ESCALB: An Effective Slave Controller Allocation-Based Load Balancing Scheme for Multi-Domain SDN-enabled-IoT Networks

Jehad Ali[1], Rutvij H. Jhaveri[2], Mohannad Alswailim[3], Byeong-hee Roh[4]

[1]Department of AI Convergence Network, Ajou University, Suwon 16499, South Korea
[2]Department of Electronics Engineering, Pandit Deendayal Energy University, Gandhinagar, India
[3]Department of Management Information Systems and Production Management, College of Business and Economics, Qassim University, P.O. Box: 6640, Buraidah: 51452, Saudi Arabia.
[4]Department of Computer Engineering, Ajou University, Suwon, 16499, South Korea
CorrespondingAuthor:bhroh@ajou.ac.kr

Abstract. In software-defined networking (SDN), several controllers improve the reliability as well as the scalability of networks such as the Internet-of-Things (IoT), with the distributed control plan. To achieve optimal results in IoT networks, an SDN can be employed to reduce the complexity associated with IoT and provide an improved quality-of-service (QoS). With time, it is likely expected that the demand for IoT will rise, and a large number of sensors will be connected, which can generate huge network traffic. With these possibilities, the SDN controllers processing capacity will be surpassed by the traffic sent by the IoT sensors. To handle this kind of challenge, and achieve promising results, a dynamic slave controller allocation with a premeditated mechanism can play a pivotal role to accomplish the task management and migration plan. Following this, we proposed an efficient slave controller allocation-based load balancing approach for a multi-domain SDN-enabled IoT network, which aims to transfer switches to a controller with idle resources effectively. Among several slave controllers for selecting a target controller, a multi-criteria decision-making (MCDM) strategy, i.e., an analytical network process (ANP) has been used in our approach to enrich communication metrics and maintain high-standard QoS statistics. Moreover, switch migration is modeled with knapsack 0/1 problem to achieve maximum utilization of the slave controllers. Our proposed scheme enabled with a flexible decision-making process for selecting controllers with varying resources. The results demonstrated with emulation environment show the effectiveness of the ESCALB.

Keywords: software-defined networking, Internet of Things (IoT), Controller, analytical network process (ANP), load balancing

1 Introduction

Software-Defined Networking (SDN) have the capability to enhance the reprogrammability access followed by modification and reconfiguration of employed Internet of Things (IoT) or their inherited applications [1]. Following this, SDN controllers are very useful to reduce the complexity of an underlying IoT network by transferring control to the centralized controller in contrast to distributed traditional architecture for the sake of computation minimization and improvements in communication metrics. On the other hand, the traditional IoT network lacks to provide such kind of facilities, therefore SDN manage the deployed IoT networks in productive manner [2]. To continue, the SDN's flexible management capabilities in coordination with network programmable applications reveal them to be an excellent choice for SDN-enabled IoT networks or their inherited applications and emerging technologies such as the Internet of Vehicles [3], Unnamed Aerial Vehicles (UAV), Internet of Medical Things (IoMT) [4], 6G [5], Industry 5.0, Industrial IoT [6], cloud computing [7], Blockchain [8] and Internet of Everything (IoE), etc [9].

The use of SDN in IoT reduces the complexity of dispersed IoT designs through the global view of the centralized controller and makes better use of available resources. For example, the control plane may function as a decision-maker when it comes to transferring traffic to

overloaded controllers through the deployment of load-controlling and network monitoring applications. Furthermore, the network's programmability and multi-tenant functionality of SDN boost resource consumption and network performance, with meeting varied application needs and ensuring seamless communication through even load distribution in a multi-domain network.

In the last decade, this emerging technology demonstrated astonishing developments in various domains, but at the same time, it can suffer from multifarious challenges that can hinder their increased growth in new deployment sectors. To highlight these constraints, the significance of load balancing, Quality of Service (QoS), data preservation, and authentication among paired devices can not be overlooked [10]. The underlined challenges have their own consequences, but load balancing among them is the most devastating one which can affects the whole communication ecosystem of these networks [11], [12].

With massive machine-type communication in SDN enabled for IoT networks, a single controller will not be able to handle the massive traffic demands, consequently, the overwhelmed traffic towards the single controller generates bottleneck in the network. As a result, to increase scalability and prevent a single-point-of-failure, the controllers with distributed control plane are deployed to enable end-to-end (E2E) communication, leveraging several controllers to segment the network into distinct domains [13]. Following that, switches are allocated to one or more controllers within these domains. Numerous controllers ensure spread geographically, nevertheless, theoretically centralized control, sharing global authority using a network perspective and collaborating to accomplish efficient management of the network for an even distribution of the traffic among the domains.

Therefore, its balancing is assumed as an influential issue in SDN-enabled IoT networks and had received great attention from the research community and enterprise market. Following this, the literature suggests numerous strategies to address this problem, but most of them lack to present a complete package for IoT networks that can be capable to take care of communication aspects followed by the network lifetime of IoT devices. Some of the existing techniques use SDN controllers as an effective technology to tackle this problem. Following this discussion, in this paper, we proposed an intelligent multi-domain SDN slave controller load balancing scheme that aims to assign the switches more efficiently to the slave controllers if the master controller load exceeds than its specified limit.

The major contributions of our proposed approach are as following.

- This paper provides a novel framework for slave controller selection using SDN in IoT networks with multi-domain scenario.
- To formulate the problem, an analytical network based slave controller selection is used as a multi-criteria decision-making strategy.
- To ensure effective migration of SDN switches, we modeled it with 0/1 knapsack problem in our proposed framework resulting in significant QoS improvement
- To validate the QoS improvement, we use Mininet emulator instead of simulation, which is more reliable for SDN experiments and ONOS controller, that is a significant SDN operating system.
- To justify novelty and contribution of our work, we compare the results of our proposed method with state-of-the-art approaches for load balanicng in IoT multi-domain network with respect to several network criteria.

The rest of the paper is structured as follows; In section 2, we describe literature review. In section 3, the motivation is described. We explain our proposed approach for load balancing in SDN-based multi-domain IoT in Section 4. The experimental framework, simulations, results and discussions are presented in Section 5. Finally, the conclusion with future direction is provided in Section 6.

2 Related Work

In this section, we evaluate the positivefollowstive impact of existing state-of-the-art load balancing schemes that had been used for IoT networks leveraging SDN, to resolve this issue.

Table 1: Summary of the literature

Schemes in literature	Related papers	Limitations
QoS-LB	[14]	Lack of evaluation in the multi-domain architecture.
Preemptive-LB	[15]	Proof-of-concept evaluation has not been conducted with experimental evaluation.
Load-aware	[16]	Concepts has not been evaluated with real internet topologies.
DALB	[17]	Specific to data center networks.
SASLB	[18]	Mapping between switches and the controllers in SDN for SASLB is static.
DLB	[19]	Threshold of the controller is not illustrated resulting in the imbalanced distribution of the traffic.
SCLB	[20]	Significance is demonstrated for fault recovery ithe nstead of load balanicng.
SMLB	[21], [22], [23]	Evaluation in the scalable network topolgies

In [14] the authors demonstrate a QoS acquainted load balancing (QoS-ALB) scheme for IoT networks utilizing the SDN framework. In the QoS-ALB method, the writers emphasized on the traffic management among multi-service-oriented servers to improve the QoS metrics during communication. In coordination with this, the authors used an Integer Linear Programming (ILP) framework to utilize the available capacity of employed servers to allocate traffic to it i.e. that have the least workload. The limitation of this was the centralized point of management because the centralized point of failure leads to full network failure. Following this discussion, reference [15] suggests a reactive-based SDN framework to ensure delay-sensitive communication in the network utilizing a preemptive load balancing strategy (Preemptive-LB). However, the proposed solution was not evaluated in the experimental environment of the SDN. In [16], the authors suggested an intelligent Load-balancing method for Wi-Fi networks utilizing an SDN-based framework. In this framework, they focused on transferring load from overcrowded access points (APs) to under-loaded APs with a multi-criteria framework utilizing the packet loss rate, received signal strength indicator (RSSI), and throughput of the network. However, this model is in a fuzzy state, it has not been tested with real wireless communication parameters, whereas the traffic dynamics may generate delay in response time.

To address the load balancing concerns of these networks, a dynamic controller allotment problem (DCAP) framework was used as an online optimization problem for SDN to minimize the overall cost of reaction and maintenance time of controllers [17]. However, during the evaluation phase, most of the communication metrics of SDN-enabled networks were overlooked.

To enhance the availability and provision an E2E performance, the works in [24], [25], [26], [27], discuss the placement of several controllers in the SDN. In [24], the capacitated placement of controllers is performed. Similarly, in [25], the controllers are placed for multiple domain SDN to improve the performance. For the design of a distributed control layer in SDN, the [26] discuss the controller allotment to the switches. In addition, the works in [27], illustrate the placement of controllers in the SDN to guarantee delay and increase resilience by considering the capacity of each controller. However, all of this only focus on decreasing the E2E delay among the switches and controllers through proper assignment of the switches to the controllers.

To continue, Abdelaziz et.al [18] incorporate a DLB model into an SDN framework by considering scalability, reliability, and fault tolerance [28]. For this, the authors used a cluster of three controllers to execute their model, and enable the one controller at a time to manage network activities [29]. In case, If an active controller fails, then the secondary controller will take over and become the primary controller based upon a predefined set of priority rules. To highlight the limitation of this model, the primary controller selection process was not modeled nor the performance metrics were demonstrated, which makes this model in the fuzzy state. To follow this, the authors present a comprehensive study related to the challenges of load balancing of IoT networks using SDN-based framework [19].

Reference [20], suggested another master-slave-based-SDN architecture, known as SCLB scheme to tackle the load balancing issued in IoT network utilizing the could employed switches to support failed controller in an operational network. It is an architecture with physically distributed controllers that work in coordination to support on failed controllers. In case, if a series of controllers fail to function, the switch is migrated to a slave controller. Moreover, it prevents the network from crashing through planning the assignment of the

slave controller. The experimental results show that in the worst cases, the controller failure in the SCLB scheme can create latency issues. Which as a whole arises the consistency issue and the implementation of the algorithm in a distributed architecture is cumbersome.

In [30] the authors proposed an architecture known as Fault Tolerant Architecture for Software-Defined Networking (FT-SDN) with distributed controllers for IoT networks that gives fault tolerance via the management of Control plan. This model uses several open-source controllers in multiple network domains to enhance the communication metrics of IoT networks. In the FT-SDN architecture, the authors also considered the scenario, when a controller is failed, then what should be adopted for his replacement. Hence, in case of more than one controller failure and external attack, it will be able to provide adequate robustness, with the least amount of packet losses in the network. The author uses multiple vendor controllers i.e. OpenDayLight, Floodlight and Ryu. However, the drawback of this approach in slave controller allocation is that Floodlight controller do not support OpenFlow version 1.2, means slave controller cannot be configured.

In [21] the authors proposed a hybrid QoS improvement scheme for SDN-based IoT network leveraging switch migration mechanism-based load balancing (SMLB). Similarly, an SMLB strategy is proposed in [22], for distributed SDN for load balancing among the controllers. In these works, the authors used threshold values of controllers to migrate switches to under-load controllers. Following this, in [23], the authors explained the slave controller allocation using the technique for order of preference by similarity to ideal solution (TOPSIS) for controllers prioritization and SMLB keeping given the controller's load. The disadvantages of this scheme was that the authors did not check the performance of the proposed of the model with scalability of the employed network. Table 1 shows the summary of the main schemes and their evaluation.

3 Motivation

In the literature, we have recorded that numerous existing SDN-enabled load balancing schemes of IoT network uses a centralized management controller, which increases the risk of network failure with respect to this centralized controller. Therefore, it is imperative to design a multi-domain SDN-enabled IoT network infrastructure, where several controllers will work in coordination to manage and address the load balancing issues of these networks. Herein, we demonstrate the problem of static allotment of the switches to the neighboring domain controller which acts as a slave/backup controller in case of the master controller failure. The master controller is the active controller for the switches which are attached to it. This concept has been elaborated in fig. 1.

For example, as shown in fig. 1, there are four controllers, which are symbolized with C, where C_1, C_2, C_3, C_4 and four domains D_1, D_2, D_3, D_4, with one to one correspondence between controllers and domains. The switches (S) are denoted with $S_1, S_2, S_3, ...S_n$. Now, let's suppose that the switches of each domain (D) are sending 2000 packets/sec generated by IoT sensors to the controller of their concerned domain. In this scenario, C_1 and C_2 can manage four switches simultaneously as their packet processing capacity is 8000 packets/sec, while C_3 and C_4 can manage five and seven switches respectively because their capacity is 10000 and 14000 packets/sec. Herein, we assume that there is a static neighbor domain controller, which is given the name and role of the slave controller for each domain. Likewise, if C_1 fails due to more load or some hardware/software failure, then, its switches will be transferred to C_2 controller. Thereafter, the C_2 has six switches, and its processing capability is only 8000 packets/sec. With this, the current traffic overwhelms the capacity of C_2 and leads to a failure of C_2 in terms of operation. As C_2 fails the switches of D_2 will be transferred to C_3 as it is next neighbor. The total demand packet rate of D_3 switches now becomes 18000 packets/sec. However, the offered packet rate of C_3 is 10000 packets/sec which is beyond its total capacity. Therefore, C_3 also fails because it cannot meet the required packet processing rate of its domain. The next neighbor assigned as a slave for C_3 is C_4 therefore the switches control will be transferred to C_4. Domain 4 will now have 12 switches with a sending rate of 24000 packets/sec to C_4. However, its processing capacity is 18000 packets/sec, which will bring down C_4 as well leading to a complete network crash. The phenomenon

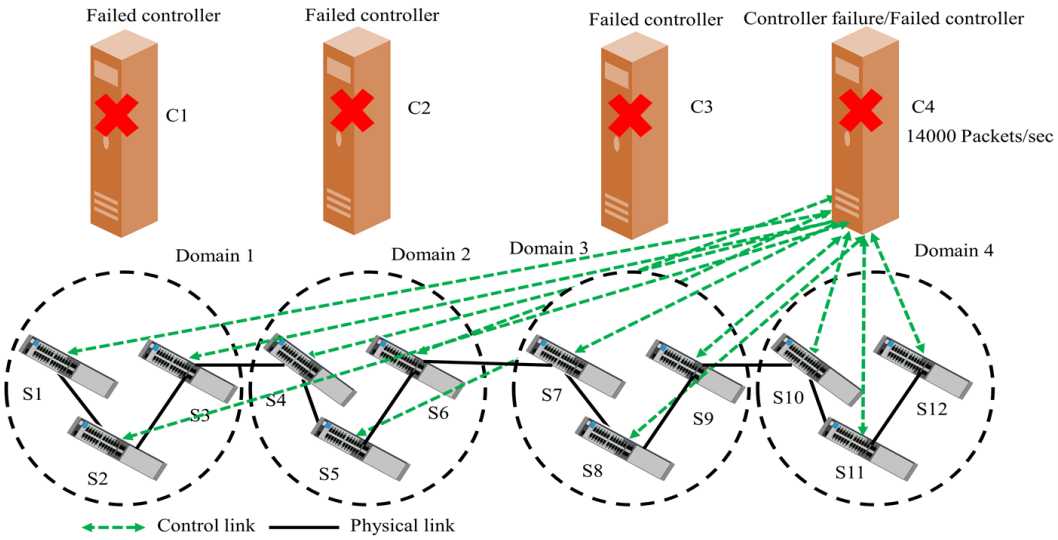


Fig. 1: Controller failures with static allotment of switches to slave controllers

explained in the example happened because of the uneven load distribution during the slave controller assignment process. Analyzing the above scenario, the assignment process might have not led to controller chain failure if a load of each controller is properly predicted. If we assign switches to the controllers according to their load and do not overwhelm them with redundant packets, then the load will be distributed fairly with the smooth operation of the network. Hence, in this paper, we address the problem of load balancing.

4 Proposed framework

To formulate the problem in the context of our proposed model, we make a hierarchical model for the control plan that consists of multiple domain controllers (DCs), and the global control (GC) plane. The GC plan comprises four sub-modules such as load calculation module (LCM), ANP Module (ANPM), switches migration module (SMM), and flows forwarding and updating module (FFUM). To visualize these all, we have added fig. 2 in the paper. To expand this discussion, an SDN is denoted with a graph $G = (V, E)$, where V denotes the vertices and E denotes the edges. Herein, the master controllers are denoted with $C = C_1, C_2, C_3, \ldots, C_N$, which are basically domain controllers DC. In addition, the slave controllers are denoted with $SC = SC_1, SC_2, SC_3, \ldots SC_K$, with constraints i.e. $K = N$, because there is at least one slave controller for a master controller. Moreover, the load criteria are denoted with LC. LC denotes the load criteria based on which slave controllers will be ranked (prioritized) leveraging ANP MCDM in case master controller i.e. C_1, C_2, C_3 or C_N exceeds the load from its threshold (T). The T=FC_{max}, where FC_{max} shows the maximum flow requests that a master controller can process. Moreover, the switches in the data plane are denoted with $S = S_1, S_2, S_3, \ldots, S_z$. Once priorities of controllers are obtained via ANPM, then switches migration starts from master controller to SC using knapsack 0/1 scheme. The details of the proposed approach are provided in the following subsections.

An SDN architecture, as described by the Open Networking Foundation [31], is made up of the application plane, the control plane, and the data plane. The data plane devices, which manage traffic flow and packets, are distinct from the control plane devices, which describe how a data plane would operate. The application layer is employed with various applications, which denote the client elements that request SDN controller for obtaining its services [32], making management of networks more flexible because the network is capable of dynamic configurations to fulfill the requirements of several applications. Our suggested

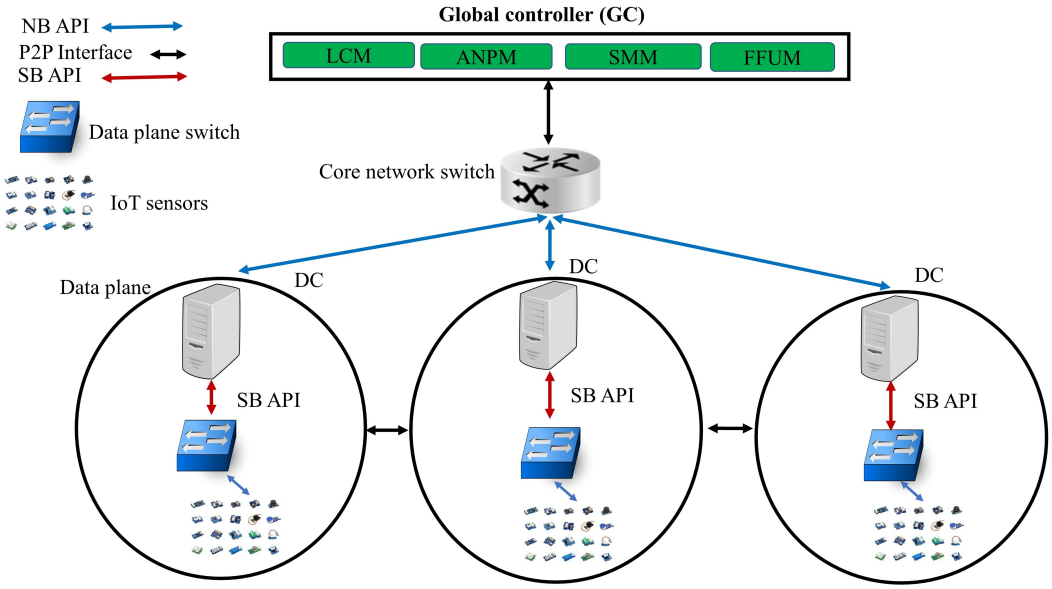


Fig. 2: Proposed framework for load balancing using ANP module with SDN for IoT

framework's architecture is organized in the subsequent sub-sections in an SDN-like manner.

4.1 Proposed SDN-based architecture for load balancing

In this section, we illustrate the SDN-based load-balancing architecture for the multi-domain environment in IoT. Our proposed architecture is based on the hierarchical organization of the control plane as shown in Fig. 2. Hence, there is a local control plane, and a global control plane as shown with DC and GC in the figure. The local controllers are connected through a backbone switch for E2E connectivity. The application plane employs modules for load balancing and monitoring (control applications) of the network. We use the applications and modules of the term interchangeably in this paper. There is LCM for monitoring the load, ANPM for ranking the controllers per their load status, SMM, which migrates the switches associated with an overload controller, and FFUM for forwarding and updating the flow rules. The data plane consists of SDN forwarding switches that communicate with DCs through southbound APIs. The IoT sensors are connected in the data plane. We employ the following modules in our proposed framework. The details and work of each module are given below.

4.2 FFUM

The flow forwarding and updating module receives the Packet-In requests $F(t)$ from the underlying data plan switches (S_i), at time t, which has flow rules FR for the traffic in the network. It works in coordination with the ANPM, and SMM, where slave controllers (SC) are prioritized, switches are migrated to it, and flow requests are sent to the slave controllers whose weight is high. Moreover, the flow rules U_{FR} which are updated will be sent by FFUM to an underlying data plan via OpenFlow protocol. The U_{FR} consists of the updated SC list with the migrated switches allocated to it.

4.3 Load calculation module (LCM)

The load calculation module receives the load status of each controller form the DCP. The load metrics are used in the ANP module for ranking the controllers. These metrics denote the criteria upon which controllers will be ranked, and then switches will be migrated to

Table 2: Notations and Explanation

Notation	Explanation
G	Denotes the network graph of SDN
E	Edges
V	Vertices
B	Importance level of a load parameter
C	Denotes the controller
LC	Load of a controller
SC	A slave controller
K	The number of slave controllers
S	The switches
z	The number of switches
n	Number of controllers
P	Packet-In requests
W	Weights computed from ANP module
CI	Consistency index
X	Eigenvector corresponding to a slave controller
CR	The consistency ratio
Y	The consistency measure
T	Threshold for load
S_i={1, 0}	Binary variable: The value 1 shows that switch is allocated to a slave controller otherwise not

it leveraging the knapsack 0/1 scheme. In the proposed scheme, the load of a controller is denoted by LC_j as shown in Eq. (1), i.e. based upon a load of its CPU usage, flow requests capacity (FRC) i.e. the total number of flow requests (Packet_In messages) a controller can process minus the current Packet_In messages, memory use and the number of switches attached with each SC. The LC_j denotes each controller, and the controller's number is denoted with n. Herein, the B shows the significance level of the load metric. However, in practice, the load is proportional to the Packet-In messages processed by a controller.

$$LC_j = B_1 * L_{cpu}(C_j) + B_2 * L_{FRC}(C_j) + B_3 * L_{MU}(C_j) + B4 * L_S(C_j) \tag{1}$$

with the following constraints:
i.e. $j = 1, 2, 3, ...n$
$L_{cpu}(C_j), L_{FRC}(C_j), L_{MU}(C_j), L_S(C_j) \in (0, 1)$
$\sum_{d=1}^{n} B_d = 1$

4.4 ANP Module for slave Controller Assignment

The ANP module is employed on the LBCP which plays an important role in load balancing of the DCP controllers. The ANP module is launched to rank the controllers. We apply ANP method of MCDM because it can make multi-criteria decisions and it considers dependency among criteria elements (load metrics) and considers feedback from alternatives (slave controllers). We consider four metrics as a criterion (LC and the alternatives are the slave controllers in the DCP (SC).

We consider central processing unit (CPU) utilization, the number of flows currently processed by a controller, memory and switches as the criteria for load measurement. The ANP model will return the priorities of the slave controllers in the DCP according to the weights i.e. from high to low based upon these four criteria. The step-by-step mathematical procedure to rank the controllers regarding their load is described below.

The ANP formulates the problem as a network model of the criteria and alternative elements, that are pairwise compared against each other. Hence, ANP prioritizes the alternatives and the criteria elements both. Herein, the ANP selects the slave controller among the controllers in multi-domain SDN for IoT. The slave controllers SC are the alternatives and the LC is the load based upon which a slave controller is allocated for the switches whose master controller is not available. The following mathematical model is used to rank the SC, and find the priority weights for the slave controllers.

– The hierarchical model is made of alternatives and criteria clusters. The criteria for load and the controllers in the DCP are depicted in Eq. (3) and Eq. (4).

$$LC = \{LC_1, LC_2, LC_3,\ ...,\ LC_n\} \tag{2}$$

$$SC = \{SC_1, SC_2, SC_3,\ ...,\ SC_K\} \tag{3}$$

– Eq. (4) reveals a comparison matrix, that identifies the relative importance of one controller (SC) over another with respect to the criteria defined in Eq. (3), denoted as the set of loads. Likewise, another pairwise comparison matrix is made showing the priority of loads regarding controllers in DCP. The quantitative value of the relative significance of one SC over another SC is derived from a 9-point scale as shown in Eq. (5), where 1 shows an equal importance level and 9 shows the extreme significance of one SC against others. Similarly, 3 shows that an SC is moderately more significant than the other SC regarding criteria such as hop count, etc. These values are given in Table III, and are incorporated in Eq. (5) for all SC with respect to every criteria [33].

$$M = \begin{bmatrix} & N_1 & N_2 & N_3 & \rightarrow & N_n \\ R_1 & 1 & a_{12} & a_{13} & \rightarrow & a_{1n} \\ R_2 & \frac{1}{a_{12}} & 1 & a_{23} & \rightarrow & a_{2n} \\ R_3 & \frac{1}{a_{13}} & \frac{1}{a_{23}} & 1 & \rightarrow & a_{3n} \\ \downarrow & \downarrow & \downarrow & \downarrow & 1 & \downarrow \\ R_n & \frac{1}{a_{1n}} & \frac{1}{a_{2n}} & \frac{1}{a_{3n}} & \rightarrow & 1 \end{bmatrix} \tag{4}$$

$$M = \begin{bmatrix} & N_1 & N_2 & N_3 & N_4 \\ R_1 & 1 & \frac{1}{3} & \frac{1}{3} & 3 \\ R_2 & 3 & 1 & 1 & 6 \\ R_3 & 3 & 1 & 1 & 9 \\ R_4 & \frac{1}{3} & \frac{1}{6} & \frac{1}{9} & 1 \end{bmatrix} \tag{5}$$

Table 3: Scale of importance

Scale	Explanation
1	Both of the SC have an equal importance
2	Once of the SC is equally up to moderately prime than the other SC
3	An SC is moderately more dominant than the other SC
4	The SC is moderately and significantly crucial than the other
5	It shows that an SC is significantly more dominant with respect to other SC
6	It indicates that a SC is significant up to remarkably prime from another
7	This shows the remarkably more dominant of a SC from the other SC
8	One of a SC is remarkable to extremely significant from other
9	A slave controller is excessively more dominant than other

– The pairwise comparison matrix (given in Eq. (5) is normalized as given in Eq. (6) to obtain the local priorities of slave controllers (SC) and load criteria metrics (LC) in the form of eigenvectors as shown in Eq. (7). The eigenvectors obtained from Eq. (7) shows the priorities of one SC over other SC in the SDN. Further, to prove the precision

of the judgments of the pairwise comparison matrices, another term consistency index (CI) is derived from these matrices. The value of $CI \leq 0.1$ shows the consistency of the pairwise judgments of one SC compared to another SC [33]. Herein, we give an example to show how the consistency in judgments matters for the final weights of the slave controllers. Let us suppose we say that $a \geq b$, and $be \geq c$. Then $a \leq c$ will lead to inconsistency in calculations. The prerequisite for CI is the consistency measure Y_j and λ_{max} which are calculated through Eq. (8) and Eq. (9). Eq. (9) shows the calculation of λ_{max} and consistency measure calculation is shown in Eq. (8). The (RI) result is put according to the criteria number or alternatives in Eq. (10), which is then put in Eq. (11). Consequently, Eq. (11) represents the CR.

$$M_\alpha = \begin{bmatrix} \frac{a_{11}}{\sum_{i=1}^{n} a_{i1}} & \cdots & \frac{a_{1n}}{\sum_{i=1}^{n} a_{in}} \\ \vdots & \ddots & \vdots \\ \frac{a_{n1}}{\sum_{i=1}^{n} a_{i1}} & \cdots & \frac{a_{nn}}{\sum_{i=1}^{n} a_{in}} \end{bmatrix} \tag{6}$$

$$X_i = \frac{1}{n} \sum_{j=1}^{n} a_{ij} \tag{7}$$

$$Y_j = \frac{M_j * X}{x_i} \tag{8}$$

$$\lambda_{max} = \frac{1}{n} \sum_{j=1}^{n} Y_j \tag{9}$$

$$CI = \frac{(\lambda_{max} - n)}{(n-1)} \tag{10}$$

$$CR = \frac{RI}{CI} \tag{11}$$

- Then, the eigenvectors for each SC are arranged in the form of unweighted super-matrix that identifies the local priorities of SC i.e. a slave controller over another SC.
- Lastly, a convergent limit super-matrix with stable weights is obtained through taking the power of weighted super-matrix till the values are converged making prioritized weights for (SC) in the DCP.

4.5 Switches Migration Module (SMM) leveraging Knapsack 0/1 problem

In this section, we describe the switch migration mechanism using knapsack 0/1 in SDN. The switches of the failed controller should be migrated to the slave controllers. However, each slave controller has a limitation for processing the incoming Packet-In requests. By solving the 0/1 knapsack problem for resource allocation, we can better utilize the controller resource by selecting the most valuable switches for a transfer to the slave controller while ensuring that they do not exceed their resource constraints. This can lead to increased efficiency, productivity, and profitability. For example, the maximum limit of an SDN controller is 9×10^6 packets/second with OpenFlow 1.2 [34]. Hence, we should migrate the switches in such a way that it may not overwhelm the slave controller. Moreover, the utilization of the slave controllers should be maximized. Therefore, we can formulate this problem as knapsack 0/1 problem, where each slave controller is assumed as a container that should be filled with objects. Hence, switches are regarded as objects. For each failed master controller, we have z switches. Moreover, we have a slave controller (SC), and m is equal to the total capacity of the slave controller i.e., the total number of Packet-In requests that it can process. In addition, a switch can be either attached with a slave controller or not i.e., switch $S_i = 0/1$ We will migrate a switch by fulfilling the following constraints. Maximize the profit in terms of throughput, so that the controller can process a large number of packet-In messages from the switches migrated to it.

$$Max \sum P_i S_i \tag{12}$$

Algorithm 1 Algorithm for Prioritization of Slave Controllers

Input:
Load of the controllers
$LC = LC_1, LC_2, LC_3, \ldots LC_N$
Output:
Prioritized slave controllers ranking (SC) i.e. weightage of slave controllers with respect to LC
Pairwise comparison of alternatives with respect to criteria
for (SC_1 to SC_K)
compare each alternative SC with respect to every criteria LC
Fill values in Eq. 5 from a nine-point scale (Table III) for each SC regarding its LC
Find the X_i (Eigenvectors) using Eq. 7
Find the CM,, Y_j using Eq. 8
Find the CI using Eq. 10
Find the CR using Eq. 11
Check condition if $CI \leq 0.1$ for matrix defined in Eq. 5
If ($CI \leq 0.1$) compare next alternative
Otherwise repeat step 3,4, 5 and 6
End If
End for loop
for (LC_1 to LC_n)
compare each criteria with respect to every alternative
Fill values in Eq. 5 from 9-point scale for each LC with respect to its SC
Find the X_i (Eigenvectors) using Eq. 7, for the LC
Find the CM using Eq. 8 regarding LC
Find the CI using Eq. 10 for LC
Find the CR using Eq. 11 with regard to LC
Check condition if $CI \leq 0.1$
If ($CI \leq 0.1$) compare next LC
Otherwise repeat step 16, 17, and 18
End If
End for loop
Compute the weighted super-matrix
Compute the limit matrix
Output the slave controllers ranking based on the weights (W_i) of the limit super-matrix

The switches migrated to the slave controller should be less than or equal to the processing capacity of the slave controller. Herein, the W_i is obtained from the ANPM, which shows the priority weight of each controller provided from the limit super-matrix.

$$\sum W_i S_i \leq m \tag{13}$$

We explain the process of switch migration in algorithm 2. The weights of the controllers are generated by the ANPM. Hence, if the load of the master controller increase than the threshold of the controller, the switches will be migrated to the slave controllers according to their weights, and the knapsac 0/1 problem will efficiently migrate it, keeping in view the current load of each controller, and the constraints given in Eq. 13, and 14. Herein, the threshold (T) is the maximum flows processing capacity of a slave controller.

Algorithm 2 Algorithm for switches migration

1: **Input:**
Weights of the controllers from the ANP
$W = W_1, W_2, W_3, \ldots W_N$
2: **Output:**
Migrated switches S_i upon fulfilling the following conditions
3: If Load of controller exceeds the Threshold (T)
4: Migrate switches fulfilling conditions given by Eq. 13, 14
5: Else if Load is less than or equal to the (T), then
6: Do not migrate the switches associated with the controller

5 Experimental setup, Results, and Discussion

We have used the Open Network Operating System (ONOS) [35] controller in our experiments to implement the proposed architecture. In addition, Mininet [36] is used for creation and emulation of network topologies. We use Abilene, DFN, OS3E, Interoute, and RedIris topologies to evaluate our proposed methodology. The details of these networks is given in Table IV. We can see a various number of switches and links in each topology such as 58, 34, 19, 110, and 11 for DFN, OS3E, RedIris, Interoute and Abilene topologies. The SDN network with these topologies is created in Mininet and five opnvswitches were used as core switches to connect the E2E domains. The experiments for the validation of our proposed scheme were performed in Mininet Emulator with real-Internet topologies. Both of these (Mininet and real-Internet topologies) provides evidence that our proposed scheme is valid for real-time because the testing is done in real-Internet topologies. Moreover, the Mininet emulation provides some significant advantages over other available simulators such as it runs the real and unmodified code which comprises of application code, the code for the control plane i.e., OpenFlow code for the SDN controller as well as the switches code, and the kernel code for the OS (operating system). In addition, it can be easily connected with the real network by presenting the interactive performance.

Then, we partition the input topologies into 5 domains, administered by 5 ONOS controllers, and assign the switches to the master controllers in each domain according to the controller placement problem illustrated in [37]. Controller placement is to determine the appropriate placements or locations for the controllers in SDN as illustrated in [38], and [39]. Each domain is managed by a single master controller. Mininet nodes communicate with ONOS controllers by reaching their IP addresses. Mininet and ONOS are both in virtual computers that operate on a server. The server is configured with an Intel Corei7 processor running at 3.80 GHz and 32 GB of RAM. Herein, operating system utilized is Linux, namely Ubuntu 20.04.

We have made a comparison of our approach with SASLB [18], DLB [19], SCLB [20], and SMLB [23] schemes. In the static assignment, each switch is allotted to a master controller of

Table 4: An illustration of the network topologies

Network Topology	No. of switches	No. of links
DFN	58	87
OS3E	34	41
RedIris	19	32
Interoute	110	149
Abilene	11	14

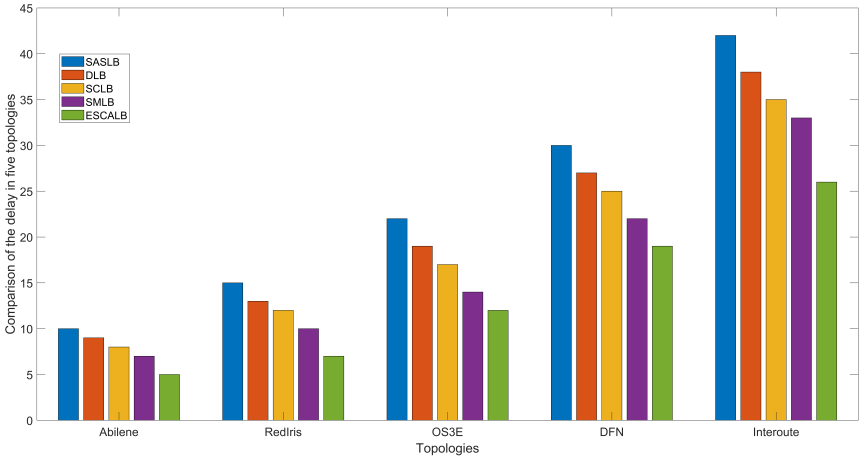


Fig. 3: Comparison of Latency in the five topologies.

the domain to which it belongs, and it selects only the neighbor domain master controller as its slave controller based on the minimum switch distance. However, in the SMLB, TOPSIS [23] is used for neighbor slave controller selection with switch migration.

To run the experiment, we have generated a load i.e. flow requests in the form packet-In messages towards these ONOS controllers. The ANP module in the load balancing plan communicates with the distributed control plan, where the SDN controllers are located. Moreover, it transfers the switches according to the ranking of the controllers, and the switch migration based on the knapsack 0/1 problem. Then the FFUM forwards the flow requests to the controllers which are prioritized by the ANP module according to their weights.

5.1 Evaluation of E2E Latency

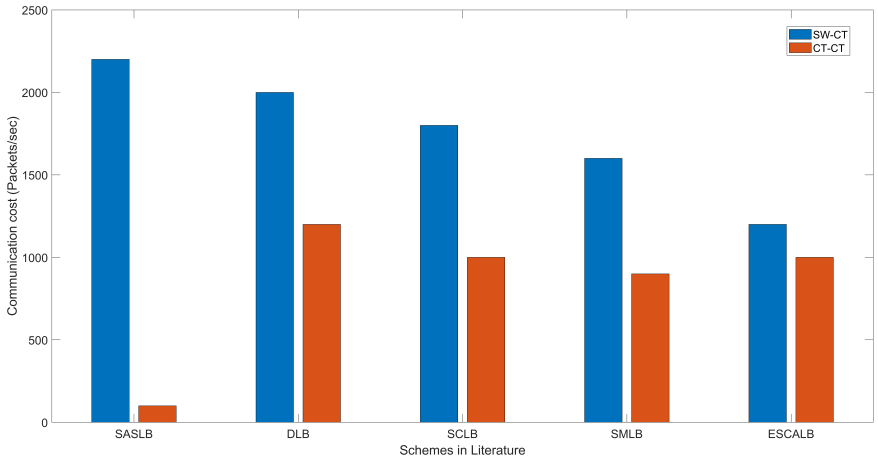
To conduct this experiment, we compute the latency of the switches connected to a slave controller in milliseconds (ms). The latency between each switch and the slave controller is calculated using the shortest path (SP between the two. Eq. (14) shows the end-to-end latency (L_{e2e}).

$$L_{e2e} = L_{SP}(S_i, SC_N) \tag{14}$$

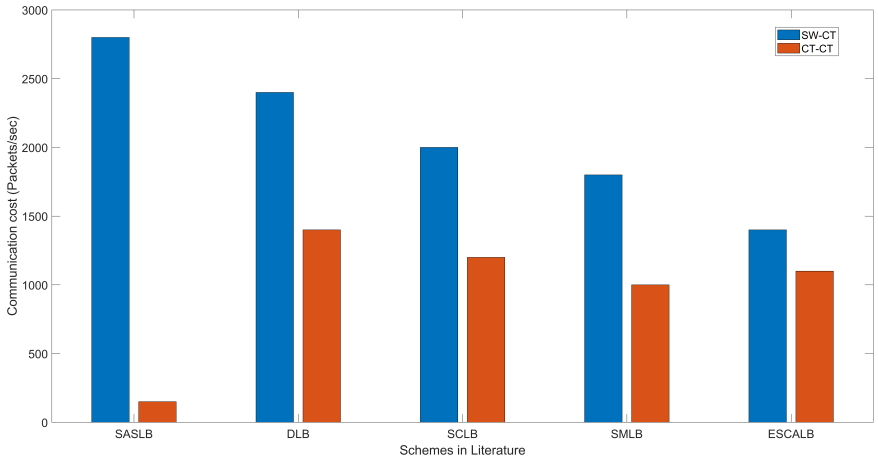
Using five different topologies, Fig. 3 illustrates the (L_{e2e}) of the proposed approach in contrast to benchmark schemes. Each of these topologies were divided into 5 domains having 5 master controllers for each domain. Based on the figure, we can observe that the (L_{e2e}) of our suggested scheme is significantly lower than SMLB and SCLB. Furthermore, as the number of nodes and links scale, an increase in (L_{e2e}) can be detected as shown from the RedIris, OS3E, DFN and Interoute topologies. The (L_{e2e}) of the suggested scheme, on the other hand, is still significantly lower than the previous models even in complex topologies. The efficient slave controller selection leveraging ANP module and switch migration has resulted in a reduction of (L_{e2e}) between the switches and the slave controllers.

5.2 Analysis of Communication cost

The communication cost is represented as the number of packets exchanged from a switch towards controller (SW-CT), and among controllers (CT-CT) as shown in Fig. 4. We evaluate it in Abilene and OS3E networks for our proposed framework. Moreover, we have compared it with prevalent models in the literature. It is important to mention that those schemes involving controllers communication among them have a high CT-CT communication cost. For example, the SASLB has a static relationship between the switches and the controllers. Hence, it has a low communication cost as compared to other schemes as revealed in Fig. 4. However, the disadvantage is the uneven packet distribution among the controllers in SDN.



(a)



(b)

Fig. 4: Comparison of Communication cost in a Abilene, and b OS3E topologies.

5.3 Analysis of Load Curve for controllers

In this experiment, we evaluate the effectiveness of the ESCALB by sending Packet-In requests to C_1. Fig. 5 shows the results for the 500 seconds. The x-axis denotes the time

in seconds and the y-axis shows the average (Avg) load. We can see that as the C_1 gets overloaded there is a sharp increase in the load of C_1 at 200 seconds. Fig. 5 (a) shows the experiment results without applying ESCALB. Fig. 5 (b) shows that when the ESCALB effectively selects C_3 as the controller to which the load is distributed.

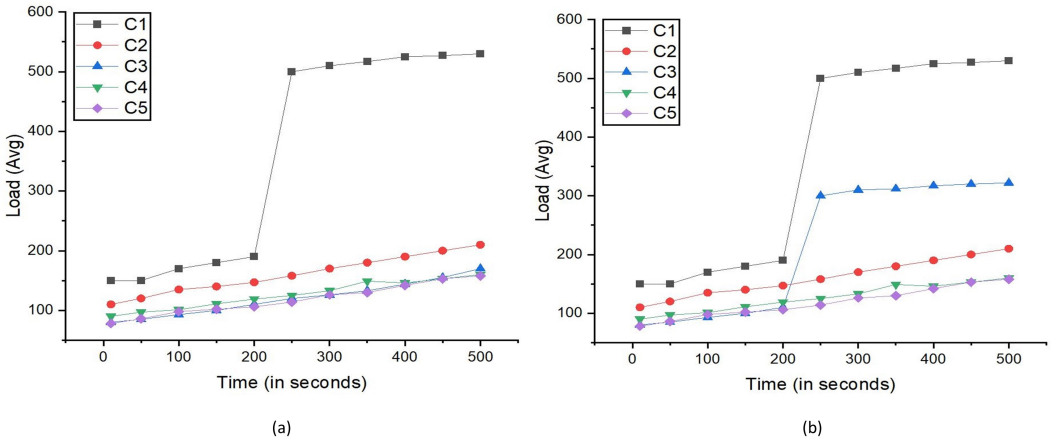


Fig. 5: Comparison of load curve without (a) applying ESCALB and (b) with applying ESCALB

5.4 Migration time

In this experiment we have evaluated the migration time (in seconds) for the ESCALB and other competitive schemes such as SCLB, and SMLB. The migration time denotes the execution time (average) for the ESCALB strategy for the various number of switches taken in the Interoute and DFN topologies emulated in Mininet. We can observe that the migration time is greater in the Interoute topology as compared to the DFN topology. Fig. 6 reveals that with increasing the number of migrated switches the time for migration also increases. However, due to less time for selecting the slave controller for the ESCALB, the overall time for migrating the switches is less in ESCALB as compared to other strategies such as SMLB and SCLB. Moreover, we can also see that in Fig. 6 (a), the time for a migration is less than in Fig. 6 (b) because the complexity of the Interoute network is more than the DFN with respect to the number of nodes in edges in each topology and the distribution of the switches.

5.5 Response Time

Herein, we compare the response time of the controller i.e. the difference between the arrival time for a Packet-In request and the reply time for a Packet-Out message for our proposed strategy ESCALB and other competitive schemes i.e. SMLB and SCLB. Fig. 7 denotes the response time (in milliseconds). We have used stable data from 20 experiments i.e. the average of the response times. Fig. 7 shows that the response time (average) is increasing as the number of arrived packets (Packet-In requests) increases. Hence, at the start when the controller is having a low load then there is less change in the response time. i.e. up to 1000 arrived packets. The results show the efficacy of ESCALB due to an effective slave controller selection strategy.

5.6 Jain's fairness index (JFI)

Fig. 8 depicts a comparison of the proposed scheme with others in the five topologies using Jain's fairness index [40]. This index indicates the allocation of traffic among the slave

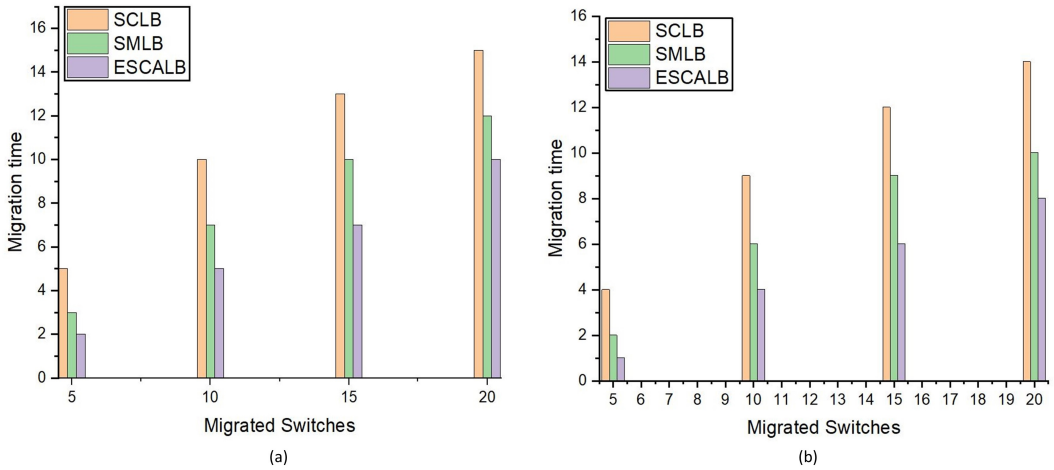


Fig. 6: Comparison of Migration time in (a) Interoute topology and (b) DFN topology

controllers in a fair and equitable manner. The JFI score of 1 indicates that the load has been distributed fairly across the controllers. Fig. 8 demonstrates that the JFI of our suggested technique is either one or very near to one in each of the five topologies tested. For topologies with a lesser number of nodes and connections (Abilene, RedIris, OS3E), we can see from the figure that the JFI is one or near to one. With a rising number of nodes and links in the network of the five topologies, the JFI, on the other hand, is becoming lower. However, our suggested strategy preserves it as near to 1 as possible, whereas the other schemes deviate more from 1. Achieving an equitable distribution of load across controllers is achieved through appropriate controller ranking and switch migration strategies.

5.7 CPU utilization

Herein, we analyze CPU utilization of our proposed strategy and the benchmark approaches. We generated traffic using Iperf and assessed CPU utilization with the sysbench tool. Fig. 9 depicts a CPU use graph with 20-second intervals. According to the graph, CPU usage does not rise from 25% to 30% for the SCLB and SMLB during peak traffic generation. However, our proposed approach, it exceeds 40% during peak traffic generation. The significant improvement in the percentage of CPU utilization demonstrates the effectiveness of our proposed scheme in comparison to alternative approaches. The effective resource utilization is obtained due to effective slave controller selection by an ANPM. Further, to efficiently transfer the switches from a master to the SC knapsack 0/1 effectively migrates the switches from the failed domain controller, by keeping in view the capacity of processing the flow requests of each controller SC.

6 Conclusion and Future directions

In this paper, we proposed a novel load balancing scheme named ESCALB for SDN-enabled IoT in a multi-domain environment to address the QoS problems with a distributed control plan utilizing SDN controllers. With this objective, the proposed ESCALB model monitors the control plan's with real-time load information to rank the slave controller and ensure the effective migration of switches in the network. An ANP was used in coordination of ESCALB model to assure the reliability of the multi-criteria decision process while ranking the controllers to effectively distribute the load among them. Following the advertised instruction, the switches were transferred to a slave controller that utilizes the 0/1 knapsack problem while keeping in view the capacity of the slave controllers. With the collaborative operation of aforestated entities, we properly transmitted switches to the slave controllers and maximized their importance and use in resource allocation and management. As a result, during

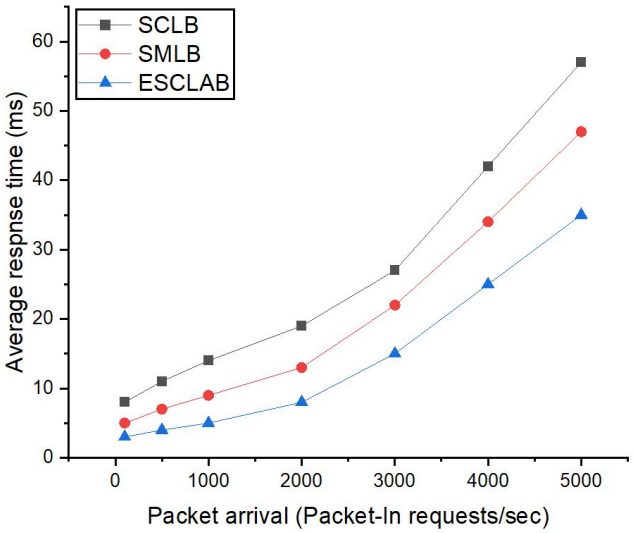


Fig. 7: Curve of Response time with respect to the controller's load.

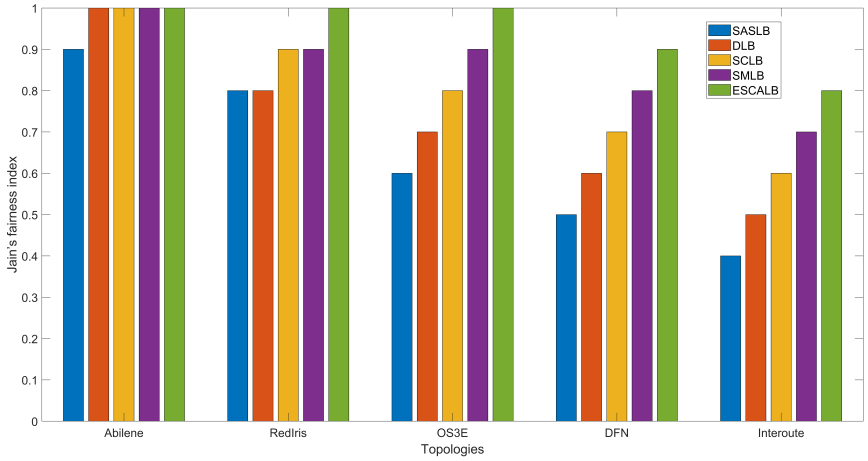


Fig. 8: Comparison of JFI in the five topologies.

simulation, we noted remarkable improvement in CPU usage, latency, communication cost, throughput, and JFI. Moreover, the migration time, response time, and controller load curve also show the effectiveness of the ESCALB. To evaluate the comparative attributes, we used Mininet—a virtual network environment composed of five real-time topologies, whereas we found that our technique overcomes SASLB, DLB, SCLB, and SMLB problems in terms of latency, throughput, CPU use, and JFI in presence of rival schemes. In the future, we will extend the framework for a multi-domain SDN-enabled IoT network with 5G and beyond networks. In the future, we will extend the works to provide more understanding of the theoretical model and extend ESCALB (Proposed) scheme to a more large-scale SDN with a diverse set of real-time traffic scenarios to analyze the performance. For example, the topology with the highest number of nodes and links in our experiment is Interoute having 110 nodes and 149 links. Therefore, we will scale it to a larger number of nodes and links. Moreover, we will extend an artificial intelligence plane that will learn about the network status and delegate the updated flow rules to the data plane.

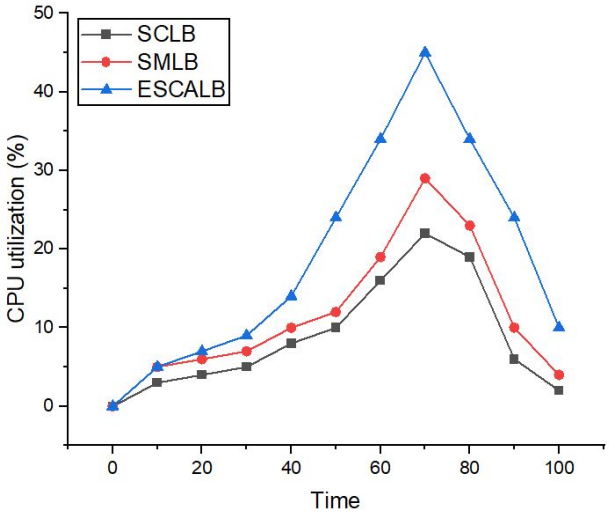


Fig. 9: CPU utilization percentage of the comparative approaches

Declaration of Competing Interest: Authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Ethical approval: This article does not contain any studies with human participants or animals performed by any of the authors.

Acknowledgment: This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2023-2018-0-01431) supervised by the IITP (Institute for Information & Communications Technology Planning & Evaluation)

References

1. D.E. Sarmiento, A. Lebre, L. Nussbaum, and A Chari. Decentralized sdn control plane for a distributed cloud-edge infrastructure: A survey. *IEEE Commun. Surveys Tuts.*
2. Xu. Lingwei, Zhou. Xinpeng, Li. Xingwang, H.J. Rutvij, and G.and Yuan Ding Thippa Reddy. Mobile collaborative secrecy performance prediction for artificial iot networks. *IEEE Transactions on Industrial Informatics.*
3. Elfatih, n.m., hasan, m.k., kamal, z., gupta, d., saeed, r.a., ali, e.s. and hosain, m.s., 2022. internet of vehicle's resource management in 5g networks using ai technologies: Current status and trends. iet communications, 16(5), pp.400-420.
4. M. k. hasan et al., "a novel resource oriented dma framework for internet of medical things devices in 5g network," in ieee transactions on industrial informatics, vol. 18, no. 12, pp. 8895-8904, dec. 2022, doi: 10.1109/tii.2022.3148250.
5. Bhattacharya, p., patel, f., alabdulatif, a., gupta, r., tanwar, s., kumar, n. and sharma, r., 2022. a deep-q learning scheme for secure spectrum allocation and resource management in 6g environment. ieee transactions on network and service management.
6. W. Han, Li. Xingwang, Rutvij H.J., G. Thippa Reddy, Z. Mingfu, A.A. Tariq, and A.K Sunder. Sparse bayesian learning based channel estimation in fbmc/oqam industrial iot networks. *Computer Communications, Elsevier*, 176.
7. Kuppusamy, p., kumari, n.m.j., alghamdi, w.y. et al. job scheduling problem in fog-cloud-based environment using reinforced social spider optimization. j cloud comp 11, 99 (2022). https://doi.org/10.1186/s13677-022-00380-9.
8. Latif, sohaib a., fang b. xian wen, celestine iwendi, f. wang li-li, syed muhammad mohsin, zhaoyang han, and shahab s. band. "ai-empowered, blockchain and sdn integrated security architecture for iot network of cyber physical systems." computer communications 181 (2022): 274-283.

9. C.N. Tadros, M.R. Rizk, and B.M. Mokhtar. Software defined network-based management for enhanced 5g network services. *IEEE ACCESS*, 8.
10. Adil, m., attique, m., jadoon, m.m., ali, j., farouk, a. and song, h., 2022. hopctp: a robust channel categorization data preservation scheme for industrial healthcare internet of things. ieee transactions on industrial informatics, 18(10), pp.7151-7161.
11. S. Ahmad and A.H Mir. Scalability, consistency, reliability and security in sdn controllers: a survey of diverse sdn controllers. *Journal of Network and Systems Management*, 29(1):1–59, 2021.
12. Djenouri, y., belhadi, a., srivastava, g. and lin, j.c.w., 2022. when explainable ai meets iot applications for supervised learning. cluster computing, pp.1-11.
13. Z. Li, Y. Hu, T. Hu, and P. Wei. Dynamic sdn controller association mechanism based on flow characteristics. *IEEE Access*, 7(4):92661–92671, 2019.
14. Montazerolghaem and M. H Yaghmaee. Load-balanced and qos-aware software-defined internet of things. *IEEE Internet of Things Journal*, 7(4):3323–3337, 2020.
15. A. Filali, Mlika. Z., S. Cherkaoui, and A. Kobbane. Preemptive sdn load balancing with machine learning for delay sensitive applications. *IEEE Transactions on Vehicular Technology*, 69(12):15947–15963, 2020.
16. M. Sohaib, Z. Chen, G. Yayu, H. Xiaojun, and C Wenqing. Towards qos-aware load balancing for high density software defined wi-fi networks. *Access IEEE*, 8.
17. T. Wang, F. Liu, and H Xu. An efficient online algorithm for dynamic sdn controller assignment in data center networks. *IEEE/ACM Transactions on Networking*, 25(5):2788–2801, 2017.
18. A. abdelaziz, a.t. fong, a. gani, u. garba, s. khan, a. akhunzada, et al., "distributed controller clustering in software defined networks," plos one, vol. 12, no. 4, 2017.
19. Y. zhou et al., "a load balancing strategy of sdn controller based on distributed decision," in proc. ieee 13th int. conf. trust security privacy comput. commun., pp. 851–856, 2014.
20. T. hu, p. yi, g. guo, j. lan and y. hu, "dynamic slave controller assignment for enhancing control plan robustness in software-defned networks," future generation computer systems, vol. 95, pp. 681–693, 2019.
21. Y. liu, h. gu, f. yan and n. calabretta, "highly-efficient switch migration for controller load balancing in elastic optical inter-datacenter networks," ieee journal on selected areas in communications, 2021.
22. Y. xu et al., "dynamic switch migration in distributed softwaredefined networks to achieve controller load balance," ieee j. sel. areas commun., vol. 37, no. 3, pp. 515–529, mar. 2019.
23. K. s. sahoo et al., "esmlb: Efficient switch migration-based load balancing for multicontroller sdn in iot," in ieee internet of things journal, vol. 7, no. 7, pp. 5852-5860, july 2020, doi: 10.1109/jiot.2019.2952527.
24. B. P. R. Killi and S. V Rao. Capacitated next controller placement in software defined networksvol. *IEEE Trans. Netw. Service Manag*, 14(3):514–527, 2017.
25. H. Aoki and N Shinomiya. Controller placement problem to enhance performance in multi-domain sdn networks. *Proc. ICN*.
26. Y. Jiménez, C. Cervelló-Pastor, and A.J García. On the controller placement for designing a distributed sdn control layer. *Proc. IFIP Netw.*
27. M. Tanha, D. Sajjadi, R. Ruby, and J Pan. Capacity-aware and delay-guaranteed resilient controller placement for software-defined wans vol. 15, no. 3, pp. 991-1005, sept. 2018. *IEEE Transactions on Network and Service Management*, 15(3):991–1005, 2018.
28. Bukhsh, m., abdullah, s., rahman, a., asghar, m.n., arshad, h. and alabdulatif, a., 2021. an energy-aware, highly available, and fault-tolerant method for reliable iot systems. ieee access, 9, pp.145363-145381.
29. M. hamdan, e. hassan, a. abdelaziz, a. elhigazi, b. mohammed, s. khan, a.v. vasilakos and m.n. marsono, "a comprehensive survey of load balancing techniques in software-defined network. journal of network and computer applications," 2021.
30. R.k. das, f.h. pohrmen, a.k. maji and g. saha, "ft-sdn: a fault-tolerant distributed architecture for software defined network," wireless personal communications, vol. 114, no. 2, pp. 1045-1066, 2020.
31. Open network foundation. (feb. 2016). sdn architecture 1.1.
32. J. liu, j. wan, b. zeng, q. wang, h. song, and m. qiu, "a scalable and quick-response software defined vehicular network assisted by mobile edge computing," ieee commun. mag., vol. 55, no. 7, pp. 94–100, jul. 2017.
33. J. ali and b. -h. roh, "an efficient approach for load balancing in software-defined networks," 2021 6th international conference on computing, communication and security (icccs), 2021, pp. 1-6, doi: 10.1109/icccs51487.2021.9776348.

34. G. cheng, h. chen, z. wang, s. chen, dha: Distributed decisions on the switch migration toward a scalable sdn control plane, in: Proc. ifip networking conference ifip, 2015, 471-477.
35. Open network operating system. https://opennetworking.org/onos/, accessed online on 10th dec, 2022.
36. R. l. s. de oliveira, a. a. shinoda, c. m. schweitzer and l. r. prete, "using mininet for emulation and prototyping software-defined networks," in proc. of the ieee colombian conf. on communication and computing (colcom), colombia, usa, pp. 1–6, 2014.
37. G. wang, y. zhao, j. huang and y. wu, "an effective approach to controller placement in software defined wide area networks," ieee transactions on network and service management, vol. 15, no. 1,pp. 344–355, 2018.
38. Wang, g.; zhao, y.; huang, j.; wang, w. the controller placement problem in software defined networking: A survey. ieee netw. 2017, 31, 21–27.
39. Ali, j.; roh, b.-h. an effective approach for controller placement in software-defined internet-of-things (sd-iot). sensors 2022, 22, 2992. https://doi.org/10.3390/s22082992.
40. R. jain, d.-m. chiu, and w. r. hawe. jain, a quantitative measure of fairness and discrimination for resource allocation in shared computer systems. dec research report tr-301, 1984.