TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

THESIS NO.: 075MSICE007

CONTROLLER LOCALIZATION IN SOFTWARE-DEFINED NETWORK USING NAKED MOLE-RAT (NMR) ALGORITHM

BY

DIWOS KARKI

A THESIS SUBMITTED TO THE DEPARTMENT OF ELECTROICS AND COMPUTER ENGINEEING IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN INFORMATION AND COMMUNICATION ENGINEERING

DEPARTMENT OF ELECTROICS AND COMPUTER ENGINEEING

August, 2021

CONTROLLER LOCALIZATION IN SOFTWARE-DEFINED NETWORK USING NAKED MOLE-RAT (NMR) ALGORITHM

BY

Diwos Karki

075MSICE007

Thesis Supervisor

Dr. Babu R Dawadi

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Information and Communication Engineering

Department of Electronics and Computer Engineering

Institute of Engineering, Pulchowk Campus

Tribhuvan University

Lalitpur, Nepal

August, 2021

COPYRIGHT ©

DECLARATION

I declare that the work hereby submitted for Master of Science in Information and Communication Engineering (MSICE) at IOE, Pulchowk Campus entitled "**CONTROLLER LOCALIZATION IN SOFTWARE-DEFINED NETWORK USING NAKED MOLE-RAT (NMR) ALGORITHM**" is my own work and has not been previously submitted by me at any university for any academic award. I authorize IOE, Pulchowk Campus to lend this thesis to other institution or individuals for the purpose of scholarly research.

Diwos Karki
PUL075MSICE007
August, 2021

RECOMMENDATION

The undersigned certify that they have read and recommended to the Department of Electronics and Computer Engineering for acceptance, a thesis entitled **"CONTROLLER LOCALIZATION IN SOFTWARE-DEFINED NETWORK USING NAKED MOLE-RAT (NMR) ALGORITHM"**, submitted by **Diwos Karki** in partial fulfilment of the requirement for the award of the degree of "**Master of Science in Information and Communication Engineering**".

Supervisor: Dr. Babu R Dawadi
Asst. Professor,
Department of Electronics and Computer Engineering
Pulchowk Campus
Institute of Engineering, Tribhuvan University

External Examiner: Mr. Kumar Pudasaini

IT Governance Consultant

Committee Chairperson: Dr. Basanta Joshi
Asst. Professor,
Department of Electronics and Computer Engineering
Pulchowk Campus
Institute of Engineering, Tribhuvan University

Date of approval: August, 2021

DEPARTMENTAL ACCEPTANCE

The thesis entitled **"CONTROLLER LOCALIZATION IN SOFTWARE-DEFINED NETWORK USING NAKED MOLE-RAT (NMR) ALGORITHM",** submitted by **Diwos Karki** in partial fulfilment of the requirement for the award of the degree of "**Master of Information and Communication Engineering**" has been accepted as a bona fide record of work independently carried out by him in the department.

Dr. Ram Krishna Maharjan

Head of the Department
Department of Electronics and Computer Engineering,
Pulchowk Campus,

Institute of Engineering, Tribhuvan University, Nepal

ACKNOWLEDGEMENT

I would like to express my very deep gratitude and appreciation to **Dr. Babu R Dawadi** for his constructive suggestion during the planning and development of this thesis. His patience, devotion and guidance has immensely helped me get here today. It has been my great honor and privilege to work under his supervision.

I am sincerely thankful to our program coordinator **Dr. Basanta Joshi** for providing appropriate platform and concise guidance relating to this thesis. . I also express my sincere gratitude to **Prof. Dr. Shashidhar Ram Joshi**, **Dr. Surendra Shrestha, Dr. Nanda Bikram Adhikari and Dr. Aman Shakya** for providing me the valuable guidance and all feedback regarding entire thesis work.

Sincerely,

Diwos Karki

075MSICE007

ABSTRACT

Software-defined Network (SDN) is the novel network paradigm where decoupling of the control plane from data plane has its inherent advantages but introduces controller placement problem (CPP). CPP involves placing optimal number of the controllers while meeting prerequisites of conflicting nature such as latency, load balancing, and computational time. To achieve scalability, deployment of multiple controllers in large scale SDN is one of the key challenges. CPP can be addressed as a multi-objective combinatorial optimization problem whose solution is a trade-off between multiple optimization parameters. In this thesis, a novel population-based meta-heuristic algorithm, Naked-mole rat (NMR) has been proposed to optimize the controller placement based on switch-controller (SC) latency, controller-controller (CC) latency while maintaining load balancing among the controllers. The ideas and mechanisms are illustrated using two publically available topologies, viz. Savvis and Ernet, from Topology-zoo. The performance of NMR algorithm is compared with Bat algorithm and NMR algorithm slightly performed better.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

BP	Breeding Probability
BFO	Bacteria Foraging Optimization
CC	Controller-Controller
CPP	Controller Placement Problem
FFA	Firefly Algorithm
IP	Internet Protocol
MOCP	Multi Objective Controller Placement
PSO	Particle Swarm Optimization
SC	Switch-Controller
SI	Swarm Intelligence
SDN	Software-defined Networking
SDWAN	Software-defined Wide Area Network
VBO	Varna Based Optimization

CHAPTER 1: INTRODUCTION

1.1 Background

With the development and deployment of novel networking paradigm: software-defined Networking (SDN), has brought its own challenges [1]. Unlike the traditional IP networks where management is done via distributed control and network protocols, SDN offers the network manageability and flexibility by bifurcation of the control and data plane. This decoupling allows the centralized and programmable control plane architecture which bolsters simplified network management, improves network utilization efficiency and support network innovation. SDN transform the legacy switches into pure forwarding elements whose flow tables are populated by the controller.

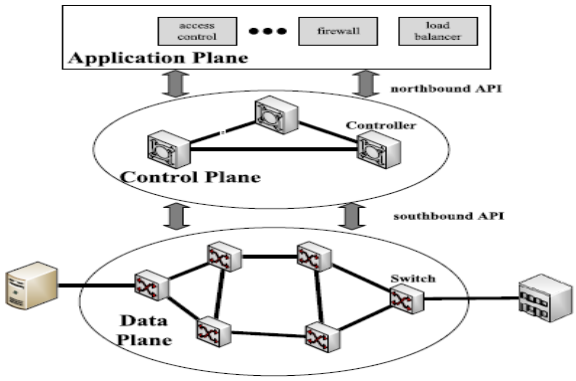


Figure 1.1: The basic architecture of SDN [1]

The southbound interface connects the controller with the data plane and OpenFlow [2] is the most widely used interface whose initial assumption is the use of only one controller in the network for the sake of simplicity. The expansion of SDN network has its own extensive management needs which cannot be fulfilled by the use of single controller. To avoid the single point of failure and enhance the reliability and the scalability [3], various network architectures with physically distributed and logically centralized multi-controller architecture have emerged, such as HyperFlow [4], Kandoo [5], Onix [6] and so on. As the number of controller and its location of deployment have significantly impacted the performance of the network in multi-controller network architecture, Controller Placement Problem (CPP) has been a research hotspot. Placement of the

controllers in order to maximize the network performance is known as CPP. It is a mathematical problem which is similar to well-known facility location problem, a non-deterministic polynomial (NP)-hard problem [7]. The solution takes exponential time to generate the appropriate solution and cannot be solved in polynomial time.

For a given network, CPP tries to address three major issues: 1) the number of controllers; 2) the position of controllers; 3) the mapping between switch and controller, with the aim to optimize objectives such as reducing the latency, minimizing cost, increasing reliability and energy efficiency and so on.

1.2 Problem Statement

With the decoupling of the control and the data plane, there has been development of some exciting prospects and implementation of SDN. The flexibility and programmability of a SDN network along with the global view of the network enhances the network manageability and innovation. A single controller is often deemed sufficient to meet the existing response time of the medium sized networks. However, for the practical deployment, factors like reliability, cost, scalability and latency affect the overall performance of the SDN network. Furthermore, a physically centralized single controller can be the point of failure of the entire network, especially for large-scale or hyper-large scale network, thus affecting the superiority of the SDN network. Therefore, deployment of multi-controller and co-operative work among them is the essential requirement for the large-scale SDN networks to achieve scalability, fault tolerance and reduced latency.

For the implementation of software-defined wide area network (SDWAN), propagation-delay limits the availability and convergence time of the network, thus determination of the appropriate locations of the controller and the mapping relations between the controller and the switch is the must. Deployment of numerous controllers on the basis of different placement metrics such as, the capacity of the controllers, load on the switch, and latencies can be done. The CPP is a NP -hard problem for which the optimal solution within a polynomial time cannot be achieved. One way to solve the NP-hard problem is by designing the efficient meta-heuristic algorithms such as particle swarm optimization (PSO), Firefly (FF), Bacteria foraging optimization (BFO) and so on. In this thesis,

Naked-mole rat (NMR) algorithm, a novel meta-heuristic swarm intelligence (SI) algorithm, is used to solve the optimization problem of controller placement. Latency and load balancing are used as a parameter to measure the performance of the network.

1.3 Objectives

The objective of this thesis is:

[1] to determine the optimum controller localization using NMR algorithm

[2] to compare the performance on NMR algorithm with bat algorithm

1.4 Scope of the thesis

The scope of this thesis is limited to finding the controller placement using NMR and Bat algorithm considering optimization parameters switch-controller latency (SC latency), controller-controller latency (CC latency) and load balancing. Load balancing is based on minimizing the difference between numbers of switches connected to different controller.

CHAPTER 2: THEORETICAL BACKGROUND

2.1 Theoretical Background

2.1.1 Controller Placement Problem (CPP):

Controller Placement Problem is the research hotspot in the realm of SDN when multiple controllers are deployed to manage the network and enforce the network policies in the controllers. Due to the bifurcation of the data and control plane in SDN, a SDN switch queries with the controller about taking actions when it encounters a new flow. Thus, the flow setup time depends upon the communication latency between the controller and the switch. Hence, the appropriate placement of the controller(s) is regarded as the critical design consideration, since the placement affects the controller latency experienced by SDN switches, which affects wide range of issues like updating routing policies, fault tolerance, load balancing, energy efficiency, scalability, quality-of-services (QoS), etc. Heller et al. [8] introduced and formulated the concept of CPP as a well-known facility location problem (controllers act as facilities and switches as demand points), thereby establishing CPP as NP hard problem.

A. Latency

The one of core factor considered in CPP is latency which depends upon distance between two nodes. Two latencies, propagation latency and processing latency affect the performance of SDWAN. Propagation latency is dependent on the distance between a switch and its assigned controller measuring the response time while load on the controller influences the processing latency.

Average switch-controller (SC) latency:

The SC latency arises due to (i) round-trip propagation delay between a switch and its designated controller, and, (ii) processing delay at the controller. The location where the controller is installed determines the SC latency, while the mapping of the switches to the controller affects the load of the controller, hence the response time of the controller.

The load of switch is measured as the total number of flow setup requests that a switch sends to the assigned controller per second. The load on the controller is measured as the total number of requests per second a controller can process without being overloaded.

Inter-controller Latency:

In order to achieve the global consistency, inter-controller or CC communication is the key. When there is a need to communicate with a switch controlled by other controller, CC communication plays a significant role. Thus, performance of end-to-end communication between switches controlled by different controllers is affected by the inter-controller latency.

B. Controller Load balancing

With the increase in number of switches controlled by the controller, the load of the controller also increases. Since, controller has a fixed capacity to handle the number of requests per second, the overload on the controller can lead to its inability to handle further more requests, which could result in queuing delays or even being able to address further requests from the switches. This could lead to increase in the communication overhead of the SC or CC latency, thus a well-balanced SC mapping is essential. However, to find the optimum placement with minimum number of the controllers and switch assignment is a convoluted task when considering the load balancing.

2.1.2 Naked mole-rat (NMR) algorithm:

NMR algorithm is a stochastic optimization algorithm based on the social behavior of the NMR which imitates the mating pattern of these animals. The key features are summarized as:

i. NMR, a eusocial animal, can have upto 295 members with average number of members being 70-80.
ii. The group is led by a female queen and the entire population is divided into breeders and workers. The most efficient NMRs among working groups are breeders and are intended for mating with the queen.
iii. All other necessary tasks are performed by the workers and the most efficient worker will be promoted to breeder's group. Thus, the workers who

outperform the other workers will become breeders while the breeders who perform the worst will be shifted back to the worker's pool.

iv. Ultimately, only the best breeder among the breeders will be the queen's mating partner.

These above mentioned rules formulize the concept of naked-mole rat algorithm (NMRA). The population of NMR is initialized during the first phase. Then the NMR population is divided into workers and breeders. The selection of the breeders is based on their breeding probability. The algorithm can be described in following steps:

1. Initialization: The population of n NMRs are randomly generated in the range of [1,2,….n]. Each NMR is represented in D-dimensional vector space. Here, D represents the number of variables or parameters to be tested in the problem. Each NMR is initialized as:

$$NMR_{ij} = NMR_{min\cdot j} + U\,(0,1) \times (NMR_{min\cdot j} - NMR_{max\cdot j} \qquad (2.1)$$

Where i ϵ [1, 2,…..n], j ϵ [1,2,….D], $NMR_{i,j}$ is the *i*th solution in the *j*th dimension, $NMR_{min,j}$, $NMR_{max,j}$ are the lower and upper bounds of the problem function respectively and U(0,1) is the uniformly distributed random number. The objective function is evaluated and its fitness is estimated after it has been initialized. B breeders and W workers are determined based on fitness, and the overall initial best solution d is calculated. After initialization, the NMR population is subjected to multiple cycles or iterations of the worker and breeder phases of the search process.

(a) Worker phase: During this phase, the workers are inclined to improve fitness to gain opportunity to become a breeder and eventually mate with the queen. The new solution based on its previous experience and its information available locally. Old solution is memorized unless the mating fitness is better than the previous one. The best fitness of worker will be remembered after the completion of search process. The equation 2.2 below is used to produce new solution from old solution given by:

$$w_i^{t+1} = w_i^t + \lambda(w_j^t - w_k^t) \qquad (2.2)$$

where, w_i^t corresponds to the *i*th worker in the *t*th iteration, w_i^{t+1} is the new solution or the worker, λ is the mating factor and w_j^t and w_k^t are two random solution chosen from the worker's pool. The value of λ is obtained from a uniform distribution in the range of [0, 1].

(b) Breeder phase: In breeder phase, the breeder NMR updates the fitness to retain its position as a breeder and improve its chances of mating. All the breeders are updated based on its breeding probability (bp) with the best fitness as their baseline. The value of bp lies in the range [0,1]. If a breeder can't update its fitness, as its breeding probability is low, then it could be pushed back to worker category. The positional update is based on equation 2.3 :

$$b_i^{t+1} = (1-\lambda)b_t^i + \lambda(d - b_i^t) \qquad (2.3)$$

Here, b_i^t corresponds to the breeder i in the iteration t, λ factor controls the mating frequency of breeders and helps in identifying a new breeder b_i^{t+1} in the next iteration.

The concept of mating pattern of breeders with the queen is used for finding the appropriate solution of the considered problem. On the basis of the idea that the best workers can be promoted to the breeders and the best breeders drift towards mating the queen is followed to devise a new solution. The algorithm focuses on determining the best breeder which can mate with queen and thus devising the potential best solution of the problem. Breeding probability controls the shifting of worker phase towards the breeding phase. The two random worker mole-rats in the close proximity to each other control the worker phase while the best mole rat and some random breeder in close proximity of the current best solution dictate the breeder phase. A simple random scaling factor, known as mating factor, governs the breeder and workers. In terms of local and global search, the worker phase of NMRA corresponds to exploration operation while the breeder phase correlates to the exploitation operation. The exploration operation is governed by two random solutions, while exploitation operation has one random solution, which is close to the current best solution.

2.1.3 Bat Algorithm

Bats, especially microbats, have been using echolocation, a type of sonar, to detect prey, avoid obstacles, and navigate their roosting cervices in the dark. They emit a series of short, ultrasonic bursts in the range of 25 kHz to 150 kHz and listen to the echo that bounces back from the surrounding objects. The loudness and intensity of such pulse emission varies according to their hunting strategies, depending on the species.

Acoustics of echolocation:

Even though, each pulse can last for only few milliseconds (8 to 10 ms), but it constitutes of constant frequency in range of 25 kHz to 150 kHz. Also, even though each pulse can last only for a few thousandths of a second, a microbat can emit about 10-20 such bursts every second. During hunting, the rate of emission can increase upto 200 pulses per second when the bats are in near proximity with their prey. This implies their fantastic ability to of signal processing and studies show that the bat ears integration time is about 300 to 400 μs. Since, the sound velocity in air is typically 340m/s, the wavelength λ of the ultrasonic sound bursts at a constant frequency is given by

$$\lambda = \frac{v}{f'} \qquad (2.4)$$

which provides the wavelength in the range of 2mm to 14mm for the frequency range of 25 kHz to 150 kHz. These wavelengths are in the same order as their prey sizes.

These emitted pulses can be upto 110 dB, but these are in ultrasonic regions. The loudness also varies as during the searching of the prey the loudness can be quiet high while the loudness is decreased when homing towards the prey. The behaviors of microbats using echolocation can be implemented in this algorithm in such a way that it can be associated with optimizing the objective function.

Thus, following rules are idealized, based on the echolocation of microbats, to develop a bat algorithm:

1. Echolocation is used by all bats to sense the distance and they know the difference between food/prey and background barriers;

2. Bats fly with a velocity v_i at position x_i with the fixed frequency $f_{min,}$ with variation in wavelength λ and loudness A_0 to search prey. They have the ability to automatically adjust the wavelength (or frequency) of their emitted pulses and adjust their rate of pulse emission $r \in [0,1]$, depending on their proximity with the target.
3. Assumption that loudness varies from large positive (A_0) to minimum constant value A_{min}

Also there is a general approximation that frequency f is in range of [*fmin, fmax*], corresponding to the wavelength of [$\lambda max, \lambda min$]. For a given problem, any frequency can be used for ease of implementation and adjustment can be made on range by adjusting the frequencies and detectable range needs to be domain specific and, then later toning down to smaller ranges.

Algorithmic equations of BA

In this algorithm, n bats are used to create bat populations which are used for iterative evolution. The location of each bat can be represented by x_i (i $=$ 1 to n) which can be considered as the solution vector of the optimization problem. Also, the bats also have their individual velocity v_i. A position vector of a bat is considered as the solution of the optimization problem in D-dimension search space with design variables considered for that search space,

$$x = [\, x_1, x_2, x_3, \dots, x_D]$$

As positions will vary with the change in iterations, x_t^i is used to represent the position vector of bat i at iteration t, while velocity is represented by v_t^i. During each iteration, each individual bat can vary its pulse emission rate r_i, loudness A_i and frequency f_i. The tuning of the frequency can be done by

$$f_i = f_{min} + \beta(f_{max} - f_{min}) \qquad (2.5)$$

where f_{min} and f_{max} are the minimum and maximum frequency range of frequency f_i for each individual bat i. Here, β is the uniformly distributed random number taken from

[0,1]. The frequency variation is used to define the velocity of the bats and is updated by using

$$v_i^{t+1} = v_i^t + (x_i^t - x_{best})f_i \qquad (2.6)$$

where x_{best} is the best solution obtained in the population at iteration t. After the velocity being updated, the position can be further updated by using

$$x_i^{t+1} = x_i^t + v_i^{t+1} \qquad (2.7)$$

For the local search part, after the selection of the best solutions on current iteration, a new solution for the bat is generated using,

$$x_{new} = x_{old} + \varepsilon A^t \qquad (2.8)$$

where, ε is the random number between [-1, 1], while $A^t$is the average loudness of all the bats in this iteration.

Pulse emission and loudness

The variation in the loudness (A_i) and the rate of pulse emission (r_i) can affect the exploration and exploitation of the search space. The pulse emission rate r_i varies from lower value to higher value monotonically while the loudness A_i can vary from higher value to much lower value. This phenomenon is because the loudness of bat decreases when the bat gets closer to the prey while the rate of pulse emission increases. Thus,

$$A_i^{t+1} = \alpha A_i^t \qquad (2.9)$$

$$r_i^{t+1} = r_i^{(0)}[1 - \exp(-\gamma t)] \qquad (2.10)$$

where, $0 < \alpha < 1$ and $\gamma > 0$ are constants.

2.2 Related Work

The CPP can be further divided based on the scope into (1) the Un-capacitated controller placement problem , where the capacity of controllers is neglected, and (2) the Capacitated Controller Capacity Problem, where the capacity of controllers' are taken into consideration.

2.2.1 Un-capacitated controller placement problem

Initially, CPP has been focused on propagation delays and ignored the consequences of traffic load balancing in the networks. Heller et al. [8] is the first paper to formulate and introduce CPP in SDN where 3 metrics were proposed for the CPP: a. minimizing average-case switch controller (SC) latency mapped to k-median problem b. minimizing the worst case SC latency for given number of controllers, similar to k-center problem c. maximizing the number of switches per controller within a latency bound, similar to set cover problem. The algorithm clustered a set of switches and a controller was placed to support such switches, thus the result was always placement dependent of the switches in the topology. Two inputs, the value of 'k' and cluster initialization were used. The analysis was done on WAN where the most significant latency is the propagation latency. The evaluation of trade-off between average SC latency and worst SC latency was done. The algorithm ran by trial and error, so for the same topology the algorithm had to run multiple times to find the best clustering, which was the major limitation of this paper.

Hu et al. [9] approached the CPP as a facility location problem and focused on the network failures like control link failure and unresponsive controllers and switches. The authors evaluated random placement, a greedy algorithm and a brute force algorithm and concluded that the greedy algorithm outperformed other placement solutions. The placement performance was found to be algorithm dependent. The shortcomings of this research were the complete ignorance of issues like SC latency, controller-controller (CC) latency and load balancing among the controllers.

2.2.2 Capacitated controller placement problem

Tanha et al. [10] studied the CPP to address the network resiliency considering deployment costs and minimizing SC latency while taking consideration of the controller capacity as well as varying switch load. The capacitated k-center algorithm was assessed on topologies of tier-1 service provider and concluded that resiliency of the controller is topology dependent. The outcomes of the experiment were ideal only for small and medium scale networks while being resource intensive.

Yao et al. [11] researched the CPP using heuristic algorithm considering the SC latency while considering the traffic load of the switches. Controller load balancing under

heterogeneous load of switches while minimizing the SC latency was the main objective of the research. Resiliency was achieved using additional controller in the network. Comparative analysis of k-center and capacitated k-center algorithm was done. This research is limited to medium size networks which was the major shortcoming.

2.2.3 Meta-heuristic approach for solving CPP

Gao et al. [12] addressed the CPP using PSO for the first time considering the SC and CC latency as the evaluation metrics. The author also addressed the concept of load balancing by implementing the controller capacity. The performance of PSO was compared with greedy algorithm and integer linear programming (ILP). For the larger network, PSO converged faster and provides optimum result when compared with greedy algorithm and ILP.

Li et al. [13] proposed the improved FF algorithm to address the CPP in multi controller environment. The average latency between SC and controller load utilization were the metrics considered for optimization. The algorithm was limited to small size network and the cross domain communication problem was not addressed.

Sahoo et al. [14] considered the CPP as the multi objective combinational problem and approached the problem using two meta-heuristic algorithms, PSO and FF. The SC and CC latency were considered as the objective function and the performance were evaluated on the publically available datasets in terms of execution time. The main shortcoming of this research was only the implementation of only latency aware placement while completely ignoring the effects of load balancing.

Zhang et al. [15] proposed the CPP for large SDN by researching on the optimal controller deployment, SC assignment and the optimum number of routing request. The researchers formulated a multi-objective optimization controller placement (MOCP) and used ABFO, a heuristic algorithm, to solve the multi objective problem. Low latency, reliability and load balancing were the metrics considered for optimization. The proposed heuristic algorithm had almost identical reliability when compared to Pareto Simulated Annealing (PSA) and heuristic but the load balancing and latency were significantly improved.

Sahoo et al. [16] addressed the link failure and latency issue of CPP to enhance the performance of the SDN. CPP was taken as multi-objective combinational optimization problem and solved using the meta-heuristics approach, i.e. PSO and FF. SC latency, CC latency and multipath connection between the switch and controller were the three metrics considered. Improvement of average delay even in single-link failure verified by simulation result was achieved. FF algorithm outperformed PSO when considering computational efficiency and generalized optimum result.

Singh et al. [17] proposed a varna-based algorithm (VBO), a heuristic algorithm, to address the total average latency between the controller and switch in SDWAN. Both the capacitated and un-capacitated controllers were taken into consideration. VBO algorithm represented switch and controller as particle and fitness of the particle was based on Karma. Evaluations were performed in Matlab simulations of the real world topologies and results showed the superiority of the VBO algorithm over other CPP technologies.

2.3 Research Gap

The metaheuristic algorithms are used to solve the CPP addressing the wide variety of issues. The NMR algorithm is one of the recently proposed meta-heuristic algorithms based on the mating pattern of the NMRs. The performance of the NMR on the benchmark test functions demonstrated the effectiveness of this algorithm and had the competitive edge when compared with other state of the art metaheuristic algorithms like PSO, FF, and so on [18]. This algorithm has never been implemented for controller placement problem in SDN, so this is the novelty for this kind of problem. So, to implement the algorithm for placement of controller while considering the suitable objective function is the major research challenge in this thesis.

CHAPTER 3: METHODOLOGY

3.1 Problem Formulation:

A network topology is represented by an undirected graph G(S, E), where S represents the set of switches and E represents the set of bidirectional physical links interconnecting the switches. The following notations represent sets, decision variables and constraints:

Sets

- S, a set of switches present in the network topology, $S = \{s_1, s_2, s_3, \ldots\ldots, s_n\}$, where n is the total number of switches
  - l(s) is the load associated with the switch
- C, a set of controllers to be installed in the network, $C = \{c_1, c_2 \ldots\ldots, c_k\}$, where k is the number of controllers to be installed
  - L(c) is the total capacity associated with the controller
- P, the set of all possible locations for controllers' placement
- $T(c)$, the set of forwarding switches connected by controller

Functions

- d(s, c) is a function that calculates the shortest path between a switch $s \in S$ to one of the controller $c \in C$
- d(c_i, c_j) is a function that calculates the shortest path from controller c_i and c_j

Decision variables

- $x_{cp} = \begin{cases} 1, & \text{if controller } c \in C \text{ is installed on location } p \in P \\ 0, & \text{otherwise} \end{cases}$

3.2 Objective Function:

The objective function of this thesis is to optimize the controller placement location based on the total capacity of the controller minimizing the SC latency and CC latency.

i. Average SC latency:

Average SC latency is the most commonly used metric in CPP. It calculates the average distance between the placed controllers' location and the switches assigned to them. It reflects the basic performance of propagation latency in the SDN by representing the average value of packet transmission latency between the switch and the controller. The mathematical expression can be expressed in equation 3.1 below:

$$L_{sc-avg} = \frac{1}{n}\sum_{s\in S} \min_{c\in C} d(s,c) \qquad (3.1)$$

ii. Average CC latency:

The subnetwork created among the controllers' is located on the controller plane, thus the latencies among these controllers is also critical aspect of SDN. Communications between the controllers are crucial for achieving the consistent view of the network's state which is utilized by network application for proper operation. The communication overhead maintained by the shared state among the controller is very significant. The controller to controller latency is mathematically represented as:

$$L_{cc-avg} = \frac{1}{k}\sum_{c\in C} \min_{c\prime\in C} d(c,c') \qquad (3.2)$$

iii. Load Balancing

It is considered that all the switches are connected to the closest controller based on latency as the metric, unless the addition of the switch exceeds the controller capacity, i.e. the controller has already the maximum number of the switches it can handle. In such scenario, the switch is connected to the second nearest controller. Taking the reference from [21], we assumed a number of OpenFlow requests (γ) in the range of 0.05–0.105 million request per second by an SDN switch, the request-handling capacity of controller (σ) as 1.1 million requests per second.

This can be mathematically represented as:

$$\sum_{s \in T(c)} l(s) \leq L(c) \qquad \forall c \in C \tag{3.3}$$

For placements of C controllers, the global average latency can be represented as:

$$G(C) = w1 \times \frac{1}{n} \sum_{c \in C} \min_{s \in S} d(s, c) + w2 \times \frac{1}{k} \sum_{c' \in C} \min_{c \in C} d(c_i, c_j) \tag{3.4}$$

where, w1 and w2 are weights, such that w1 + w2 = 1.

The objective function of capacitated controller placement problem for global average latency is:

$$\textit{Minimize } G(C) \tag{3.5}$$

Subject to:

$$\sum_{s \in T(c)} l(v) \leq L(C) \qquad \forall c \in C$$

For k number of controllers to be deployed in a SDN, the goal is to optimize the placement of controllers C such that the value of $G(C)$ is minimized subjected to the constraint present in equation (7) and |C| = k.

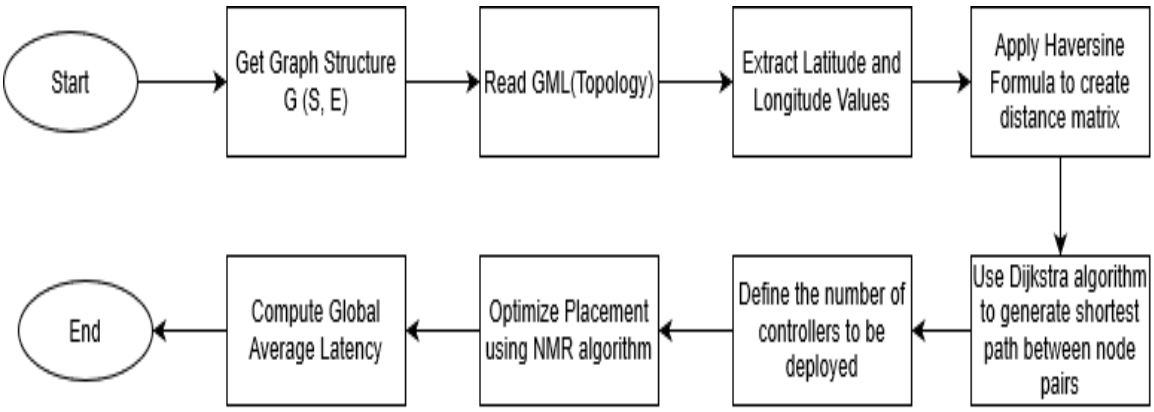


Figure 3.1: Proposed methodology for controller placement

3.3 Dataset

The datasets used in this thesis are the real topologies that are obtained from the Topology Zoo. It is the collection of most up-to-date network data that are made publically available by network operator. The network topology available are undirected graph *G(N,E)* abstracting the connectivity of data communication network. The topology information is available in Graph Markup Language (GML) and GraphML. Additionally, the network maps contains meta-data, for instance, longitude and latitude of nodes, link types and speed and so on, which increases the efficacy of network topology.

For this thesis, topology Savvis and Ernet are used. Savvis is the network topology representing the backbone network of USA which consists of 19 nodes and 20 edges connection between them. Topology Ernet is the backbone network of India which consists of 16 nodes and 18 edges connection between them. The following figure 3.2 represents the schematic view of topologies.

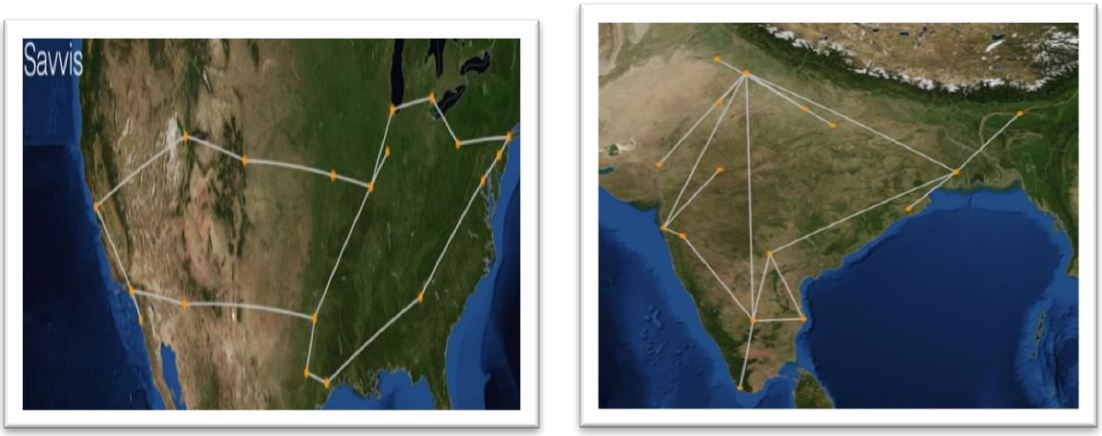


Figure 3.2: Topology Savvis and Ernet

At first, these network topologies are read in GML format and using networkx the longitude and latitude is extracted. Thus, each GML file will provide us the node, the connection between the nodes and edges, their longitude and latitude and the link bandwidth between them. The latitude and longitude is used to calculate the shortest distance matrix between each node.

3.4 Haversine Distance Approach:

The great circle distance between the pairs of switch is computed using Haversine distance approach. The shortest distance between two points on a sphere, measured along the earth surface is known as great circle distance. Alternative to this approach is Law of cosines which is actually accurate only for shorter distance. The equation 3.6 is used to compute the great circle distance as defined by Haversine approach, where Ψ_1 and Ψ_2 represent the latitudes of the point 1 and point 2, λ_1 and λ_2 represent the longitude of point 1 and 2 respectively and r is the radius of the earth at a constant 6371 km.

$$distance = 2(r)\arcsin\sqrt{\sin^2\left(\frac{\Psi_2-\Psi_1}{2}\right) + cos(\Psi_1)\cos(\Psi_2)\sin^2\left(\frac{\lambda_2-\lambda_1}{2}\right)} \qquad (3.6)$$

3.5 Switch-controller mapping relationship

The mapping of the switch to controller is the relationship of how the switches are controlled by the controller. The assumption is that one switch can be controlled by only one controller and is based on the shortest distance between the switch and the controller. Also, the switch positions where the controllers are placed are controlled by default by the deployed controller. The selection of the second nearest controller is only deemed necessary in situation when the load of the controller exceeds the switch handling capacity. An index list is created which represents the mapping relationship of controllers with the switch as shown in figure 3.3 below.

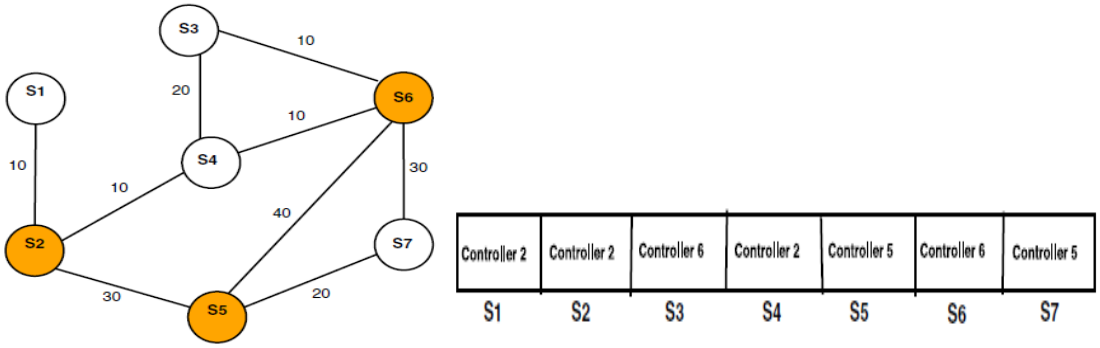


Figure 3.3: Switch to controller mapping

3.6 Naked-mole Rat Controller Placement Problem (NMRCPP)

NMR algorithm is used for optimizing the controllers' placement with the motive to minimize CC and SC latency while keeping the load balancing as the constraint. The CPP can be addressed by using NMR algorithm by the following way.

In NMR algorithm, the solution of the optimization problem is determined by the position of the NMRs. First, n numbers of NMRs are initialized, with each NMR representing one of the combinations of k possible placement of controllers to be deployed. Given the k number of controllers to be deployed, each NMR is represented as a 2k dimensional vector and each dimension represents the one of the controller position. After initializing the NMRs, each of the controllers' position is assigned to the switches in the network based on the shortest distance to the controllers and label for switches mapping is created for every controller. After mapping of the switches to the controller, the load of the controller is measured by using equation 3.3. If the capacity of controller exceeds its limitation, the switch is assigned to the second nearest controller. As a switch is assigned to the possible nearest controller, the delay from a node to a nearest controller is measured in terms of distance. Then, the objective function is calculated using equation 3.5, which represents the fitness of the NMRs. The best 'b' breeders are selected based on the best fitness value and remaining NMRs are selected as workers. Also, the best position of the NMR is set as global best position.

Then, until the termination criterion is met, the position of breeders and workers are updated. For all the breeders, the position is updated using equation 2.3. Breeder NMRs are used for exploitation of search space, thus the search space of breeders is situated around those controllers' position which obtains the best fitness value. Hence, breeder NMRs position is always directed by the best NMR position. After the position is updated, the fitness of the breeders is updated using equation 3.4 and if it improves the fitness, then the new position is remembered.

Similarly, for all the workers, the position is updated using equation 2.2. The main purpose of workers NMRs is exploration of the search space. The exploration of the search space is based on the two random worker NMRs position, thus the update of position vector of worker NMRs is directed by the two randomly selected workers

position. Thus, like a random search in the search space, worker NMRs performs the exploration part. After updating the position of workers and breeders, their fitness is evaluated and only if the new fitness is better than the previous one, it is stored in the memory. Then, evaluation of the fitness of the NMRs is done and only the best NMRs are selected as the breeder for next iterations. The best among the breeders is memorized as the best solution for that iteration. Finally, when the termination criterion is met, the best breeder NMR presents the solution of the controller placement problem. Thus, the solution contains the position of the controllers, which essentially means the longitude and latitude of the controllers with switch to controllers mapping relationship and the best minimized value of the objective function.

Algorithm 1 Naked-mole Rat Controller Placement Problem

Input: *LatLong, nCont, SrtPathMtrx //latitide & longitude, number of controllers, the shortest distance between each nodes*

Output: *best_latency(optimized cost), FinalCtrlPos, MappingS2C, //best value of minimized latency, latitude and longitude of the controllers, switch to controller mapping relationship,*

Initialization:
nSwitch ⟵ size of Latlong // total number of switches
maxIter⟵ maximum number of iterations
nPop ⟵ NMR swarm size // total number of NMR population
bp⟵ breeding probability // the breeding probability of the breeder NMR
n_B⟵ nPop/5 // population size of the breeder NMR
for *i = 1 to nPop* ***do****,*
nmr(i).Pos⟵randomly select nCont from nSwitch
label ⟵ calculate mapping of switch to controller
fitness⟵ evaluate latency of nmr(i)
end for
Optimization:
do until *iter < maxIter*
sort NMRs according to fitness in ascending order
X_{best} ⟵NMR with best minimum latency
L_{best} ⟵ label of the best NMR //mapping of controller and switches
for *i = 1 to n_B, do*
if U(0,1) > bp
update position using: $b_i^{t+1} = (1-\beta)b_i^t + \beta(X_{best} - b_t^i)$ *// β refers to random (0,1)*
if *latency of* $b_i^{t+1} < b_i^t$ ***then***
update the new position of b_i^{t+1}
end for
for *i = n_B + 1 to nPop,* ***do***
update position using: $w_i^{t+1} = w_i^t + L \times (w_j^t - w_k^t)$ *// L is the levy flight step*
if latency of $w_i^{t+1} < w_i^t$ ***then*** *//only update position if fitness of new position of worker is better than previous position*
update the new position of : w_i^{t+1}
end for
update iter+1
FinalCtrlPos⟵ position of X_{best}
MappingS2C ⟵ label of L_{best}
best_latency ⟵ fitness value of X_{best}

Algorithm 2 Evaluating Fitness

Input*: ShrtPathMtrx , nmr.pos //shortest path matrix containing distance between the nodes, // the original controller position*

Output*: fitness, label // the average SC latency, mapping of switch to controller*

```
for each nmr, do
    for each node
        find the shortest distance path to the controller;
        label = assign the node nearest to it
        Compute controller load capacity using equation (6)
        while controller load exceeds the capacity, do
            assign switch to the next nearest controller
            Update the label
    end for
    for each controller, do
        find the shortest distance path to other controller;
        Evaluate the controller-controller latency using the path
    end for
fitness = the combined global average latency using equation (7)
end for
return fitness, label
```

Algorithm 1 elucidates the flow of how the NMRA works for solving the controller placement problem. The positional update of worker is based on L, which is the levy flight steps and the two randomly selected worker NMRs while breeders are updated based on β which is the random number between 0 and 1 and the best breeder, which is basically the best NMR. Here, each NMR represents the solution of the controller placement containing the co-ordinates of the nodes to be deployed.

CHAPTER 4: IMPLEMENTATION AND ANALYSIS

4.1 Result and Analysis

For the evaluation of this work, the experimental work was performed in Google colab for the optimization of CPP. Python programming language was used for implementation of NMR and bat algorithm. The experiment runs on the system that consists of Windows 8.1 (64 bit) having Intel core i5-3317U CPU @ 1.7 GHz and 6 GB RAM. The population size for the optimization methods is set to be 40 for all experiments. The results were obtained by running the algorithms 30 times while providing the network topology from Topology-Zoo as input. For the experimental setup, two topology (Savvis and Ernet) containing 18, 16 nodes were taken and the best controller placement was analyzed based on SC latency and SC+CC Latency combined. Also, in some scenario, where load was unbalanced, load balancing was also maintained.

The two algorithms NMR and Bat algorithm were analyzed on three aspects: (i) effect on average SC latency with increasing number of the controllers, (ii) effect on global latency with increase in number of controllers (iii) effect of algorithm execution complexity with increase in controllers to find optimal solution

4.1.1 Average SC latency VS number of controllers

Average SC latency in SDWAN refers to the average time taken by the switches to receive its flowtable instruction from its nearest associated controllers, which is proportional to the distance between the switch and the controller. The placement of the controller based is analyzed on the following topology:

A. Ernet

Ernet is the customer backbone network topology of India containing 16 nodes in various cities of India. The topology is analyzed by using NMR and Bat algorithm.

Case I: SC latency

In this case, both the NMR and Bat algorithm performed better in terms of the exact same placement for the controllers as shown in figure 4.1 and 4.2 below. For single controller placement, i.e. k = 1, the controller is placed at Delhi (node 8) which was our first master controller. With the increase in controller, further placements of the controller were

determined as illustrated in figure below. For k = 2, the placement of controller was at Delhi (node 8) and Bengaluru (node7). The controller situated at Delhi controlled switches (node 4, 5, 8-14) while controller located at Bengaluru controlled 7 switches located at node 0-3, 6, 7 and 15. Similarly, for placement of three controllers, the controller location was situated at Delhi, i.e. node 8, Bengaluru, i.e. node 7, and Kolkata, i.e. node 13. The controller situated at Delhi controlled 6 switches located at node 4, 5, 8, 10, 11 and 12, while controller at Bengaluru controlled 7 switches located at node 0, 1, 2, 3, 6, 7 and 15 and controller at Kolkata controlled 3 switches at node 9, 13 and 14.

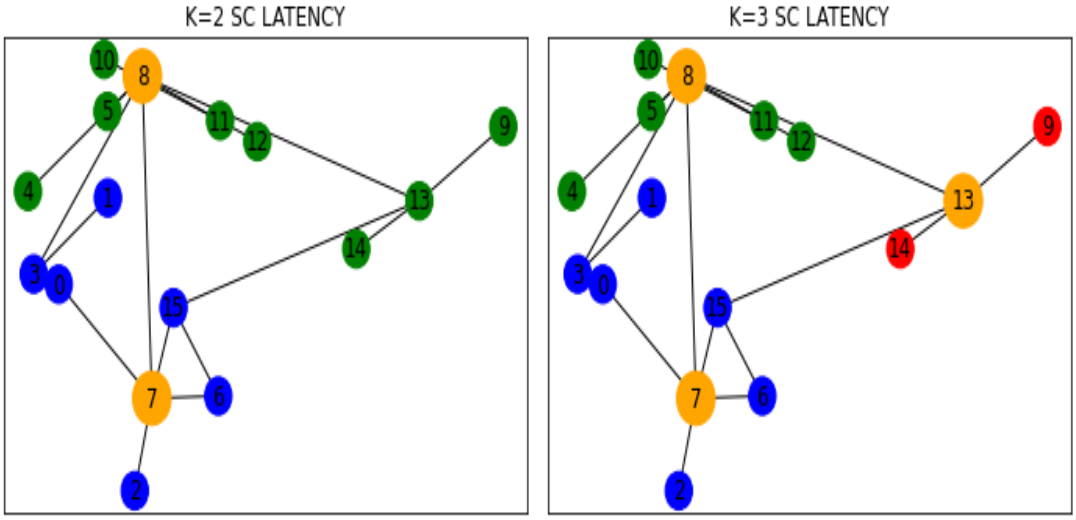


Figure 4.1 Controller Placement for k=2 and k=3

When the 4 controllers were needed to be placed at the topology, the placements were at Delhi (node 8), Bengaluru (node 7), Kolkata (node 13) and Mumbai (node 3). The controller located at Delhi controlled 6 switches located at node 4, 5, 8, 10, 11 and 12, while controller at Bengaluru controlled 4 switches at node 2, 6, 7 and 15 and controller at Kolkata controlled 3 switches at node 14, 13 and 9, with controller at Mumbai controlling 3 remaining switches situated at node 0 and 1.

Similarly, for 5 controllers to be placed the controllers is placed at Delhi (node 8), Bengaluru (node 7), Kolkata (node 13), Pune (node 3) and Allahabad (node 12) with controller load of 3, 4, 3, 3, 3 switches respectively. Controller at Delhi controlled switches at node 4, 5 and 8 while the controller at Bengaluru controlled switches at node 2, 6, 7 and 15, controller at Kolkata controlled switches at node 9, 13 and 14. Similarly,

controller at Pune controlled switches at node 0, 1 and 3, while controller at Allahabad controlled switches at node 10, 11 and 12.

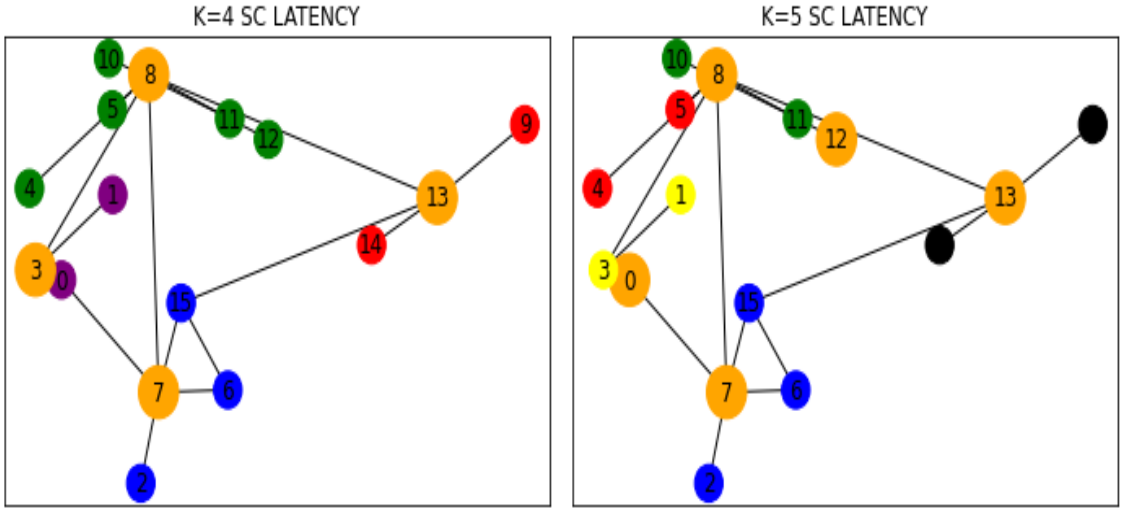


Figure 4.2 Controller Placement for k=4 and k =5

The figure 4.3 illustrates the ramifications on average SC latency with the increase in the number of the controllers. As shown in figure, when a single controller has been placed the average SC latency was the maximum of 6.28 ms, as only a single controller manages all the switches, thus increasing the propagation distance between switch-controller and impacting the latency. When the controller number was increased to 2, i.e. k = 2, the average SC latency decreased to 3.75 ms, a 40.3% reduction in average SC latency when compared to k = 1. This decrease in average SC latency was due to increase in number of controllers, which reduces the propagation distance between switch-controller, as switches have the flexibility of connecting to the controller located at their proximity. If the number of controllers was increased to 3, i.e. k = 3, the average SC latency further decreases to 2.52 ms, a 32.6% average SC latency reduction compared to two controllers. This decline in average SC latency has occurred due to the presence of more controllers, which results in smaller propagation distance between switch-controller, thus affecting the average SC latency. For 4 controllers' placement, i.e. k = 4, the average SC latency was 1.80 ms, a 28.7% reduction of average SC latency compared to placement of three controllers. While for placement of 5 controllers, the average SC latency was 1.53 ms, a 15% lower average SC latency when compared to four controllers' placement scenario.

Table 4.1: Average SC latency in Ernet

Number of controllers (k)	Latency (ms)
1	6.28
2	3.75
3	2.52
4	1.80
5	1.53

Thus, it can be concluded that, with the increase in number of the controllers', the average SC latency keeps on decreasing, until there is a controller for a switch. Table 4.1 shows the decrease in SC latency with increase in controller number. The paramount change in SC latency was seen when the controller number is two when compared to single controller with almost 40% curtailment of latency. The further considerable decrease in SC latency is seen until k = 4, as each controller resides in the nodes position with maximum degree available.

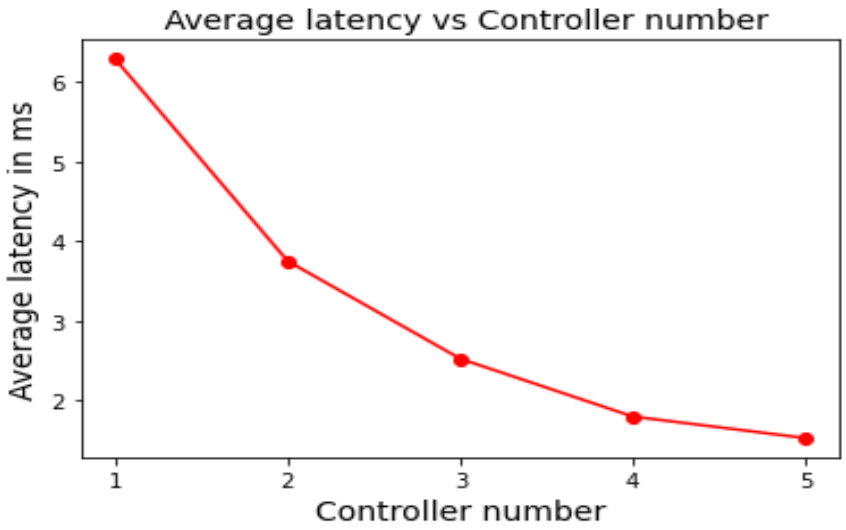


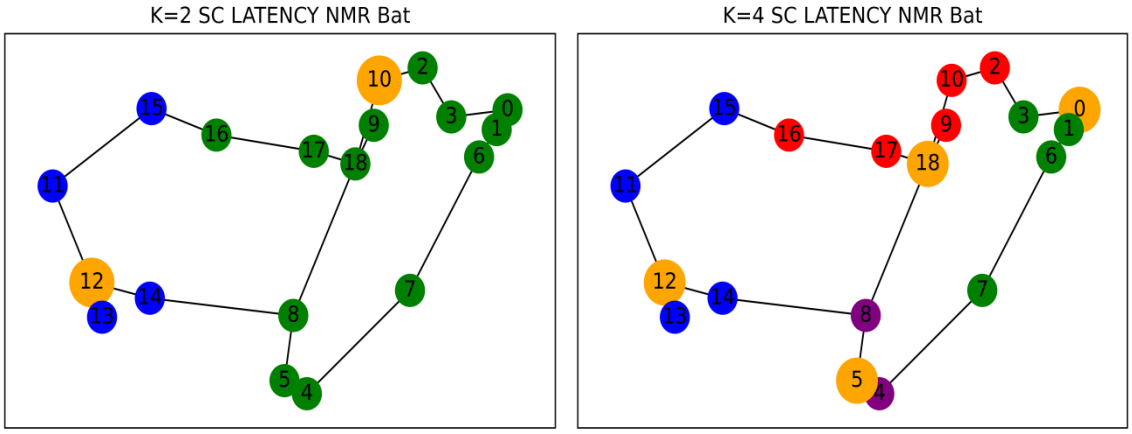
Figure 4.3: Average SC latency with increase in controllers

B. Topology Savvis

Topology Savvis is the customer backbone network of United States which consists of 19 nodes.

Case I: SC Latency only

When considering SC latency only, the placement of the controller was same for k = 1, k = 2, and k = 4, thus both the algorithms has achieved the same SC latency. Figure 4.4 (a) and 4.4 (b) below shows the placement of both NMR and Bat for k = 2 and k = 4. As seen in figure, the priority was only given for SC latency and load balancing was not considered, thus resulting in unbalanced controller placement. For k = 2, the controller was placed at Chicago (node 10) and Los Angles (node 12). The controller at Chicago connected 14 switches at node 0-10, 16, 17 and 18, whereas controller at Los Angles controlled 5 switches at node 11, 12, 13, 14 and 15.

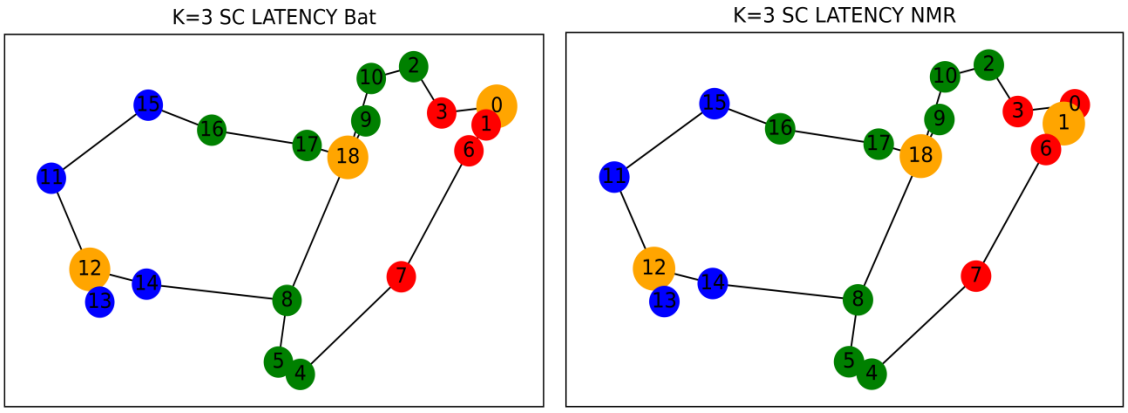


a. b.

Figure 4.4: (a) Placement of the controllers considering SC latency for (a) k=2 (b) k=4

For four controllers as in figure 4.4 (b), i.e. k=4, the controllers were placed at Saint Louis (node 18), New York (node 0), Los Angles (node 12) and Austin (node 5) connecting 6, 5, 5 and 3 switches respectively. The controller at Saint Louis connected switch located at nodes 2, 9, 10, 16, 17 and 18, whereas New York's controller has covered node 0, 1, 3, 6 and 7. Similarly, controller at Los Angles has connected switches at node 11, 12, 13, 14 and 15, while controller at Austin controlled switches at node 4, 5 and 8. With the increase in controller's number, the average SC latency decreased, as

more number of switches was connected to the controllers, thus decreasing distance between switch and controller.



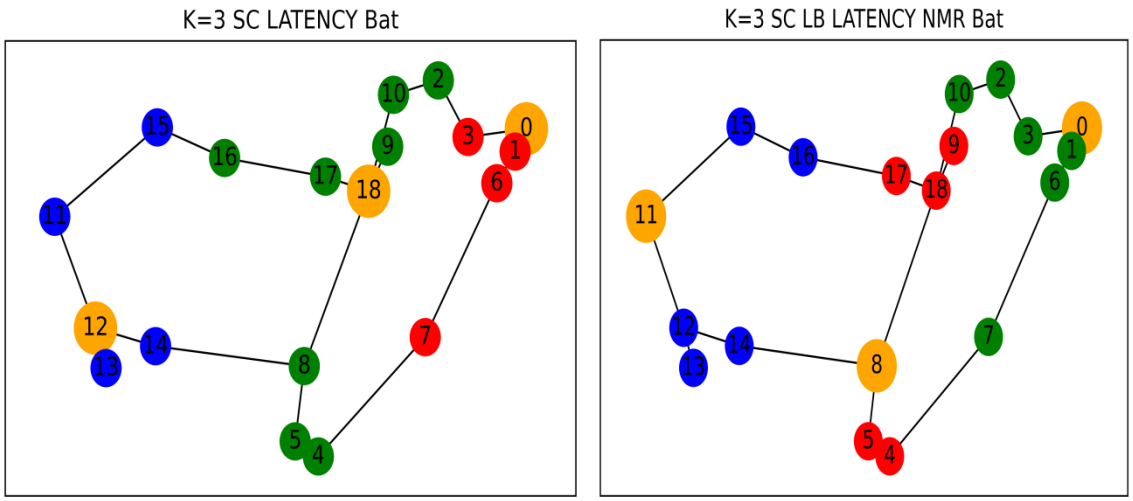
a. b.

Figure 4.5: (a) Controller placement for bat algorithm at k=3 (b) Controller placement for NMR algorithm at k=3

In figure 4.5, the difference in placement of controller for bat algorithm and NMR algorithm is shown. Out of three controllers located, both bat algorithm and NMR algorithm have selected Los Angles (node 12) and Saint Louis (node 18) as common nodes to place the controllers, while bat algorithm selected New York (node 0) as third controller's location whereas NMR selected Philadelphia (node 1) as controller placement location. The total switches controlled by the controller at Los Angles and Saint Louis was 5 and 9 respectively. Five switches were controlled by the controller at New York in case of bat algorithm whereas NMR opted to select Philadelphia. The average SC latency of NMR was slightly better than bat algorithm (3.01 ms vs 3.05 ms).

Load Balancing Scenario:

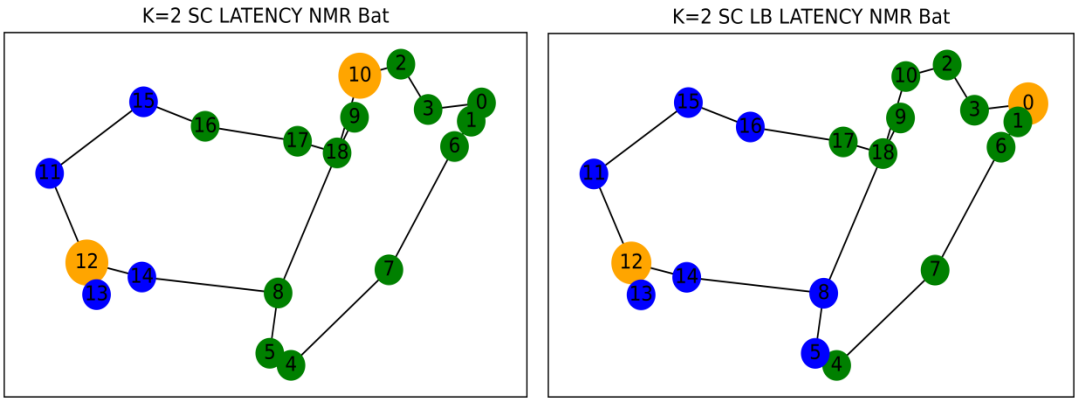
During the consideration of SC latency, the load balancing is neglected in some scenario, as the algorithms have been focused on minimizing SC latency only. In a SDN, load balancing is also a key parameter for smooth operation of the architecture. The figure 4.6 below shows the unbalanced and balanced load scenario for 3 controllers focused on minimizing SC latency.



a. b.

Figure 4.6: (a) Controller placement for k = 3 focusing only on SC latency (b) Controller placement focusing on Load balancing

As seen in figure 4.6, during an unbalanced load case, the controllers were located at Saint Louis (node 18), Los Angles (node 12) and New York (node 0), controlling 9, 5 and 5 switches respectively. The load distribution was uneven as the maximum number of switches controlled was 9 and minimum number of switches connected was 5. Thus, to mitigate this drawback, a load balanced controller placement is shown in figure 4.6 (b). The initial placement was slightly tweaked, placing controller at San Francisco (node 11), Dallas (node 8) and New York (node 0), with San Francisco controlling 6 switches at nodes 11-16, while Dallas controlled 6 switches at nodes 4, 5, 8, 9, 17 and 18, whereas the controller at New York controlled 7 switches at node 0, 1, 2, 3, 6, 7 and 10. The node 16 controlled by the controller at Saint Louis (node 18) in figure 4.6 (a) was now controlled by controller at San Francisco (node 11). The controller at Saint Louis (node 18) is shifted to Austin (node 8) and controlled 6 switches node, thus reducing the load at the controller. Switches at node 10 and node 2 were now controlled by the controller at New York (node 0). Thus, a relatively balanced controller load distribution was achieved in the latter scenario. The SC latency of unbalanced condition was 3.01 ms while the balanced condition latency was 3.49 ms, thus at the cost of 15% increase in the SC latency, a better well balanced controller placement was achieved.



a. b.

Figure 4.7: (a) Controller placement at k =2 focusing SC latency only (b) balanced load condition k = 2

As shown in figure 4.7(a), during the placement of the controller while just considering the SC latency resulted in unevenly distributed load in the controller, as controller at Los Angles (node 12) only connected 5 switches only and controller at Chicago (node 10) controlled 14 switches. As focus was only on SC latency, the closely placed switches were kept under same partition. However, in figure 4.7 (b) a slightly better load balanced controller placements was achieved by placing controllers at Los Angles (node 12) controlling 8 switches and New York (node 0) controlling 11 switches. Switches at node 16, 8 and 5 were assigned to controller located at Los Angles (node 0). The average SC latency of scenario at figure 4.7(a) was 5.08 ms while for the scenario 4.7(b) was 5.61 ms, thus an increase in latency by 10% was observed at the cost of slightly better load distribution among controllers.

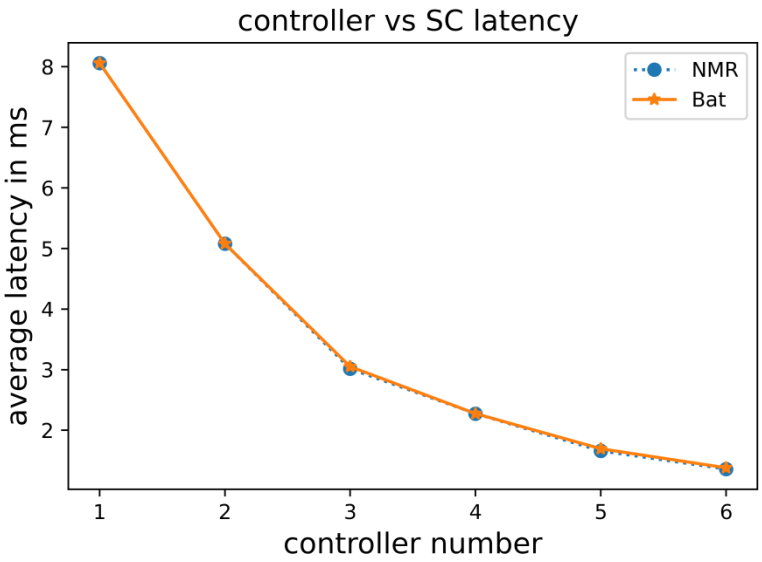


Figure 4.8: Average SC latency vs number of controller

As in figure 4.8, the average SC latency decreases with increase in number of the controller because of decrease in distance between the switch and the controller. The table 4.2 below shows the average SC latency with increase in the controller number.

Table 4.2: Average SC in topology Savvis

Number of controllers (k)	Latency (ms)
1	8.06
2	5.08
3	3.01
4	2.27
5	1.69
6	1.38

When a single controller was placed, the average SC latency was 8.06 ms, while for two controller placement, i.e. k = 2, the average SC latency was 5.08 ms, a 36% reduction in average SC latency when compared to k = 1. This decrease in average latency was because two controllers were placed in such a way that the switch-controller distance is minimized, which in effect decreased the average SC latency. For k=3 scenario, the average SC latency achieved was 3.01 ms for NMR algorithm and 3.05 ms for bat algorithm, almost a 40% reduction in average SC latency when compared to k = 2

scenario. Further, when 4 controllers were used, the average SC latency further decreased from 3.01 ms to 2.27ms, a 24% decrease in average latency when compared to k = 3 scenario. At k = 5 scenario, the average SC latency of NMR algorithm was 1.66 ms while for bat algorithm was 1.69 ms, and when compared to k = 4 scenario, almost 26% decrease in average SC latency. When k = 6, the average SC latency was 1.38 ms, an 18% decline in average SC latency, when compared to k = 5 scenario. In average SC latency at topology Savvis, NMR algorithm slightly performed better at finding the minimum average latency when compared to bat algorithm.

4.1.2 Average SC and CC latency VS number of controllers

If the consideration was given to combined SC latency and CC latency, there were two scenarios: (i) 0.9 CS and 0.1 CC (ii) 0.8CS and 0.2 CC. The analysis was topology specific and mentioned below:

A. Ernet

CASE I: 0.9 CS and 0.1 CC:

In this scenario, 90% priority was given to average SC latency and 10% to average CC latency. For two controllers, i.e. k = 2, and three controllers, i.e. k = 3, both NMR and bat algorithm had the exact same placement, thus the same latency as shown in figure 4.9 below. However, due to addition of the CC latency, the algorithms focused on the CC latency as well, which can be seen on the figures 4.9 and 4.10 below.

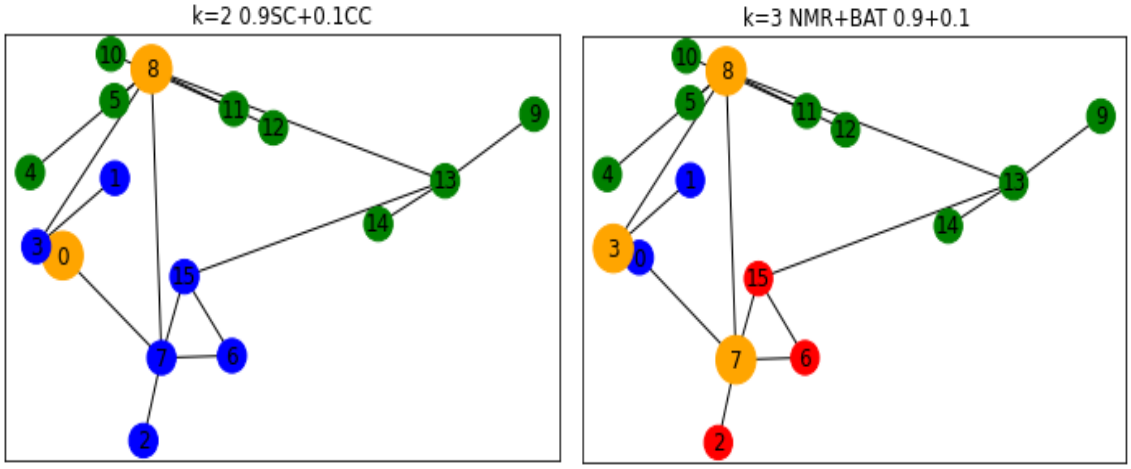
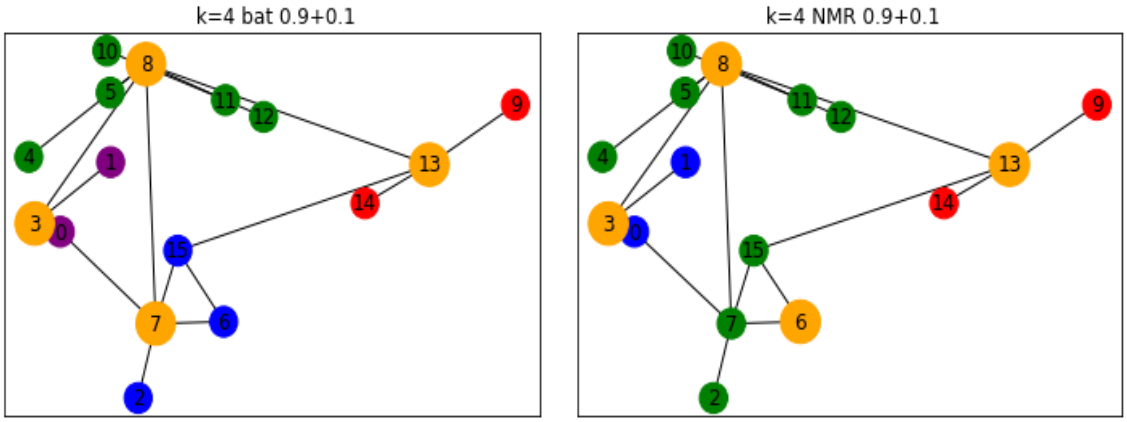


Figure 4.9: Controller placement for k = 2 and k = 3 for NMR and bat algorithm

In two controllers scenario, i.e. k = 2, the controllers was placed at Delhi (node 8) and Pune (node 0). The controller at Delhi controlled 9 switches located at node 4, 5, 8-14 while the controller at Pune controlled node 0-3, 6, 7 and 15. For three controllers scenario, i.e. k = 3 the controllers were placed at Delhi (node 8), Mumbai (node 3) and Bengaluru (node 7). 9 switches located at node 4, 5, 8-14 were connected to the controller located at Delhi, while 3 switches situated at node 0, 1 and 3 were connected to the controller in Mumbai, whereas 4 switches at node 2, 6, 7 and 15 were connected to controller in Bengaluru.



a. b.

Figure 4.10: Controller Placement for (a) k = 4 for bat algorithm (b) k = 4 for NMR algorithm

As shown in figure 4.10, the placement of four controllers' is illustrated. Both NMR and Bat algorithm has three controller placement same, i.e., Delhi (node 0), Mumbai (node 3) and Kolkata (node 13) with each controller connected to 6, 3, 3 switches respectively. However, NMR selected fourth node as Chennai connecting switches at node 15, 7 and 2 while bat algorithm selected Bengaluru connecting node 15, 6 and 2. The latency of NMR was better than bat algorithm at k = 4.

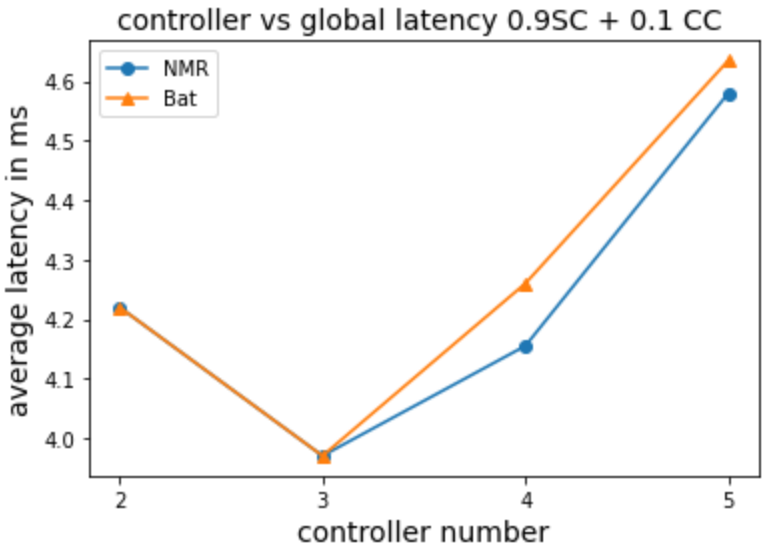


Figure 4.11: comparison of NMR and Bat algorithm on 0.9SC + 0.1 CC global latency

Table 4.3: 0.9SC + 0.1CC latency in Ernet

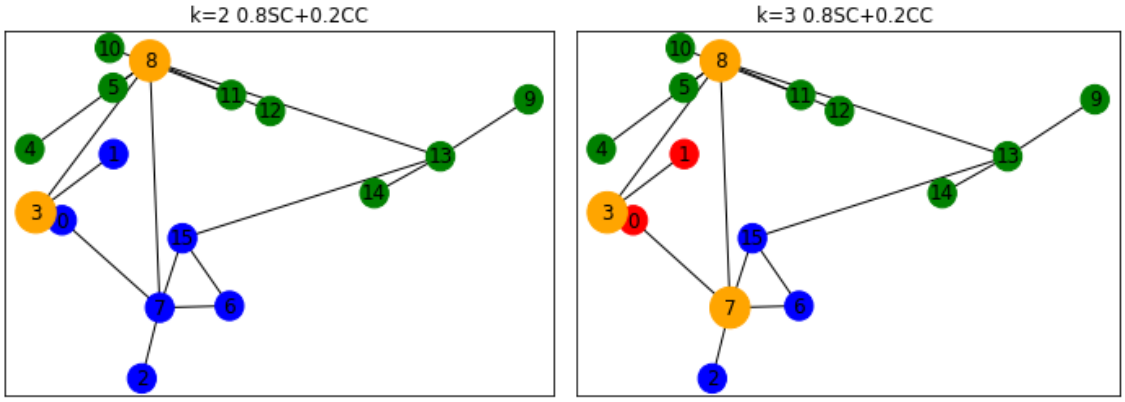
Number of controllers (k)	NMR Latency (ms)	Bat Latency (ms)
2	4.22	4.22
3	3.97	3.97
4	4.15	4.18
5	4.58	4.64

From the figure 4.11 and table 4.3, it can be observed that latency actually decreases from k=2 scenario to k=3 scenario, which was due to decrease in SC latency with increase in controller, while addition of a single controller did not increase much CC latency. Further addition of the controllers, i.e. k = 4 and k = 5, the CC latency increased due to addition of control path between the controller while there was decrease in SC latency due to addition of the controller, thus overall increasing the global latency with increase in controller. When comparing NMR with bat, the global latency of NMR was slightly better than bat algorithm in scenario when 90% priority is given to SC latency, while 10% priority is given to CC latency.

Case II : 0.8 CS + 0.2 CC

In this scenario, 80% priority was given to SC latency and 20% priority was given to CC latency. For k = 2 to 5, both NMR and bat algorithm had the same placement of

controller and thus have same global average latency which can be demonstrated in figure 4.12 below.



a. b.

Figure 4.12: (a): controller placement for k=2 (b) controller placement for k =3 for NMR and bat algorithm at 0.8 SC + 0.2 CC scenario

In figure 4.12 (a), the two controllers were placed at Delhi (node 8) and Mumbai (node 3). Controller at Delhi was connected to 9 switches located at node 4, 5, 8-14 while Mumbai controller controlled 7 switches, i.e., node 0-3, 6, 7 and 15. For three controllers scenario, i.e. k = 3, the controllers were placed at Delhi (node 8), Mumbai (node 3) and Bengaluru (node 7). The controller located at Delhi controlled 9 switches situated at node 4, 5, 8-14, while the controller at Mumbai controlled switches at node 0, 1 and 3, whereas the controller at Bengaluru connected switches at node 2, 6, 7 and 15. With the increase in priority on CC latency, the placements were somewhat dictated by the CC latency, as with the increase in the controller, the control path keeps on increasing even though the SC latency decreases with increase in the controllers and its ramifications was seen at global average latency.

Table 4.4: 0.8SC + 0.2CC latency for topology Ernet

Number of controllers (k)	Latency (ms)
2	4.43
3	4.825
4	5.825
5	6.50

Due to the same placement of the controllers in this scenario, both NMR and bat algorithms have same global average latency as shown in table 4.4 and figure 4.13 below. When the controllers' were increased from k = 2 to k = 3, the global average latency has increased by 8% from 4.43 ms to 4.825 ms, while for scenario when additional controller is added, i.e. k = 4, the average latency increased by almost 20% from 4.825 ms to 5.825 ms and when there were 5 controllers, there was increase in latency by almost 11% compared to 4 controllers scenario upto 6.50 ms. The observed increase in latency was due to additional path cost due to increase in number of controller. Whenever a new controller was added, the shortest path from all the previous controllers needs to be assessed and calculated, which ultimately contributes to increase in global average latency.

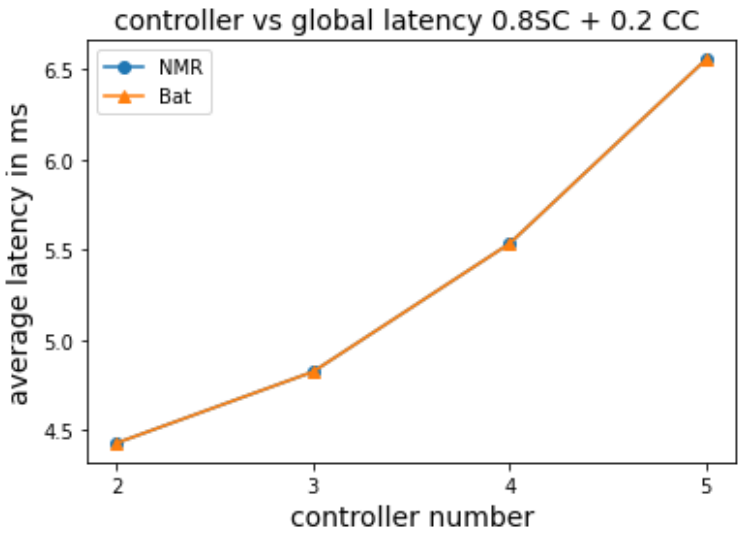


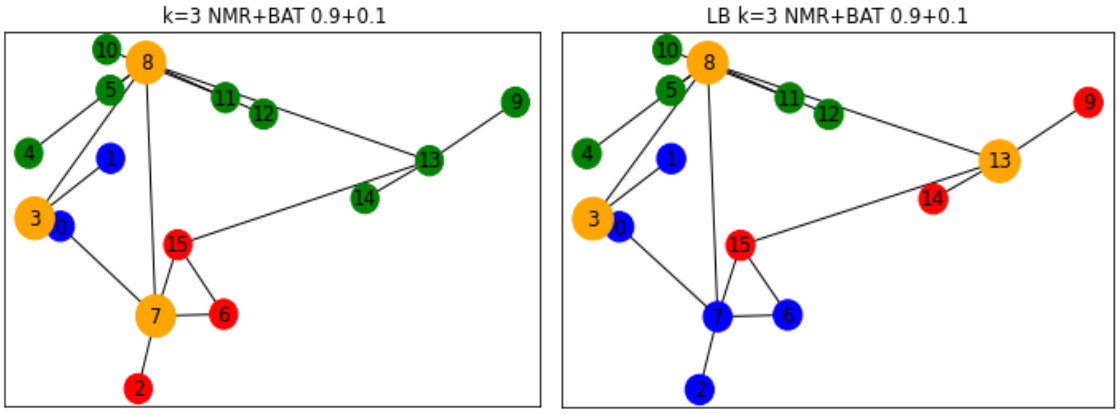
Figure 4.13: Change in global average latency with increase in number of controllers at 0.8SC + 0.2 CC scenario

Case III: Load balancing

In most of the scenarios, both NMR and bat algorithm provided the controller placement with almost balanced load condition. Here a balanced load is considered, when there is minimum difference between maximum numbers of switches controlled by the controllers in a certain placement of controllers. However, there were some specific

scenarios where algorithms completely disregard the load balancing and just focus on minimizing the latency, thus creating unbalanced load scenarios.

As shown in figure 4.14(a), when three controllers were placed to minimize the global average latency, the minimum latency was observed when controllers were placed at Delhi (node 8), Mumbai (node 3) and Bengaluru (node 7) while controlling 9, 3 and 4 switches respectively with the latency of 3.97 ms. However, when considering load distribution in above scenario, there is uneven distribution of switches in the controller with the controller at Delhi connecting maximum 9 switches whereas the controller at Mumbai connects to only 3 switches. This unbalanced load situation can be mitigated in figure 4.14(b), where the third controller was placed in Kolkata (node 13) instead of Bengaluru (node 7) and the controller at Mumbai was connected with additional 3 switches. Thus, the controller at Delhi connected switches located at node 4, 5, 8, 10, 11 and 12, whereas the controller at Mumbai was connected to switches at node 0, 1, 2, 3, 6, 7 and 15, while controller at Kolkata managed switches at node 9, 13 and 14. The load balancing scenario comes at a price of slightly increased global average latency of 4.26 ms, a 5% increase in latency, while having improved load balancing among the controllers.



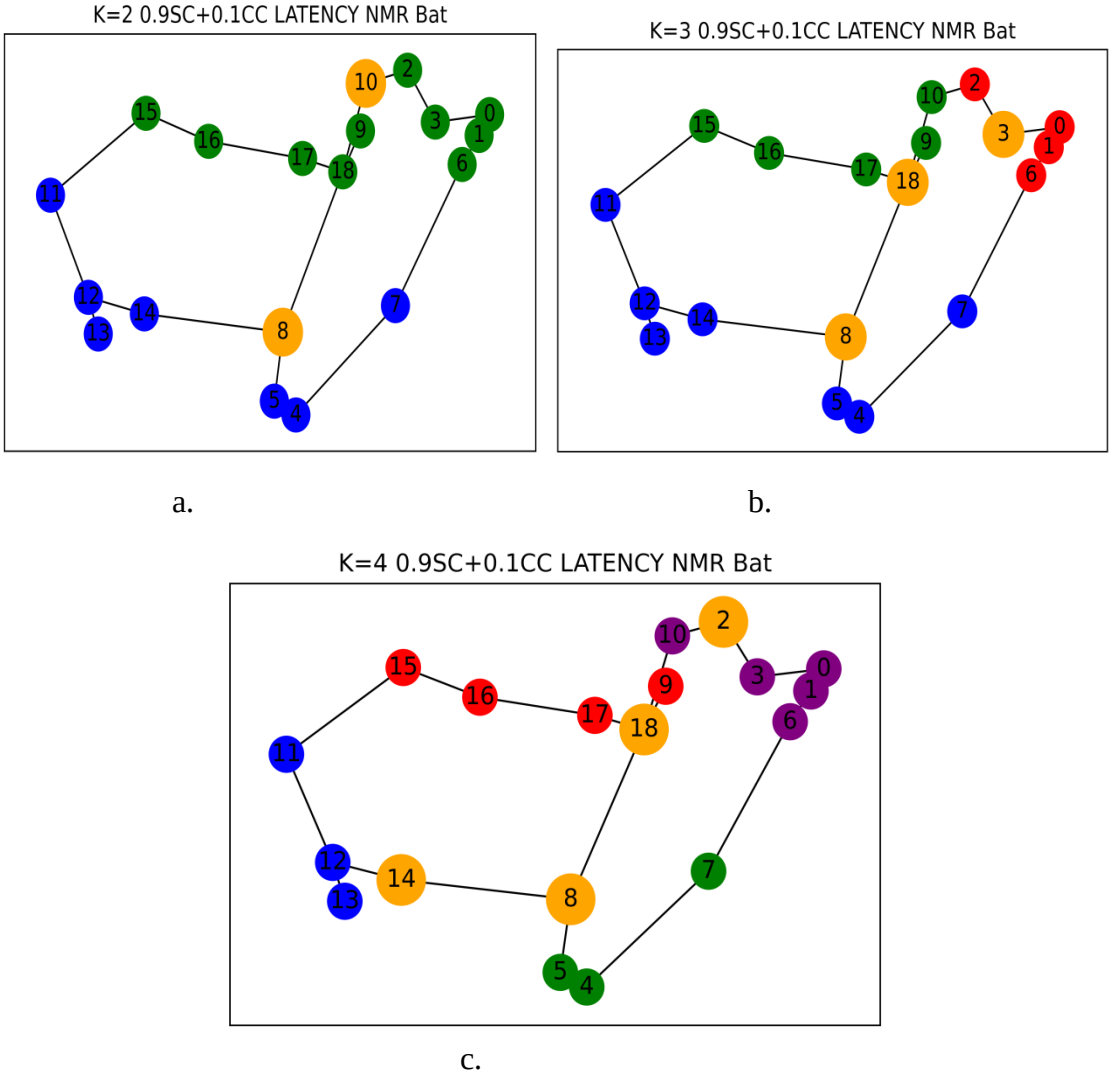
a. b.

Figure 4.14: (a) Unbalance controller placement for k =3 at 0.9 SC + 0.1 CC scenario (b) balanced controller placement at same scenario

B. Topology Savvis

Case I: 0.9 SC + 0.1 CC

The 90% priority was given for SC latency while 10% priority was given for CC latency in this scenario. Both the algorithms, i.e. NMR and bat, have found the exact same placement for all of the scenarios (k = 2 to k = 5). In the figure 4.15, the placement of controller for two, three and four controllers are shown below. On general visualization of controller placement in these scenarios, it can be seen that due to slight focus on CC latency, the controllers were placed in close distance with one another and with clear control paths between them.



a. b.

c.

Figure 4.15: Controller placement in 0.9CS + 0.1CC scenario (a) k = 2 (b) k = 3 (c) k =4

For two controllers, the placement was done at Chicago (node 10) and Dallas (node 8) with control of 11 and 7 switches respectively. Chicago's controller connected switches at node 0, 1, 2, 3, 6, 9, 10, 15, 16, 17 and 18 while Dallas controller covered switches located at node 4, 5, 7, 8, 11, 12, 13 and 14. For three controllers' placement scenario, the controllers were placed at Dallas (node 8), Saint Louis (node 18) and Pittsburg (node 3). The controller at Chicago with 11 controllers was further sub divided into two controllers with Saint Louis controlling 6 switches and Pittsburg controlling 5 switches. Further, when adding another controller to k = 4 scenario, the placement of controllers were at Dallas (node 8), Saint Louis (node 18), Detroit (node 2) and Phoenix (node 14). The 8 switches controlled by Austin in k = 3 was further subdivided into 4 switches each controlled by Phoenix and Austin. The fourth controller was placed at Detroit (node 2) which now controlled 6 switches at node 0, 1, 2, 3, 6 and 10, while controller at St. Louis controlled 5 switches at node 9, 15, 16 and 18.

Table 4.5: 0.9SC + 0.1CC latency for topology Savvis

Number of controllers (k)	Latency (ms)
2	5.79
3	5.4
4	5.49
5	5.81

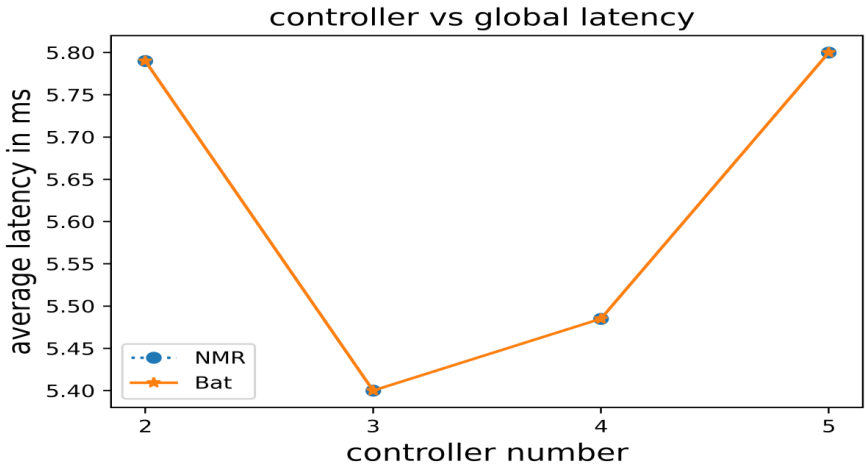
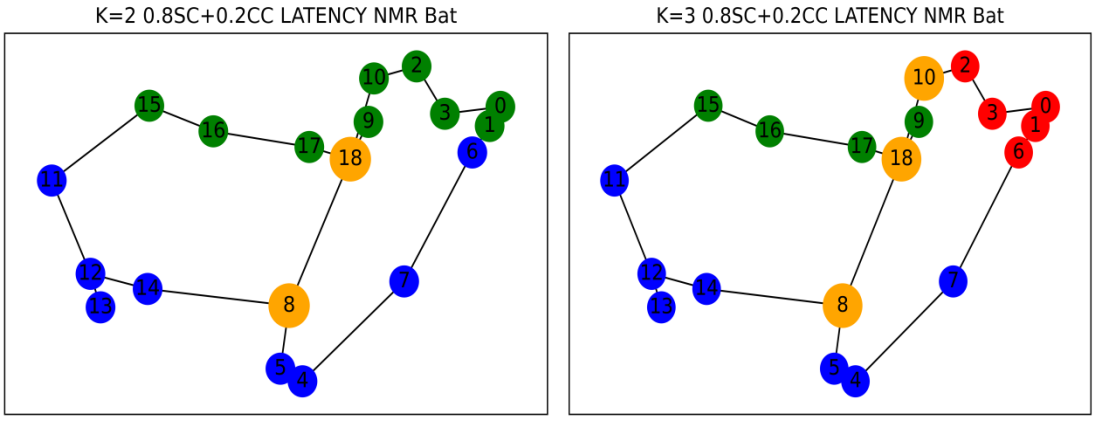


Figure 4.16: Global latency variation with number of controllers in 0.9SC + 0.1 CC

As shown in figure 4.16, when the value of k =2, the global average latency was 5.79 ms and on the addition of another controller, the global average latency reduced to 5.4 ms, which is 6% reduction when compared to k = 2. This was due to the fact that the addition of another controller largely affects the SC latency due to decrease in controller to switch distance, while addition of another controller adds the CC latency as control path increases, but as the major part of global latency constitutes of SC latency, the overall latency decreased. On addition of fourth controller, the global average latency increased to 5.49 ms, a slight increase in latency compared to when k = 3. This was because of further addition of controller causes increase in control paths between the controllers resulting in increase in CC latency, even though the addition of another controller reduces the CS latency. For k = 5, the global average latency further increased to 5.81 ms, as further addition of the controller increased the average CC latency, while SC latency does not reduce in great proportion. This can be further illustrated by table 4.5.

Case II: 0.8 SC + 0.2 CC

In this scenario, 80% priority was given to SC latency, while 20% priority was given to CC latency, which in total constitutes of global average latency. The placement of controllers for all the scenarios (k = 2 to 5) both NMR and bat algorithm were exactly same, thus the algorithms gave the same global average latency. The placement of controllers for k = 2 and k =3 is shown in figure 4.17 below.



a. b.

Figure 4.17: Controller Placement for 0.8SC+0.2CC scenario for (a) k = 2 (b) k = 3

As shown in figure 4.17(a), the placements of the controller for two controllers were at Saint Louis (node 18) and Dallas (node 8). Saint Louis controlled 10 switches while Dallas controlled 9 switches. When another controller was added, then the placement of third controller was at Chicago (node 10) and this addition of the controller subdivided the Saint Louis controller into two partitions, the controller at Saint Louis controlling switches at node 15, 16, 17 and 18, whereas controller at Chicago controlled switches at node 0, 1, 2, 3, 6 and 10. The remaining switches were controlled by the controller at Dallas. The global latency at k = 2 is 5.66 ms, while at k = 3 is 5.86 ms. Even though there is increase in controller number, the global average latency has not increased much because with increase in controller the average SC latency decreases, which contributes in 80% of global latency whereas the average CC latency increase, thus ultimately increasing the global latency at three controller scenario.

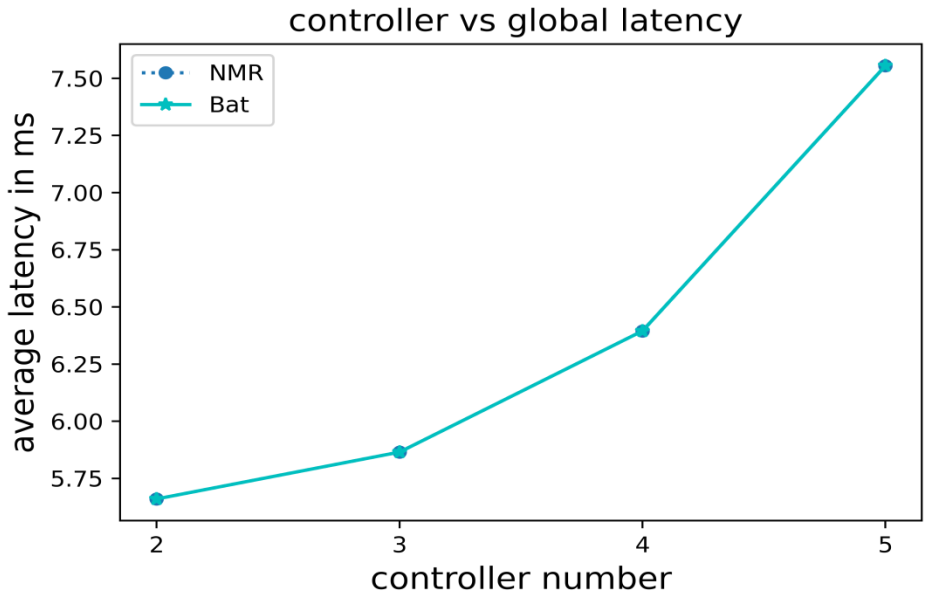


Figure 4.18: Change in global average latency with increase in controllers

As shown in figure 4.18, with the increase in number of controllers, the global average latency is increasing. At k = 2, the global average latency was 5.66 ms, while for k = 3, the global average latency was 5.86, just a mere 3% increase in the latency compared to two controllers scenario. The increase in latency was due to the addition of another controller, whose repercussion was the addition of control path, thus adding average CC

latency. Even though the additional controller mitigates the average SC latency, the average global latency increases, and the effect of considering 20% priority to average CC latency is felt. On further addition of controllers to k = 4 and k =5, the average CC latency affects the global average latency, as the value increases with increase in number of the controllers. The global latency at k = 4 is 6.395 ms while at k = 5 is 7.55 ms, almost 18% increase in global average latency compared to former. With the increase in number of the controllers, the best control paths needs to be located with all the controllers, thus even a 20% contribution on global latency added a significant part of average CC latency to average global latency. The change in latency with increase in the number of the controller can be illustrated by the table 4.6 below.

Table 4.6: 0.8SC + 0.1CC latency for topology Savvis

Number of controllers (k)	Latency (ms)
2	5.66
3	5.86
4	6.395
5	7.55

Comparison of different Scenarios

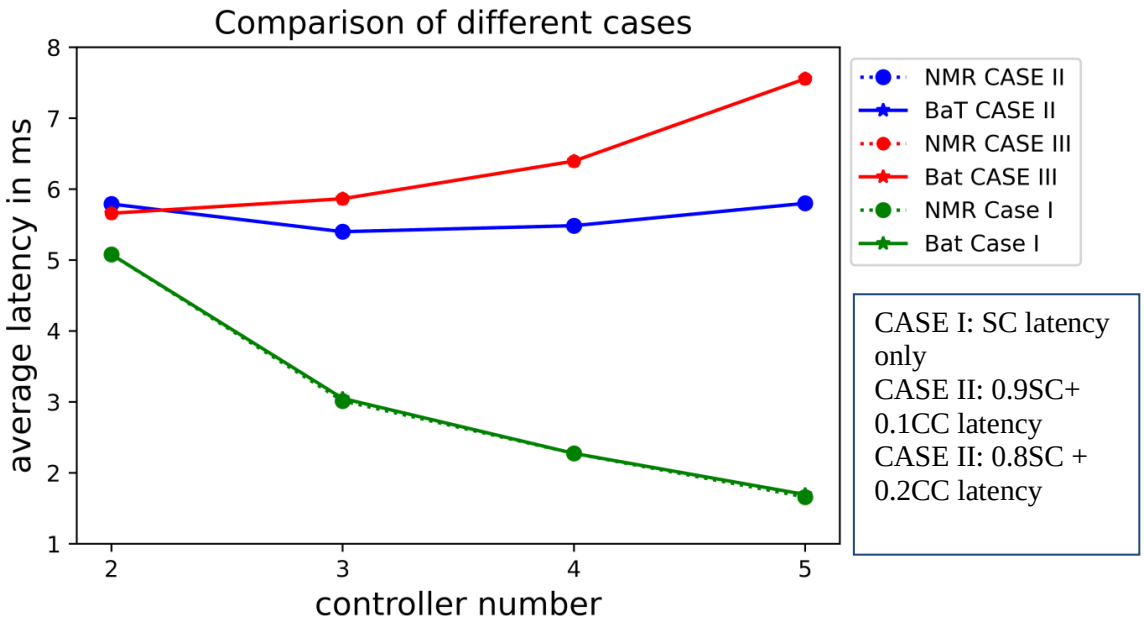


Figure 4.19: Comparison of different cases used in Topology Savvis

As in figure 4.19, three different cases of controller placement for two different algorithms are compared for topology Savvis. When considering just SC latency, i.e. case I, the average SC latency keeps on decreasing with the increase in controller number, since with increase in the controller, the average distance between the switch and the controller decreases. During case II, i.e. 0.9CS+0.1CC, with increase in controller number, the global average latency is not decreasing much with slight decline seen in k = 2 to k = 3 scenario. This decline is seen because with addition of controller, further control path is added increasing CC latency, but this addition of controller reduces CS latency. Since, CS has major contribution (90%) in global average latency, the slight decline has been observed. After k = 4, there has been slight increase in global average latency, as CC latency increases significantly with further addition of the controller.

In case III, i.e. 0.8CS+ 0.2CC, it has been observed that, with just addition of another controller, the global average latency starts to rise. This is because with increase in the controller number, the CC latency increases. Since 20% priority is given to CC latency, the effect of increase in CC latency can be seen when k has been increased from k = 2 to

k = 5. Now, even if there has been substantial decrease in SC latency with increase in the controller, the global average latency starts rising up due to influence of CC latency.

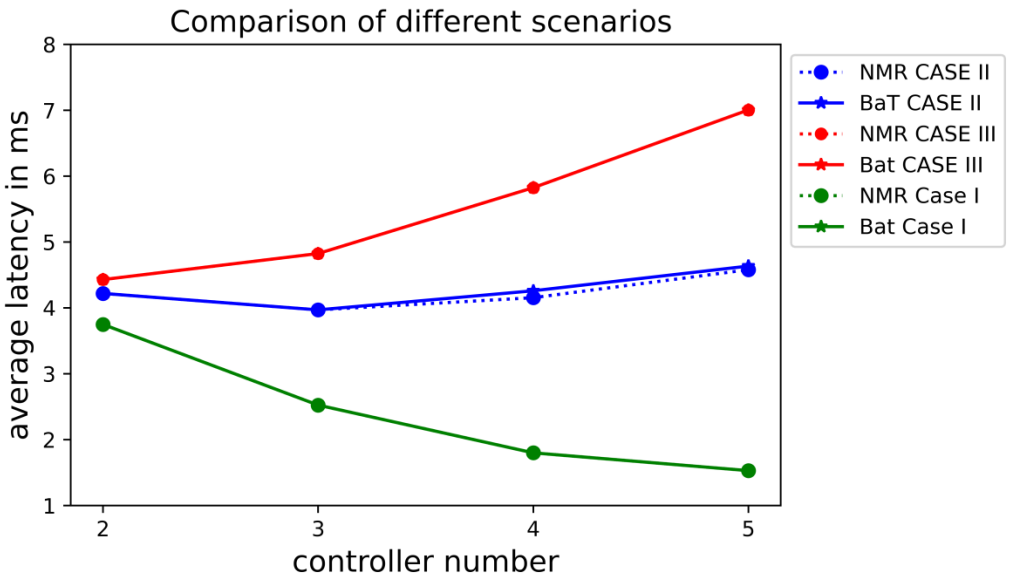


Figure 4.20: Comparison of different scenarios in topology Ernet

Three different scenarios for controller placement in Ernet are shown in figure 4.20. In the first case, i.e. considering average SC latency only represented by green line, the effect of additional controller further reduces the average SC latency. This is because with increase in the number of the controller, average distance between SC decreases and switches are more closely placed to the controller, thus reducing the propagation latency between the switch and the associated controller. Hence, when there is addition of the controller (k = 2 to 5), the average SC latency decreases.

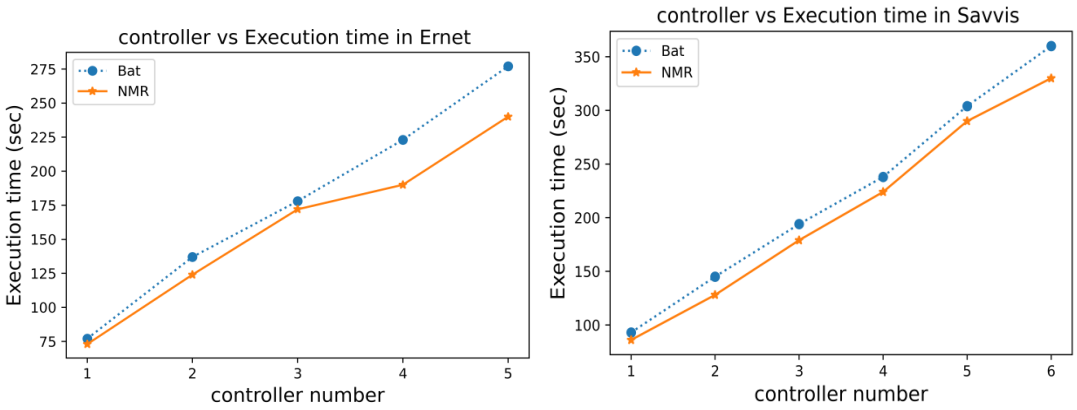
When considering the second case, i.e. 0.9SC + 0.1CC, the global average latency slightly reduces when the third controller is added (k = 2 to 3). This decline is due to decrease in average SC latency when third controller was added even though there is addition of the control path when third controller was added, increasing average CC latency. As average SC latency constitutes 90% of global average latency, thus a major decrease in SC latency decreases the overall latency even though further controller path was added. However, on addition of fourth controller (from k = 3 to 4), the global average latency starts to increase, as addition of another controller increases the control

path, thus increasing average CC latency. Although, the decrease in SC latency is seen with further addition of the controller, the overall global average latency however increases.

In the third case, 0.8SC + 0.2 CC, the global average latency starts to increase with the further addition of the controllers (k = 2 to 5). The average CC latency rises when the controllers are added as the number of control path also increases. As the average CC latency constitutes 20% of global average latency, the overall latency increases, even though the SC latency decreases with the addition of further controllers. Although, the average SC latency accounts for 80% of global average latency and with further controller addition decreases, the 20% of average CC latency affects the overall global average latency.

4.1.3 Execution Complexity

Execution complexity refers to time taken by algorithms to find the best placement of the controllers. The figure 4.20 illustrates that with the increase in number of controllers, the execution time increases, as with the increase in number of controllers, the placement matrix in the algorithm increases.



a. b.

Figure 4.21: Execution time of algorithms with increase in controller number in (a) Ernet (b) Savvis

Due to this increase in placement matrix, the possible combinations of the placement increases which results in complexity of algorithms. As seen in figure 4.21, the NMR

algorithm slightly performed better. Also, when comparing two topologies, the algorithm complexity increases with increase in number of nodes as the execution time of topology Savvis is higher than topology Ernet.

CHAPTER 5: CONCLUSION AND FUTURE WORK

5.1 Conclusion

The placement of controllers in two different topologies, i.e. Ernet and Savvis, using two different metaheuristic algorithms is analyzed in this thesis. The placement is based on global average latency, which is the sum of average SC latency and average CC latency. Three different scenarios of global latency, i.e. average SC latency only, 0.9SC + 0.1CC and 0.8SC + 0.2 CC, are analyzed for the placement of the controller. In almost all cases, both NMR and Bat algorithm performed relatively well in finding the best possible placements in the topology. Further, a placement where unbalanced load condition is seen has been optimized to load balancing, by adding the capacity of the controller and limiting the maximum number of switches the algorithm. Both the algorithms did find the best placement for the controller in almost every scenario with minimum latency. Only when considering the execution time, the performance of the NMR is slightly better than bat algorithm.

5.2 Future work

This thesis is limited to finding the controller placement considering average SC latency, average CC latency and load balancing. Load balancing is based on the assumption that every controller has a fixed capacity upto which the processing latency can be considered negligible. Thus, if the difference between the maximum and minimum number of switches connected can be reduced, a slightly load balanced placement can be achieved, as it reduces the maximum controller utilization for each controller. Further works can be done to bolster the reliability, fault tolerance and minimize the cost of placement. As increasing the number of parameters to optimize is a tradeoff since with increase in parameters results in placement trying to focus on multiple objectives but ultimately compromising one parameter to focus on another parameter. The work can be extended further by considering the processing latency of the controller which is the major limitation of this thesis.

References

[1] D. Kreutz, F. Ramos, P. E. Veríssimo, C. E. Rothen, C. E. Rothenberg, S. Azodolmolky and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE,* vol. 103, pp. 14-76, Jan. 2015.

[2] N. McKeown, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, pp. 69-74, April 2008.

[3] Z. Guo, "STAR: Preventing flow-table overflow in software-defined networks," *Computer Network,* vol. 125, pp. 15-25, October 2017.

[4] A. Ganjali and Y. Tootoonchian, "HyperFlow: A distributed control plane for open flow," *Proc. Internet Netw. Manage. Conf. Res. Enterprise Netw.*, pp. 3-10, 2010.

[5] S. Ganjali and H. Yeganeh, "Kandoo: A framework for efficient and scalable of offloading of control applications," *1st Workshop Hot Topics Softw. Defined Netw.*, pp. 19-24, 2012.

[6] T. Koponen, "Onix: A distributed control platform for large-scale production networks," *OSDI,* pp. 1-6, 2010.

[7] B. P. Rao and R. Killi, "Controller placement in software defined networks: A comprehensive survey," *Computer Networks,* vol. 163, no. 106833, 2019.

[8] B. Heller, R. Sherwood and N. Mckeown, "The controller placement problem," *inProc. HotSDN,* pp. 7-12, 2012.

[9] Y. Hu, W. Wendong and X. Gong, "Reliability-aware Controller Placement for Software-Defined Network," *China Univeristy Telecommunication Post,* pp. 92-97, 2012.

[10] M. Tanha, D. Sajjadi and J. Pan, "Enduring node failures through resilient controller

placement for software defined networks," *Global Communications Conference (GLOBECOM)*, pp. 1-7, 2016.

[11] G. Yao, J. Bi and Y. G. L. Li, "Reliability-aware controller placement for software-defined networks," *Integrated Network Management*, pp. 1339-1342, 2014.

[12] C. Gao, H. Wang, F. Zhu and L. Zhai, "A Particle Swarm Optimization for Controller Placement Problem in Software Defined Network," *Algorithm and Architectures for Parallel Processing*, vol. 9530, 2015.

[13] Y. Li, W. Sun and S. Gaun, "A Firefly Inspired Controller Placement Problem in Software Defined Network," *2nd International Conference on Computer and Communication Engineering Technology(CCET)*, pp. 254-258, 2019.

[14] K. S. Sahoo, A. Sarkar and S. K. Mishra, "Metaheuristic solutiions for solving controller placement problem in SDN-based WAN architecture," *ICETE-2017*, pp. 15-23, 2017.

[15] L. C. W. W. a. Y. Z. Y. Zhang, "A survey on software defined networking with multiple controllers," *Netw. Comput. Appl.*, vol. 103, pp. 101-118, Feb. 2018.

[16] K. S. Sahoo, S. Sahoo, A. Sarkar, B. Sahoo and R. Dash, "On the placement of controllers for designing a wide area software defined networks," *IEEE Region 10 Conf. (TENCON)*, pp. 3123-3128, Nov. 2017.

[17] A. k. Singh, . S. Maurya and S. Srivastava, "Varna-based optimization:," *Varna-based optimization:*, vol. 14, no. 3, 2020.

[18] R. Salgotra and U. Singh, "The naked mole-rat algorithm," *Neural Computing and Application*, pp. 8837-8857, 2019.

[19] F. J. R. a. P. M.Ruiz, "On reliable controller placements in software defined networks," *Computer Communications*, vol. 77, no. 41-51, 2016.

[20] Y. Li, S. Guan, C. Zhang and W. Sun, "Parameter Optimization Model of Heuristic Algorithms for Controller Placement Problem in Large-Scale SDN," *IEEE Access* , vol. 1, no. 1, 2020.

[21] J. Zhao, H. Qu and J. Zhao, "Towards controller placement problem for software-defined network using affinity propagation," *Electronics Letters,* vol. 53, no. 14, pp. 928-929, 2017.

[22] S. Knight and M. Roughan, "The Internet topology zoo," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 9, pp. 1765-1775, Oct. 2011.