International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 5
DOI: https://doi.org/10.17762/ijritcc.v11i5.6609
***Article Received**: 22 February 2023 **Revised**: 24 March 2023 **Accepted**: 18 April 2023*

Controller Placement in Vehicular Networks: A Novel Algorithm Utilizing Elite Opposition-Based Salp Swarm and an Adaptable Approach

Sanjai Pathak[1], Ashish Mani[2], Mayank Sharma[3], Amlan Chatterjee[4]

[1]Amity University Uttar Pradesh Noida, India
e-mail: pathak.sanjai@gmail.com
[2]Amity University Uttar Pradesh Noida, India
e-mail: amani@amity.edu
[3]Amity University Uttar Pradesh Noida, India
e-mail: msharma22@amity.edu
[4]California State University, Dominguez Hills, USA
e-mail: achatterjee@csudh.edu

Abstract -The rapid advancement of networking technology has enabled small devices to have communication capabilities, but the current decentralized communication system is not ideal for heterogeneous networks like vehicular networks. The integration of routing, switching, and decision-making capabilities in the same network device limits innovation and impedes performance in decentralized networks, especially in vehicular networks where network topologies change frequently. To address the demands of such networks, Software-Defined Networking (SDN) provides a promising solution that supports innovation. However, SDN's single-controller-based system may restrict the network's operational capabilities, despite being programmable and flexible. This paper suggests two methods to tackle the complex problem of controller placement in SDN: an adaptable approach based on OpenFlow protocol in OpenNet and an evolutionary algorithm called Elite Opposition-Based Salp Swarm Algorithm (EO-SSA) to minimize propagation latency, load imbalance, and network resilience. Multiple controllers increase the network's capabilities and provide fault tolerance, but their placement requires a trade-off among various objectives. The proposed methods have been evaluated and analyzed to confirm their effectiveness. The current decentralized network system is not adequate for vehicular networks, and SDN offers a promising solution that supports innovation and can meet the current demands of such networks.

Keywords - Salp Swarm Algorithm, Adaptive Approach, VANETs, Software Defined Network.

I. INTRODUCTION

Vehicular Networking (VANETs) refers to using wireless communication technologies to create a network between vehicles, infrastructure, and other devices in the transportation system. In VANETs, each device in the network can perform routing, decision-making, and forwarding functions, which can help improve road safety, traffic efficiency, and driver experience. However, this decentralized nature of VANETs poses challenges in meeting the network requirements of heterogeneous networks, especially in vehicular environments where the network topology changes constantly and communication demands are high. To address these challenges, programmable networking, such as Software-Defined Networking (SDN), is a promising solution providing more flexible and efficient network management. The placement of controllers is a crucial problem in SDN for VANETs that involves balancing multiple objectives such as propagation latency, network resilience, and load distribution. Swarm intelligence algorithms, such as the Elite-opposition based Salp Swarm Algorithm (EO-SSA), can solve this problem effectively. Vehicular Ad-hoc Networks (VANETs) provide a solution by relying on neighboring devices to act as forwarding elements and maintain network connectivity, despite their limited transmission range. However, VANETs face challenges such as load imbalance due to their distributed nature. The deployment of ad-hoc networks can be complex and challenging due to their decentralization, dynamism, variability, connectivity, reliability, and individual routing behavior. A distributed control plane, as shown in Fig.1 [1], can be implemented to address these challenges.

The architecture of a Software-defined network (SDN) offers agility, flexibility, and innovation in communication technology. It provides dynamic configurations that adapt to the changing needs of computation networks and enables vehicular networks to cope with their dynamic nature. SDN enhances networking capabilities, including resource management and multiple network applications, and extends the networking framework

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 5
DOI: https://doi.org/10.17762/ijritcc.v11i5.6609
***Article Received**: 22 February 2023 **Revised**: 24 March 2023 **Accepted**: 18 April 2023*

with programmable capabilities to address the various challenges of vehicular networks. SDN comprises three components: application plane, data plane, and control plane, as shown in Fig.2. The SDN controller communicates with OpenFlow-enabled switches using the southbound interface to make routing and switching decisions. Controller placement is a crucial task in SDN and involves identifying controllers' optimal number and location to achieve optimal performance. In vehicular networks, multiple controllers are needed, and each controller's location affects various network performance parameters [3,8].

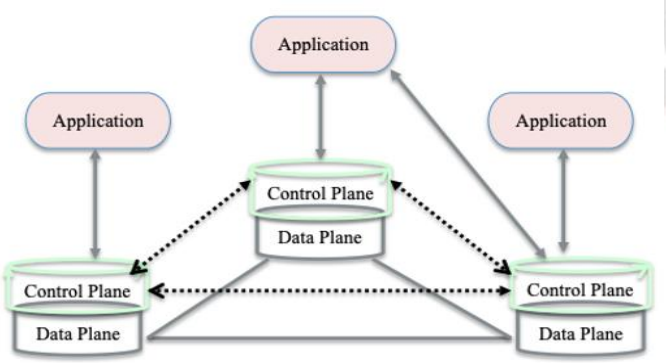

Figure 1. Existing Network Architecture.

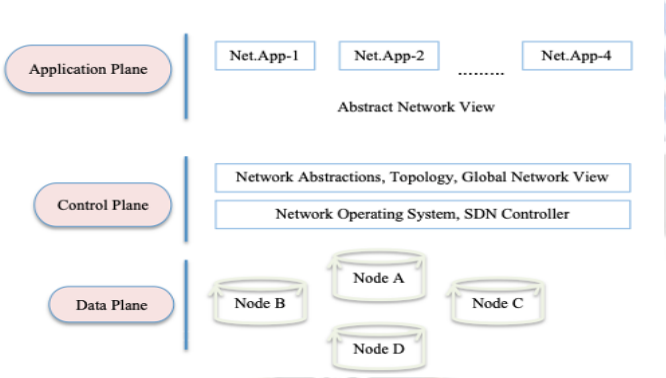

Figure 2. The logical view of SDN.

Heller et al. first introduced the controller placement problem (CPP) in Wide Area Networks (WANs) to mitigate the transmission delay between SDN controllers and their associated switches [2]. Their research focused on optimizing propagation latency using an evolutionary algorithm. The placement of SDN controllers is akin to the facility location problem, known as NP-hard. The process of developing an algorithm that can produce all possible solutions for controller placement is arduous and time-consuming. It is because the search space is extensive, and multiple search agents must be considered when optimizing the algorithm under the constraints. In the case of CPP, to place k SDN controllers for n network switches, all the feasible combinations could be $C(r,k)$ where $k < r$. For example, to locate the position of 7 controllers in a 70 nodes network topology, a total of 1.19877472 E+9 feasible controller placements to obtain an optimal placement. An evolutionary algorithm can be a viable solution for efficiently tackling these types of cases. By traversing a smaller subspace of the entire search space, this algorithm can produce solutions close to the optimal solution [4].

Based on a comprehensive literature review and an evaluation of performance and resilience metrics, it has been established that there is no one-size-fits-all solution for controller placement that would be ideal for all network requirements. However, based on the Topology and conditions of the network, a trade-off between these metrics could be deemed acceptable [5]. Due to the dynamic nature of vehicular networking, an effective system must be able to manage the addition and removal of controllers from the network. Researchers must consider various issues during the controller placement process, such as the limited number of controllers, flow request distributions, and placement locations, to ensure optimal network performance. This paper introduces a flexible method for placing controllers in vehicular networks, where the positioning of controllers is continuously reassessed to meet the communication requirements of the network and ensure low latency. The proposed method was practically implemented on OpenNet with a POX controller, using a comprehensive approach to controller placement. However, given the rapid pace of change in the vehicular networking environment, there is a need for a faster process. As mentioned, an evolutionary algorithm is a more efficient alternative for handling such cases. Further research into the frequency and severity of changes in the vehicular networking environment could be interesting, formulating the controller placement problem as a dynamic optimization problem and solving it using an evolutionary method with an analysis of the resulting outcomes.

Salp Swarm Algorithm (SSA) is a meta-heuristic optimization algorithm introduced by Mirjalili et al. in 2017. It is inspired by the swarming behaviour of Salp in the ocean. The algorithm is based on the idea of emulating the social behaviour of Salp to find the optimal solution for a given problem. The algorithm has been used for solving various optimization problems and has shown promising results. The algorithm mimics the behaviour of Salp swarms in which each Salp follows a set of rules for searching food and reproducing. The algorithm uses these rules to find the optimal solution for a given problem [6]. Like any other meta-heuristic algorithm, in many ways, SSA works intelligently to avoid entrapment into local optima for several real-world applications due to the tendency to create a Salp chain during the optimization process [4]. The complexity of real-world applications, such as controller placement, can make it challenging to find the global optimum using standard meta-heuristic algorithms like SSA. Optimization must account for multiple framework elements, including latency, load

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 5
DOI: https://doi.org/10.17762/ijritcc.v11i5.6609
***Article Received**: 22 February 2023 **Revised**: 24 March 2023 **Accepted**: 18 April 2023*

balancing, resilience, and deployment cost, which can further limit SSA's optimization capability. This issue is mainly due to the actual entrapment behavior of SSA in local optima and its poor searchability. To overcome these limitations, we introduce EO-SSA, a new optimizer incorporating an EOL technique to enhance SSA's optimization capability for controller placement. This paper employs an Elite-opposition-based Learning (EOL) technique to improve SSA's exploration and exploitation propensity for the controller placement problem, known as a new optimizer EO-SSA [15]. The EOL technique is a recently developed method in computational intelligence that has demonstrated superior performance in solving optimization problems with a practical and cost-effective implementation. The EOL technique generates an initial solution by utilizing the current solution for the next generation, along with the initial control parameters of standard SSA such as the number of search agents and maximum generation [7]. The paper presents several significant contributions, which are outlined below:

- The paper reviews the challenges traditional vehicular networks face and provides a comprehensive literature review on integrating Software-Defined Networking (SDN) in vehicular networks. The study mainly focuses on how SDN manages the dynamic nature of Vehicular Ad hoc Networks (VANETs) and provides solutions to the associated challenges.
- The paper proposes an adaptable approach for controller placement in vehicular networking, considering the dynamic nature of VANETs and their varying traffic demands. Unlike fixed placement approaches, this alterable approach aims to continuously evaluate and adjust the placement of the controller to ensure optimal network performance based on current traffic conditions.
- The paper presents a sample network topology implemented in OpenNet using OpenFlow-enabled switches to demonstrate how the network's behavior changes by integrating an SDN-enabled controller. The paper considers different scenarios, such as the dynamic addition or deletion of OpenFlow-enabled switches or SDN controllers and the impact on the SDN controller's performance due to network expansion.
- The paper presents an enhanced version of the standard SSA algorithm, called EO-SSA, for solving a single-objective controller placement problem in vehicular networks. The objective is to minimize the latency between the SDN controller and associated nodes, a critical metric for ensuring efficient network performance. EO-SSA integrates the Elite-opposition-based Learning (EOL) technique to improve the algorithm's exploration and exploitation capabilities and achieve better global optimization. The proposed approach is evaluated using a set of experiments, demonstrating its effectiveness in optimizing the controller placement in vehicular networks.

The paper is structured as follows: Section 2 presents a review of related work. Section 3 proposes an alternative approach for solving the controller placement problem in vehicular networks using EO-SSA. Section 4 discusses the simulation results and computational analysis conducted in a network environment that was set up with OpenNet and the POX controller. Then, Section 5 concludes the paper.

II. Related Work

In software-defined networks (SDN), the SDN controller plays a crucial role in managing and controlling the network by performing low-level operations. The placement of SDN controllers in the network is challenging, and researchers have proposed different models to address the SDN scalability problem. These models consider various network metrics such as propagation latency, inter-controller latency, load imbalance, and network resiliency to optimize controller placement and improve network performance. Heller et al. [2] identified the controller placement problem that includes finding the location of controllers and the required number of controllers in respect of network topology. Wang et al. [9] proposed a novel architecture, i.e., software-defined internet-of vehicles (SDIV), by adopting the SDN approach to address the limitations of traditional networks and leveraging the separation mechanism of control and data plane.

Hock et al. [5] introduced a framework to evaluate the possible placements of controllers in a given network topology, known as Pareto-based Optimal Controller-Placement (POCO). After assessing multiple performance and resilience metrics, it has been discovered that no single optimal controller placement solution is suitable for all types of networks. Instead, a compromise between these metrics is required to achieve an optimal controller placement solution for a particular network. Lange et al. [10] proposed an extended version of the POCO framework to address the requirement for large-scale network infrastructures. They employed a heuristic approach for evaluating the framework, which is less accurate but requires less computation time. Bari et al. [11] experimented with large-scale WAN network deployment, where centralized network architecture reported many network performance and scalability issues. The study's primary goal was to dynamically select the controller's location and configuration in the network based on the changing network conditions. This objective aimed to minimize the communication overhead and flow setup time. Müller et al. [12] introduced a controller placement strategy called the "Survivor" technique, which aims to tackle network challenges by explicitly considering paths from different

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 5
DOI: https://doi.org/10.17762/ijritcc.v11i5.6609
***Article Received**: 22 February 2023 **Revised**: 24 March 2023 **Accepted**: 18 April 2023*

network ranges and fault tolerance mechanisms during network design.

Wang et al. [13] proposed a concept to partition the network to decrease the whole latency of the network, including queuing latency. The Clustering-based Network Partition Algorithm (CNPA) is a method introduced in the paper to partition the network into clusters, where a designated SDN controller manages each cluster. The algorithm aims to reduce the maximum end-to-end latency between the SDN controller, and OpenFlow switches in the network. By partitioning the network into clusters, the CNPA ensures that each cluster is optimized for its specific traffic and load characteristics, improving network performance.

TABLE I. SOFTWARE-DEFINED NETWORK VS. EXISTING NETWORK ARCHITECTURE

Sr. no.	Parameters	Software-Defined Network	Existing Network
1	Monitoring and controlling the network	More accessible due to the decoupled control plane	Complex due tightly coupled control plane and data plane
2	The global view of the network	Centralized view due to the controller	Distributed view
3	Network maintenance cost	Lessor	Higher
4	Required time to update/error or resolve issues	Relatively easy due to centralized network controller implementation	Difficult and time-consuming
5	Optimal utilization of the controller	Important	It's not relevant
6	Integrity and consistency	Important	Not important
7	State of network and forwarding tables	Important	Important
8	Accessibility of Controller	Important	It's not relevant
9	Optimal utilization of resources	High	Less

III. FOLLOWING APPROACHES PROPOSED FOR THE CONTROLLER PLACEMENT PROBLEM IN VANET'S

Designing a network for controller placement involves addressing the challenge of scalability, particularly in large-sized networks where a single controller may not be sufficient to perform optimally. The solution to this challenge often involves placing multiple controllers. To achieve this, designers typically seek answers to the following questions.

- What is the estimated number of controllers required for the network?
- Where should the controllers be placed?
- What is the maximum number of devices that can be efficiently connected to a controller?

In addition to the considerations mentioned above, the capacity of the controller is another crucial parameter to consider. The issue of capacity arises in two scenarios: (1) when the capacity of a controller is unlimited and, therefore, no constraint, which is known as the incapacitated controller placement problem (ICPP), and (2) when every controller has a limited capacity due to finite resources, which is known as the capacitated controller placement problem (CCPP). To ensure optimal performance of SDN, it is important to use an effective method to determine the best placement for the controller [20].

Table 1 provides a comparison between software-defined networks and traditional networks, highlighting relevant network parameters that should be considered when placing controllers. It is essential to conduct a thorough network analysis when performing controller placement. In 2012, Heller formulated the controller placement problem as an incapacitated CPP. It was an objective to minimize the latency between the OpenFlow switch and the SDN controller in both the average and worst cases as $k-median$ and $k-centre$, respectively. It highlights the importance of considering network metrics when designing the controller placement strategy.

The primary objective of solving the controller placement problem is to determine the appropriate number of controllers, denoted by k, needed to be placed in the network to meet the desired network performance parameters. The average latency between forwarding elements (S) and controller (C) can be measured and represented as $\pi^{avg-latency}$ in equation (1), which is known as the optimization problem of minimum $k-median$ [2].

$$\pi^{avg-latency}(\mathrm{C}) = 1/|S| \sum_{s \epsilon S} \min d(s,c) \quad \ldots (1)$$

Where $d(s,\mathrm{c})$ represents the shortest path from switches s to the associated controller $c \epsilon C_i$. The objective is to solve a single-

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 5
DOI: https://doi.org/10.17762/ijritcc.v11i5.6609
***Article Received**: 22 February 2023 **Revised**: 24 March 2023 **Accepted**: 18 April 2023*

objective controller placement problem using EO-SSA to realize the lowest latency mean.

A. An Alterable Method for The Controller Placement Problem

SDN controllers can be used to execute routing applications or broadcasting commands that can alter the behavior of the network. However, relying on a single controller to handle all network traffic can negatively impact performance. Determining the optimal placement of a single SDN controller within a network can be challenging as performance metrics must be considered alongside the network topology. This can be particularly difficult if the network topology contains multiple areas with varying traffic levels at different times of the day. For example, network traffic during the daytime may primarily consist of browsing for information, while at night, it may be used for backups and installations.

In a scenario where controllers are already implemented, if the network experiences growth, there is currently no way to expand it without changing the network architecture, which can be costly and time-consuming. While some research has focused on minimizing the cost of network expansion, other essential concerns remain unaddressed. These include the bandwidth of the connection between OpenFlow switches and controllers, the processing time for flow requests at the controller, and the latency between controllers. Addressing these concerns is crucial for effectively planning and deploying high-performing networks.

A reliability problem exists between the controller and OpenFlow switch in a network domain [14]. OpenFlow switches serve as terminal nodes in a network, while the controller acts as a source node. Both links and nodes in the network can be in either an active or inactive state. An active form signifies that the node or link is functioning correctly and is dependable, while a dormant state indicates that the node or link is not reliable and has failed. With the given setup, the problem involves determining the placement of k controllers in a network of n OpenFlow enabled switches, where k is less than or equal to n. The problem is formulated based on the graph $G\ (V, E)$ of the network, where V represents the set of nodes, k is the number of controllers to be placed, and E is the set of edges representing the connection link between nodes.

Further, we consider that there may be multiple paths between nodes $u \epsilon V$ and $v \epsilon V$, and we need to determine the shortest path, denoted as $d(u, v)$. The physical link between node u and node v can be represented as (u, v). We can represent the physical link between node u and node v as (u, v). To minimize network latency at the SDN controller, we opted for connecting each switch to its closest controller through the shortest path mechanism of the graph. Suppose there are m paths in a given network, and for each physical element $e \epsilon (V \cup E)$, the failure probability of element e is defined as P_e. Without loss of generality, we assume that a failure probability of element e breaks the control path D_e. The path loss (δ), which represents the reduction in available paths due to failures, can be calculated as a percentage equation (2).

$$\delta = {}^{1}/_{m} \sum D_e P_e \qquad \dots (2)$$

where e ϵ (V $\cup$ E) and $m = n + k\,(k-1)/2$.

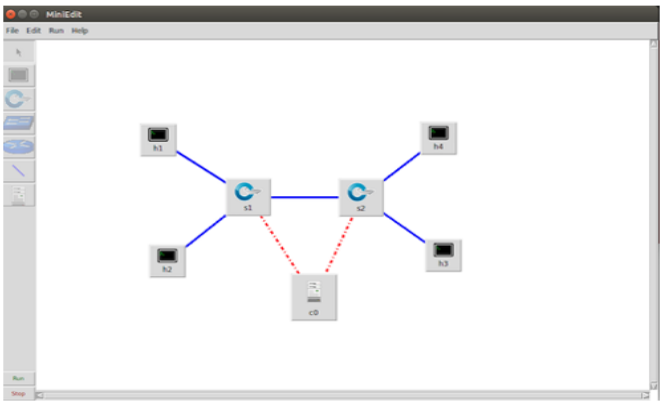

Figure 3. Basic Topology with 4 h, 2 s, and 1 c.

The effectiveness of the alterable approach of CPP, in which the place of the controller is not fixed and may change based on network dynamics, is confirmed with a sample network topology developed in OpenNet with the POX controller, as depicted in Fig. 3. The topology consists of four hosts and two OpenFlow switches with an SDN controller. In the simulation, h1 pings to the other hosts, h3, and h4, and the ping request is sent to the network forwarding device (OpenFlow switch). As there is no entry in the forwarding table of the switch for the 1st request on the respective switch, it is considered a table miss request for the controller. The controller will raise a command on receiving a request from the switch for the switch to update the flow table with the routing decision by the controller and the required action to be taken by the switch. The OpenFlow switch forwards the request to the appropriate destination, h3 and h4. Since there is a flow entry in the forwarding table, the OpenFlow switch can make routing decisions without contacting the SDN controller for consecutive similar requests at the OpenFlow switch.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 5
DOI: https://doi.org/10.17762/ijritcc.v11i5.6609
***Article Received**: 22 February 2023 **Revised**: 24 March 2023 **Accepted**: 18 April 2023*

Algorithm 1 Salp Swarm Algorithm Based on Elite-opposition Learning

```
procedure EO-SSA(Max_Iteration, Iteration, lb, ub, dim, fObj)
    Initialization: Search Agents, Algorithm Parameters, Optima, FitnessVal
    while Iteration ≤ Max_Iteration do                    ▷ Main Loop
        if Iteration ≥ 1 then
            Position update using the equation of standard SSA
            IntelligentAdjustment()
            Calculate Fitness
        else
            standardSSA() to build Salp chain
        end if
    end while
    Optima                                  ▷ Returns the Best Solution
end procedure
```

B. An Evolutionary Algorithm (EO-SSA) Method for The Controller Placement Problem

The SSA algorithm's ability to balance exploitation and exploration effectively for various real-world optimization problems makes it a compelling choice for controller placement problems. The SSA algorithm is efficient, straightforward, workable, easy to implement, and simple to deploy on complex optimization problems. To further enhance the optimization algorithm's performance in this study, an Elite-opposition-based Learning (EOL) approach was employed. The EOL method was applied to the EO-SSA algorithm to improve its overall exploration ability [15] quickly. In general, Algorithm 1 shows the execution of EO-SSA.

EO-SSA is based on the Salp Swarm Algorithm (SSA) and includes an intelligent adjustment step to increase the algorithm's exploration ability and decrease the likelihood of getting stuck in local optima [15]. The fitness evaluation of the population, which consists of the possible placements of controllers, is performed using equation (1) to minimize the latency between the network nodes and their associated controllers. The intelligent adjustment in the position of Salp is performed using equation (3) to improve the algorithm's search capabilities.

$$X^{i}_{j+1} = \left\{ c_j * \left((Ub_j - Lb_j) - X^{I}_{j} \right) \right. \quad \ldots (3)$$

where c_j is a uniformly distributed random number in [0, 1]. X^{i}_{j+1}, an elite solution is obtained through an intelligent adjustment with a nearby solution X^{I}_{j} in the search space. In the first iteration, the Salp chain is built using the equation proposed in [6]. In the subsequent iterations, the optimization process is continued using the EO-SSA method. This intelligent adjustment increases the exploration ability of the algorithm and reduces the probability of being trapped in local optima.

IV. Environment Setup And Results Analysis

The optimal placement of an SDN controller in a network eliminates the limitations of traditional networking, where network operators manually support it. Controllers provide a global view of the entire network, leading to effective resource utilization and better network management. To implement this approach, OpenNet, OpenFlow v.1.0, and POX as remote controllers have been used in the proposed alterable method. This section provides a step-by-step guide to the implementation process, including the necessary configuration setup and installation of required packages into the system, as outlined in [16]. To begin with, the following standard packages need to be installed using Linux command on Ubuntu v14.04 [17] [18]: git, python-dev, mercurial python-setuptools git, python-pygccxml, and python-urllib3.

A. Network Simulation Setup

The experiment's simulation was conducted on a computer with 16 GB of RAM, an Intel® Core™ i7-3520M CPU @ 2.90 GHz, and Ubuntu v.14.04 Linux development environment. The EO-SSA algorithm was implemented using MATLAB R2017b in this study, with the corresponding code developed for the alterable approach.

B. Simulation of Network Topology using OpenNet

To perform a native installation of OpenNet as a superuser using the source code, please log in to Ubuntu v14.04, open the terminal, and follow these steps:

- Download the OpenNet source code [18][19].
- Extract the source code and navigate to the extracted directory.
- Run the configure script to configure the installation.
- Run the make command to build the executable files.
- Run the make install command to install OpenNet.

To create the network topology for the experiment, use the MiniEdit tool, which can be found in the examples folder under the MiniNet source code repository. To launch the MiniEdit GUI, run the appropriate command in the terminal. Once the MiniEdit GUI appears on the screen, which is providing the necessary tools to develop a sample topology similar to the one depicted in Fig. 3.

```
sudo examples/miniedit.py
```

MiniEdit is a graphical user interface (GUI) tool with an integrated set of tool buttons on the left side of the interface. These buttons allow users to create the Topology by placing different components, such as hosts, OpenFlow switches, legacy switches, legacy routers, network links, and controllers, using the selector tool to adjust these components according to their requirements. The simulation can be started or stopped using the buttons provided on the MiniEdit GUI screen. The GUI provides a convenient way for SDN developers to design and configure network topology. It also makes it easy to debug and observe the simulated behavior of the network.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 5
DOI: https://doi.org/10.17762/ijritcc.v11i5.6609
***Article Received**: 22 February 2023 **Revised**: 24 March 2023 **Accepted**: 18 April 2023*

In this experiment, the SDN controller is configured as a remote controller supporting the OpenFlow v.1.0 protocol. The configuration is done using MiniEdit, as shown in Fig.4. To enable the command line interface (CLI), the preference window in MiniEdit needs to be configured accordingly. The controller (c0) must be set as a remote controller by selecting the properties window on the right-click menu. The emulated OpenFlow switched network in this simulation is configured to search for the SDN controller with the loopback IP address of 127.0.0.1 and the default OpenFlow port number of 6653 since the network hosts are set up in running mode. This allows the SDN controller and the emulated OpenFlow switched network to run on a single machine.

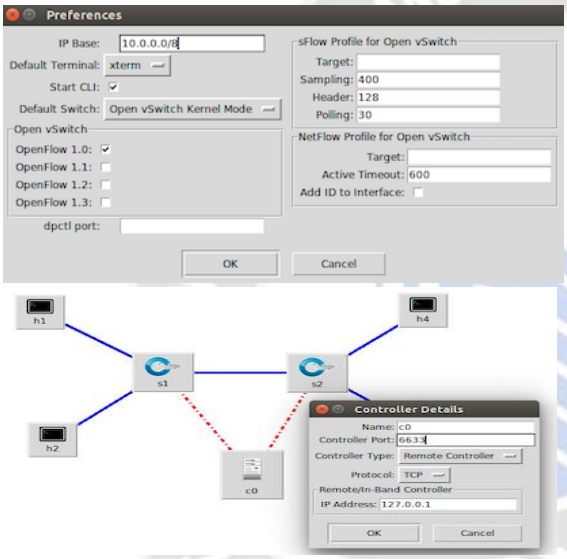

Figure 4. Preferences and Controller Configuration Basic.

To begin the simulation in MiniEdit, the user can click on the "Run" icon in the MiniEdit toolbar, which will display simulation data on the console window. However, if no SDN controller is listening, no traffic can be passed between the hosts, resulting in an "unreachable" message when using the ping utility. The user can run a specific command in the command window to enable POX as a remote controller and receive dump messages on the POX window.

sudo ~/pox/pox.py forwarding.l2_pairs info. packet_dump.

C. Examination of the Alterable Method

Based on the analysis of ping requests from h1 to h2, h3, and h4 in Fig.5, the following observations were made:

- In the absence of the POX controller, when ping requests were sent, none of the hosts were reachable, resulting in no communication response being observed.
- The initial packet of 64 bytes takes longer than subsequent packets when the POX controller is active and listening at port 6653. It is due to the installation of flow rules. When the ping operation is initiated from h3, the POX controller estimates the optimal path from h1 to h3 and installs flow rules into the forwarding tables of OpenFlow switches s2 and s1. However, subsequent requests do not require further communication with the POX controller.
- If the controller goes down, communication from h1 to h3 would still work because the OpenFlow switch has the flow entry for this request. However,
  the ping request from h1 to h2 would fail because the OpenFlow switch does not have the forwarding rule to direct the traffic from h1 to h2. In SDN architecture, the controller is responsible for installing the forwarding rules into the switches. Since the controller is down, it cannot install the necessary forwarding rule, resulting in the failure of the ping request.

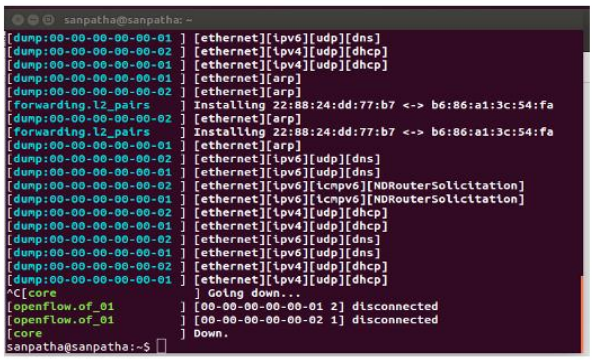

Figure 5. Controller Window

We carried out experiments to test various scenarios depicted in Fig.6:

- Traffic generation when the controller is not started yet.
- Starting the controller and listening to traffic on port 6653.
- Conducting ping operation from h1 to h3 with the controller up and analyzing the results.
- Running ping operation from h1 to h3 with the controller down and analyzing the impact.
- Conducting ping operation from h1 to h2 with the controller down and observing the results

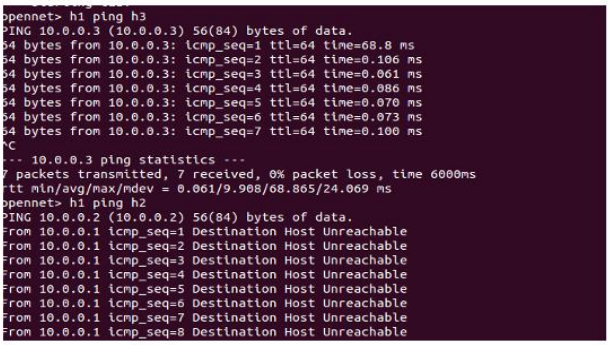

Figure 6. Ping h1 to h3, h2, and h4

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 5
DOI: https://doi.org/10.17762/ijritcc.v11i5.6609
***Article Received**: 22 February 2023 **Revised**: 24 March 2023 **Accepted**: 18 April 2023*

Based on the conducted experiments in various scenarios, the results reveal that the OpenFlow switch does not make any routing decisions without first consulting the SDN controller, even on the initial request. The SDN controller is responsible for determining the optimal path and updating the flow rules on the OpenFlow switch. In the suggested adaptable approach, the latency of each node is computed, and the node with the lowest latency is chosen as the ideal location for the controller. If the calculated latency is higher than the expected latency due to environmental changes in vehicular networking, the process is repeated to ensure that the network's anticipated latency is achieved.

TABLE II. FEASIBLE PLACEMENTS OF CONTROLLERS (K=4)

Run #	Possible Controller Placement (k = 4)	Latency (Best case)
1	[9 21 11 30]	0.29018
2	[12 33 25 30]	0.08699
3	[13 32 15 25]	0.13526
4	[2 24 20 6]	0.092294
5	[6 17 18 20]	0.10211
6	[25 6 18 2]	0.10109
7	[26 3 25 22]	0.16259
8	[28 12 21 23]	0.11234
9	[10 26 30 13]	0.17758
10	[2 12 22 5]	0.095963
11	[24 4 13 10]	0.17525
12	[31 16 26 24]	0.18177
13	[19 25 12 10]	0.16094
14	[3 30 34 1]	0.12703
15	[20 11 32 7]	0.10465
16	[20 11 32 7]	0.10465
17	[15 3 2 8]	0.092661
18	[13 32 15 25]	0.13526
19	[1 26 23 17]	0.089306
20	[32 31 13 23]	0.1192
21	**[29 5 13 10]**	**0.08699**
22	[13 26 25 24]	0.15203
23	[13 32 15 25]	0.13526
24	[34 1 33 17]	0.086439
25	[9 5 31 25]	0.11522
26	[31 26 29 28]	0.092625
27	[31 27 7 30]	0.15043
28	[16 23 33 13]	0.078515
29	[10 26 1 30]	0.12317
30	[13 31 5 19]	0.10181

D. An Evolutionary Algorithm (EO-SSA) Method for The Controller Placement Problem

To solve the controller placement problem, a combinatorial optimization approach is proposed in this paper, which utilizes an evolutionary algorithm called EO-SSA. The proposed approach is tested on the Internet2 OS3E network topology with 34 forwarding elements and 4 controllers that need to be positioned. The primary aim of the study is to determine the most efficient placement of controllers in the network, with the goal of minimizing latency between the nodes and their respective controllers. This objective is expressed mathematically in equation (1). This paper suggests that when designing network software, it is crucial to consider recommendations and research in the literature on controller placement strategies. Factors such as the controller's ability to handle real-time events and push actions in advance on connected switches should also be considered [2]. To address the optimal placement of k controllers in a network of $n - forwarding$ elements, this paper proposes using EO-SSA to solve this combinatorial optimization problem. Finding all possible placements of SDN controllers is an exhaustive and time-consuming process when considering the objective function.

Finding all possible placements of SDN controllers given a certain number of controllers to be positioned in the network can become computationally intensive. It can even exceed the available RAM capacity. This is due to the search space, which includes all possible combinations of placements subject to constraints. The number of search agents required for optimization algorithms can also become large, depending on the number of nodes n and the number of controllers k to be placed. A combinatorial relationship that can be used to find all possible numbers of placements is represented by Equation (4).

$$\binom{n}{k} = \frac{n!}{k!\,(n-k)!} \quad \ldots. (4)$$

Performing an exhaustive search for all possible placements becomes increasingly time-consuming and computationally demanding with an increase in the number of controllers (k), even for relatively small network sizes (n). Therefore, in a dynamic and flexible network, where adaptability to changes in the network environment is crucial, time becomes a limiting factor for such an approach.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 5
DOI: https://doi.org/10.17762/ijritcc.v11i5.6609
***Article Received**: 22 February 2023 **Revised**: 24 March 2023 **Accepted**: 18 April 2023*

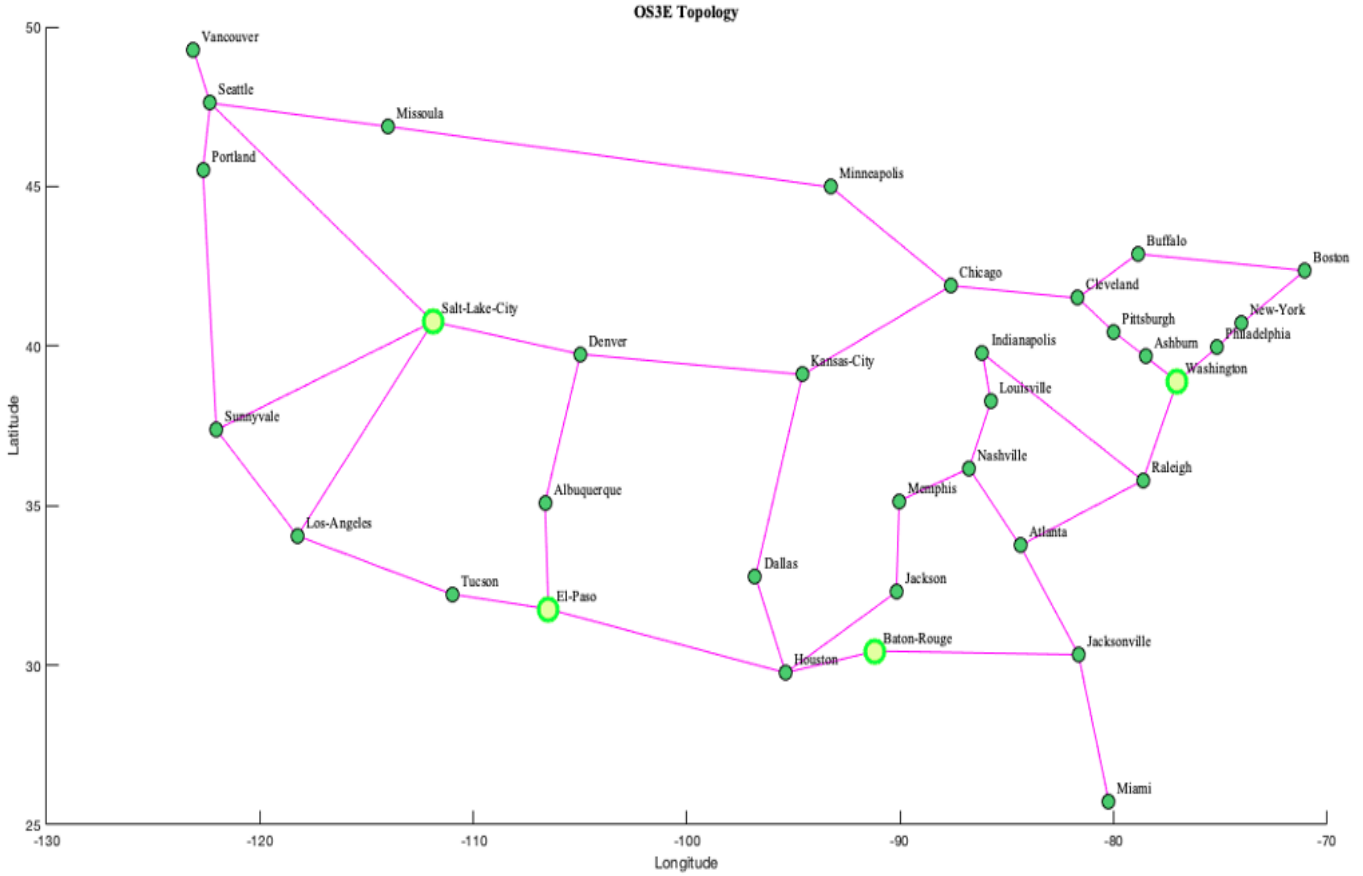

Figure 7. Internet2 OS3E Topology with Controller (k=4) Positioned [21]

The average latency of 30 independent runs was evaluated using the EO-SSA algorithm to analyze all possible placements of controllers in the Internet2 OS3E Topology, which has 34 network nodes and four controllers. Table 2 presents the results. The optimal placement of controllers, which was discovered in the 21st iteration, is shown as a green node in Fig. 7 and is underlined. This placement, which involves placing the controllers in Salt-Lake City, El-Paso, Baton-Rouge, and Washington, was determined through an exhaustive evaluation of all possible placements, which can be a time-consuming and computationally expensive task. However, the EO-SSA algorithm was able to efficiently search the space of controller placements and identify the optimal solution.

V. Conclusion

Finding the optimal number and placement of SDN controllers is crucial for the efficient functioning of software-defined networking. The algorithm used in this work, EO-SSA, provides a promising method for evaluating the possible placements of controllers in SDN. The primary goal of this study was to enhance fault tolerance in software-defined networks (SDNs) by taking into account network performance metrics such as flow setup latency, network latency between the OpenFlow switch and SDN controller, link bandwidth, connectivity between controllers, and path loss reduction. The experimental results obtained from the OpenNet sample network topology, implemented with the OpenFlow protocol and POX controller, demonstrated the SDN controller's adaptive behaviour. The OpenFlow switch first requested the installation of flow setup rules when the controller was ON. The network remained operational even if the controller was OFF for subsequent requests.

Overall, this work provides an alternative approach for controller placement in SDN and demonstrates the effectiveness of EO-SSA for evaluating all possible placements of controllers in a network topology.

That sounds like an excellent idea for future studies. Dynamic optimization would benefit a dynamic network environment such as vehicular networking, where network conditions and topologies can change rapidly. Additionally, other performance metrics besides latency could be optimized, such as network throughput or energy efficiency. The proposed EO-SSA algorithm could be adapted to consider multiple objectives, leading to a more comprehensive optimization solution. Overall, there is much potential for future research in this area. It will be interesting to see how the proposed approach evolves to address new challenges and performance metrics in SDN controller placement.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 5
DOI: https://doi.org/10.17762/ijritcc.v11i5.6609
***Article Received**: 22 February 2023 **Revised**: 24 March 2023 **Accepted**: 18 April 2023*

REFERENCES

[1] P. Ghosekar, G. Katkar, Dr P. Ghorpade, "Mobile Ad hoc Networking: Imperatives and Challenges," in International Journal of Computer Applications IJCA, vol. 1, pp. 153-158, 2010.

[2] Heller B, Sherwood R, McKeown N. "The controller placement problem, "in Pro-ceedings of the First Workshop on Hot Topics in Software Defined Networks, ACM, Jul.2012.

[3] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," in SIGCOMM Comput. Commun. Rev., vol. 38, no. 2, pp. 69–74, 2008.

[4] Abdelhamied A. Ateya, Ammar Muthanna, Anastasia Vybornova, Abeer D. Algarni, Abdelrahman Abuarqoub, Y. Koucheryavy, Andrey Koucheryavy, "Chaotic salp swarm algorithm for SDN multi-controller networks," in Engineering Science and Technology, an International Journal, Volume 22, Issue 4, Pages 1001-1012, Aug. 2019.

[5] D. Hock, M. Hartmann, S. Gebert, M. Jarschel, T. Zinner, and P. TranGia, "Pare-to-Optimal Resilient Controller Placement in SDN-based Core Networks," in ITC, Shanghai, China, Sep.2013.

[6] Seyedali Mirjalili, Amir H. Gandomi, Seyedeh Zahra Mirjalili, Shahrzad Saremi, Hossam Faris, Seyed Mohammad Mirjalili, "Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems," in Advances in Engineering Software, Volume 114, Pages 163-191, Dec. 2017.

[7] Andersen V , Nival P . A model of the population dynamics of salps in coastal waters of the Ligurian Sea. J Plankton Res 1986;8:1091–110.

[8] N. B. Truong, G. M. Lee, Y. Ghamri-Doudane, "Software-defined networking-based vehicular Adhoc Network with Fog Computing," in IFIP/IEEE International Sym-posium on Integrated Network Management (IM), pp. 1202-1207, May 2015.

[9] X. Wang, C. Wang, J. Zhang, M. Zhou, C. Jiang, "Improved Rule Installation for Real-Time Query Service in Software-Defined Internet of Vehicles," in IEEE Trans-actions on Intelligent Transportation Systems, vol. 18, no. 2, pp. 225-235, Feb. 2017.

[10] Lange S., Gebert S., Zinner T., Tran-Gia P., Hock D., Jarschel M., Hoffmann M., "Heuristic approaches to the controller placement problem in large scale SDN net-works," in, IEEE Transactions on Network and Service Management, 12(1), 4-17, Feb.2015.

[11] M. F. Bari et al., "Dynamic Controller Provisioning in Software Defined Net-works," in Proceedings of the 9th International Conference on Network and Service Management (CNSM 2013), Zurich, pp. 18-25, Oct.2013.

[12] 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 surviva-bility," in IEEE Global Communications Conference, Austin, TX, pp. 1909-1915, Dec.2014.

[13] G. Wang, Y. Zhao, J. Huang and Y. Wu, "An Effective Approach to Controller Placement in Software-Defined Wide Area Networks," in IEEE Transactions on Network and Service Management, vol. 15, no. 1, pp. 344-355, Mar. 2018.

[14] Liu, Jiang and Xie, Renchao, Reliability-based controller placement algorithm in software-defined networking," in Computer Science and Information Systems, pp. 547-560, Jun. 2016.

[15] S. Pathak, A. Mani, M. Sharma and A. Chatterjee, "A New Salp Swarm Algorithm for the Numerical Optimization Problems Based on An Elite Opposition-based Learning," in 2021 Asian Conference on Innovation in Technology (ASIANCON), 2021, pp. 1-6, doi: 10.1109/ASIANCON51346.2021.9544105.

[16] S. Pathak, A. Mani, A. Chatterjee and M. Sharma, "Software Defined Network Simulation Using OpenNet for Vehicular Network," in 3rd International Conference on Communication and Electronics Systems (ICCES), Coimbatore, India, pp. 170-175, Oct.2018.

[17] MiniNet Walkthrough for Linux (Ubuntu, Mint) [Online] http://mininet.org/walkthrough/

[18] OpenNet Source Code and Installation Information for Linux Ubuntu 14.04.5 [Online] Available: https://github.com/dlinknctu/OpenNet.

[19] M. C. Chan, C. Chen, J. X. Huang, T. Kuo, L. H. Yen and C. C. Tseng, "OpenNet: A simulator for software-defined wireless local area network," in IEEE Wireless Communications and Networking Conference (WCNC), Istanbul, pp. 3332-3336, Apr.2014.

[20] AK Singh, S Srivastava," A survey and classification of controller placement problem in SDN," in International Journal of Network Management, vol. 28, Mar.2018.

[21] L. Mamushiane, J. Mwangama and A. A. Lysko, "Given a SDN Topology, How Many Controllers are Needed and Where Should They Go?," 2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Ve-rona, Italy, 2018, pp. 1-6, doi: 10.1109/NFV-SDN.2018.8725710.