This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/OJCOMS.2020.3028971, IEEE Open Journal of the Communications Society

An Improved Switch Migration Decision Algorithm for SDN Load Balancing

OLADIPUPO ADEKOYA, ADEL ANEIBA, MOHAMMAD PATWARY *(Senior Member, IEEE)*
Birmingham City University
Birmingham, United Kingdom
CORRESPONDING AUTHOR: O. ADEKOYA (email: oladipupo.adekoya@mail.bcu.ac.uk)

Dynamic and Adaptive Load Balancing (DALB) and Controller Adaption and Migration Decision (CAMD) frameworks are the recently developed efficient controller selection frameworks that solved the challenge of load-imbalance in Software-Defined Networking (SDN). While CAMD framework was established to be efficient over DALB framework yet it was not efficient when the incoming-traffic load was elephant flow, hence, leading to a significant reduction in the overall system performance. This study had proposed an Improved Switch Migration Decision Algorithm (ISMDA) that solved the network challenge when the incoming load is elephant flow. The balancing module of the switch migration framework, which runs on each controller, is initiated during the controller load imbalance phase. The improved framework used the controller variance and controller average load status to determine the set of underloaded controllers in the network. The constructed efficient migration model was used to, simultaneously, identify both the migration cost and load-balancing variation for the optimal selection of controller among the set of underloaded controllers. The controller throughput, response time, number of migration space and packet loss were used as the performance comparison metrics. The average controller throughput of ISMDA increased with 7.4% over CAMD framework while average response time of the proposed algorithm improved over CAMD framework with 5.7%. Similarly, the proposed framework had 5.6% average improved migration space over CAMD framework and the packet-loss of ISMDA had average 6.4% performance over the CAMD framework. It was concluded that ISMDA was efficient over CAMD framework when the incoming traffic load is elephant flow.

***Index Terms*—SDN, load balancing, switch migration, algorithm, distributed controllers**

I. INTRODUCTION

SOftware Defined Networking (SDN) is known to be a promising architecture that separates control planes and the data plane. One of the applications of SDN is the adaptive energy consumption of IoT infrastructures in the deployment of Internet of Everything [1]. SDN come with a new network architecture that simplifies the network through rapid innovation, flexible management, and programmability such that the behaviors of the system are dynamically customized [2]. The single centralized controller, such as (Floodlight [3], and Ryu [4] implemented on the control plane cannot withstand today's current traffic requirements with the rapid growth in network size. Therefore, researchers have introduced multiple controller architecture (logically centralized but physically distributed), which divides the networks into several subdomains with distinct controllers. Some of the controller architectures are (Kandoo [5], HyperFlow [6], and Onix [7]).

The multiple controller had introduced the advantages of controller scalability and reliability to the network community. However, the state of the network flow is not the same as suggested by [8]. Its vary in space and time. The mapping between switches and controllers are static. Consequently, there will be an uneven distribution of traffic flow in the network with the dynamic change of traffic flow requests. This static mapping between switch and controller can easily lead to availability of limited amount of control resources in order to satisfy switch requirements. This process would lead to limited use of control resources. Hence, leading to the challenge of controller load imbalance.

Controller load balancing method is classified into controller optimization and switch migration methods. The controller optimization class determines the optimized number and the optimized position of controllers within the network for the purpose of load balancing distribution. However, this approach suffers from the real-time change since it only optimizes controller nodes. On the other hand, the switch migration approach is used to balance the load of the controller. These two methods improve the flexibility of networks and load balance dynamically, however, the methods do not consider an efficient switch selection and target controller for load shifting. Hence, there is need to develop an efficient method that would optimize the switch selection process and the target controller selection for load shifting in the network.

The remaining part of this study is arranged as below: Related work is discussed in Section II. The system model proposed in this study along with the load judgment, switch selection, and controller selection is discussed in Section III. The experimentation of the proposed framework is discussed in section IV. Similarly, the analyses, the simulated result and discussion of the proposed framework are shown in section V while the conclusion is presented in section VI.

II. RELATED WORKS

The conventional application of SDN relies on the concept of a centralized control plane which has several drawbacks

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/OJCOMS.2020.3028971, IEEE Open Journal of the Communications Society

related to scalability and reliability [6]. The launch of OpenFlow 1.3 protocol by Open Networking Foundation (ONF) [9], helps in providing means for the implementation of switch migration in SDN. The OpenFlow 1.3 protocol defines three different roles of SDN controller as master, equal, and slave roles. A switch in SDN can be connected to only one controller at a time in the master role state, while it can be connected to more than one controller in either slave or equal role state. The role request message is used by every controller to communicate its role to the switch, and every switch in its capacity must remember the position of each controller connections. If an active controller (in master state role) gets overloaded or the traffic flow requests of connected switches grow bigger than expected, the reassignment of switches to other domains can easily be achieved. Due to the significant benefits of OpenFlow version 1.3, researchers had proposed switch migration approach for SDN controllers load balancing.

The work in [10] proposed an efficient load balancing strategy towards the utilization of resources on the controller to minimize power consumption by switching off a light-loaded controller among the cluster pool of distributed controllers. The study extended the work of [10]. It developed an enhanced and detailed distributed control plane achieved by switch migration protocol in [11] to resolve the challenge of high response time of overloaded controllers. The response time goes down when the number of packet-in messages surpasses a predefined threshold. The work of [10] adopted an instance pool approach of the controller where the pool increases or decreases based on the load arrival from the data plane. However, the framework, randomly, chooses switches for migration without considering the effect on the target controller. Similarly, the study did not describe the process of target controller selection for the acceptance of load shifting.

Similarly, the work in [12] developed a load balancing mechanism that balanced load distribution among distributed controllers. The device recommended a coordinator which is a centralized controller, that periodically collects load statistics among controllers and decides either to perform switch migration or not. However, this method had several limitations that may reduce the performance of the systems. These limitations included high memory, bandwidth, and CPU power due to frequently exchange messages with other controllers. Similarly, if the centralized coordinator collapses, the whole balancing mechanism will go down, which does not conform with the availability and scalability features of distributed controllers in SDN.

Furthermore, [13] developed a safe migration mechanism by selecting the nearest controller as their target controller, considering the distance between the migrated switch and the target controller to have a reduced migration cost. This approach may reduce packet-loss and low response time in the whole system. However, considering the nearest controller as the target controller for switch migration without proper consideration of the target controller workload will lead to another controller load imbalance.

To achieve an active load balancing among distributed controllers, the work of [14], expressed switch migration problem as a resource maximization problem, taking an approximation algorithm approach in solving each of the distributed controllers to find an optimal solution. However, this approach randomly chooses both the migrated switch and the target controller, which may lead to high response time and low network throughput.

In addition, the work in [15] proposed a Dynamic and Adaptive Load Balancing (DALB) framework. This framework, during switch migration, selects the nearest neighbour's controller among the set of under-loaded controller to accept load shifting. DALB was designed to accept elephant flows (traffic load between 501p/s and 5000p/s, inclusively) as against the mice flow (traffic load between 1p/s and 499p/s, inclusively). The strength in DALB framework is the choice of the nearest controller for accepting load shifting. This implies that at each phase of migration, the nearest neighbor controller will always be considered for migration. This can lead to fast execution time and brings about a reduction in the cost of migration. However, the process leads to traffic congestion in accepting load shifting.

Controller Adaption and Migration Decision (CAMD) framework were proposed by [16], as the Least-loaded controller selector among the set of the under-loaded controllers for accepting load shifting during switch migration. This approach is efficient and accepts more loads when the incoming traffic is mice flow (traffic load between 1p/s and 499p/s, inclusively). It was ascertained that CAMD algorithm outperformed DALB [15] and Elasticon [11] algorithms with respect to throughput, response time, and less migration cost under Least-loaded controller. However, switch selection with the lightest load for migration in CAMD approach would lead to low Load Balancing Rate (LBR). Similarly, CAMD approach produces reduction in the network efficiency, low throughput in the network, high response time and high packet loss.

DALB [15] is associated with the problem of increase in congestion when chosen the target controller while CAMD [16] approach of switch selection is associated with the problem of switch selection inefficiency, low throughput, high response time and high packet loss. This study aims to address the aforementioned problems associated with the CAMD and DALB frameworks. The proposed algorithm (ISMDA) is expected to select a heavily-loaded switch for migration (just like DALB) from the overloaded controller but will migrate it to the most appropriate controller during migration so that that maximum clustered resources would be left free.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/OJCOMS.2020.3028971, IEEE Open Journal of the Communications Society

III. PROPOSED ISMDA FOR SDN CONTROLLER LOAD BALANCING OVERVIEW

The first subsection introduces the various notations used in this study while the subsequent subsections illustrate the ISMDA design.

A. Notations

The SDN is constructed as an undirected graph $G = (V, E)$, where V denotes the set of nodes and E composes a set of links. This study takes into account, the assumption that the controller had been optimally deployed in the topology such that a set of switches is managed by each controller. The main notations used in this study are recorded in Table 1.

Table 1. Main Notations used in the study

Notation	Definition
C = N	Controllers set
S = K	Switch sets
(β')	Load collection threshold
ρ	The load balancing rate
CL_{cur}	Current controller load
(C_{UL})	The set of light-loaded controllers
(C_{OL})	The set of over-loaded controllers
C_{UL} *and* $C_{OL} \in C$	Both sub-controllers are member of C
J^{th}	The controller that belongs to C_{UL}
i^{th}	The controller that belongs to C_{OL}
$P(s_k)$	The set of switches controlled by C_{OL}
$P(s_k)$	The set of switches controlled by C_{OL}
mc_{ij}	The mapping relationship between s_k and c_j

B. ISMDA LOAD BALANCING STRATEGY

One of the effective ways of managing overload controller is load balancing through switch migration mechanism. This study introduces a framework that balances load distribution among the controllers, through dynamic switch migration system. This approach optimizes migration efficiency and improves the rate of load balancing.

1) ASSOCIATED ASSUMPTIONS

The associated assumptions for the use of the proposed framework are presented below.

- There is a means of communication among all the controllers and the load can be shared easily among all these controllers. This means that switches connected in the network can easily access every other controller in the network and migrate to the appropriate controller when the load balancing strategy gets triggered.
- A distributed data store (Hazelcast) that connects sets of controllers together to give a logically centralized controller already exists in the network. Similarly, a messaging library protocol (Zookeeper) for seamless communication also exists.
- Once a particular switch had been chosen for migration, it cannot go back to its original controller until the process of migration had been completed.

2) The Flowchart of ISMDA framework

The flowchart of proposed ISMDA framework is depicted in Fig. 1. ISMDA introduced in this work runs on every controller, and they work together to ensure load balance among controllers.

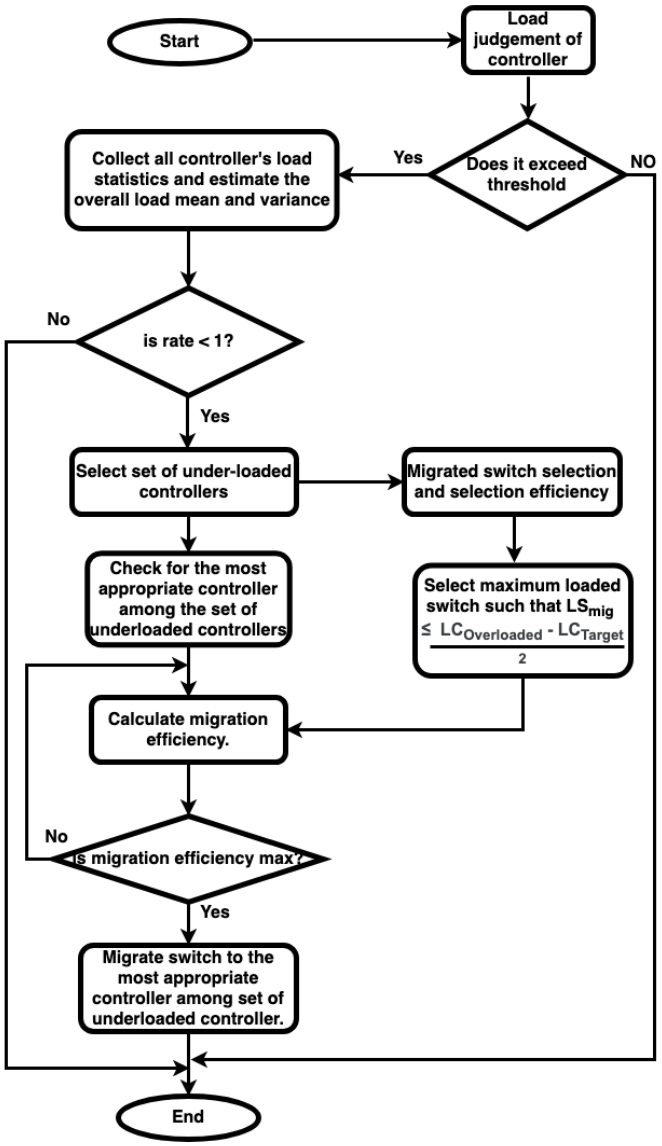

Fig. 1: The Flowchart of ISMDA framework

The ISMDA runs as a module on the controller. The controller intermittently checks to know when the load goes beyond the predefined threshold (see algorithm 2). During switch migration operation the set of an underloaded controllers, according to ISMDA design, accepts more of elephant flow than mice flow which improves the overall throughput and brings more stable balanced networks. If the load surpasses the predefined threshold, the load judgment module (see module 1) gets initiated. The controller collects other controllers load statistics with its load information and compute the estimated load mean and variance. The controller simultaneously initiates the load judgment module (see module 1), the switch selection module (see module 2) and target controller module (see module 3). The computed

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/OJCOMS.2020.3028971, IEEE Open Journal of the Communications Society

variance is used to identify the set of underloaded controllers and the most appropriate underloaded controller is selected. Subsequently, the load judgment module (see module 1) then passes the flow operation to the switch module of the ISMDA. The switch with the highest load request would be chosen for migration such that the load reduction of the overloaded controller is not more than the increase of the target controller load in order to have a successful balanced system. The controller ensures that the weight on the switch to be migrated is equal or less than the difference between the half of the overloaded controller and the target controller. The ISMDA formulated two efficiency model to evaluate the selection of the switch and the controller efficiency. The overloaded controller after switch selection with the help of zookeeper (message library protocol) will then migrate the selected switch (maximum loaded) to the most suitable controller among the set of underloaded controllers such that the maximum clustered resources are left free.

3) Load balancing mechanism for ISMDA strategy

Fig. 2 depicts the system architecture of ISMDA, which consists of three different modules that run locally on each controller in the network topology. From Algorithm 1, the load judgment module (in module 1) is triggered immediately when the load capacity of the controller is more than its predefined threshold. Each controller, in the network, collects all other controller loads with its load and calculates the controller mean and variance. These determine the set of underloaded controllers in the topology such that any controller whose weight is less than or equal to the average of controllers load is known as underloaded controller set.

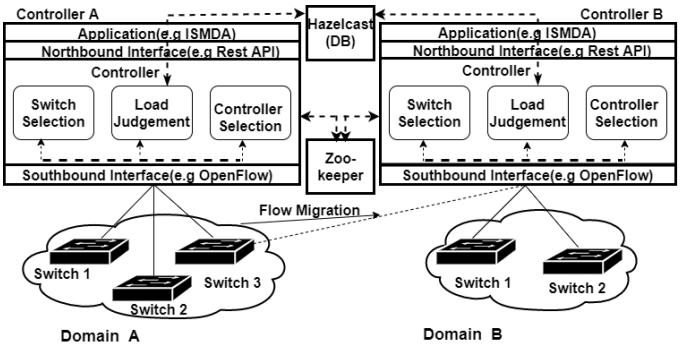

Fig. 2: Load balancing mechanism for ISMDA strategy

Consequently, the switch selection module (in module 2) also gets triggered along with the load judgment module in (module 1) and its main job is to select the heavily-loaded switch from the overloaded controller for migration, which improves selection efficiency based on the high load balancing rate. On the other hand, the controller selection module (in module 3) also gets started along with module 1 and 2. It ensures that the most appropriate controller of the underloaded controller set is selected, which is carefully done with migration efficiency in mind and ensures that maximum clustered resources are left free. The distributed database (Hazelcast) is a storage application where all shared information (controller threshold and load statistics) between the distributed controllers stored for easy access, and the Zookeeper is a coordinating service that glues the communication between the controllers. During uneven load distribution in the network, the flow of switch 3 in Fig. 2 of controller A in domain A will migrate its flow to the controller B of domain B.

This study, further, describes the different modules of ISMDA and its main functions. The main function of ISMDA is shown in Algorithm 1, and the adjustable controller threshold is shown in Algorithm 2. When the current load status of controller goes beyond a specified threshold (β'), the algorithm immediately calls $Load\_Judgement\_module$ (in module 1) to calculate the load status of all controller in the network. There is, also, a simultaneous call on both $Switch\_Selection$ (in module 2) and $Target\_Controller\_Selection\_module$ (in module 3) for Switch and Controller selection respectively. Consequently, the algorithm takes a decision and the switch then migrate to the target controller. The Overview of ISMDA is presented in 4. This study also presented a dynamic controller load threshold in 5. that enhances the update of controller threshold in the network.

4) ISMDA Algorithm (Algorithm 1)

Algorithm 1 comprises of dynamic controller threshold algorithm (Algorithm 2) and the three modules (modules 1,2 and 3) that are used in this study. ISMDA Algorithm uses Algorithm 2 to check if the controller load surpasses the predefined threshold set by the user. Consequently, If the condition is true, then the three modules would be triggered for respective operations.

Algorithm 1: Depiction of the overview of ISMDA.

Input: N, β', ρ, CL_{cur}
Output: Balanced Distributed Controllers
if $CL_{cur} > \beta'$ **then**
    $Initialize\ Load\_Judgement\_module$
    $Initialize\ Switch\_Selection\_module$
    $Initialize\ Target\_Controller\_Selection\_module$
    $Switch\_Migration(c_j \leftarrow s_k)$
    Update N (current and target)
end if
return $Balanced\ Distributed\ controllers$

5) Dynamic Controller Threshold (Algorithm 2)

Algorithm 2 performs the dynamic update of the controller threshold. The value of the threshold is stored in the distributed database. Every controller in the network checks to know if its load surpasses the predefined threshold. If the condition is true; it estimates the mean load of all connected controllers in the network. If the estimated mean is less than or equal to the predefined initial threshold, it returns the initial threshold. Otherwise, the estimated average load is set as the new threshold and updates itself dynamically. ISMDA needs to collect every other controller load in the system in order to make a proper load balancing decision. However, frequent collection of load statistics messages in the network to make load balancing decisions can reduce system performance with

a static controller threshold approach.

Algorithm 2: ISMDA Dynamic Controller Threshold.

Input: Controller's load list $\{CL_1, ..., CL_n\}$
Output: Adjustable (β')
$Find\ mean\ load\ \overline{C}L(c_i) = \frac{1}{n}\sum_{i=1}^{n} CL(c_i)$
Let Initial (β') = 1600
if $\overline{C}L(c_i) \leq Initial(\beta')$ **then**
    (β') = 1600
    if $\overline{C}L(c_i) \geq Initial(\beta')$ **then**
        $(\beta') = \overline{C}L(c_i)$
    end if
end if
return (β')

In order to solve the problem of reduced system performance, this study had proposed a dynamic load collection threshold approach (presented in Algorithm 2). This is stored in the distributed database where every controller can easily read it.

6) Load judgement Module (Module 1)

This module tracks and reports, in real-time, load statistics of each controller. The statistics include controller resources like memory usage, CPU usage, and bandwidth usage. When the controller load surpasses a predefined threshold (in algorithm 2), it estimates the average load of all other controllers and then identifies the under-loaded controller using a system variance (system performance error). The lower the system performance error the better the system. Consequently, all controllers which their current load is less than or equal to the system average value is considered as under-loaded.

There are basically two methods in getting the load status of SDN controllers as described by the author in [18]. These methods are centralized load collection algorithm and the distributed load collection algorithm. In a centralized load collection algorithm, the load balancer is deployed to track and collect the load status of the controller. This approach is not efficient enough and can only be applicable to a smaller network environment. However, every controller node in the distributed load collection algorithm can act as a load balancer. It can also easily track, estimate, and report its own load information regardless of the size of the network which makes it more appropriate and applicable in a large-size network.

The load on the controller can be measured in either of two ways. These two ways are switch input metrics and performance metric. Metrics like response time, average message arrival rate, and the flow table entries are generally used by switch input metrics in describing the load condition of the controller. However, this study does not make use of switch input metric since it will directly affect the resources on the controller. The performance metrics are related to the controller resources and are considered in this study to describe the load status of a controller. The metrics of the controller used are bandwidth usage, memory usage, and CPU usage. The total control event message from switches to the controller is taken as the input load from a switch. Since the demand of switch on controller differs from one another, and the resources on the controller are multi-dimensional, it becomes difficult to compare their requirements. A weighted sum is, therefore, assigned to integrate different types of resources which adjusts the weight in order to change the utilization fraction contributed by controller resources.

Every controller in the domain estimates its total load of $CL(c_i)$. This can be achieved in real-time based on the metrics specified above with their corresponding assigned weights which can be defined as

$$CL(c_i) = \begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix} \begin{bmatrix} CL_{band}(c_i) \\ CL_{mem}(c_i) \\ CL_{cpu}(c_i) \end{bmatrix}. \quad (1)$$

Where w_1, w_2 and w_3 are the weights of control plane resources (memory, bandwidth, and CPU) and $CL_{band}(c_i)$, $CL_{mem}(c_i)$ and $CL_{cpu}(c_i)$ represent the actual bandwidth, memory, and CPU utilized on the control plane by switch flow request.

Assuming that θ_k is the $packet\_in$ message generated by switch s_k to controller c_i during the time slot t. Since these resources domicile on the same control-plane (homogeneous systems), there is need to introduce a weighting (w_1, w_2 and w_3) system. Weight is assigned to control-plane resources here to unify the differences and similarities between resources. It will help in knowing the true consumption of resources by the switch. The main importance of weight to controller resources is to minimize bias. Without the weight, the inconsistency will shoot up and increase general error.

Hence, the accumulated load on the controller as a result of the $packet\_in$ control messages can be expressed as

$$CL(c_i) = \sum_{s_k \in c_i} \theta_k(t)\alpha_{ki}. \quad (2)$$

$CL(c_i)$ represents the current load status of controller (c_i), α_{ki} represents control plane resource utilization value and $\theta_k(t)$ represents the number of control events messages sent by the switch s_k to controller c_i at time frame t. This study takes α_i to represent the resource utilisation value of the controller c_i and is defined as

$$\alpha_i = \sum_{s_k \in c_i} \alpha_{ki}. \quad (3)$$

The value of α_{ki} is expressed as the weighted sum of network utilisation generated by each switch s_k to controller c_i and can be depicted as

$$\alpha_{ki} = \left(\omega_i^x \frac{\theta_k * x_i}{\mu_i} + \omega_i^y \frac{\theta_k * y_i}{\nu_i} + \omega_i^z \frac{\theta_k * z_i}{\tau_i} \right). \quad (4)$$

Where θ_k represents the control load events (Packet_in message) generated by the connected switches. The CPU

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/OJCOMS.2020.3028971, IEEE Open Journal of the Communications Society

usage, memory usage, and bandwidth usage of c_i per control event for switch s_k are represented by x_i, y_i and z_i, respectively. The value ω_i^x, ω_i^y and ω_i^z represent the estimated weights of controller resources. The CPU usage, memory usage and the bandwidth usage must satisfy the condition $\omega_i^x + \omega_i^y + \omega_i^z = 1$. Also, μ_i, ν_i and τ_i represent the CPU, memory, and bandwidth capacity of controller (c_i), respectively.

Module 1: Depiction of the Load_Judgement module of ISMDA

Input: n(number of controller load), ρ_i, β'
Output: C_{UL}
max = NULL
for $i = 1$ *to* n **do**
    read controller load $CL(c_i)$
    $\max = max\ + CL(c_i)$
    if $CL(c_i) > \beta'$ **then**
    end if
end for
Find mean load $\overline{C}L(c_i) = \frac{1}{n}\sum_{i=1}^{n} CL(c_i)$
Find load variance $\lambda = \frac{1}{n}\sum_{i=1}^{n}\left(CL(c_i) - \overline{C}L(c_i)\right)^2$
for $i = 1$ *to* n **do**

    $\rho_i = \frac{CL(c_i)}{\frac{1}{n}\sum_{i=1}^{n} CL(c_i)}$
    if $\rho_i < 1$ **then**
        $C_{UL} \leftarrow\ c_i$
        if $\rho_i > 1$ **then**
            $C_{OL} \leftarrow\ c_i$
        end if
    end if
end for
return C_{UL}

The concept of population variance is introduced to estimate the control plane equilibrium level since all the controllers are considered. This variance is expressed as

$$\lambda = \frac{1}{n}\sum_{i=1}^{n}\left(CL(c_i) - \overline{C}L(c_i)\right)^2. \tag{5}$$

Where λ denotes the variance, n (in this case, is the population size) denotes the number of controllers, $CL(c_i)$ indicates the controller load status in real-time and $\overline{C}L(c_i)$ denotes controller mean load. Consequently, the load balancing module will trigger a load balancing strategy according to the variance. Also, the switch migration will be triggered when there is controller load imbalance.

The module will collect other loads in the network and find the average value of the load factor which can be determined by

$$\overline{C}L = \frac{1}{n}\sum_{i=1}^{n} CL(c_i). \tag{6}$$

A controller load balancing rate, denoted by ρ_i, is used express to the ratio of the controller's current load status to their mean load. This is denoted by

$$\rho_i = \frac{CL(c_i)}{\frac{1}{n}\sum_{i=1}^{n} CL(c_i)}. \tag{7}$$

The module then computes the following three ratios upon which its decision making for migration is based on

- ρ_i < 1: This means that the current status of load on the controller is not up to the current average load. This implies that the controller is under-loaded and can still accommodate more loads or receive a switch migration
- ρ_i > 1: This implies that the current status of the load on the controller is more than the current average load. This means that the controller is overloaded and would lead to network overhead and imbalance.
- ρ_i = 1: This means that the current status of load on the controller is equal to the average load of all active controllers. Hence, the controller is neither in overloaded mode nor under-load mode. Consequently, the controller will not accommodate any additional load.

Based on the above procedures, controllers are, therefore, grouped into the set of the overloaded controller and the set of the under-loaded controller. This is represented mathematically as

$$\begin{cases} \rho_i > 1 & \text{controller in overload mode} \\ \rho_i = 1 & \text{controller in unstable mode} \\ \rho_i < 1 & \text{controllerin under-load mode.} \end{cases}$$

7) Switch selection module (Module 2)

This module selects the maximum load switch from the overloaded controller for migration purposes. Some of the useful statistics reported by the load judgment module were the round trip time, switch flow table entries, and the packet arrival rate. Generally, round trip time and the flow table entries are typically used for controller selection, while the average message arrival rate is used for both controller selection and threshold estimation.

According to the OpenFlow protocol, these three aforementioned statistics are helpful for the switch selection. The controller load is in proportion to the number of messages sent from the connected switches at time-span (t). Different messages such as Packet_in messages, Hello message, Echo messages, and others can be forwarded by the switch to the controller. However, the Packet-in flow messages do contribute more to the weight on the controller. This study had considered the average Packet-in message pushed from the switch to the controller to depicts the controller load. The goal of ISMDA is to spread the load among distributed controllers.

This study assumed that the load reduction of the overloaded controller, in an ideal condition, should not be more than the increase of the target controller load to have a successful balanced system. Subsequently, this study concluded that in the selection of switch for migration purposes, the weight on the switch to be migrated should be equal or less than the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/OJCOMS.2020.3028971, IEEE Open Journal of the Communications Society

difference between the half of the overloaded controller and the target controller. Hence, the choice of selection is made under the condition of

$$SL_{Migrate} \leq \frac{CL_{Overloaded} - CL_{Target}}{2}. \quad (8)$$

$SL_{Migrate}$ represents the load of the migrated switch from the overloaded controller, $CL_{Overloaded}$ represents the load of the overloaded controller, and the CL_{Target} represents the load of the most appropriate controller which is the target controller.

This concept emerges from the NSGS Procedure for Indifference Zone method [17]. This procedure, immediately, notifies switch migration module of its selection. NSGS Procedure abridges the Ranking and Selection (R&S) procedure and the fully sequential procedure. NSGS is an acronym from **Nelson**, **Swann**, **Goldsman**, and **Song**, which were the authors' first name in the work of [17].

Module 2: Depiction of Switch_Selection module of IS-MDA.

Input: Overloaded controller C_{OL}
Output: set of migrated switches s_k from C_{OL}
$Initialise\ switch\ set\ P = \{\}$
for $\forall\ s_k\ \in\ C_{OL}$ **do**
  for $\forall\ Flows\ \in\ C_{OL}$ **do**
    $Find\ migrated\ switch$
  end for
end for
$\left(\frac{1}{n}\sum_{i=1}^{n}(C_i)\right) / \max_{i=1}^{n}(C_i)\ to\ find\ selection\ efficiency$
$s_k = \underset{s_k \in C_{OL}}{\operatorname{argmax}}\{\mathrm{P_{sk}}\}$
$Migrate\ SL_{Migrate} \leq \frac{CL_{Overloaded} - CL_{Target}}{2}$
Return $switch\ set\ P\{\}$

In this study, it is expected that a switch can be connected to multiple controllers at the same time. However, a master controller is the only one that will be fully operational and attending to the request of the switch at a time, while the others can be in slave or in equal mode state. The change in the role is achievable with the help of a messaging library protocol like ZeroMQ(ZMQ) and Zookeeper while the HA-TCP is used to ensure message delivery between the switch and controller.

8) Controller Selection module (Module 3)

The purpose of this module is to select the most suitable controller to accept the load from the overloaded controller so as to ensure that the maximum clustered resources are reserved. For the optimal selection of controller, this study constructed a migration efficiency model that simultaneously identified both migration cost and load balancing variation. Migration cost and the load balancing variation are defined to further enhance their use on efficiency model constructed.

When switch migration occurs, the load balance of the network increases. However, it does cause extra migration cost to the network. Hence, the concept of migration cost needs to be introduced. Migration costs, primarily, comprises two main components namely, the increase in load on the controller and the number of a packet exchanged between the controllers. During migration of s_k from c_i to c_j. Migration cost is expressed as

$$mc_{s_k c_j} = mc_{ex} + mc_{lc}. \quad (9)$$

Where mc_{ex} is the message exchange cost and the mc_{lc} denotes the load increased cost. The load increased cost is the increase in load that happens when the target controller accepts the migrated switch from the overloaded controller. This is expressed below as

$$mc_{lc} = \begin{cases} \theta_i mc_{s_k c_j} - \theta_i mc_{s_k c_i}, & \mathrm{mc}_{s_k c_j} > mc_{s_k c_i} \\ 0, & \mathrm{mc}_{s_k c_j} \leq mc_{s_k c_i}. \end{cases} \quad (10)$$

Where θ_i denotes the packet_in control events messages produced from switch s_k.

Consequently, there is need to introduce the technique of load variance as a selection factor. This study used the load variance of controller as the balancing factor and $\overline{C}L(c_i)$ as the mean load of N controllers. This selection factor of a network before switch migration is re-expressed from equation (5) as

$$\vartheta = \frac{1}{n}\sum_{i=1}^{n}\left(CL(c_i) - \overline{C}L(c_i)\right)^2. \quad (11)$$

Where ϑ denotes network load variance before switch migration occurs. $CL(c_i)$ denotes load on controllers and $\overline{C}L(c_i)$ denotes average load of controller n. The load selection factor of the network after switch migration ensures that system biasness is reduced to a minimum. This is defined as

$$\vartheta^* = \frac{1}{n}\sum_{i=1, i\neq j}^{n}[CL^*(c_i) - \overline{C}L^*)^2 + CL^*(c_j) - \overline{C}L^*)^2], \quad (12)$$

where $CL^*(c_i)$ =$\big(CL(c_i)$ - $\theta_{sk} mc_{s_k c_i}\big)$ and $CL^*(c_j)$ = $CL(c_j)$ + $\theta_{sk} mc_{s_k c_j}$.

In equation (12), ϑ^* simply denotes the load variance of the network after switch migration had occurred. Migration efficiency is formulated to further enhance the selection of the target controller and also to indicate a trade-off between load balancing rate and cost of migration. Migration efficiency in this study is defined as the ratio of load balancing rate to the migration cost. It is expressed as

$$\Delta = \frac{\mid \vartheta^* - \vartheta \mid}{mc_{s_k c_j}}, \quad (13)$$

$$\forall s_k \in S, c_j \in C, J(s_k, c_j) = \{0, 1\} \quad (14)$$

$$\forall s_k \in S, \sum_{c_i \in C} J(s_k, c_j) = 1, \quad (15)$$

$$\exists c_i \in C, CL(c_i) \leq \beta' \quad (16)$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/OJCOMS.2020.3028971, IEEE Open Journal of the Communications Society

Equation (14) limits the connections among devices, while equation (15) ensures that each switch at a time (t) is managed by the master controller while Equation (16) ensures that it is not possible for all controllers in the network to be overloaded at the same time.

Module 3: Depiction of the Target_Controller_Selection module of ISMDA

Input: s_k, C_{UL}, mc_{lc}
Output: Target Controller c_j
$Get\ the\ current\ load\ on\ C_{UL}$
for $\forall$ $controller\ c_j\ \in\ C_{UL}$ **do**
  for $j = 1\ to\ n \in C_{UL}$ **do**
    $Estimate\ variance\ \vartheta\ and\ \vartheta^{*}using\ (11)and(12)$
    $Calculate\ migration\ efficiency\ using\ (13)$
  end for
end for
$Check\ that\ CL(c_i) + SL(s_i) \leq CL(c_j)$
$C_{TS} = \underset{c_j \in C_{UL}}{\text{argmax}}\{C_{UL}\} : CL(c_i) + SL(s_i) \leq CL(c_j)$
update $\{C_{UL}\}$ $with\ current\ load\ status$
return $target\ controller, c_j$

C. Example for illustration

This study illustrates a simple example to prove the significance of load variance and migration efficiency in switch migration conditions under the same controller capacity. Assuming the controller C = $\{C1, C2, C3, C4\}$ and their respective load is represented as LS = **{80, 30, 50, 40}**p/s. The average controller load of $\overline{LS}$ = $50p/s$. Controller Threshold (CT) is assumed as **60**p/s. Due to the above computation, $C1$ will be the overloaded controller. Generally, variance determines the deviation of the set of numbers from its mean value, in a given distribution. The individual controller variance are $v(C1)$ = 900, $v(C2)$ = 400,$v(C3)$ = 0, and $v(C4)$ = 100, and **variance** of the load of the controller before migration is **350**. The overloaded controller $C1$ can either shift the extra load of **20** to $C2$ or $C4$. If controller $C1$ chooses $C2$, individual variance becomes **{**$100, 0, 0, 100$**}**, and its global load **variance** for the network is **50**$(p/s)^2$. Subsequently, if C4 is chosen, the individual variance then becomes **{100, 400, 0, 100}**p/s, and its load variance becomes **150**p/s^2. Therefore, the migration efficiency for $C2$ approximately becomes **15**, whereas, for $C4$, the migration efficiency becomes 10. It is therefore concluded, that this study considered a higher migration efficiency controller as the best choice. $C2$ will be considered as the most appropriate controller because the overall load balance will be in a stable mode.

Table 2. Switch to controller flow arrival rate

Controller set	C1			C2				C3	
Switch set	s1	s2	s3	s4	s5	s6	s7	s8	s9
Switch Flow rate (KB/s)	400	300	300	300	500	400	700	200	300
Controller Threshold	1600								

Lemma 1: A switch with the maximum loaded flow requests on the overloaded controller is considered in this study during the migration stage, as this will Load Balancing rate (LBR) of the network.

Proof 1: The degree of closeness to the real load distribution can be measured with a load balancing rate. In this study, Table II, shows the flow arrival rate of different switches to the controller. Controller C2, at time (t), has the total load of $1900p/s$ which makes it an overloaded controller.

Controller C3, with higher migration efficiency, will be selected as the target controller for load shifting. This study describes how the LBR is affected by choosing a switch with less flow requests. This study had considered equation (17) for the evaluation of LBR which is defined as

$$\textbf{LBR} = \left(\frac{1}{n}\sum_{i=1}^{n}(C_i)\right) / \max_{i=1}^{n}(C_i). \tag{17}$$

Where $\{C_i, ..., C_n\}$ constitutes the load list of the connected systems which includes overloaded controller's load within the co-domain of 0 and 1. Before migration, the LBR is computed as 0.59. For instance, if controller C2, selects switch s4, its lightest loaded switch for migration, the LBR in equation (17) after migration will be 0.71. Meanwhile, if C2 decides to select its highest loaded switch s7 for migration, the LBR becomes 0.94. From the above analysis, this study revealed that selecting a switch with the highest flow request will bring about significant improvement in the network efficiency.

IV. EXPERIMENTATION

The experimentation of this study is discussed in this section. This study used a simulation approach and the assessment of throughput, number of migration, response time, and packet loss for the performance evaluation. The python code used for this experiment is made available on github code repository https://github.com/dipokoya2003/ISMDA.git.

In setting up the simulation environment, this study was carried out on a PC equipped with an Intel Core i7-6700HQ CPU @2.60GHZ with 16GB of 1600MHz DDR3 memory and a Windows 10 professional system. The experiment was conducted with a Jupyter notebook compiler, a free and open source software(FOSS) that allows creation of live code. In the algorithm, it was assumed that the controller threshold was set at **2000**p/s. The current load on the controller A-D before any migration phase was assumed to be 500p/s, processing rate was set to 70%, and the total sum of incoming load generated was initiated from (10000-26000)p/s with the size of 20.

For each of the iterations, the predefined Controller Threshold (CT) which the proposed ISMDA used in taking its migration strategy was set to be **2000**p/s on all of the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/OJCOMS.2020.3028971, IEEE Open Journal of the Communications Society

simulated controllers. The current load on controller A, B, C, and D before any migration phase was set to be $500p/s$. The processing rate to simulate a real control scenario was set to be 70%. In each of the iterations, the experiment was conducted with the total incoming load between (10000-26000)p/s, while each of these iterations was executed 1500 times in order to obtain the accuracy of the experiment. The proposed ISMDA will trigger Module 1 when the computation of the load is more than the predefined controller threshold. Subsequently, the ISMDA will call the switch selection (module 2) and the controller selection module (module 3). These modules will select the switch that is heavily loaded for migration and select the most appropriate controller to accept the incoming load, respectively. This study proposes an ISMDA that ensures that at each stage of the migration phase, the maximum clustered resources are left free. Hence, leading to a significant improvement over the baseline frameworks when the incoming traffic is elephant flows. The elephant flow in this study is defined between $501p/s$ and $5000p/s$ while the mice flows is defined as between $1p/s$ and $499p/s$

V. RESULT AND DISCUSSION

For the performance of this study to be validated, the ISMDA strategy is introduced and compared with the other two similar works in the literature by [15] and [16]. DALB [15] (also referred to as nearest controller selection approach) easily gives rise to traffic congestion for accepting load shifting. It increases the communication overheads between switches and controllers since it is possible for multiple switches to migrate into the nearest controller at the same time. CAMD [16], (also referred to as Least-loaded controller selection approach) may only be efficient in accepting load shifting when the incoming traffic flow is mice flow. Also, CAMD choice of a switch, with the lowest flow request rate for migration, is not suitable as compared to heavily-loaded switch. Selecting a switch with a minimum flow request, as proved in the last paragraph of section 3, may not improve the efficiency of the network as compared to choosing the heavily-loaded switch. Due to this reason, ISMDA selects a heavily loaded switch for migration. It carefully selects the most appropriate controller with the migration indexing factor to accept the load. The results of the simulation are discussed as follows, with a brief definition of each performance metric.

- Controller throughput: This refers to the maximum amount of packets that can be processed successfully by a controller. The average controller throughput was estimated based on the traffic generated during the simulation process. The total incoming load used in this study was between (10000 - 26000)p/s. Fig. 3, 4, and 5 shows the controller throughput of each of the reviewed studies and the proposed ISMDA. It was established from the graph that the number of flow requests successfully processed by the proposed ISMDA is more than the reviewed works. The Fig. 6 shows the average throughput comparisons of different Algorithms.

  In Fig. 3, the DALB approach processed more flow requests and seemed consistent at the peak of the load

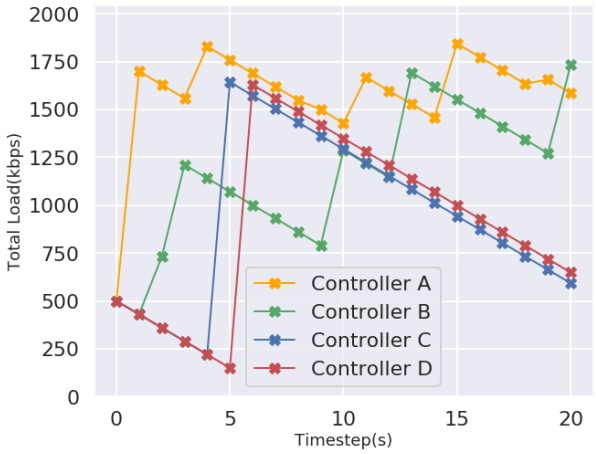

Fig. 3: DALB Controller throughput

with a reduced decline rate as compared with the CAMD approach. Though, this was not consistent as compared to the proposed ISMDA, as shown in Fig. 5 at the same level.

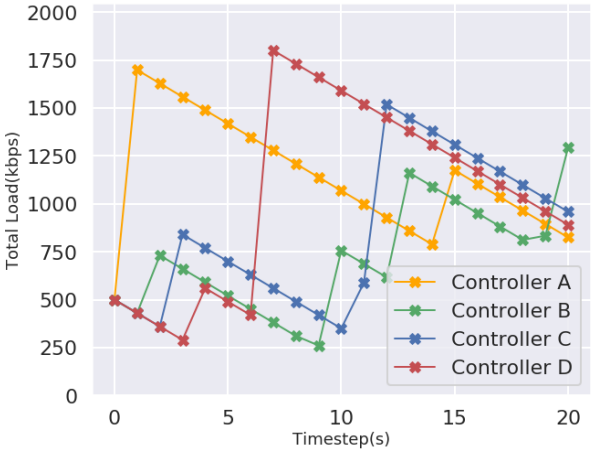

Fig. 4: CAMD Controller throughput

In Fig. 4, the CAMD algorithm could not maintain consistency at the peak of the load, The decline rate goes as low as 250 as compared to the proposed ISMDA and DALB. This, consequently, reduced the total throughput. In Fig. 5, ISMDA successfully processed more flow requests than both DALB and CAMD. It is more consistent at the peak of the load, and the decline rate is more reduced as compared with both DALB and CAMD. Fig. 6 shows the average throughput comparison for the three algorithms.

Fig. 6 reveals that the proposed ISMDA accepted more traffic than CAMD and DALB algorithms. The average controller throughput of the proposed framework in Fig. 6 rises to about 444p/s, which is an increase of approximately 7.4% over the CAMD and about 1.1% over the DALB. While, there is a similar throughput between DALB and ISMDA, it is shown that the proposed ISMDA has an improvement over DALB and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/OJCOMS.2020.3028971, IEEE Open Journal of the Communications Society

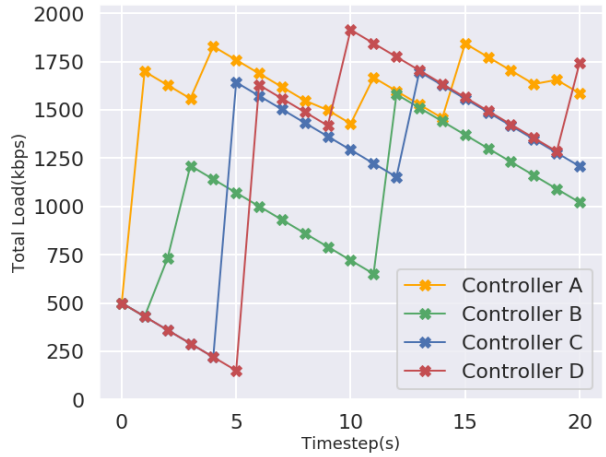

Fig. 5: Proposed ISMDA Controller throughput

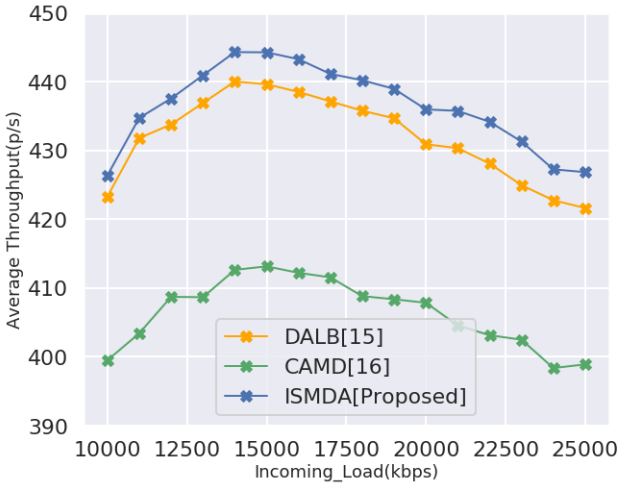

Fig. 6: Throughput comparison of different Algorithm

CAMD algorithms.

- Response time: It is expected that when there is a controller load imbalance, the response time in a given network will increase. For response time computation, this study adopted equation (18) developed by Netforecast [19]. All variables, in this equation, were fixed during estimation since this study was only interested in measuring the effect of packet-loss on response time. The number of rejected packets was used for packet-loss values. The adopted equation is presented as

$$R = R_d + R_t, \tag{18}$$

where

$$R_d = 2\Big[D + C_c + C_t\Big] + \Big[D + \frac{[C_c + C_s]}{2}\Big]\frac{T-2}{m} + Dl_n\Big[\frac{T-2}{m} + 1\Big] + KT(\frac{L}{1-L}), \tag{19}$$

and

$$R_t = \frac{MAX\Big[8p\frac{1+OHD}{B} * D\frac{p}{w}\Big]}{1-\sqrt{L}}. \tag{20}$$

In equation (17), R is the response time, R_d depicts propagation delay time, and R_t depicts the transmission delay time. In equation (18), D depicts round-trip delay; C_c depicts current processing time; C_t depicts server TCP processing; C_s depicts server processing time; T depicts application turns; m depicts multiplexing factor; Dl_n depicts packet-loss ratio and K depicts TCP timeout. Similarly, In equation (19), P depicts payload length; OHD depicts overhead ratio; B depicts minimum path bandwidth and W depicts the effective window size.

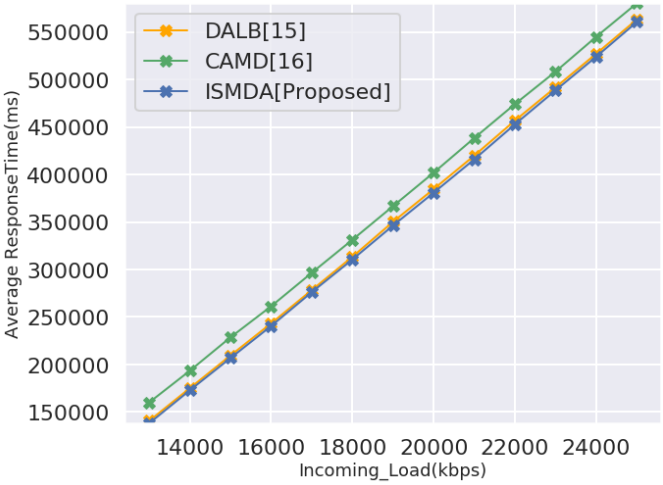

Fig. 7: Response Time comparison of different Algorithm

This study used average response time to compare the three algorithms in the simulation analysis. When an incoming load increases, the average response time of the three algorithms increase also. Fig. 7 reveals that the response time of the proposed ISMDA performs better than the CAMD and DALB with about 5.7% and about 1% less, respectively. Effective mechanism deployed in this study for selecting a controller to accept incoming load from the overloaded controller played a significant role. ISMDA selects the most appropriate controller at any phase of migration such that the maximum clustered of resources are left free. Consequently, ISDMA ensures that resources were well utilized which brought about a reduction in the number of rejected packets and significantly increased controller response time.

- The number of Migration space: This is the number of times each algorithm has to perform migration during controller load imbalance or overloading. The average count of the rejected packet was used to analyze and estimate the performance of ISMDA, DALB, and CAMD. Fig. 8 shows the number of times on an average that each of the algorithms had to perform the migration. It reveals that increase in the incoming load would lead to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/OJCOMS.2020.3028971, IEEE Open Journal of the Communications Society

increase in the average number of migration for the three algorithms. The proposed framework performs better than DALB and CAMD with 1.7% and 5.6% less, respectively. This implies that the proposed framework will bring about reduction in the number of times it migrates a switch during controller load imbalance, as compared to DALB and CAMD.

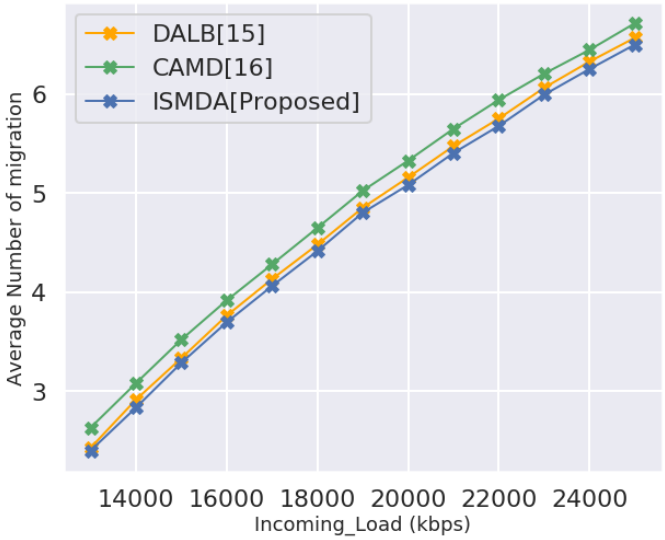

Fig. 8: Migration time comparison of different Algorithm

The efficient mechanism deployed and the choice of most suitable controller at every phase of migration are considered to be responsible. This led to a significant reduction in the number of times migration took place.

- Packet Loss: the number of packets that get dropped during transmission is regarded as packet-loss. In this study, the average number of rejected packets were estimated for the performance analyzes of the three algorithms.

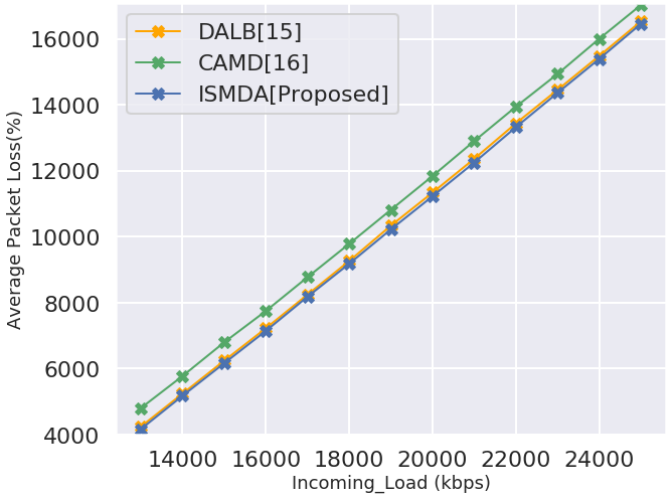

Fig. 9: Packet-loss comparison of different Algorithm

Fig. 9 reveals that as incoming load increases, the average packet lost increases for the three algorithms. However, CAMD had the highest average packet loss which ISMDA and DALB had similar average packet loss. Analysis estimated ISMDA to be 1.1% and 6.4% efficient over DALB and CAMD, respectively, with respect to average packet loss. This is so due to the effective mechanism brought forward in this study. ISMDA accepts more load than any of CAMD and DALB. Consequently, it had a fewer number of packet rejected.

VI. Conclusion

This study had proposed ISDMA to solve the challenge of SDN load imbalance when there is elephant flow in the incoming traffic load during controller load imbalance. ISDMA used controller load adjustment module, switch selection module and the target controller selection module. When the current load of a controller is above a predefined threshold, the balancing module of the switch running on each controller in the network is called upon to assure proper load balancing among controllers. The balancing module finds the unloaded controller among the controller set using variance and the mean load status of the controllers. The proposed algorithm, efficiently, migrates heavily loaded switches from an overloaded controller to the most appropriate controller among the underloaded set of controllers such that the maximum clustered resources are left free. The study, also, constructed a migration efficiency model, which showed a trade-off between variation in load balancing and the migration cost.
Results revealed that ISMDA outperformed both CAMD and DALB approaches in terms of throughput, the number of migration phases, packet-loss, and response time. Based on the simulation conducted, the proposed ISMDA is more efficient over CAMD and DALB under the condition of elephant traffic flow. The proposed ISMDA, the resource utilization of the distributed controller is now more balanced. It also improves controller throughput, results in less migration time in the network, improves the packet-loss and response time of the control plane. Future research shall focus on the implementation of a topology-aware switch migration intelligent system on an actual SDN controller. It is expected that the future research will predict the load of the control plane in advance, hence, improve the scalability and reliability of the network in a real large-scale environment.

References

[1] C. Tipantuna, and X. Hesselbach, NFV/SDN Enabled Architecture for Efficient Adaptive Management of Renewable and Non-Renewable Energy. IEEE Open Journal of the Communications Society, 1, pp. 357-380, 2020.
[2] T. Hu, Z. Guo, P. Yi, T. Baker and J. Lan, "Multi-controller Based Software-Defined Networking:A Survey," in IEEE Access, vol. 6, pp. 15980–15996, 2018.
[3] Chao Qi,Shuo Zhao. "An Aware-Scheduling Security Architecture with Priority-Equal Multi-Controller for SDN," China Communications, vol. 14, no. 9, pp. 144–154, 2017.
[4] T. Hu, P. Yi, Z. Guo, J. Lan and J. Zhang, "Bidirectional Matching Strategy for Multi-Controller Deployment in Distributed Software Defined Networking," in IEEE Access, vol. 6, pp.14946–14953, 2018.

[5] Yeganeh, Soheil Hassas, and Y. Ganjali. “Kandoo:a framework for efficient and scalable offloading of control applications.” The Workshop on Hot Topics in Software Defined Networks ACM, 2012, pp. 19–24.
[6] Tootoonchian, Amin, and Y. Ganjali. “HyperFlow:a distributed control plane for OpenFlow.” Internet Network Management Conference on Research on Enterprise NETWORKING USENIX Association, 2011, pp. 3-3.
[7] Koponen, T., Casado, M., Gude, N., Stribling, J., Poutievski, L., Zhu, M., Ramanathan, R., Iwata, Y., Inoue, H., Hama, T. and Shenker, S., Onix: A distributed control platform for large-scale production networks. In OSDI Vol. 10, pp. 1-6, 2010.
[8] Guo, Zehua, et al. “STAR: Preventing Flow-table Overflow in Software-Defined Networks.” Computer Networks, 2017, pp. 1–11.
Guo, Z., Liu, R., Xu, Y., Gushchin, A., Walid, A. and Chao, H.J., STAR: Preventing flow-table overflow in software-defined networks. Computer Networks, 125, 2017, pp.15-25.
Guo, Z., Liu, R., Xu, Y., Gushchin, A., Walid, A. and Chao, H.J., 2017. STAR: Preventing flow-table overflow in software-defined networks. Computer Networks, 125, pp.15-25.
[9] OpenFlow switch specification version 1.3.0 [OL]. https://www.opennetworking.org/images/ stories/openflow-spec-v1.3.0.pdf.
[10] Dixit, A., Hao, F., Mukherjee, S., Lakshman, T.V. and Kompella, R, "Towards an elastic distributed SDN controller", In Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking, 2013, pp.7–12.
[11] Dixit, Advait, Fang Hao, Sarit Mukherjee, T. V. Lakshman, and Ramana Rao Kompella."ElastiCon; an elastic distributed SDN controller." In Architectures for Networking and Communications Systems (ANCS), ACM/IEEE Symposium on Architectures for Networking and Communications Systems, 2014, pp. 17-27.
[12] Liang, C., Kawashima, R. and Matsuo, H., "Scalable and crash-tolerant load balancing based on switch migration for multiple open flow controllers",Second International Symposium on Computing and Networking, 2014, pp. 171-177.
[13] Aly, Wael Hosny Fouad, and Abeer Mohammad Ali Al-anazi. "Enhanced Controller Fault Tolerant (ECFT) model for Software Defined Networking." Software Defined Systems (SDS), Fifth International Conference on. Software Defined Systems (SDS),2018, pp. 217-222.
[14] Cheng, G., Chen, H., Wang, Z. and Chen, S., May. DHA: Distributed decisions on the switch migration toward a scalable SDN control plane. IFIP Networking Conference (IFIP Networking), 2015, pp. 1–9.
[15] Zhou, Y., Zhu, M., Xiao, L., Ruan, L., Duan, W., Li, D., Liu, R. and Zhu, M, "A load balancing strategy of sdn controller based on distributed decision", IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications, September 2014, pp. 851-856.
[16] Sahoo, K.S. and Sahoo, B., CAMD: a switch migration based load balancing framework for software defined networks. IET Networks, 8(4), 2019, pp.264-271.
[17] Nelson, B.L., Swann, J., Goldsman, D. and Song, W, "Simple procedures for selecting the best simulated system when the number of alternatives is large" Operations Research, 49(6), 2001, pp. 950-963.
[18] LI, J.Q., SUN, E.C. and ZHANG, Y.H, "Multi-Threshold SDN Controllers Load Balancing Algorithm Based On Controller Load", DEStech Transactions on Computer Science and Engineering, (CCNT), 2018.
[19] Miyagi, M., Ohkubo, K., Kataoka, M. and Yoshizawa, S, "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, April 2004, pp. 905-906.

Adekoya Oladipupo (Member of IET) received the B.Sc. in Computer Science degree from Olabisi Onabanjo University, Nigeria, in 2007. Also, he received his M.Sc. in Data Network and Security from Birmingham City University, United Kingdom, in 2018. He is currently pursuing his Ph.D. degree with the School of Computing Engineering and Built Environment at Birmingham City University, United Kingdom. His research interests include SDN/NFV and Artificial Intelligence.

Adel Aneiba (Member of IET) received a B.Sc. in Computer Science from the University of Benghazi, Nigeria. He obtained an M.Sc. in e-commerce at Staffordshire University in 2003 and a Ph.D. in computing in 2008. Currently, he is an Associate Professor in the Internet of Things (IoT) at Birmingham City University, United Kingdom. His research interests are IoT, computer network simulation, evaluation, optimization and block-chain.

Mohammad Patwary (Fellow of IET) received his B.Eng. (Hons) in Electrical and Electronic Engineering in 1998 from the Chittagong University of Engineering and Technology in Bangladesh in 2005, a Ph.D. in Telecommunication Engineering from the University of New South Wales in Sydney, Australia. He led both the Intelligent Systems and Network Research groups in the School of Computing and Digital Technology as a Professor of Telecommunication Networks and Digital Productivity and a Head of Research at the Centre for Cloud Computing (CCC) at Birmingham City University. His research interests are Telecommunication networks, Intelligent system design, and Innovative business modeling for D-Econo.