Journal of Cases on Information Technology
Volume 25 • Issue 1

An Improved Switch Migration Method-Based Efficient Load Balancing for Multiple Controllers in Software-Defined Networks

Muktar Abdella Jiru, Adama Science and Technology University, Ethiopia
Ketema Adere, Adama Science and Technology University, Ethiopia*
T. Gopi Krishna, Adama Science and Technology University, Ethiopia
https://orcid.org/0000-0002-9552-1575
Janaki Ramulu Perumalla, Adama Science and Technology University, Ethiopia

ABSTRACT

The present work proposed an improved switch migration method (ISMM) for balancing the controller load to migrate the switch from overloaded to under-loaded controller. This method is used on each controller during controller load balance in order to decide the state of controllers. The implementation was conducted using the Mininet simulation tool. The performance of the proposed solution has been evaluated by using throughput, response time, and packet loss metrics. Accordingly, the simulation results indicated that this method has improved throughput by 7.6% over SSMS and 1.8% over CAMD, improved response time 8.3% over SSMS and 4% over CAMD, and improved the packet loss 8% over SSMS and 1.3% over CAMD during the incoming traffic load between 501p/s and 5000p/s. Thus, the performance of ISMM has shown efficient results over SSMS and CAMD among all assessed metrics by balancing the load between controllers.

KEYWORDS

CAMD, Controller Load Imbalance, ISMM, Multiple Controller, SDN, SSMS

1. INTRODUCTION

The internet has led to the development of a digital society in which almost everything is networked and accessible from anywhere. This has been grown rapidly and used for a variety of real-world applications such as web services, database access, real-time audio-visuals communications, and broadcasting. However, despite their widespread use, traditional IP (Internet Protocol) networks are very difficult to manage and still a complex. It is a challenging to configure the network according to the pre-defined policies and to reconfigure and also leads to load imbalance between network devices (Kreutz et al., 2014).

In order to overcome the limitation of traditional networks, the concept of SDN has been emerged recently. It is the new network architecture to enhance the traditional network, centralized management,

DOI: 10.4018/JCIT.326136 *Corresponding Author

reduced hardware costs both CAPEX and OPEX, a highly scalable, vendor-neutral that build to enable operators, error-free, flexible management of networks, dynamically configurable network that easily adapts to changing network requirements (Eissa et al., 2019; Neghabi et al., 2018; Zhou et al., 2014). SDN is a new paradigm of a network that separates the control plane from the data plane for simpler and flexible management of the network. It makes networks programmable via Application Programming Interfaces (APIs). In traditional network both control and data planes embedded together in the same network device whereas SDN separate these planes and use the OpenFlow protocol for communication between the control and data planes. The data plane comprises the network devices, in which the controller acts as a brain that centralizes network intelligence. SDN contains two types of the controller in its architecture i.e. single (only one centralized controller) and multiple controllers.

In the early stages of SDN design, a single controller manages the entire network. In Fig. 1 that adapted from (Paliwal et al., 2018), controller (C1) manages four switches in the network. When the source host sends a new packet to the switch (S1), the switch cannot perform the forwarding function due to the lack of sending information for the new packet. Then, the S1 sends messages packet-in to the C1 to receive the sending for the new packet. C1, the switch forwards the packet to the next device. Finally, the packet successfully reaches the destination host. The controller plays a significant role in the transmission of the data traffic. However, such controller has several problems such as single point of failure (SPoF), which cannot handle a huge number of requests from switches because of limited controller capacity.

Therefore, it is necessary to deploy multiple distributed SDN controllers to overcome the limitation of being limited to a single SDN controller (Hamdan et al., 2021).

Multiple controllers in SDN referred where more than one controller are working in collaborate manner. In this regard, the large-scale network and heterogeneous SDN must be split into several sub-domains, and those domains require at least one controller. Compared to single controllers, the multiple distributed controllers have several advantages in terms of scalability, reliability, availability, and high performance with even the increased demand for requests. In Fig.2 adapted from (Muluye, 2020), multiple controllers became a new SDN paradigm that solves the single controller problem. In the below figure, C1 and C2 share a certain logic in a logically uniform way, so that both C1 and C2 explicitly activate sending routes on all relevant switches when new packets arrive at S1, which means that it can adequately relieve the flow processing load of a single controller. Further, when network traffic increases, load balancing among the given controllers is required to be addressed (Mamushiane et al., 2018).

The available controller load balancing method is commonly grouped into controller optimization and switch migration methods (Adekoya et al., 2020). Under controller optimization technique, the

Figure 1. Single controller

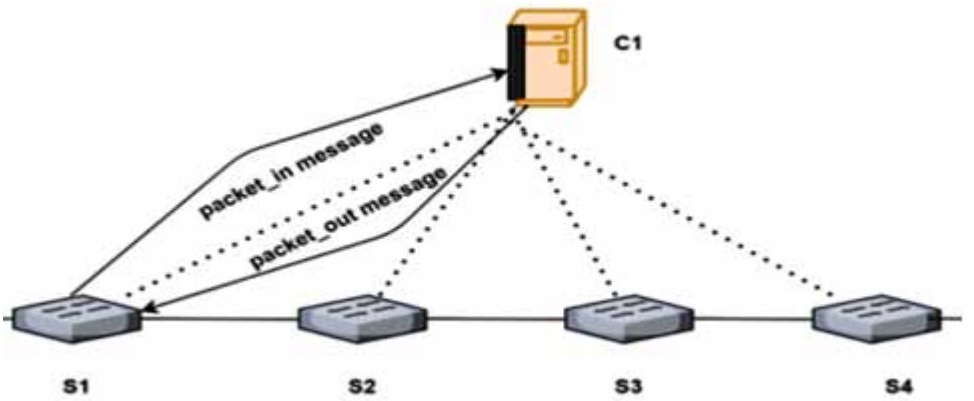
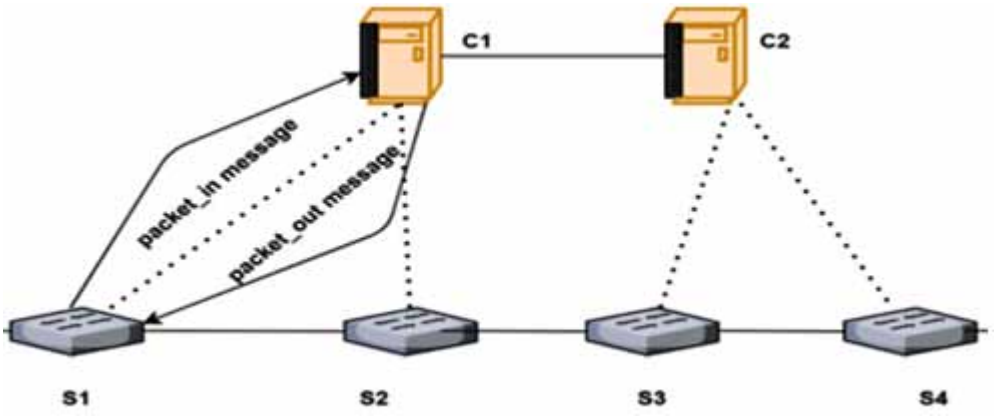


Figure 2. Multiple controllers



optimized position and the optimized number of controllers will be determined within the network, nevertheless, this may suffer in the real time change since it only optimizes controller nodes. On the other hand, to balance the traffic load of the controller the switch migration approach is considered in which an efficient switch selection and target controller for load shifting will be demanded.

Therefore, the problem of load balancing in the SDN with multiple controllers has become an interesting research challenges. In multiple controllers, if the incoming traffic load to the controller was not properly balanced and monitored between each controller, it would cause overloading in some controllers whereas the other controllers are under loaded (Filali et al., 2019). That could be brought an uneven load distribution among controllers that reduces the expected performance achievements. So, there is an urgent need for an efficient load balancing method which can able to solve the above problem in order to improve system performance. By taking this into account, two developed SSMS and CAMD methods are tried to address these problems. However, both methods were not effective when the incoming traffic load is high that lead to a performance reduction in the overall system.

To overcome such problems, this paper proposed dynamic switch migration method also called ISMM. First, the load is measured and judgment made, second, the switch is selected; third a target controller must be specified to send the migration request, then the request is checked by the controller and if possible it is accepted then, carryout switch migrations from the overloaded controller to under loaded controller. The proposed solution then balances load distribution among the controllers to achieve efficient load balancing and improve the network performance.

2. RELATED WORKS

The implementation of SDN faced several challenges with regard to the load balancing between multiple controllers, which give an overview of the whole network (Neghabi et al., 2018). Zhou et al (Zhou et al., 2014) suggested DALB algorithms for solving the problems. During the switch migration, this method was selected the nearest controller from the set of underloaded controllers to accept the load change. In this algorithm traffic load between (501p/s and 5000p/s, inclusive) and traffic load between (1p/s and 499p/s, inclusive) force on the DALB algorithm was the choice of the next controller to accept the load change.

Wang et al. (2018) proposed an initial model for all switch migration techniques. The exchange between controllers to have the life and safety properties according to the OpenFlow standard messages include start migration, role request, role reply, flow mod, barrier messages. Due to the exchange of messages between controllers before the migration, the response time on this was increased

considerably controller also in the overload state. This limitation model due to the load on the target controller was not taken into account in this model.

Xu et al. (2019) investigated about BalCon and BalConPlus which are the dynamic allocation between switches and controllers to improve the efficiency of handling fluctuations in traffic load. The authors suggested that BalCon and BalConPlus achieve controller load balancing between SDN controllers with low migration costs. BalCon is an appropriate for situations where the network was not require serial processing of switch requests. The simulation showed that BalCon and BalConPlus decrease the load imbalance problems when only incoming traffic load was mice flow with little computing effort.

Hu et al. (2017) a two Staged Switch Migration Strategy (SSMS) is proposed in order to dynamically adjust the controller load. First migration domain has been addressed by genetic algorithm then sequential migrating of multiple switches is integrated. This method decide overloaded and underloaded controller in order to migrate several switches, however, the evaluation of controller response time, throughput and packet loss are not considered. It is also inefficient when incoming traffic is 500-5000p/s. Further, in Hu et al. (2019) EASM is proposed to balance controller loads and improves migration efficiency. The core of EASM is to migrate switches and balance controller loads considering performance, response time, load balancing rate, and migration efficiency. The outcomes depict that EASM at once achieves a low controller response time, high controller performance, low migration costs, and a better load balancing rate. Nevertheless, not consider number packet loss and also not efficient under high load incoming traffic loads.

Wang et al. (2017) raised migration efficiency improvement by using SMDM. Based on the best migration efficiency situations, the SMDM algorithm was designed by the greedy method, and the algorithm consists of two phases which are load balancing detection and migration actions generation. A greedy-based algorithm was developed that offers the possible migration action options when a load imbalance occurs between the controllers. The load was calculated from the number of packet-in messages and the minimum path costs from the switch to the controller. The decision to migrate the switch is based on this. The output of this step provides a set of immigration controllers and emigration controllers.

Sahoo and Sahoo (2019) proposed was discussed based on the CAMD algorithm that the least loaded controllers are selected to accept the load change during switch migration. With this method, when a controller becomes overloaded, several witches must migrate from the domain of the current controller to another domain of the unloaded controller. After the static migration, the new controller act as the master controller and the old controller as the slave controller, however, it is not effective when the incoming load traffic is under elephant flow.

In addition to the above-listed related work, more papers were reviewed by considering switch migration strategies toward controller load balancing (Abdelaziz et al., 2017; Al-Tam & Correia, 2019; Al-Tam & Correia, 2019; Al-Tam & Correia, 2019; Hu et al., 2018; Sahoo & Sahoo, 2019; Xue et al., 2019; Zhou et al., 2018). In this regard, still the existing works have several shortcomings in order to realize better performance of controller outputs such as throughput, response time, and packet loss rates. Considering the importance of such matters into account, the present work tried to fill the gap by incorporating better strategies in order to execute the switch migration.

3. PROPOSED ISMM FOR EFFICIENT LOAD BALANCING

In order to address the major research challenges related to load balancing in multiple controllers, the following methods and techniques were developed. First model formulation and assumption will be discussed followed by the overall design of the proposed ISMM. Then, the flow chart and generalized system pseudocode will be discussed in details.

3.1 Model Formulation

The network was described according to the relevant knowledge of the graph concept. The entire topology is presented by an undirected graph, i.e., G = (V, E), where V and E are a set of nodes and links respectively. There are M controllers in the network, and the controller set is C and N switches in the networks, and the switch set S, so |V| = M+N. It assumes that a load of all controllers has been improved in advance and each controller manages several switches in the sub domain. Let, Ck = {C1, C2, . . ., Ck} be the set of controllers and Si = {S1, S2,. . .,Sm} be the set of switches, in general V = C ? S. The load on the controller is dynamic in nature. We select the switch Si from a set of switch Si = {S1, S2,. . ., Sm} in the overloaded controller with the highest load and migrate it to the target controller Ck. Then, let, Si?Ck, represents the switch set Si managed by the controller Ci. When the controller gets overloaded, a set of switches Si, migrate to target controller Ci. In the switch migration mechanism, after load shifting, the current control domain becomes the new master domain whereas the previous one automatically goes to the slave mode.

The following general assumptions have been considered under our proposed system (Adekoya et al., 2020; Hu et al., 2019; Neghabi et al., 2018): the controllers can talk with each other to share the load and each switch in the network can access each of the controllers to migrate the switch from one controller to another, once a switch has chosen to migrate it cannot revert to the previous controller's domain until the new controller's domain has met all of its control requirements. only one controller is assigned to each switch as master controller and the rest runs in equal or slave mode, all the controllers cannot be overloaded for the period t, i.e., only one switch can migrate, and finally, to simplify our experiment, all controllers have the same computational capacity

Further, we assume that controller is usually a plane in SDN design that executes requests to improve network management and application performance. It uses protocols to express switch where to send packets. Controllers in SDN send load according to forwarding rules set by a network operator, which minimizes manual configurations for single network devices. Any communication between the application and the data plane must go through the controller (Mostafavi et al., 2020; Zhu et al., 2019). The controller exchanges data with applications such as load balancers or firewalls through northbound interfaces. The network devices communicate with controller using an South Bound Interface (SBI), i.e., OpenFlow protocol (Abdullah et al., 2018).

OpenFlow protocol is the most popular southbound API protocol that permits communication among control and data plane via the OpenFlow channel. OpenFlow protocols have been managed since 2011 by ONF, an organization dedicated to introducing and promoting SDN and NFV. There is a different version of OpenFlow, but a multi-controller architecture is only defined beginning from version 1.2 of the OpenFlow protocol to improve communication among controllers and network devices.

Today, there exist many SDN controllers which include Open Day Light (ODL), Open Network Operating System (ONOS), RYU, Beacon, Floodlight, NOX, POX, Maestro, Kandoo, and others (Miyagi et al., 2004). From the provided list of controllers, ODL and ONOS support multiple controller architectures while the others are only supporting a single and centralized controller. On the other hand, when we compare the latter two, there are many reasons to choose ODL as it is the most suitable controller that can be customized and automated networks of any size and scale. Under such context, our developed system becomes a robust and open-source code that meets most SDN and NFV requirements in order to support the powerful IT industrial demands. It also provides novel functionalities as a new platform, and implements some of innovative issues and modularity concerning the other controllers.

3.2 Overall System Design

In the multiple controllers SDN network, there is static traffic that varies with temporal and spatial characteristics that are prone to controller load imbalance. There is also unbalanced load distribution

between multiple controllers in which the current work has improved the existing switch migration aspects. Thus, in the present work, we have proposed an improved switch migration method that efficiently balances traffic loads across each controller. Figure 3 shows the overview of the switch migration process between two controllers, namely, controller 1 and controller 2. In this switch migration process, switch 3 is migrated from controller 1 to controller 2 to balance the load between them.

The overall system design of the proposed method is shown in Figure 4, which consists of four modules that run locally on each controller in the network topology. The modules in the design include

Figure 3. Overview of switch migration method

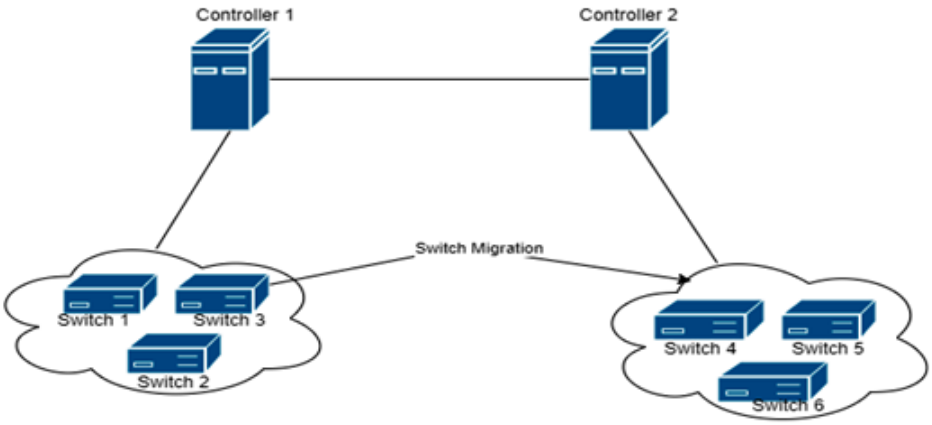
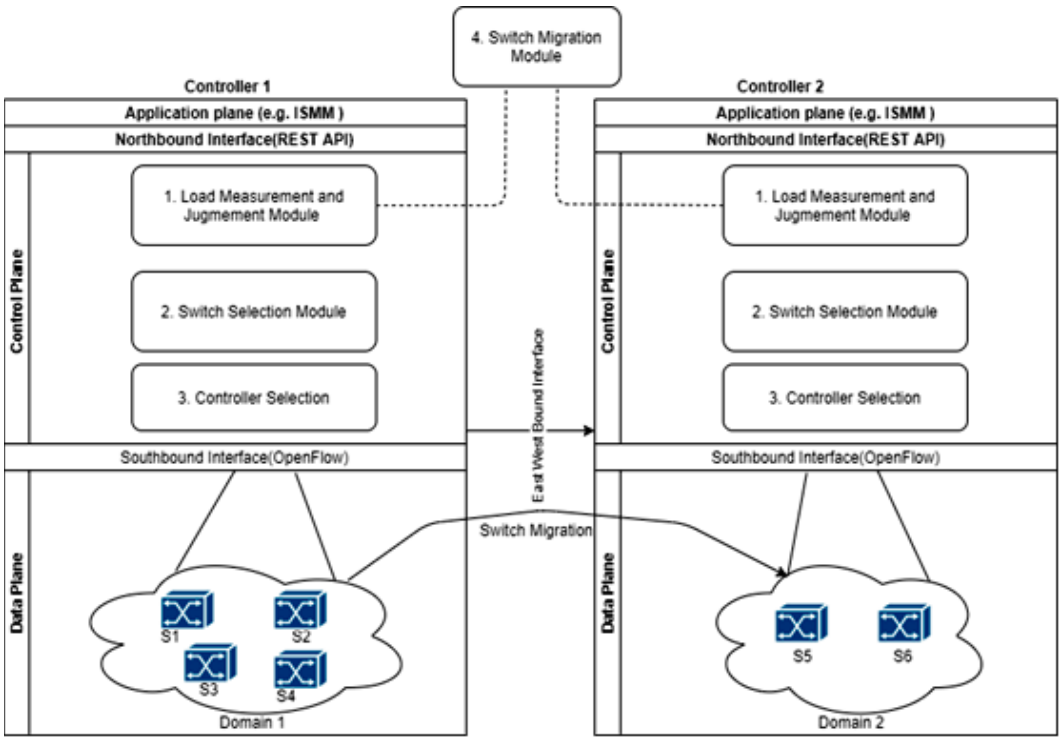


Figure 4. Overall system design of proposed solution



the load measurement and judgment module, the switch selection module, the controllers' election module, and the switch migration module. The load measurement and judgment module checks immediately whether the controller load surpasses the predefined threshold and make decisions. Each controller in the network design collects all controller loads and calculates the controller's load mean and variance. These decide the set of underloaded controllers in the network topology. Based on this the controllers which load capacity is less than or equal to controllers load is defined as a set of underloaded controllers.

3.3 Flow Chart of Proposed ISMM

Proposed ISMM, calculates a load of each controller regularly, which is calculated from the incoming traffic loads (flow rate). When the controller load exceeds the threshold, the switch with the highest load and target controller is selected and turn on the migration, the request will be sent to the target controller, if the target controller can accept the new switch request, the migration request will be accepted by the target controller and the switch will be migrated to the target controller and the role of controller is changed from master to slave and vice versa. The flowchart of the proposed ISMM is revealed in Figure 5.

Figure 5. The flow chart of proposed ISMM

3.4 Generalized Proposed Pseudocode

The generalized pseudocode of the proposed ISMM for efficient load balancing in SDN multiple controllers was described in algorithm 1.

Algorithm 1: Pseudocode of the Proposed ISMM

```
Step 1: Input: C, T, CLcur
Step 2: Output: Balanced Multiple Controllers
Step 3: if (CLcur > T) then
Step 4: prepare the Load Measurement and Judgment Module
Step 5: prepare Switch Selection Module
Step 6: prepare Target Controller Selection Module
Step 7: Send Role Request to Controller
Step 8: if (Role Request Accepted) then
Step 9: Switch Migration Module do (Cj, Sk)
Step 10: Update C (both current and target controller)
Step 11: end if
Step 12: end if
Step 13: Return Balanced Multiple Controllers
```

where, C: a set of Controller, T: Controller load Threshold, CLcur: Current Controller Load. The packet-in message from switches to the controller is taken as the input load from a switch. All controller in the domain guesses its total load of CL(ci).

This can be achieved in real-time based on the metrics stated above with their equivalent assigned weights which can be defined in Equation 6.

$$CL_{(ci)} = \left[w1w2w3\right]\begin{bmatrix} CLband\left(ci\right) \\ CLmem\left(ci\right) \\ CLcpu\left(ci\right) \end{bmatrix} \tag{1}$$

where, w1, w2, and w3 are the weights of control plane resources (memory, bandwidth, and CPU), and CLband(Ci),CLmem(Ci), and CLcpu(Ci) represent the actual bandwidth, memory, and CPU utilized on the control plane by switch flow request.

Algorithm 2: Pseudocode for Load Measurement and Judgment Module of ISMM

```
Step 1: Input: n, T
Step 2: Output: CUL and COL
Step 3: max= NULL
Step 4: for i=1 to n do
Step 5: Read Controller Load CL (ci)
Step 6: max = max + CL (ci)
Step 7: if CL (ci) > T then
Step 8: end if
Step 9: end for
Step 10: Find Mean Load CL(Ci) = 1/n Σ_{i=1}^{n} CL(ci)
Step 11: find load variance λ = 1/n Σ_{i=1}^{n} (CL(Ci) − CL(Ci))2
Step 12: if CL (ci) < T
Step 13: CUL← Ci
Step 14: if CL (ci) > T
Step 15: COL ← Ci
Step 16: end if
Step 17: end if
Step 18: Return CUL and COL
```

where, n: Number of Controller Load, T: Threshold, COL:Over-loaded Controller, CUL: Underloaded Controller, CL(Ci): Load of Controller Ci at the time, while others can be in slave or equal mode.

$$CL\left(ci\right) = \sum_{Sk \in Ci} \theta k\left(t\right) \alpha ki \tag{2}$$

Selection efficiency is calculated as depicted in Equation 3

$$\left(\frac{1}{n}\sum_{i=1}^{n}\left(Ci\right)\right) / \max_{i=1} n\left(Ci\right) \tag{3}$$

Algorithm 3: Pseudocode for Switch Selection module of ISMM

```
Step 1: Input: C_OL
Step 2: Output: A set of Migrated Switches S_k from C_OL
Step 3: Sort Switches by their incoming packet-in load in ascending
            order.
Step 4: initialize switch set P= {}
Step 5: for ∀ S_k∈ C_OL do
Step 6: for ∀ packet-in flows S_k∈ C_OL do
Step 7: find migrated Switches
Step 8: end for
Step 9: end for
Step 10: Select Maximum Loaded Controller from C_OL based on selection
            efficiency.
Step 11: S_K = max {P_sk}, S_k∈ C_OL
Step 12: Migrate SLmigrate ≤ (CLoverloaded − CLtarger) / 2
Step 13: Return switch set P={}
```

where C_{OL}: Overloaded Controller S_k: a set of migrated switches

Algorithm 4: Pseudocode for Target Controller Selection Module of ISMM

```
Step 1: Input: S_K, C_UL
Step 2: Output: Target Controller C_j
Step 3: Get the Current Load on C_UL
Step 4: for Controller C_j∈ C_UL do
Step 5: for j=1 to n ∈ C_UL do
Step 6: Calculate load Variance using ν = (1/n) Σ_{i=1}^{n} (CL(Ci) − CL̄(Ci))²
Step 7: end for
Step 8: end for
Step 9: Check that CL (Ci) + SL (Si) ≤ C_L(C_j)
Step 10: C_TS = max {C_UL}: (C_i) + SL (Si) <CL (Cj), C_j∈ C_UL where C_j∈ C_UL
Step 11: Update C_UL with current Load Status
Step 12: Return Target Controller, C_j
```

where COL: Overloaded Controller CUL: Underloaded Controller Sk: a set of migrated switches Cj: a set of controller.

Algorithm 5: Dynamic Controller Threshold using ISMM

Step 1: ***Input:*** CL_1. $CL_2, \ldots, CL_n$
Step 2: ***Output:*** Adjustable T
Step 3: Find Mean Load $\bar{C}L_{(Ci)} = \frac{1}{n}\sum_{i=1}^{n} CL(Ci)$
Step 4: Let initial T = 1800
Step 5: If $\bar{C}L_{(Ci)} \leq$ initial T then
Step 6: T=1800
Step 7: If $CL_{(Ci)} \geq$ Initial T then
Step 8: $T=\bar{C}L_{(Ci)}$
Step 9: **end if**
Step 10: **end if**
Step 11: Return T

where T: Threshold CL1. CL2…, CLn: Controller Load List n: number of controllers

4. SIMULATION AND PERFORMANCE EVALUATION

4.1 Simulation Setup

The following simulation setup was employed in order to assess the performance of proposed method. This has been conducted on Mininet simulation tools with java based ODL controller which supports OpenFlow v1.3. The general simulation setup and parameters used are listed in Table 1.

In this study, since it is freely accessed an open-source and popular in SDN as supporting Open Flow protocol, the Mininet simulator was adopted in order to conduct the simulations. It is also a python-based simulator that runs on any platforms such as Linux, and Windows. Accordingly, to obtain the set of results, the implementation has assumed four controllers and twelve switches. The C1 contains 3 switches, C2 contains 4 switches, C3 contains 2 switches and C4 contains 3 switches designed on MiniEdit GUI. Based on the investigation C2 is overloaded and C3 is underloaded, so

Table 1. Simulation setup parameters

Parameters Description	Value
OpenFlow protocol	OF v1.3
Simulation tool	Mininet 2.3.0
Controllers	ODL
Number of switches	12
Number of controllers	4
Controller threshold	1800 p/s
Total incoming load capacity	10,000-25,000 kbps
Mice flow	1 p/s – 499 p/s
Elephant flow	501 p/s – 5000 p/s
Experiment iteration executed times	1500 times
Performance evaluation metrics	Throughput, Response time, and Packet loss

its need to migrate the switches with the highest load to underloaded controller C3. Through the simulation run, the controller threshold is set to initial and dynamically updated if needed based on the bandwidth, memory, and CPU of controllers. The traffic loads between 501p/s- 5000p/s were used in studies that act as elephant flows in network topology because it improves the overall throughput and brings more stable balanced networks that mice flow.

The network topology of the current developed Mininet simulator with 4 controllers and 12 switches in MiniEdit GUI (Figure 6). The multiple controllers were designed network topologies in MiniEdit GUI. For multiple controllers, the configured the controllers in MiniEdit with the port number 6633, controller type OpenFlow reference, and remote IP address 127.0.01. After configuration, the capacity is filled with a different number of packet-in messages. The switch in this topology communicates with four controllers to balance the load between controllers. The load in multiple controllers' topology is balanced in a better way. Lastly, the designed topology was run and gets the result.

Figure 7 shows the remote IP, Controller Type, remote port, Protocol of each controller after run on MiniEdit GUI. The proposed model is for the starting 4 controllers. The capacity of each

Figure 6. Network topology in MiniEdit GUI of 4 controllers and 12 switches

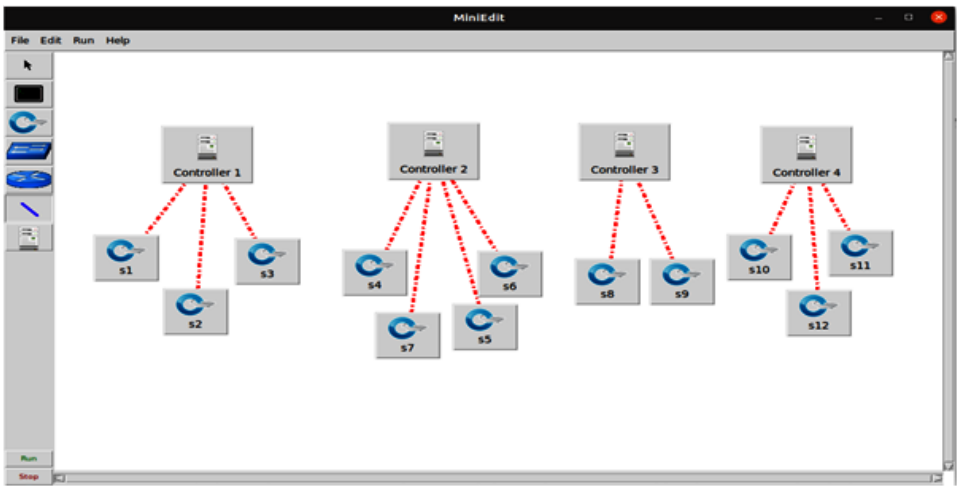
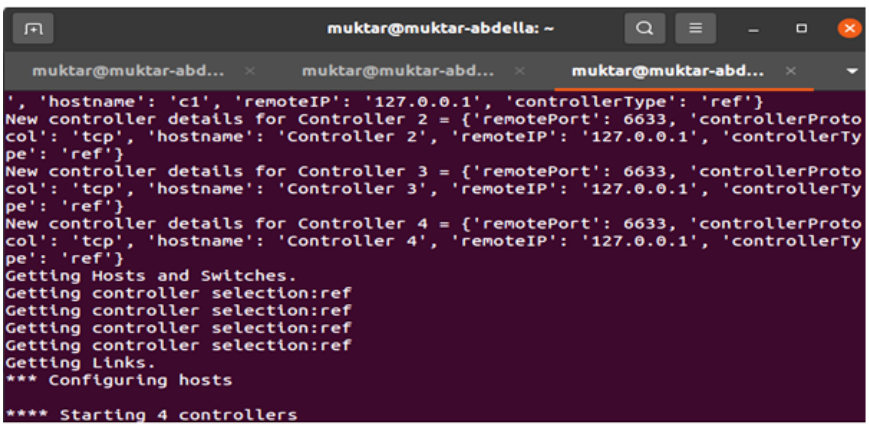


Figure 7. Sample of four controllers execution



controller is equal, but the load or packet-in message sent from switches to each controller is differ because traffic varies from time to time. After 4 controllers start initial controller threshold is set as 1800 p/s and the experiments execute 1500 times.

Another issue is the reason of metrics selection for the study. The authors have selected throughput, response time, and packet loss based on the gaps of the studies in the related works and to study filling the gaps. As the number of controller increases cost of configuration is also increase so we limit out design on 4 controllers and twelve switches in our works.

4.2 Performance Evaluation and Discussion

The ISMM will run all four modules on each controller to balance the load between each controller. First, the load measurement and judgment module are started for the computation of the load. If the load computed load is more than the predefined controller threshold, the controller is overloaded and dynamically updates the controller threshold. If controller load is less than controller threshold, subsequently, the ISMM will start the switch selection module and the controller selection module simultaneously, and then perform the switch migration to balance the load. The switch with the highest load is selected for migration and the most appropriate controller is also selected for accepting the incoming load. Therefore, ISMM improved over other methods.

The simulation results show the comparison of the SSMS and CAMD with the proposed ISMM. CAMD is the least loaded controller from the underloaded controller that may only be efficient in accepting load shifting when the incoming traffic flow is mice flow. SSMS selecting the optimal migration objects from the underloaded controller for accepting load shifting that used two-stage to perform the migration. The below comparison of three methods was measured based on four controllers by setting the threshold for each controller and dynamically updating the controllers' threshold based on the computational capacity of the controllers. The total in-coming loads for each controller are between 10,000kbps-25,000kbps and the initial controller threshold is 1800p/s. More of their comparison result was described in the below section.

In this study, evaluation metrics such as controller throughput, packet losses, and response time are used for conducting performance comparison.

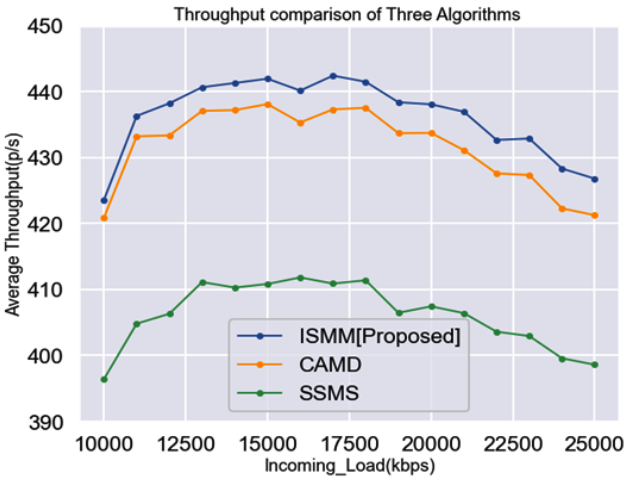
a) Controller Throughput

Throughput is the number of bits received in a given period which is the sum of tasks completed in one unit of time after performing load balancing. It determines the rate at which the computational work is done using the ISMM. The higher throughput, the better controller performance. Throughput is expressed in the below expressions.

$$\sum_{requesti}^{n}\left(time, t\right) \tag{4}$$

Figure 8 depicts the simulation results of the controller throughput of each of the reviewed works and the proposed ISMM that shows the maximum amount of packets that can be processed successfully by a controller at a given period. In our simulation, the controller throughput was predictable based on the traffic generated during the simulation process. The total incoming load used in this work was between 10,000kbps – 25,000kbps and the average throughput was between 390p/s-450p/s. The result shows the controller throughput assessments with SSMS and CAMD and proposed ISMM receive more traffic than CAMD and SSMS algorithms. The controller throughput of the proposed method increases by approximately 7.6% over the SSMS and about 1.8% over the CAMD. So, there is a big difference between the throughput of SSMS, CAMD, and ISMM results. This indicates that the proposed method achieves greater efficiency with maximizing throughput under balanced load distributions.

Figure 8. Throughput comparison of the three algorithms



b) Packet Loss

Packet loss is the number of packet lost that happen during sending packets from switches to the controller. When network switches are busy processing incoming packets, they will likely drop some of the incoming packets. The ISMM aims at achieving minimum packet loss to show better controller performance. Therefore, the packet loss can be calculated using the Equation 5:

$$P\left(loss\right) = \frac{Tx\ \ packets - Rx\ \ packet}{Tx\ \ packet} \tag{5}$$

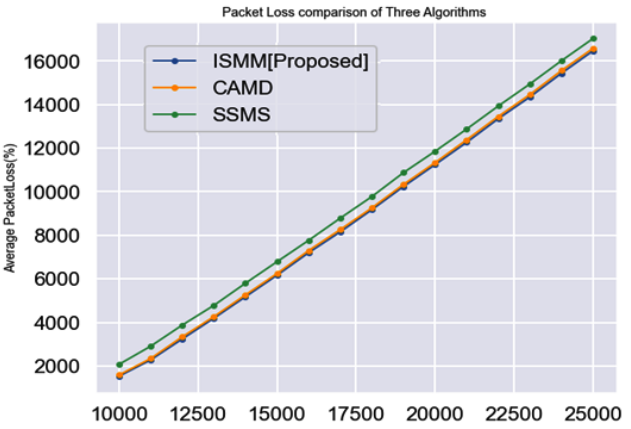
where Txpacket: Total number of transferred packets in OpenFlow switches, Rx packet: Total number of the packet that is as received, and P (loss): Packet Loss

Figure 9 depicts the simulation outcomes for the packet loss. The total incoming load used in this work was between 10,000kbps – 25,000kbps and the average packet loss was between (1 - 16000)%. As incoming load increases, the average packet lost increases for the three algorithms as described in the Figure. However, SSMS had the highest average packet loss when compared with ISMM and CAMD had little improvement in average packet loss compared with ISMM. The analysis assessed that the ISMM was 8%efficient over SSMS and 1.3% efficient over CAMD. This is so due to the effective mechanism taken improvement in this study. Accordingly, it had a fewer number of packets lost. So, the gap raised in the state of the art was filled by minimizing the number of packet losses in our works.

Response time is defined as the summation of the time request taken for the user to retrieve the results of a request. The ISMM aims at achieving the smallest controller response time to show better controller performance. It is estimated that when there is a controller load imbalance, the response time in a given network will be maximized. The ISMM aims at achieving the smallest response time to show better controller performance. All variables, in this equation, were fixed during the assessment. The accepted equation is written as:

$$\mathbf{R} = \mathbf{Rd} + \mathbf{Rt} \tag{6}$$

Figure 9. Packet loss comparison of the three algorithms



where:

$$\mathrm{Rd} = 2\left[\mathrm{D} + \mathrm{Cc} + \mathrm{Ct}\right] + \left[\mathrm{D} + \frac{\left[\mathrm{Cc} + \mathrm{Cs}\right]}{2}\right]\frac{\mathrm{T}-2}{\mathrm{m}} + \mathrm{Dln}\left[\frac{\mathrm{T}-2}{\mathrm{m}} + 1\right] + \mathrm{KT}\left(\frac{\mathrm{L}}{1-\mathrm{L}}\right) \tag{7}$$

$$\mathrm{Rt} = \frac{\mathrm{MAX}\left[8\mathrm{p}\,\frac{1+\mathrm{OHD}}{\mathrm{B}} * \mathrm{D}\,\frac{\mathrm{P}}{\mathrm{w}}\right]}{1-\sqrt{\mathrm{L}}} \tag{8}$$

In equations 6, 7, and 8, where P: pay load length, OHD: overhead ratio, B: minimum path bandwidth and W: effective window size, R: response time, Rd: propagation delay time, Rt: transmission delay time, D: round-trip delay, Cc: current processing time, m: multiplexing factor, Dln: packet-loss ratio, K: TCP timeout, Ct: server TCP processing, Cs: server processing time, T: application turns.

Figure 10 reveals the response time of the simulated three algorithms. The total incoming load used in this study was between 10,000kbps – 25,000kbps and the average response time was between (150000–550000) measured by milliseconds. ISMM selects the most appropriate controller at any phase of migration such that the maximum clustered resources are left free. Consequently, ISMM indicates that resources were well consumed and significantly reduce the controller response time. Generally, ISMM reduces the response time of the requests than the other two methods to achieve its objective and fill the gap in related works.

The comparison of three different algorithms after load balancing is revealed in Figure 11 based on traffic flow between 1 p/s – 499p/s. It shows the load of four controllers before balancing, then after using ISMM a load of each controller is balanced by migrating the switches, and it also shows the improvement of ISMM over SSMS and CAMD methods.

This shows that our work is also efficient under small-scale traffic flows. We compare ISMM with SSMS and CAMD to validate the performance of load balancing. Before applying any methods, C2 is overloaded. Both SSMS and CAMD select C3 as the underloaded controller to reduce the

Figure 10. Response time comparison of the three algorithms

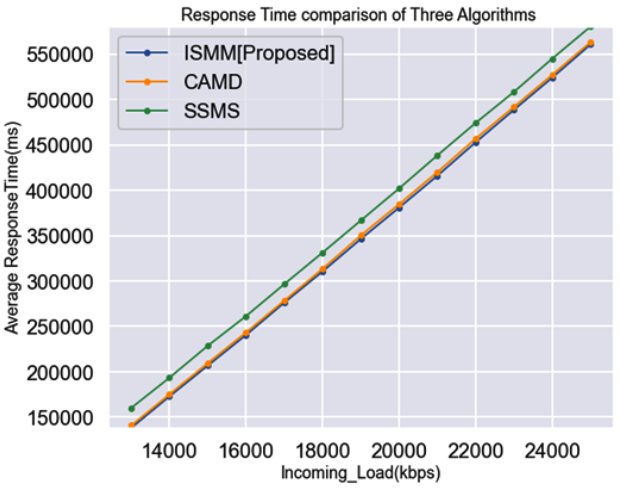
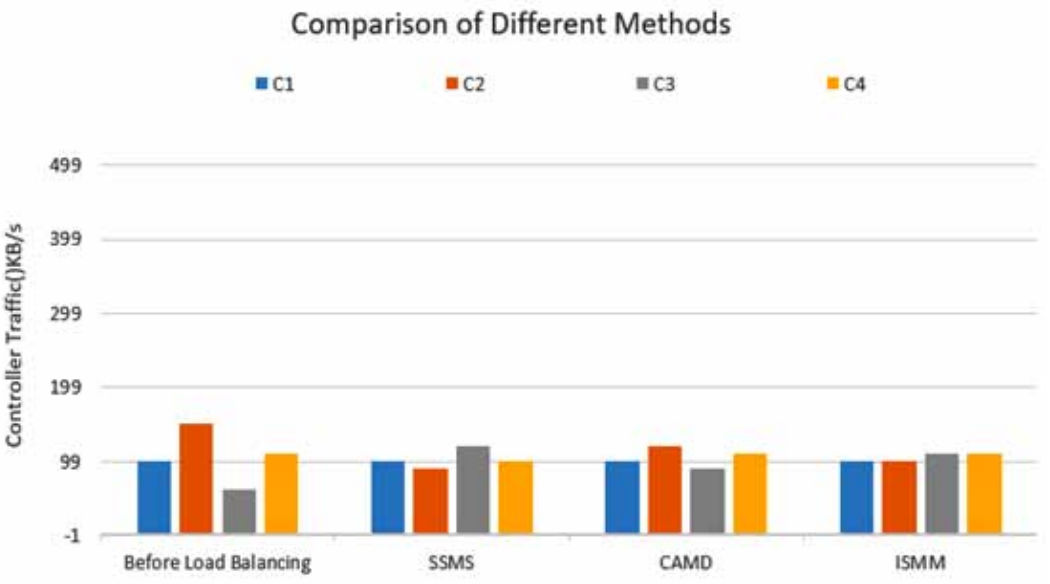


Figure 11. Comparison of three algorithms



controller load of C2. While multiple switches are migrated into C3, C3 is easy to become a new overloaded controller, so, load measurement and judgment module check the loads whether it surpasses the threshold or not, then check underloaded controllers from the controllers, then select target controller to complete migration processes to balance the load between the controller. In this case, SSMS selects minimum loaded switch from the overloaded controller which makes the switch be migrated from the overloaded controller to underload more than one. This reduces the migration efficiency and increases migration cost because it needs to migrate more than one to reduce the load on the controllers. CAMD selects the maximum loaded switch from a set of overloaded controllers

to reduce its load. This method is somewhat better than SSMS but not efficient under both low and high traffic loads.

The performance evaluation result for the three algorithms is described generally in Table 2 and the results are taken by using the average value of each result and executing the simulation 1500 times to obtain the accuracy of the experiment. The information result is taken from the simulation result directory file and the MiniEdit GUI. Based on the obtained results, one can conclude that the performance of proposed ISMM achieved better than the two previously known methods namely, SSMS and CAMD in terms of evaluation metrics such as packet losses, response time, and controller throughput. In Table 2 the individual improvements of the three methods are summarized.

This verified that the proposed ISMM was effective in balanced loads and improve the overall performance of the networks.

Therefore, this study used a simulation approach and the assessment of throughput, response time, and packet loss for the performance result assessment. The current load on every 4 controllers before any migration period was set to be 500p/s means that the initial controller's load. The processing rate to simulate a real control scenario was set to be 70%. The total incoming load was between 10000p/s – 26000p/s, while each of these iterations was executed 1500times to obtain the accuracy of the experiment. The threshold of the controller is set as 1800p/s and it is updated dynamically based on a load of a controller. The reason why we take this threshold was based on initial packet-in requests sent to controllers. All this taken information is taken for justifying our result and performed for each controller in the network domain.

As shown in Table 3, the ISMM improves given metrics with the best efficiency in comparison with SSMS and CAMD. That percentages of each metric is considered for evaluations. Therefore, the proposed solution improves controller throughput by 7.6% over SSMS and 1.8% over CAMD, while the controller response time is improved by 8.3% over SSMS and 4% over CAMD. Lastly,

Table 2. Summary of three algorithm evaluation

Average Throughput(p/s)	**Throughput of Three Algorithm**		
	SSMS	**CAMD**	**ISMM**
	392 p/s	414 p/s	422 p/s
	395 p/s	421 p/s	429 p/s
	403 p/s	428 p/s	437 p/s
	410 p/s	436 p/s	444 p/s
Average Packet Loss (%)	**Packet Loss of Three Algorithm**		
	SSMS	**CAMD**	**ISMM**
	¯ 3171	¯1981	¯ 1891
	¯ 7367	¯ 6167	¯ 6087
	¯ 12820	¯ 11620	¯ 11540
	¯ 15600	¯ 14400	1 4320
Average Response Time(ms)	**Response Time of Three Algorithm**		
	SSMS	**CAMD**	**ISMM**
	¯ 206600 ms	¯ 182600 ms	¯ 160400 ms
	¯ 346400 ms	¯ 322400 ms	¯ 300200 ms
	¯ 348600 ms	¯ 346400 ms	¯ 324000 ms
	¯ 518000 ms	¯ 494000 ms	¯ 471800 ms

Table 3. General improvement of three method by percentage

No	Performance Evaluation Metrics	SSMS	CAMD	ISMM
1.	Throughput	91.1%	96.8%	98.6%
2.	Response Time	94.1%%	89.8%	85.8%
3.	Packet Loss	97.5%	90.8%	89.5%

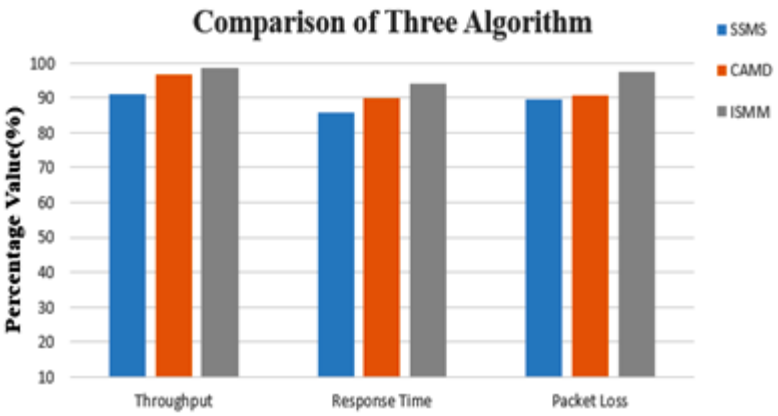
packet loss is improved by 8% over SSMS and 1.3% over CAMD, this indicates the improvement of the proposed method all aspects when we compared with the well-known methods.

Generally, Figure 12 describes the summary of the above table percentage values. It shows by what percent does the related works and proposed ISMM increase the controller performance and decrease the controller loads between four controllers in our implementation. As the graph shows there was improvement between the existing CAMD, SSMS, and the proposed ISMM in three performance metrics. The result describes that as the number of load incoming increases all three performance metrics decreased. But, this issue was handled by our proposed ISMM, in this algorithm overloaded controller migrates the switch of its domain to the underloaded controller if it surpasses the maximum thresholds. To do this migration our proposed ISMM uses a four-module includes a load judgment and measurement module, switch selection module, a controller selection module, and a switch migration module.

To sum up, the assessment of the result simulated on the Mininet simulation tool cross-checked with other methods indicates that the ISMM is the best method for solving the controller load balancing problem. For the validating the performance, the ISMM is compared with other similar works in the related research area. The study used throughput, response time, and packet loss for the performance evaluation. In the algorithm, it was assumed that for each iteration, the proposed ISMM was set the controller threshold at1800p/son all of the simulated controllers. The current load on each controller means 4 simulated controllers before any migration phase was set to be 500p/s. The processing rate to simulate a real control scenario was set to be 70%. The experiment was conducted with the total incoming load produced between 10000p/s-26000p/s with the load-size of 20, while each of the iteration was executed 1500 times to obtain the accuracy of the experiment.

The proposed ISMM solves the problem of controller load imbalance between multiple controllers. The overall performance was increased after switch migration from overloaded controllers to loaded controllers. All this process is done based on the four-module of designed ISMM which includes load

Figure 12. Comparison of three algorithms summary



measurement and judgment module, switch selection module, appropriate controller selection module, and switch migration module. The proposed ISMM will activate Module 1 when the computation of the load is more than the predefined controller threshold. Subsequently, the ISMM will call the switch selection module and the controller selection module. After that, the switch with maximum load is selected for migration, and the most appropriate controller is selected to accept the incoming load, respectively. Then, the migration of the switch is held based on the design ISMM. Hence, leading to important enhancement over the baseline works when the incoming traffic is elephant flows. The elephant flow in this study is defined between 501p/s and 5000p/s while the mice flow is defined as between 1p/s and 499p/s. The reason why the elephant flow is selected to show the ISMM works under high traffic loads or work for the large-scale network.

5. CONCLUSION

In this study, the problem of controller load balancing in which some controllers are overloaded while other are underloaded has been addressed. Such issue happened in multiple controllers SDN when unbalanced load distribution among the deployed controllers have reported and ineffective with existing static switch and controller mapping. Different kinds of literature reviews were considered to find a better solution for the given problem.

Then, dynamic ISMM is proposed in order to meet our research objective. In this regard, given switches are migrated from an overloaded to underloaded to balance the traffic load. The proposed method used four modules including load measurement and judgment module, switch selection module, a controller selection module, and switch migration module. The design of ISMM runs locally on each controller. The load measurement and judgment module in this design check immediately whether the controller load surpasses the predefined threshold and make decisions. Each controller in the network design collects all controller loads and calculates the controller's load mean. This is used to decide the set of underloaded controllers and overloaded controllers in the network topology. If the current load on the controller is above a predefined threshold it's overloaded and if the current load on the controller is below a predefined threshold it's underloaded. Secondly, the switch selection module is also becomes activated which is used to select the highest incoming loaded switch from the set of overloaded controllers for the migration procedure which improves the selection efficiency. Thirdly, the controller selection module also gets started beside modules 1 and 2 to ensure the most suitable controller from a set of underloaded controllers is selected. Lastly, the fourth module accomplishes switch migration and balance loads between controllers.

Lastly, the performance of the ISMM is also studied after simulation considered on Mininet tool. ODL controller is used because it supports a network of any size and scale, modular, and robust. Thus, we conclude that based on the conducted simulation, the obtained result is revealed that the ISMM improved the performance evaluation metrics such as throughput, packet loss and response time of the control plane when compared with SSMS and CAMD between the traffic loads of 501p/s-5000p/s.

DATA AVAILABILITY

There are no significant data in this work, but all design files are available in the thesis file.

CONFLICTS OF INTEREST

The authors declare that they have no conflicts of interest.

FUNDING STATEMENT

This works don't have any funding assistance.

AUTHORS CONTRIBUTION

Muktar Abdella Jiru: Investigating and executing the entire objective pf the work, using proper research methodology.

Ketema Adere Gemeda: Topic selection, supervising, executing the research as per the scientific principles.

T. Gopi Krishna: Development of Algorithms as per the research work and executing the program.

Perumalla Janaki Ramulu: Mathematical understanding, programs verifications, literature execution and compiling the data.

REFERENCES

Abdelaziz, A., Fong, A. T., Gani, A., Garba, U., Khan, S., Akhunzada, A., Talebian, H., & Choo, K. K. R. (2017). Distributed controller clustering in software defined networks. *PLoS One*, *12*(4), e0174715. doi:10.1371/journal.pone.0174715 PMID:28384312

Abdullah, M. Z., Al-Awad, N. A., & Hussein, F. W. (2018). Performance Comparison and Evaluation of Different Software Defined Networks Controllers. *International Journal of Computing and Network Technology*, *6*(2), 36–41. doi:10.12785/IJCNT/060201

Adekoya, O., Aneiba, A., & Patwary, M. (2020). An improved switch migration decision algorithm for SDN load balancing. *IEEE Open Journal of the Communications Society*, *1*, 1602–1613. doi:10.1109/OJCOMS.2020.3028971

Al-Tam, F., & Correia, N. (2019). Fractional switch migration in multi-controller software-defined networking. *Computer Networks*, *157*, 1–10. doi:10.1016/j.comnet.2019.04.011

Al-Tam, F., & Correia, N. (2019). On load balancing via switch migration in software-defined networking. *IEEE Access : Practical Innovations, Open Solutions*, *7*, 95998–96010. doi:10.1109/ACCESS.2019.2929651

Al-Tam, F., & Correia, N. (2019). Fractional switch migration in multi-controller software-defined networking. *Computer Networks*, *157*, 1–10. doi:10.1016/j.comnet.2019.04.011

Eissa, H. A., Bozed, K. A., & Younis, H. (2019). Software Defined Networking. *19th Int. Conf. Sci. Tech. Autom. Control Comput. Eng. STA*, 620-625.

Filali, A., Cherkaoui, S., & Kobbane, A. (2019, May). Prediction-based switch migration scheduling for SDN load balancing. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)* (pp. 1-6). IEEE. doi:10.1109/ICC.2019.8761469

Hamdan, M., Hassan, E., Abdelaziz, A., Elhigazi, A., Mohammed, B., Khan, S., Vasilakos, A. V., & Marsono, M. N. (2021). A comprehensive survey of load balancing techniques in software-defined network. *Journal of Network and Computer Applications*, *174*, 102856. doi:10.1016/j.jnca.2020.102856

Hu, T., Guo, Z., Yi, P., Baker, T., & Lan, J. (2018). Multi-controller based software-defined networking: A survey. *IEEE Access : Practical Innovations, Open Solutions*, *6*, 15980–15996. doi:10.1109/ACCESS.2018.2814738

Hu, T., Lan, J., Zhang, J., & Zhao, W. (2019). EASM: Efficiency-aware switch migration for balancing controller loads in software-defined networking. *Peer-to-Peer Networking and Applications*, *12*(2), 452–464. doi:10.1007/s12083-018-0632-6

Hu, T., Zhang, J., Wang, L., & Qiao, D. (2017, March). An improved switch migration approach to controller load balancing in sdn. In *2017 2nd International Symposium on Advances in Electrical, Electronics and Computer Engineering (ISAEECE 2017)* (pp. 436-442). Atlantis Press. doi:10.2991/isaeece-17.2017.83

Kreutz, D., Ramos, F. M., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S., & Uhlig, S. (2014). Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, *103*(1), 14–76. doi:10.1109/JPROC.2014.2371999

Mamushiane, L., Lysko, A., & Dlamini, S. (2018, April). *A comparative evaluation of the performance of popular SDN controllers. In 2018 Wireless Days (WD)*. IEEE.

Miyagi, M., Ohkubo, K., Kataoka, M., & Yoshizawa, S. (2004, April). Performance prediction method for web-access response time distribution using formula. In *2004 IEEE/IFIP Network Operations and Management Symposium (IEEE Cat. No. 04CH37507)* (Vol. 1, pp. 905-906). IEEE. doi:10.1109/NOMS.2004.1317792

Mostafavi, S., Hakami, V., & Paydar, F. (2020). Performance Evaluation of Software-Defined Networking Controllers: A Comparative Study. *Computer and Knowledge Engineering*, *2*(2), 63–73.

Muluye, W. (2020). A review on software-defined networking distributed controllers. *Int. J. Eng. Comput. Sci.*, *9*(2), 24953–24961. doi:10.18535/ijecs/v9i2.4439

Neghabi, A. A., Navimipour, N. J., Hosseinzadeh, M., & Rezaee, A. (2018). Load balancing mechanisms in the software defined networks: A systematic and comprehensive review of the literature. *IEEE Access : Practical Innovations, Open Solutions*, *6*, 14159–14178. doi:10.1109/ACCESS.2018.2805842

Paliwal, M., Shrimankar, D., & Tembhurne, O. (2018). Controllers in SDN: A review report. *IEEE Access : Practical Innovations, Open Solutions*, *6*, 36256–36270. doi:10.1109/ACCESS.2018.2846236

Sahoo, K. S., & Sahoo, B. (2019). CAMD: A switch migration based load balancing framework for software defined networks. *IET Networks*, *8*(4), 264–271. doi:10.1049/iet-net.2018.5166

Wang, C. A., Hu, B., Chen, S., Li, D., & Liu, B. (2017). A switch migration-based decision-making scheme for balancing load in SDN. *IEEE Access : Practical Innovations, Open Solutions*, *5*, 4537–4544. doi:10.1109/ACCESS.2017.2684188

Wang, H., Xu, H., Huang, L., Wang, J., & Yang, X. (2018). Load-balancing routing in software defined networks with multiple controllers. *Computer Networks*, *141*, 82–91. doi:10.1016/j.comnet.2018.05.012

Xu, Y., Cello, M., Wang, I. C., Walid, A., Wilfong, G., Wen, C. H. P., Marchese, M., & Chao, H. J. (2019). Dynamic switch migration in distributed software-defined networks to achieve controller load balance. *IEEE Journal on Selected Areas in Communications*, *37*(3), 515–529. doi:10.1109/JSAC.2019.2894237

Xue, H., Kim, K. T., & Youn, H. Y. (2019). Dynamic load balancing of software-defined networking based on genetic-ant colony optimization. *Sensors (Basel)*, *19*(2), 311. doi:10.3390/s19020311 PMID:30646575

Zhou, Y., Zheng, K., Ni, W., & Liu, R. P. (2018). Elastic switch migration for control plane load balancing in SDN. *IEEE Access : Practical Innovations, Open Solutions*, *6*, 3909–3919. doi:10.1109/ACCESS.2018.2795576

Zhou, Y., Zhu, M., Xiao, L., Ruan, L., Duan, W., Li, D., . . . Zhu, M. (2014, September). A load balancing strategy of sdn controller based on distributed decision. In *2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications* (pp. 851-856). IEEE. doi:10.1109/TrustCom.2014.112

Zhu, L., Karim, M. M., Sharif, K., Li, F., Du, X., & Guizani, M. (2019). *SDN controllers: Benchmarking & performance evaluation.* arXiv preprint arXiv:1902.04491.

Muktar Abdella works at Werabe University as Lecturer and Researcher. He is looking forward to work on Computer Networking and Security, Cloud Computing, Machine Learning, and Data Science. He is interested in the Software Defined Network and Network Function Virtualization research areas.