Optimal controller selection in Software Defined Network using a Greedy-SA algorithm

Kshira Sagar Sahoo, Bibhudatta Sahoo, Ratnakar Dash, Nachiketa Jena
Department of Computer Science and Engineering.
National Institute of Technology
Rourkela, India, 769008
{kshirasagar12@gmail.com, bibhudatta.sahoo@gmail.com, ratnakar@nitrkl.ac.in, nachiketa.jena@gmail.com}

***Abstract* – Software Defined Network is one of the most recent Internet technology that manages the large scale network. Both from implementation and performance point of view SDN will improve the next generation networking services. It is important to find a solution to the controller placement problem is remaining a key issue in SDN based architecture. It decides where to place the controllers with a limited amount of resources within the network. This paper illustrates a preliminary work on the controller placement in SDN environment using an existing heuristic technique. More formally, a network is given by a set of elements (either switches or routers) they must be managed by the controller(s), the algorithm finds the number controller(s) require to cover all the network elements within the network in a optimal way. The primary criteria is the distance between all nodes and selected controllers is minimized. Controller's capacity is a constraint of the controller, that restricts a controller to manage an unlimited number of data plane devices. We have proposed and implemented simulated annealing algorithm with greedy heuristic to solve the controller placement problem.**

Keywords –SDN, Controller placement, Greedy-SA

NOMENCLATURE

Simulated Annealing (SA), Software Defined Network (SDN),

Optimal Controller Placement Problem (OCLP)

I. INTRODUCTION

Software Defined Network (SDN) is one of the new revolutions in the networking field that makes the behavior of the network of the network devices (such as router /switches) programmable and allows them to be controlled by a central element, thus offering advanced customizability of network control and forwarding behavior [1][12]. This new paradigm has created the interest from both industry and academia since last couple of years. It is an approach to computer networking in which network control is decoupled from the hardware and given to a software application called a software controller. In other words, this architecture decouples the control function and forwarding function and enables the network control to become directly programmable and the underlying infrastructure to be abstracted from applications.

The key elements of SDN incorporate isolated control and information planes, intelligently unified system controller, programmability of the control plane, and standard application programming interfaces (APIs). SDN is relied upon to significantly affect future systems administration through empowering an open programmable network platform that gives awesome flexibility to supporting different applications. Partition of control and data planes and unified controller in SDN additionally offer a promising way to facilitating inter-domain end-to-end QoS provisioning in the future Internet. The split of data and control planes permits making the forwarding (data) platform more simple, and brings system's intelligence into various controllers that manage the switches. Because of a few reasons arrangement of SDN in large system may divide into smaller regions. These reasons might require protection, adaptability, security, etc. [2]. Every small SDN region keeps running by one or more controllers, for example, NOX, Floodlight, POX and so forth. Placing of the controller(s) in a network affect the performance and the expense of the system, whether the SDN environment is having single or multiple controllers. So we have discussed a complete model that will locate the best position of the controller(s) and interconnect the nodes for better execution with least cost [3].

The controller placement problem in SDN scenario was discussed in [10]. In this paper optimization is carried out regarding latency from assigned controller to switch. This optimization problem is an NP-hard problem.

The objective of this article is to minimize the cost of the SDN in terms of minimizing the global latency while placing a controller by considering the distance of the node to the controller as a constraint.

The remainder of the paper is organized as follows. Section II, discussed on various works done on the controller placement problem so far. The problem statement and proposed model are introduced in section III, followed by a greedy simulated annealing algorithm for the controller section in Section IV and simulation results is also discussed in this section. The conclusion and future work are presented in Section V.

II. RELATED WORK

To set up an SDN environment requires proper planning. Various papers have already been dealing with this issue. They have tackled the problem in different aspect compared to different parameter. In [4], the authors have addressed the issues of the controller placement. The authors in [14], have used k-critical algorithm to solve the placement problem where the used metric is the average latency between the network devices.

The proposed mathematical formulation of the said problem, determines the optimal number of controllers. More constraints like cost of the link, cost of equipments, latency of path setup, and pattern of the network traffic will be considered as the future work of this paper.

III. PROBLEM STATEMEMNT

Many algorithms have proposed earlier on Optimal Controller Placement Problem (OCLP) in a wireless network, which was an important problem in designing the cellular network. Our proposed method somehow mimics the said problem discussed in [8], [5]. Considering M nodes in an SDN environment, a set of N controllers must be placed in order to manage the network traffic. The constraint that is N<M must be followed while considering the placement problem. Before discussing the following information are required to know to formulate the mathematical model.

• The location of all the network elements in the network must know along with from each switch how much traffic goes to the controller.

• The behavior of various types of controllers.

• The length and the bandwidth available from each switch and the controllers.

A. Notations:

For a given network represented in a graph *G(V, E)*, where V is the set of network elements, E the set of available links. Let n be the number of nodes where n=|V|. Let k denotes the number of controllers to be placed in the network.

$S = \{s_1, s_2, .. s_n\}$ and $S \subset V$,set of switches present in the data plane. $C = \{c_1, c_2, ..., c_n\}$ and $C \subset V$ set of controller in the control plane. $L = \{l_1, l_2, .. l_n\}$ and $L \subset V$, set of possible links between the controllers and switch. *P*, set of possible locations for the controller.

α^c , is the price for the controller of type $c \varepsilon C$.

β^c , is the available controller of type $c \varepsilon C$. x_{cp} , is a binary variable. x_{cp} =1 if controller of type $c \varepsilon C$ at position $p \varepsilon P$

The possible controller to controller paths denoted by n_c . Let $E(c)$ denote the forwarding devices controlled by the controller c and $L(c)$ is the capacity of the controller *c*.

B. Proposed Model:

Minimization of the global latency to manage the SDN enable network by choosing a suitable place for the controller is the objective of this paper. It might include $C_c(x)$ cost to setup the controller. $C_l(v)$ the cost related to connecting the controllers to the switches and the cost related to inter controller connection i.e. $C_t(z)$.

$$C_c(x) = \Sigma \alpha^c \Sigma x_{cp} \qquad (1)$$

$$C_v(x) = \sum_{s \varepsilon S} \sum_{p \varepsilon P} dist(s, c) \qquad (2)$$

$dist(s, c)$, is the Euclidean distance between the switch (s) and its corresponding controller (c).

The controller placement problem can be modeled such that the following cost function must satisfy:

$$min(C_c(x) + C_l(x) + C_t(x)) \qquad (3)$$

subject to constraint

$$\sum_{c \varepsilon C} x_{cp} \leq 1$$

But prior to minimizing the above cost function we are minimizing the global latency of the network considering the distance between switch and controller as a parameter which has given in the equation 4. Available ports of a controller should be less than the connected switches.

$$G(C) = \frac{1}{n} \sum_{v \in V} \min_{c \in C} d(s, c) + \frac{1}{n_c} \sum_{c_i, c_j \in C} d(c_i, c_j) \quad (4)$$

Global Latency of the SDN controller must be minimize i.e.

$$\min G(c) \qquad (5)$$

In this article we have done half of the work i.e we have implemented and tested a greedy-simulated annealing algorithm to solve the said problem on different topologies and tried to minimize the global latency and comparing its results with the K-median algorithm [7].

IV. A GREDDY-SA ALGORITHM

A. SA Algorithm

SA has been used to solve many optimization problems for a long time [6]. Annealing method is a physical process where metals are gradually cooled to reach a stage to become a strong one. SA is an analogous method used for optimization problem; by approximating the global optimum of a given function. This probabilistic technique is a process where the temperature is reduced slowly, starting from a random search at high temperature.

Initial temperature lowering down to a moderate stage until the system comes to a balance point, where no more changes required. In each stage changing has happened several times, until reaches to a thermal equilibrium point. Next stage begin with a lower temperature. SA keep the current assignment values to variables. In the subsequent step it take a variable along with value.

The Standard SA algorithm follows the below procedure for the outcome: A new arrangement is formed by the random movement of the present one. If new arrangement is better than the previous one immediately replaces the present one, otherwise it might replace the present one probabilistically. This possibility of replacement is high at the beginning of the algorithm, and reduces at each stage. This probabilistic technique allows the framework to move toward the best solution. Despite the fact that SA is not ensured to discover the global optima, it is still better from others algorithm in getting away from local optima.

In this paper, we have applied the SA technique, considering N number of nodes which will be considered as controllers for the system. The encoding of the designs is accomplished by a method for twofold strings, in such a way that number of 1's in the string implies that the number of controller has been chosen, whereas a 0 implies that the corresponding network element is not a controller, but rather serve as a switch. Because we must choose N nodes to be the controllers, SA looks for the strings with exactly N 1s on them. Standard SA can't deal with the fixed set of 1s from the string, for this an additional operator (op) must be added to the standard SA. The additional operation will act like this.

$$op(x)=\begin{cases} \textit{if } (k<N) \textit{ then, place } (N-k) \textit{ number of 1's randomly} \\ \textit{otherwise remove } (k-N) \textit{ number of 1's randomly} \end{cases}$$

After the random process in the SA, the individual string z will have k number of 1s which might be different from the actual number of 1s in z i.e N. If $k < N$ the search operator adds $(N-k)$ number of 1s in random positions. In other sense, if $k > N$, the restricted search operator randomly selects $(k-N)$ 1s and removes them from the binary string. With this method it is guaranteed that all the binary strings managed by the SA has exactly N . This method is called restricted search.

Table 1: Notation used in the algorithm

Notation	Description
T_0	Initial temperature
$s(T)$	Binary string
$Maximum\_temp\_changes$	maximum temperature changes
$Max\_mutations$	maximum number of mutation in each iteration
C_{ij}	distance from a node i to the closest controller j

The pseudo-code of the standard SA algorithm is described here. The initial temperature of the system T_0 is chosen in such a way that the initial probability of acceptance worse solutions is 0.8, a standard value for the SA. This probability value decreases with the temperature of the system. Objective function decides the states of $S(T)$ and $S_{mut}(T)$, will be better or worse than the other. We use a Greedy algorithm to obtain the objective function in equation 2.

Algorithm 1. ***Standard simulated Annealing Algorithm***

```
Set initial temperature of the system $T_0$ which is very high.
Generate initial state $S(T_0)$ randomly
$i \leftarrow 0$
for $i = Maximum\_Temp\_Change$ do
    $T_i = \frac{T_0}{i+1}$
    for Maximum_Mutation do
    Apply mutation to the current state $S(T_i)$
    find fitness value of the current state ( $S(T_i)$ )
      if $S_{mut}(T_i)$ is better than $S(T_i)$ then
        $S(T_i) = S_{mut}(T_i)$
      else
      Generate a random number $u$ , $u \varepsilon [0,1]$
       endif
      if $e^{\Delta E/T_i} > u$ then
        $S(T_i) = S_{mut}(T_i)$
      else
        Discard mutated Solution $S_{mut}(T_i)$
    endif
  end for
  $i = i+1$
end for
```

B. GREEDY ALGORITHM

Greedy algorithm always takes the best immediate, or local solution while finding an answer. In [8] author had used a Greedy method for tackling terminal assignment (TA) problem. We use this method for solving the controller placement problem. This approach begins from a random permutation of $\pi(K_{M-N})$. Then, the cost function $C_l \leftarrow C_{ij}$ is the Euclidean distance between switch i and controller j is calculate and the nodes are allocated to the closest controller. If the controller do not have the capacity to handle the node, the algorithm searches for the following nearest controller and carry out the same operation. This process will continue until an controller found and allocate rest of the node to the controller. The greedy solution provide by the algorithm fails when no controller can accommodate the required capacity, that case is considered as the worse case.

Algorithm 2. ***Greedy Heuristic Algorithm for controller selection***

```
for each Binary string z of SA
Choose N number of 1 as controllers
Choose $M-N$ number of 0 as the nodes
Select a permutation $\pi(K_{M-N})$ at random
for for every terminal $\pi(K_i)$ do
        Determine $C_{ij}$ = distance from $\pi(K_i)$ to the closest
controller $c_j$
            $c_j \leftarrow \pi(K_i)$
            $C_l \leftarrow C_{ij}$
            Find $F(z) = \Sigma C_l$
end for
```

V. RESULT AND FUTURE SCOPE

In order to test our approach, we have taken several instances of network topology. To generate the network, we have used Gephi software as it is described in [9]. The load of the controller, which is generated by the switch has been traced. The proposed algorithm is written in python version 2.7 and execute the programs on a machine equipped with Intel Core i3 4-Core processors and 8 GB RAM. We have the average solution by executing the algorithms for 50 times on every randomly generated topology.

For evaluation and measure the performance of the proposed algorithm, we run two other algorithms simultaneously, for comparison purpose. The two algorithms are integer linear programming algorithm (ILP) in [13] and random placement algorithm respectively. For solving ILP we have used IBM ILOG CPLEX. Most of the time the linear programming algorithms are considered as the optimal solution for reference. So in our simulation, we mainly consider the ILP algorithm as a reference. We execute all three algorithms on various topologies, based on the selected locations, for 50 times, and select the placement that produces the best performance. In the simulation we characterize the latency and computing time against the number of controllers.

Figure 1 shows the result of average latency while the number of the controllers are gradually increasing under the same topology in each strategy. The result shows that the average delay gained by Greedy –SA algorithm is always relatively stable compared to other two.

Figure 2 assesses computational time with increasing controllers in a given topology while the size of the network is the same. It has seen that, deploying more number of controllers, the convergence of time becomes larger. When the number of controllers increases the growth rate of computation also increase.

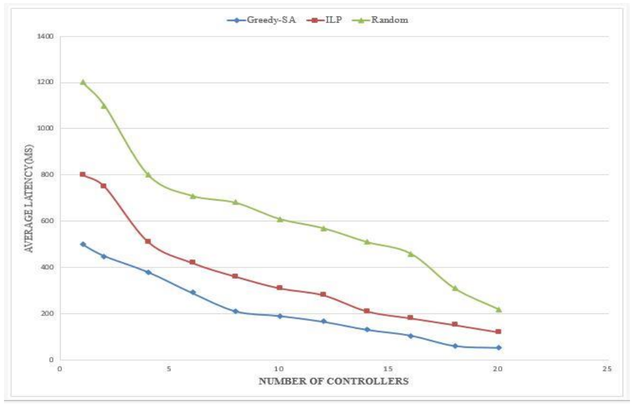

Figure 1: Number of controllers and impact on average latency

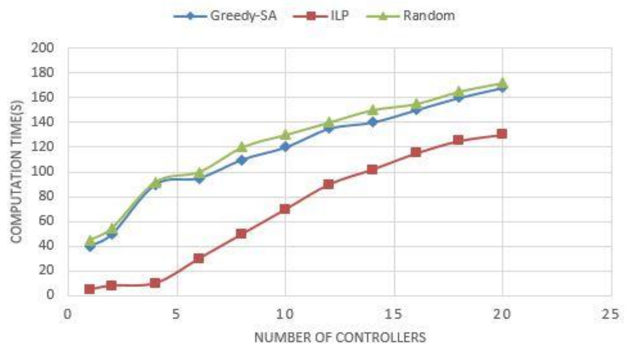

Figure 2: Number of controllers and impact on computation time

VI. CONCLUSION AND FUTURE WORK

Designing the control plane is a major challenge in SDN based architecture. Though the total number of controllers in the control plane has been known before, but their positions have a major impact on many real time issues. We have addressed the issue and presented a mathematical model this problem. Brute force approach to this problem is practically feasible for small and medium size network, an alternative mechanism is required for large instance of networks. Usually heuristic approach involves to meet the time and resource demand.

This paper must be considered as the first step towards solving controller placement problem where distance is the metric. However, in the future work we will consider it as a multi objective problem where latency and load distribution to be considered as different metrics. In future work we will consider the arrival and execution of real time task as aother important metric for the said problem [13] and implement the cost function on a real time environment.

.

REFERENCES

[1] Hu, Fei, Qi Hao, and Ke Bao. "A survey on software-defined network and openflow: from concept to implementation." Communications Surveys & Tutorials, IEEE 16.4 (2014): 2181-2206.

[2] 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,J. China Univ. Posts Telecommun., vol. 19, no. S2, pp. 92171, Oct. 2012.

[3] Sallahi, Afrim, and Marc St-Hilaire. "Optimal Model for the Controller Placement Problem in Software Defined Networks." Communications Letters, IEEE 19.1 (2015): 30-33.

[4] B. Heller, R. Sherwood, and N. McKeown, â€œThe controller placement problem,in Proc. 1st Workshop Hot Topics Softw. Defined Netw., 2012, pp. 712.

[5] Salcedo-Sanz, S., et al. "A Hybrid Greedy-Simulated Annealing algorithm for the optimal location of controllers in wireless networks." Proceedings of the 5th WSEAS Int. Conf. on Artificial Intelligence, Knowledge Engineering and Databases, Madrid, Spain. 2006.

[6] Kirkpatrick, Scott, C. Daniel Gelatt, and Mario P. Vecchi. "Optimization by simulated annealing." science 220.4598 (1983): 671-680.

[7] Cervello-Pastor, Cristina, and Aurelio J. Garcia. "Defining a network management architecture." Network Protocols (ICNP), 2013 21st IEEE International Conference on. IEEE, 2013.

[8] Abuali, Faris N., Dale A. Schoenefeld, and Roger L. Wainwright. "Terminal assignment in a communications network using genetic algorithms." Proceedings of the 22nd annual ACM computer science conference on Scaling up: meeting the challenge of complexity in real-world computing applications: meeting the challenge of complexity in real-world computing applications. ACM, 1994.

[9] M. Bastian, S. Heymann, and M. Jacomy Gephi: an open source software for exploring and manipulating networks,Proceedings of International AAAI Conference on Weblogs and Social Media (ICWSM).,pp. 361-362. California, USA, 2009

[10] Heller, Brandon, Rob Sherwood, and Nick McKeown. "The controller placement problem." Proceedings of the first workshop on Hot topics in software defined networks. ACM, 2012.

[11] Cervello-Pastor, Cristina, and Aurelio J. Garcia. "On the controller placement for designing a distributed SDN control layer." *Networking Conference, 2014 IFIP*. IEEE, 2014.

[12] Sahoo Kshira Sagar, Sahoo Bibhudatta , and Panda Abinas. "A secured SDN framework for IoT." *Man and Machine Interface Conference*, IEEE, 2015.

[13] Sahoo, Sampa et la. "Execution of real time task on cloud environment." 12th *IEEE India International Conference (INDICON 2015).*

[14] Yao, Guang, et al. "On the capacitated controller placement problem in software defined networks." *Communications Letters, IEEE* 18.8 (2014): 1339-1342.