllers. In addition, a failing element aims to solve the problem of controller disappointment [11].

2. RELATED WORKS

In order to define the controller problem, a mathematical model is required to represent the behavior of SDN traffic [12]. We have used a solver called CPLEX to returns the optimal solution or the best solution found when the time limit is reached. Since the first introduction of the SDN controller issue by Heller *et al.* [7], many researchers have proposed different algorithms for dealing with one of the most difficult problems facing an engineer with respect to deploying an SDN network: the placement of controllers in the network.

A formidable challenge associated with solving SDN controller placement problems is that all of the algorithms proposed involve a tradeoff among scalability, resilience, and model expansion [13, 14]. With respect to investigations of the SDN controller problem, the technical paper published by Heller *et al.* [15] is one of the most cited. The authors proposed a heuristic approach to finding the ideal role for controllers in large SDN organizations. In this study, the main prediction was the normal inactivity scenario, which is considered essential in determining the inertia estimates required for large-scale SDN use [16]. The approach is dependent primarily on propagation delay, with the location of a controller being based on the shortest path between switches and controllers that have been assigned in the network topology [17]. This study offered the most accurate solution for addressing the problem. An interesting conclusion was that increasing the number of controllers does not necessarily decrease the average latency between switches and assigned controllers [18].

3. RESEARCH METHOD

We received an undirected organization topology $G(S,)$, where S indicates the arrangement of switches and E is the arrangement of edges in the middle. Let L speaks to the arrangement of dynamic controllers and N is the quantity of controllers, which is 1 as a matter of course. X=$[xik]$ $|S|\times N$ means the task relations among switches and controllers, in which each section xik=1 if switch i interfaces with controller k and xik=0 something else. For the gathered insights, tkn speaks to the preparing time for occasion n dealt with by controller k and i is the normal number of streams requiring for arrangement at switch i in current time.

3.1. Constraints

Our objective is to minimize the number of controllers' N considering a series of constraints, including as,

$$\forall\, k \in L, k < Cmax \tag{1}$$

and

$$\forall\, k \in L, k < Mmax \tag{2}$$

and

$$\forall\, k \in L, k < Emax \tag{3}$$

and

$$\forall\, k \in L, k < Dmax \tag{4}$$

and

$$\forall\, k \in L, \Sigma tknnEk < tmax \tag{5}$$

Finally;

$$\forall\, i \in S, \Sigma xikk = 1 \tag{6}$$

As shown in (1) to (5) for each controller specify the maximum CPU usage, memory usage, normal number of hours per second and normal number of dropped packets, and normal packet preparation time. Missing packet include both the packet provided by the controller and the associated switches. A packet issued by the driver ensures that the driver will be overloaded. Assuming we do not think about good and up-down switch times, the packet that removes the associated switches also shows that the controller ignores the flow setting problems. As shown in (6) determines whether each switch is assigned to one controller and one controller [19, 20].

3.2. Evaluation function

In order to evaluate the utility of a controller, an evaluation function is defined a function of the 5 metrics for each controller:

$$Uk = (Ek, Ck, Mk, Dk, tkn) \tag{7}$$

Each of the 5 metrics reflects the controller performance. We design the evaluation function as follows though there is more than one possible expression.

$$Uk = \alpha Ek + \beta Ck + \gamma Mk + \mu Dk + \omega \Sigma tknnEk \tag{8}$$

In this formula, α, and ω are coefficients that can be redone to alter the general centrality of the 5 measurements. Uk is standardized to [0,1] expecting to rearrange the count. The ordinary range of Uk is indicated as [$Ulower$], where $Ulower$>0 and $Uupper$<1. When Uk>$Uupper$, it implies controller k is over-burden and subordinate switches ought to be reassigned to different controllers. On the off chance that no dynamic controllers have enough limit (regarding parcels handling amount), another controller will be enacted to assume control over the unassigned switches. When Uk<$Ulower$, it implies the related switches of controller k can be converted to those of other controller to chop down assets. Thinking about the requirements, when the estimation of a measurement goes past the ordinary area, the estimation of the assessment capacity should see strange consequently. Subsequently, the assessment capacity can be changed as (8) [21-23].

$$Uk = Uk * ||(Ek > Emax)||(Ck > Cmax)||(Mk > Mmax)||(Dk > Dmax)||(\Sigma tknnEk > tmax) \tag{8}$$

where $Uk*$=αEk+βCk+γMk+μDk+ω$\Sigma tknnEk$ and ‖ is the logical operator OR. If any metrics violate the constraints, Uk will become 1 immediately and trigger the centralized scheduler program.

3.3. Reassignment

In the attached schedule, the contacts associated with each driver are sorted by traffic key. Meanwhile, the controls are set at your fingertips [24]. The largest stacking switches in the overload controller are initially switched to another dynamic controller with the highest available limits for example, packet processing probability and are intended to regulate steel between controllers over a period of short time [25, 26].

3.3.1. Algorithm 1: Reassignment algorithm

```
Input: Network topology $G$, Switch set $S$, Active controller set $L$, Traffic set $A$=[□$s$] |$S$|,
Previous assignment $X$
Output: New assignment $X*$
1: $Lv$← List of the outstanding controllers for $X$
2: for each $l$ from $Lv$ do
3: $Hc$← Sorted list (descending capacity) of controller excluding $l$
4: $Hs$← Sorted list (descending loading) of the switches for controller $l$
5: $s$← First switch in $Hs$
6: while statistics of $l$ violate the constraints (1) to (6) do
7: $k$← First controller in $Hc$
8: if $s<Emax-Ek$ then
9: update $X$← reassign switch $i$ to controller $k$
10: recalculate $Hc$, $Hs$
11: else if $s$ is the last item in $Hs$ then
12: update $X$← add a new controller
13: recalculate $Hc$
14: reset $s$← first switch in $Hs$
15: else if $Hs=\emptyset$ then
16: update $L$,← remove $l$
17: break
18: else
19: $s$← Next switch in $Hs$
20: end if
21: end while
22: end for
23: $X*$←$X$
```

3.4. Failover

In the failover component, the selection program regularly checks the pulse of each controller as well as the connected status of each switch in each time unit [27]. When you identify inactive controllers or unspecified switches, you set up a unified scheduling program to resolve the issue. We address the issue in two stages: i) check that there are controllers available between actors and partners for unspecified changes; ii) If no accessible controller is found, start another controller to check the switches. The connection cycle is similar to the redefinition cycle, for example, the most important requirement is the heaviest steel switches and upper limit controllers [28].

In our case, all controllers have only an environmental perspective on the topology of the network. Since in our situation we would expect the controllers to only deal with layer 2 learning and startup problems, they can, of course, learn the MAC address and specify the address. Thus, after the redefined cycle, controllers can become aware of the topological change and adapt as needed. To start at level 3, the controller must find a way to obtain information about the new topology at any time when re-mapping the switch, which will be considered in the next step.

4. RESULTS AND DISCUSSION

4.1. Evaluation one - Failover

As mentioned in the assessment system, the statistics application is acquired by the manager for the statistics application of the topology partner Mininet, which always examines the position of the association between the switches and the activity of the manager. If there is no chance that the connection status will stop, an immediate opportunity for the external scheduler is created in the scheduler. Hence, the scheduler starts immediately and designs another/current controller for users who compare switches. Our basic evaluation results show that the switches can be reconnected to the ACTION counter in 3 seconds. One of our future tasks is to maintain a design table for the connection between controls and switches. Accordingly, the "DOWN controller" capability of the scheduler can be used as soon as the scheduler can justify designing another/current controller for switches originally designed for the DOWN controller. This saves up to 3 seconds to achieve takeover results.

4.2. Evaluation two

In normal situation, it does not create enough load to run a "pingall" command on the Mininet topology head mentioned above (27 hosts, 13 switches) to overload the POX l2_learning controller. One of the issues in this review was finding the normal way to recreate situations where the administrator has to request an event load (normal packet response time sometimes occurs completely, or even events occur.). We first tried extending the stack with a handler by adding additional data and changing the topology of Mininet, but "pingall" executes the "ping" command separately between 2, so PACKET_IN- events are usually the same than the size of topology, ie. We decided to look for an answer to limit the administrator's ability to handle PACKET_IN can be adapted to accelerate administrator overload. It turned out that the Linux utility "cpu limit" could definitely mess up the processors, and tests show that it worked perfectly for our needs as shown in Figure 1.

Another problem was that the package preparing the details for the Mininet switch could change from topology to topology, but there is only one PACKET_IN control pause of the controller, regardless of the topology of the hidden switches. To make the administrator aware of the various provisions that violate the work with their hidden switch, our responses identify the possibility of a break from the perspective of a topology change. Our response took advantage of Mininet's integrated "ovs-dpctl" tool when our general audit was performed on the Mininet (OVS) [2]. The return of the "ovs-dpctl" tool has a field called "hit" that shows the number of packets preparing for the pause that takes place at the OVS level. In our response, the validated content was sent to the Stats Collector application in the Mininet topology tool, which again controls the ovs-dpctl number and will send "Overloaded controller", which activates program capabilities a to perform the reclassification calculation when the number of "hits" increases. This is the method to enable the ability to perform reclassification algorithm.

In this review, another “Controller-Overloaded” indicator is that the manager's regular preparation time PACKET_IN has been halved from the previous overview. To do this, the scheduling application is scheduled at regular intervals to check the normal processing time of the PACKET_IN apartments collected by the Stats Collector program in the staffing table. When you recognize half the slope of the previous view, it performs the transfer calculation. This is a research method to perform the transfer calculation. Figures 2 and 3 describes these two conditions.

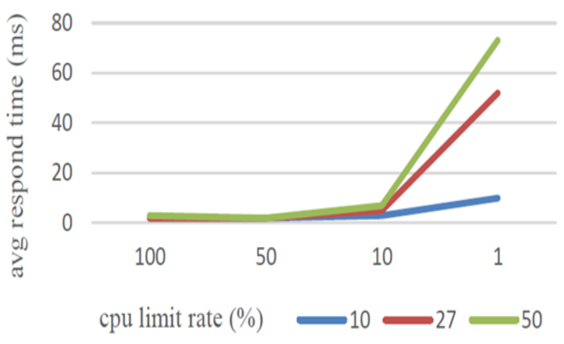

Figure 1. Graph for average response time and CPU ra

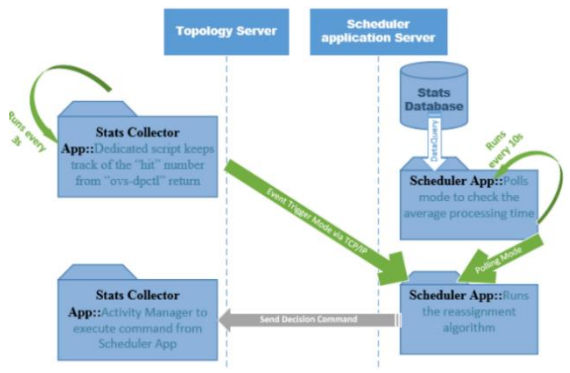
Topology Server
Scheduler application Server
Stats Database
DataQuery
Runs every 3s
Stats Collector App::Dedicated script keeps track of the "hit" number from "ovs-dpctl" return
Runs every 10s
Scheduler App::Polls mode to check the average processing time
Event Trigger Mode via TCP/IP
Polling Mode
Stats Collector App::Activity Manager to execute command from Scheduler App
Send Decision Command
Scheduler App::Runs the reassignment algorithm

Figure 2. Two modes of app

Our underlying review results show that it is possible to switch to a typical controller that is normally stacked within 8 seconds in lift mode, but this can usually be stopped within 3 seconds in a single trigger mode. Advanced SDN tools, such as Mininet and OVS, give us the convenience of re-creating the SDN system. Either way, it is not easy to emulate a huge organization with minites with limited resources. Initially, we tried to run Mininet topology and many POX controllers in a similar virtual machine (VM), but the resource value between Mininet topology and the regulator shares the regulator exposure, especially as the size of the organization evolves. This problem can be understood when the CPU and Mininet memory can be separated from the POX controllers.

Later, we proposed running Mininet and POX controllers on individually connected virtual machines. The loading of the simulated network could be easily adjusted by simply modifying the number of bridged VMs, which increased the flexibility of the simulation. To simulate the performance of the controller, we have the "cpulimit" tool to restrict the correct processor for each controller. It will be more advantageous that this limitation of processor usage may later become an indirect part of SDN tools.

Some visualization tools are suitable for Mininet regulators. In any case, it seems that most of them do not defend the powerful topology of the network, which is a major problem with SDN research. Moreover, they are not feasible under the compatible circumstances. The video display and information display select more adaptable and user-friendly means of data analysis.

4.3. CPLEX optimization

As the problem size increases, it becomes very difficult (not to say impossible) to solve the problems manually and that is why a solver is needed. The optimization for this scenario is run on IBM's ILOG CPLEX Optimizer version 12.5. The optimizer runs on a single thread. The process that is shown in Figure 3 is run on a computer that has the solver installed.

We are already aware of the solution to this problem but the optimizer will validate the above solution. Furthermore, the optimizer suggests the time taken to get the solution, which is an important information when the problem size increases. The solution is plotted and shown in Figure 3. The links connecting the switches to the controllers are shown with either solid blue or green colour. A blue colour indicates the cost for the links of type one (1 Mbps). The green colour is the cost for the links of type two (10 Mbps). Similarly, the controller placements are coloured and are placed on top of the possible placement markers. The yellow colour indicates the first controller type and the red colour indicates the second controller type. As we can see, the result of the optimization matches the solution found earlier. The links between controllers are always the same type of colour, since we assume that it takes minimal bandwidth to have controllers communicate with each other. Because the goal of the planning model is to place controllers on a network, traffic between controllers is not considered and in our model, the cheapest link speed should be good enough for connecting controllers together.

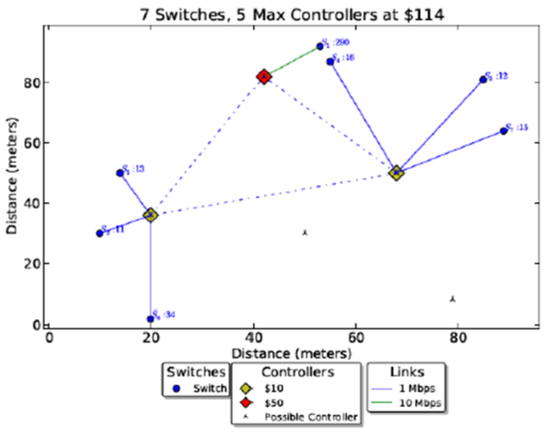

Figure 3. Optimal solution found by CPLEX for the planning problem

4.4. Small to large input sizes

The optimization should be measured using a variety of methods. One way to measure, it is to keep track of the cost for diff t ranges of solutions. Another method is to keep track of the time taken by the solver to fi an optimal solution. The cost of the solution should be increasing as the number of switches increases. This is because the input increase means more controllers may be placed and all those switches must be connected. The number of switches in the topology is directly related to the number of links placed by the solver between switches and controllers. Since every switch must be connected to a controller, it increases the cost of the solution.

The input for the model is very important and, in this section, we show what the input for controllers may be. Today many controllers exist and have been tested in real scenarios. Each of these controllers has their own specifications. The specification of each controller type depends on the hardware used and the implementation of the programming language. Table 1 shows four types of controllers that are used by the solver.

The specification of the link types that connect controllers and switches together is easy to determine. On any networking sales website, one can determine the cost per meter and the types of links that are available. Currently, a meter of 100 Mbps ethernet is about $0.25 and $0.63 for a meter of 1 Gbps ethernet. Furthermore, over 10 Gbps fi er optic cable is on average $29 per meter. Table 2 shows the link input available for optimization.

Table 1. The type of controllers that are used as input to the planning model

	Type 1	Type 2	Type 3	Type 4
Cost ($)	1200	2500	6500	1200
Number of Ports	8	32	64	64
Processing Time (Seconds)	2500	4000	8000	15000
Number of Controllers	20	15	10	6

Table 2. The type of links that are used as input to the planning model

	Type 1	Type 2	Type 3
Cost/meter ($)	0.25	0.63	29
Bandwidth	10Mbps	1Gbps	10Gbps

When the error bars are wide, it indicates that the results are less reliable because the range covers a wider set of values. This is shown in Figures 3 and 4. We can improve the error bars by running more than four instances for each problem. Decreasing the error bars is possible if we decrease the standard deviation since standard deviation is related to the square root of the number of instances each problem is repeated. Looking at the figure, when |P | is 20 and |S| is 100, the error bars are very wide. To make the bars shorter, we need to make the standard deviation smaller by running each problem more than 4 times. If we wish to decrease the standard deviation by half, each problem would have to run 16 times. When |P | is 20 and |S| is 100, each problem instance takes 16 hours to complete. This would mean that for 16 runs, it would take 256 hours to complete all the 16 instances. The standard deviation for four instances is 8,490 seconds and the confidence interval is ±27,019 seconds. Assuming the same mean is used, if we were to run the problem 16 times, the confidence interval would decrease to ±9,056 seconds (because standard deviation decreased by half). This is 33% of the confidence interval when we compare it to the four instances from our result.

Also, in Figures 3-4, we can see that the error bars are wide for |P|≥15 and ISi 2:50. We can improve the reliability of the result for problem 31 by running an instance of the problem 16 times. Since the average time for that problem is about 13 hours (46,219 seconds), running it 16 times may take 205 hours (15 fold increase). The results show that the total time taken to find optimal solutions for every problem run four times is 19 days (1,656,776 seconds). Improving the reliability by running all of the problems 16 times is not practical because it would have taken 306 days to find the solutions. The solution time was expected to be high because integer programming problems fall into NP-Hard problem types.

Figures 3-5 show the plots of the results grouped by IPI- The plots show the solution time and solution cost against the number of switches. The figures show more details than the figures that have all the IPI in one graph. The reason for this is that when IPI is small, the range of values for the solution time and the cost are much lower than when IPI is of a higher value and this allows us to better analyze the graphs. For example, if we look at the time taken to find an optimal solution when ISi is 30 and IPI is 5, we cannot see any details in Figures 4 and 5 but we see considerable details in Figures 3-5.

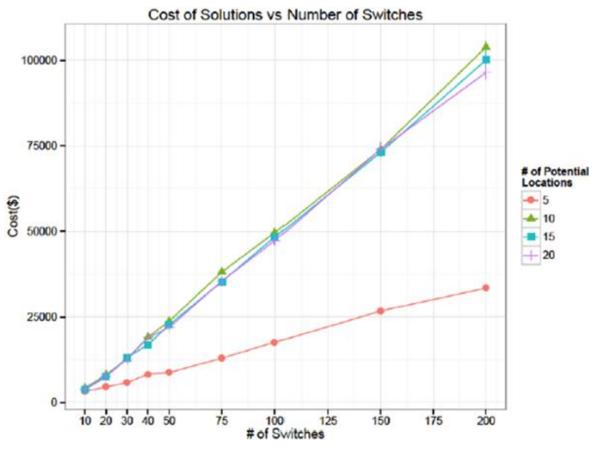

Figure 4. Cost of solutions colored against maximum number of controllers

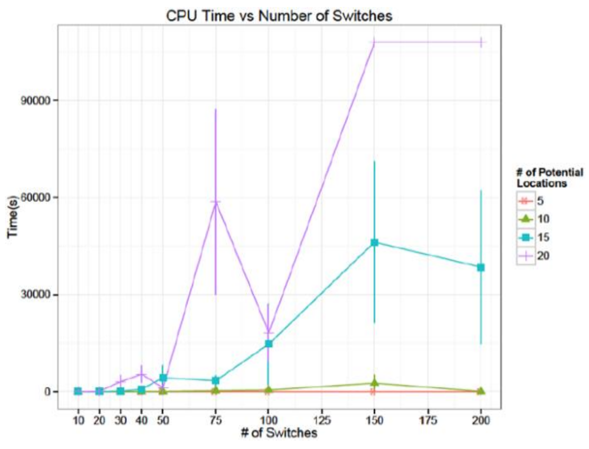

Figure 5. Solutions time against the number of switches with 95% confidence interval

5. CONCLUSION

It is essential to develop methods for finding the preferred controllers in a network, because SDN networks transmit motion to a controller. This study finds an ideal place for network mentors by suggesting the controllers. Current network is augmented in SDN with a mathematical model that reduces the cost of setting up controllers. The goal is to find out the minimum cost of having controllers while considering flow setup, controller processing, link bandwidth controllers and network latency for switches and find connections between controllers. Existing SDN networks that require a planning process are formed by another mathematical model that accepts pre-installed equipment.

In terms of cost, we can also see some general trends. The first thing we can observe is that the cost of solutions is higher as the input size increases. As the number of possible controller locations increased, for the same number of switches, the optimal cost was found to be slightly cheaper (up to a certain point). The planning model consists of a binary integer program that minimizes the cost of placing controllers on a network while keeping the flow setup latency under a threshold and respecting the inventory that is available for the planning process.

Our expansion includes two types of experiments. The first one investigates the result of the expansion model by adding and removing switches. The second experiment investigates how the existing network changes with different cost to remove existing inventory from the network. Before any expansion topology can be optimized, planning the network by the planning model must be done. For the first experiment, 3 different networks are planned randomly that form the topologies that our expansion model performs modifications on. Then for each of the topologies, 15, 10, and 5 switches are removed from the network and 5, 10, and 15 switches are added to the network before expansion is run. The results of the optimal solutions show that removing items from the network is inexpensive in cost and time. However, when adding 5, 10 and 15 switches to the network, the solution time and cost of the network increases. However, when switches are added, to expand the network, the solver takes time but it is a quick operation. The time to expand an existing network by adding 15 switches ranges from 1 to 4 seconds. The cost of the solutions depends on whether an existing item is reallocated, if so, the optimal solution cost is much more expensive.

The second expansion experiment involves in adding a set of switches to an existing topology with the cost to remove the existing inventory that starts at 100 times the planning cost and goes down to 1 of the planning costs. From the results, we observe that the optimal solution when the removal cost of the existing inventory is 100, the existing network is not changed at all. A controller is placed for the additional switches that were added. As the cost to remove existing inventory decreased, the existing network started changing and the optimal solution cost also decreased. Furthermore, the cost to remove existing inventory decreased, changes to the existing network became more frequent. For future work, this work can be expanded to apply more advanced heuristic algorithms such as Tabu search. This would help the algorithm pass the local minimum and find solutions that are closer to the optimal solution.

REFERENCES

[1] N. Feamster and R. Jennifer, "The road to SDN, An intellectual history of programmable networks," *ACM Queue*, vol. 11, no. 12, pp. 20-22, 2013.

[2] N. McKeown *et al*., "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69-74, 2008.

[3] B. Heller *et al*., "The controller placement problem," P*roceedings of the first workshop on Hot topics in software defined Networks*, 2012, pp. 7-12.

[4] 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, Switzerland, 2013, pp. 18-25.

[5] Y. N. Hu *et al*., "On the placement of controllers in software-defined networks," *The Journal of China Universities of Posts and Telecommunications*, vol. 19, no. 2, pp. 92-171, 2012.

[6] J. I. Naser and A. J. Kadhim, "Multicast routing strategy for SDN-cluster based MANET," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 5, pp. 4447-4457, 2020.

[7] L. Yang *et al*., "Forwarding and control element separation (forces) framework," *RFC3746*, pp. 5-30, 2004.

[8] J. Medved *et al*., “Opendaylight: Towards a model-driven sdn controller architecture,” *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*, Sydney, NSW, Australia, 2014, pp. 1-6.

[9] B. Nunes *et al*., "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.

[10] M. Kind *et al*., "Split architecture: Applying the software define networking concept to carrier networks," *World Telecommunications Congress (WTC)*, pp. 1-6, 2012.

[11] B. Heller, "The controller placement problem," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 473-478, 2014.

[12] L. Li *et al*., "Toward software-defined cellular networks," *2012 European Workshop on Software Defined Networking*, Darmstadt, Germany, 2012, pp. 7-12.
[13] D. Venmani, "Demystifying link congestion in 4g-lte backhaul using openflow," *2012 5th International Conference on New Technologies, Mobility and Security (NTMS)*, Istanbul, Turkey, 2012, pp. 1-8.
[14] T. Luo *et al*., "Sensor openflow: Enabling software-defined wireless sensor networks," *IEEE Communications Letters*, vol. 16, no. 11, pp. 1896-1899, 2012.
[15] D. Simeonidou *et al*., "Enabling the future optical internet with openflow: A paradigm shift in providing intelligent optical network services," *2011 13th International Conference on Transparent Optical Networks*, Stockholm, Sweden, 2011, pp. 1-4.
[16] S. Gringeri, "Extending software defined network principles to include optical transport," *IEEE Communications Magazine*, vol. 51, no. 3, pp. 32-40, 2013.
[17] M. Shirazipour *et al*. "Openflow and multi-layer extensions: Overview and next steps," *2012 European Workshop on Software Defined Networking*, Darmstadt, Germany, 2012, pp. 13-17.
[18] S. Das *et al*., "Packet and circuit network convergence with openflow," *2010 Conference on Optical Fiber Communication (OFC/NFOEC), collocated National Fiber Optic Engineers Conference*, San Diego, CA, USA, 2010, pp. 1-3.
[19] Quintero-Duran, M., Candelo-Becerra, J. E., and Soto-Ortiz, J. D., "A modified backward/forward sweep-based method for reconfiguration of unbalanced distribution networks," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 1, pp. 85-101, 2019.
[20] M. Shirazipour *et al*., "Realizing packet optical integration with sdn and openflow 1.1 extensions," *2012 IEEE International Conference on Communications (ICC)*, Ottawa, ON, Canada, 2012, pp. 6633-6637.
[21] M. Channegowda *et al*., "Experimental evaluation of extended OpenFlow deployment for high-performance optical networks," *2012 38th European Conference and Exhibition on Optical Communications*, Amsterdam, Netherlands, 2012, pp. 1-3.
[22] N. Handigol, "Where is the debugger for my software-defined network," *Proceedings of the 1st HotSDN Workshop*, 2012, pp. 55-60.
[23] Canini, Marco, and Dejan Kostic, "Systematic Software Testing Meets Networking," *Conference Open Networking Summit, Research Track (ONS)*, 2013, pp. 1-2.
[24] M. K. Shin *et al*., "VeriSDN: Formal verification for softwaredefined networking (SDN)," *Telecommunication Review*, pp. 1-2, 2013.
[25] B. Lantz *et al*., "A network in a laptop: rapid prototyping for software-defined networks," *Proceedings of the 10th ACM Workshop on Hot Topics in Networks. HotNets 2010*, Monterey, CA, USA, pp. 19-26, 2010.
[26] P. Guimaraes *et al*., "Experimenting content-centric networks in the future internet testbed environment," *2013 IEEE International Conference on Communications Workshops (ICC)*, Budapest, Hungary, 2013, pp. 1383-1387.
[27] B. B. Bezabeh and A. D. Mengistu, "The effects of multiple layers feed-forward neural network transfer function in digital based Ethiopian soil classification and moisture prediction," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 4, pp. 4073-4079, 2020.
[28] H. Attia, "Artificial neural network-based unity power factor corrector for single phase DC-DC converters," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 4, pp. 4145-4154, 2020.