*Computers, Materials & Continua*
DOI:10.32604/cmc.2021.014627
***Article***

Tech Science Press

# Load Balancing Algorithm for Migrating Switches in Software-Defined Vehicular Networks

**Himanshi Babbar[1], Shalli Rani[1,*], Mehedi Masud[2], Sahil Verma[3], Divya Anand[4] and Nz Jhanjhi[5]**

[1]Chitkara University Institute of Engineering and Technology, Rajpura, 140401, India
[2]Department of Computer Science, College of Computers and Information Technology, Taif University, Taif, 21944, Saudi Arabia
[3]Department of Computer Science and Engineering, Chandigarh University, Mohali, Punjab, 140413, India
[4]Department of Computer Science and Engineering, Lovely Professional University, Phagwara, 144001, India
[5]Taylor's University, Selangor Darul Ehsan, Malaysia
[*]Corresponding Author: Shalli Rani. Email: shalli.rani@chitkara.edu.in; shallir79@gmail.com
Received: 04 October 2020; Accepted: 02 December 2020

**Abstract:** In Software-Defined Networks (SDN), the divergence of the control interface from the data plane provides a unique platform to develop a programmable and flexible network. A single controller, due to heavy load traffic triggered by different intelligent devices can not handle due to it's restricted capability. To manage this, it is necessary to implement multiple controllers on the control plane to achieve quality network performance and robustness. The flow of data through the multiple controllers also varies, resulting in an unequal distribution of load between different controllers. One major drawback of the multiple controllers is their constant configuration of the mapping of the switch-controller, quickly allowing unequal distribution of load between controllers. To overcome this drawback, Software-Defined Vehicular Networking (SDVN) has evolved as a configurable and scalable network, that has quickly achieved attraction in wireless communications from research groups, businesses, and industries administration. In this paper, we have proposed a load balancing algorithm based on latency for multiple SDN controllers. It acknowledges the evolving characteristics of real-time latency *vs.* controller loads. By choosing the required latency and resolving multiple overloads simultaneously, our proposed algorithm solves the load-balancing problems with multiple overloaded controllers in the SDN control plane. In addition to the migration, our algorithm has improved 25% latency as compared to the existing algorithms.

**Keywords:** Software-defined networking; load balancing; multiple controllers; ryu controller; mininet

## 1 Introduction

In the past few years, the use of Internet-enabled smart devices in various applications such as vehicular networks, Internet of Vehicles (IoV), data centers, and Internet-of-Things (IoT) has been growing tremendously [1]. The SDVN is a standard implementation of the Internet of Things (IoT) innovation in the smart transport network (STN) market as shown in Fig. 1. In the IoV, different services are required to be delivered, including collision warning, road traffic detection, traffic management, infotainment, and so on. It allows the traffic to be convenient for traffic and people. Thus, supporting such applications requires a powerful data center. Interaction between smart devices is utilized for computing and communicating around the world which can lead to the development of heavy data internet traffic [2]. A decentralized method of control called SDN is commonly used to manage this huge amount of data congestion to determine the correct service delivery, facilities, and implementations to the smart devices' end-users. SDN is an evolving network standard and it is among the key prominent research areas, which controls the network in such a systematic, unified, and programmable way by disaggregating the control plane and data plane and hence a desirable solution to an IoV [3]. The huge traffic generated in this process is controlled by the datacenter raises numerous problems such as response time, unbalance in load, system overload, and the disruptive traffic flow for Internet-enabled smart gadgets in the immense scaled networks.

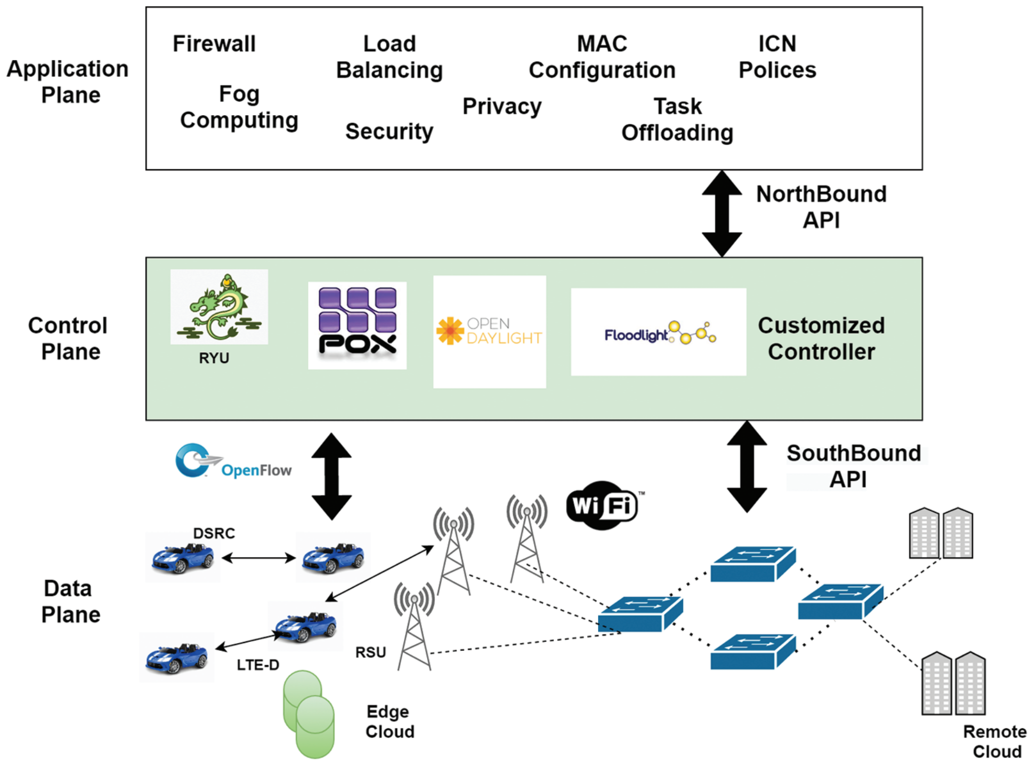


**Figure 1:** Software-defined vehicular networks

The method adopted in conventional networks had some constarints, and for these constraints solution is SDN. SDN has a few core characteristics:

1) SDN has been approached by many academies and businesses in the past several years and had several benefits due to the network virtualization technologies [4].

2) SDN is most notable aspect is the segregation of control and data planes. Data plane manages specific network modules that are capable of handling and controlling the various data plane features.
3) Scenario of SDN can be observed from Fig. 1. SDN offers services to all the problems in the conventional network that requires separating the control plane for every network device [5].

Over the past several years, there is enormous increase in the amount of data centers that have switched to Software Defined Networking (SDN) to satisfy the needs of an evolving cloud-driven industry [6]. With the current study, it has been examined that the market of SDN has accounted for: $10.80 billion in 2015 is now expected to reach $135 billion in 2022 which is growing at a Compound Annual Growth Rate (CAGR) of 43% from 2015 to 2022. SDN is continuously rising at a greater speed. The entire area of SDN is composed to surge at a CAGR of 42.41% during the tenure of 2019–2024. The expanding growth in big data is greatly increased with the adoption of software-defined data centers across all the areas that include retail, IT, telecom, etc. Software-Defined infrastructure aims at implementing the automation processes that run smoothly in organizations [7]. Instead of handling a plethora of appliances independently, SDN allows the administrator to handle their entire network as a single unit and make the required changes from a single point of access whenever required.

SDN load balancing has also brought forward the concept of multiple-controller SDN. For example, the SDN southbound interface protocol, OpenFlow v.1.3, can technically handle controller replication and load-balancing. There are three main phases of an SDN controller: master, slave, and equal [8]. A controller could alter its location at any moment but only the controller in the master state has complete access to the switch. It supports flexible switch migration of many distributed controllers, therefore it is a simple and efficient strategy for load-balancing. The control plane's tentative design uses only one controller. Though the benefits of a central controller could alter its location at any moment, however only the controller in the master state has maximum switch visibility. SDN network control confronts some issues which challenge its nature (i.e., centralized control) due to a growing daily operations-today network demands.

Therefore, in this paper, we proposed a load balancing switch migration scheme using the multiple controllers in software-defined vehicular networks to overcome the failure of a single controller on one server which results in a highly available and scalable multi-controller in SDN. To the best of our knowledge, to simultaneously solve the problem of why, what, and how; the load balancing will be performed on a heavily loaded switch to immigrate to the least loaded controller. There are many methods proposed in the literature but our scheme gives the best results as compared to the previously implemented algorithms.

## *1.1 Research Contributions*

This paper differentiates from the existing body of literature in terms of:

i. Irrespective of the conventional load balancing algorithms for vehicular networks, this paper advertises switch migration for vehicular networks in a vehicular environment.
ii. We designed a load balancing approach to measure the load of controllers and switches linked to the controller and notifying the load information systematically to the super controller through transmitting messages which is the essential part of the load balancer controller.

iii. We evaluated the response time operations *vs.* controller load that can assess the development tendency in latency for the controller as load increases.
iv. We used latency to measure controller load, and we acquire an appropriate latency threshold to better identify overload controllers to manage them in advance.
v. To enhance the failure of single controller load balancing, we proposed a new switch migration algorithm to handle multiple loaded controllers. Whenever there is huge load on one controller in a single load-balancing operation for software-defined vehicular networks, then the proposed methodology handles the load smoothly.

### 1.2 Paper Organization

The remaining of the paper is organized as: Section 2 presents the literature survey on SDN load balancing and laid the focus on the algorithms of load balancing. Section 3 explains the multi-controller load balancing in SDN. Section 4 discusses the details of the preliminary network setup of our proposed algorithm. Section 5 focuses on the implementation and results analysis and finally, in Section 6 conclusion of the paper is presented.

## 2 Literature Survey

Load balancing in vehicular networks needs a lot of improvement at its early stage. There are multiple proposals of research that exist in the literature survey which resolves the issues of load imbalance amongst the servers and various controllers. For instance [9], authors analyze a flexible control method to determine which switch and where to relocate to a balanced control plane and recognize it as a switch migration problem (SMP). The switch migration method is being proposed by [10] which perceives the switch migration as a signature selection problem but is developed as the Earth Mover's Distance (EMD) framework to safeguard technically critical network controllers and later developed the heuristic load balancing method which is efficient and is scalable for the immense scale networks.

The authors of [11] evaluated SDN based on an optimization load balancing algorithm that utilizes the effect of the linear increase in the load and diminish the weight of inertia to improve the execution of constrained swarm optimized which further would reduce the response time and enhance the Quality of Service (QoS) metrics. In [12] authors developed the load balancing strategy based on switches for multiple controllers. This strategy balances the load between switches and controllers as well as resolves the back and forth of the load and helps to enhance the efficiency. Trestian et al. [13] implements the OpenFlow load balancing algorithm for the data centers network that allows the efficient and effective use of resources at the least cost and complexity. The reliable and load balance aware multi-controller in [14] proposed this approach to initiate the multi-controllers and discuss the related QoS for the reliable load balancer and later the algorithms permit the switches to controllers to balance the load distribution among the various controllers. Nkenyereye et al. [15] framed the concepts that implement the software-defined vehicular networks based system which is based on the modeling and execution schemes. Later the authors presented the different architectures and their respective system models for LTE-V2X interactions.

The authors of [16] proposed the multi-controller load balancing approach for the SDN termed as Dynamic Clustering. In this whenever an overloaded cluster is detected then the Super Controller (SC) runs the partition algorithm that rearranges the Regular Controller's (RC's) into the clusters of RC's and therefore, it simplifies the operation of load balancing and divides the dependency between the SC's and RC's during the periodical load balancing. Askar [17] proposed an adaptive load balancing technique uses the OpenFlow switches that were programmed to work using the function it can be a switch, hub, or router. OpenFlow switches operate in the charge of the controller that is interconnected to the entire switches and has an overview of the complete network and its corresponding resources. The authors of [18] facilitated the controllers of the SDN optimization problem for balancing the load in a mathematical model. Later authors proposed a Dijkstra's Routing algorithm in the SDN nodes which can suggest the separated multipath routing.

Authors of [19] load balancing problem was formulated as a Mixed Integer Non-Linear Programming (MINLP) to minimize the mean reaction time between both the switch and the controller. Nevertheless, a load-balancing better solution is needed to handle the flow communication between OpenFlow switches so that the switch migration costs and controller load variance are shared. Tennakoon et al. [20] developed an approach known as Q-Learning for balancing the load in Software Defined Networks to degrade the number of users who are not satisfied in a 5G network. This solution combined Q-Learning techniques with a fairness function to improve the user experience at peak traffic conditions. The authors of [21] framed the proposed EASM (Efficiency Aware Switch Migration) for effectively migrating the switches. In this, we have evaluated the performances of EASM algorithms that are against the baseline schemes. EASM diminishes the response time for effective migration.

## 3 Multi-Controller Load Balancing in Software-Defined Networking

In this section, we emphasized the multi-controller load balancing research in SDN. We firstly introduced the concept of a single controller and then illustrated the multi-controller load balancing.

### *3.1 Load Balancing in SDN*

The servers in the network get loaded up with the load as the request from users are rising. Therfore, the load must be balanced to provide a better service and meet QoS requirements. If all this problem is overlooked, then it results in the link failure and server crash [8]. Compared to conventional networks, the switches only have a data plane with them while splitting all control planes from the switches and progressing them to a central system in SDN called a controller. Therefore, an enormous volume of data needs to be collected and uploaded to the cloud. The networking resources have to be distributed continuously and effectively for improved user satisfaction and better productivity when accessing the data [12]. A load balancer function is to maximize bandwidth to boost efficiency, with low latency, using the resources effectively with no deadlocks, but not imposing extra overhead on the network. Although there are issues with reliability and scalability with a single controller. Thus, to solve this issue, getting distributed multiple controllers is an alternative for east-west interfaces that allows those controllers to interact among themselves.

### *3.2 Single and Multi-Controller Load Balancing*

During the initiation phase of SDN architecture, the overall network is controlled by a single controller [22]. In Fig. 2, four network switches are controlled by a controller (c1), if the source host wishes to transmit a new packet to switch (s1), the switch could not perform the forwarding function as no routing information is available for the new packet. The switch (s1) instead exchanges (Packet-in) messages to the controller (c1) to enable the routing for the new packet. Upon obtaining a controller response, migrate to another network, and then transmit the packet. Eventually, the packet effectively achieves the destination host. In the flow of transferring the packet, the controller plays an essential role [13]. Because of the limited capacity of the controller, a single controller is not able to deal with the huge volume of requests transferred from the switches. Therefore, switches cannot organize the routing information of the newly received packets which unfortunately would affect the communication and network applications.

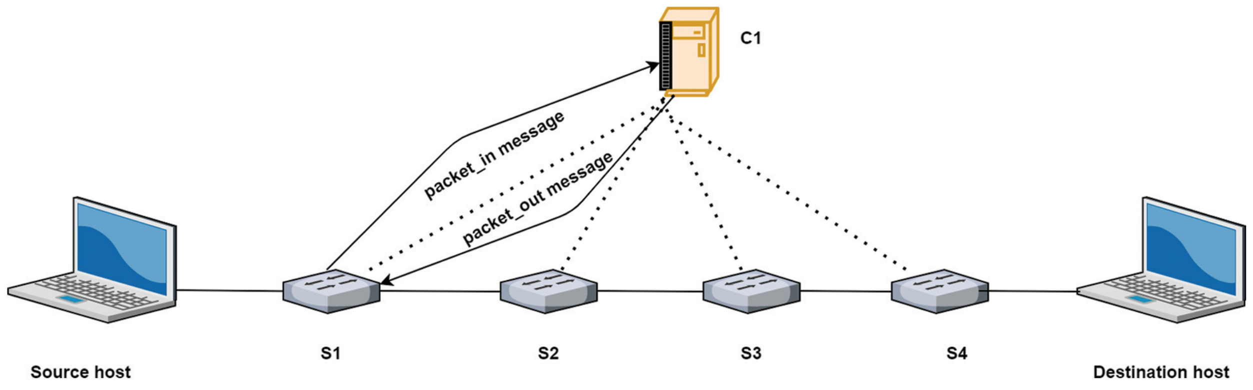


**Figure 2:** Single-controller load balancing

Due to the constraint in the implementation of OpenFlow has suggested the idea of master and slave controllers where one switch can interact with one controller and various slave controllers [23]. In Fig. 3 multi-controllers became a new SDN paradigm that resolves the issue of a single controller. There are two Ryu SDN controllers in the topology given below and every controller handles the network. In this case, (c1) and (c2) share a certain logic in a logically unified way, so that both (c1) and (c2) will enable routing paths explicitly in all relevant switches when new packets arrive at (s1). That means, it can appropriately relieve a single controller's flow processing load [14]. To resolve the issue of a single point of failure, the two Ryu controllers are the backup for each other. Based on the above analysis, it is observed that multi-controller design can efficiently improve the performance of the network of SDN and in the future, the utilization of multi controllers became modern research.

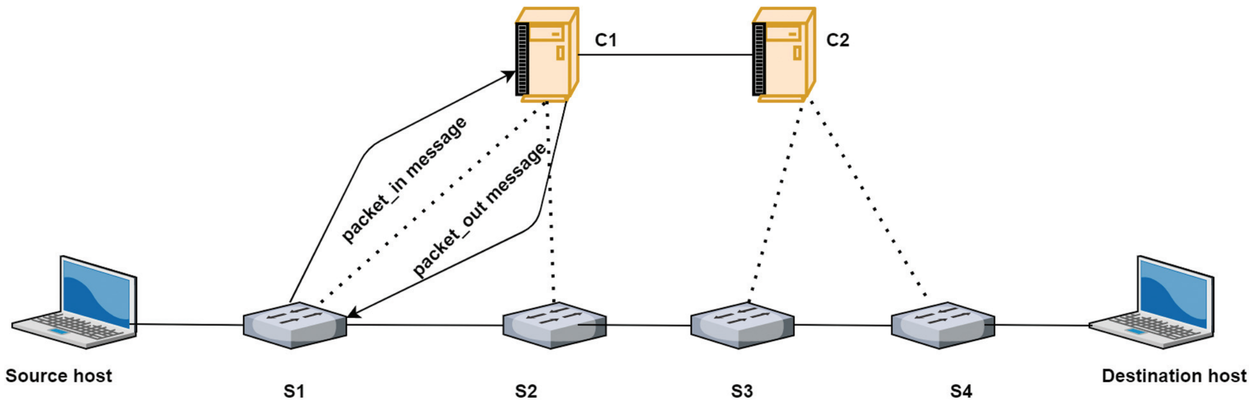


**Figure 3:** Multi-controller load balancing

Therefore, to balance the load among the switches in the various controllers, the important aspect is how the load can be balanced, on which parameters the load will be balanced, and how the load can be calculated to facilitate the minimum results in respect to latency, the volume of packet-in messages are arriving from the switches, how much memory is consumed [24], what is a load of rule installation this will give rise to the effect of load on the multi-controllers and finally it will produce the minimum migration cost and time [25].

## 4 Preliminary Network Setup for Proposed Algorithm

In this section, we proposed the algorithm for load balancing that explains the design.

### *4.1 System Model Designing*

In this section, we present the major components of load balancing that are the essential part of our network design followed by our switch migration mechanism. Normally, when a switch encounters a packet, designed to match a non-corresponding flow table entry in an SDN network, the incoming packets must be summarized in a PACKET IN message and moved to their master controller for routing and flow table entry execution. PACKET IN message processing is commonly considered to be the most significant aspect of controller load. Thus, the massive distributional difference between PACKET IN messages could lead to imbalance workloads amongst multiple controllers. Within this section, we introduce the load balancing algorithm based on migrating switches in the Software-Defined Vehicular Networks technique to balance the load with various overloaded controllers, making a fine-grained based on latency for overloaded controllers.

We designed a network of SDN Graph G comprised of X controllers as shown in Fig. 4 where $A =$ "$\{A_1, A_2, A_3, \ldots, A_X\}$" and Y switches $B =$ "$\{B_1, B_2, B_3, \ldots, B_Y\}$". Such controllers serve the same purpose and split the network into N regions. The traffic flow and controller volume of work are dynamically altered in each region. Let us assume $R_{Am} \in B$ denotes the switch set is handled by the master controller $A_m$. In this section, we have described the design of the algorithm, the rest of the section consists of problem formulation and measurement of load.

### *4.2 Measurement of Load and Acquiring Latency*

An actual outcome for recurrent statistics is introduced into our scheme calculating the latency at the same time interval, Z. Let us say $B_y$ and $Z_x$ to indicate the yth switch and xth time, respectively. Assume $z_{arrive}$ indicates the PACKET IN message arrival time and $z_{reply}$ indicates the time till we receive the reply from the controller in the form of PACKET OUT message to the corresponding switch. Therefore, it is easy to receive the latency to the PACKET IN request as defined in Eq. (1).

$$z_{response} = z_{reply} - z_{arrive} \tag{1}$$

In the meantime, the volume of messages requested is recorded from $B_y$ in $Z_x$ (indicated as $(gB_yZ_x)$) which is used to represent the workload received by $B_y$ in $Z_x$ (indicated as $Load_{ByZx}$). Thus, the total load of the switches in $R_{Am}$ is defined as the volume of requests messages received $Load_{AmZx}$ of controller Am in the xth time and is represented as in Eq. (2):

$$Load_{AmZx} = \sum_{B_y \in R_{Am}} (gByZx) \tag{2}$$

We signify the latency as $z_{B_y}$ *response* to the entire PACKET IN requests forwarded from $B_y$ in $Z_x$. Consequently, $A_m$ in $Z_x$ can easily access the overall latency of all messages recognized. After receiving these data, we could even acquire the average latency directly to a single PACKET IN message with an interval of time Z in Eq. (3):

$$z_{A_m Z_x} = \frac{\sum B_y \in R_{A_m} z_{B_y} \text{ response}}{\text{Load}_{A_m Z_x}} \tag{3}$$

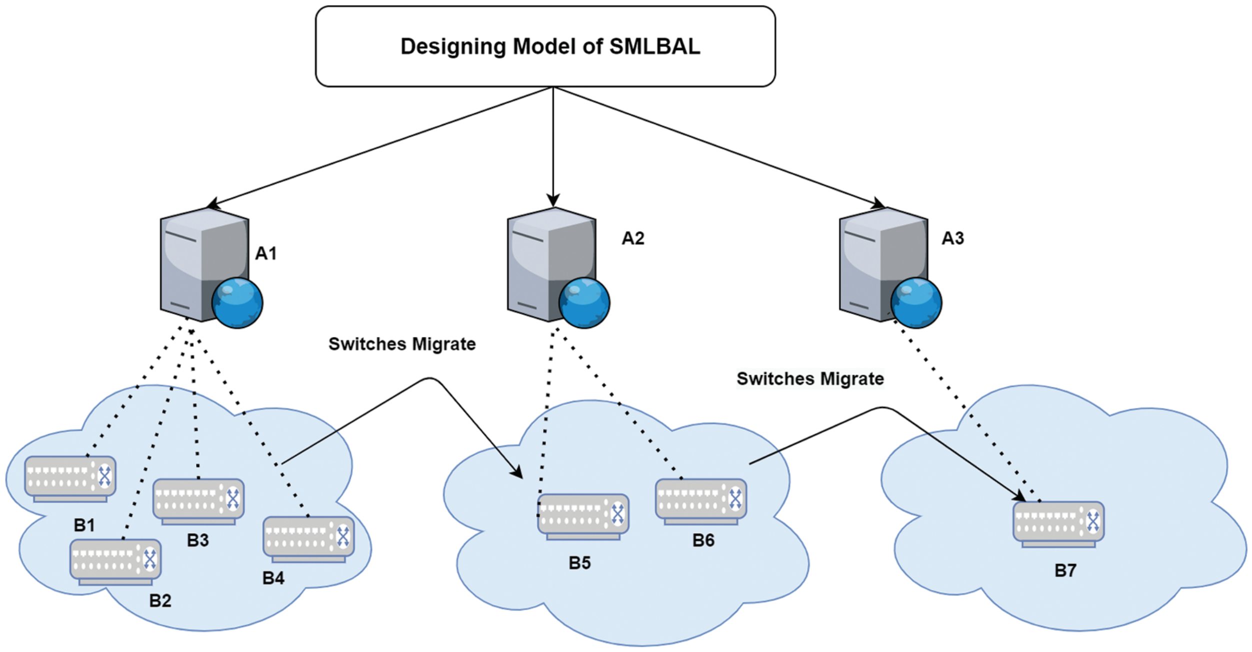


**Figure 4:** Design model of migrating switches

### *4.3 Threshold Based on Latency*

To identify the overloaded controller and load imbalance occurrences, and to identify the necessary migration time, a threshold value is needed. The threshold value isn't static in the present section, since attempts have been made to balance the entire load of the control plane as precisely as possible among all the controllers. If the threshold value is less than the controller's load then it is likely to unbalance the controller due to the massive traffic which leads to the high cost of migration and faulty stability of the network. Therefore, choosing the threshold value is important for the precise load imbalance judgment.

For this, we need to calculate the controller's load based on latency. The experiment testbed was implemented using a Linux Ubuntu server system. The experiments ran on an Ubuntu Server 18.04 (64-bit) LTS with an Intel i7 core 8th generation and 16 GB of memory. The hypervisor was used in VirtualBox 5.2, Mininet, Ryu, and tested load balancing algorithms. Python 2.7 is used to write the main logic in Ryu as well as the topology used in Mininet. Communication between the controller and switches occurred on port 6653 and utilized the OpenFlow 1.3 protocol. Each experiment runs for 1000 s or 10 min.

### *4.4 Controller's Load Detection*

The sudden and unceasing rise in latency is used to identify the best load-balancing time. In this, what we have to do is begin to compare the latency of all controllers with the required threshold to figure out which controller is easy to access and tends to be in a huge load state. To get the overloaded controllers, set the latency of all the controllers present in a given time. The two sets Source Controllers and Destination Controllers are used to efficiently choose overloaded controllers, and lower load shifting pair complexity. Source and Destination controllers are designated as SRC_C and DST_C respectively. Therefore, we examine the controller's latency and determine which controllers should be chosen as SRC_C and which should be chosen as DST_C.

- If $z_{AmZx} >$ threshold value; $A_m$ is added to SRC_C.
- If $z_{AmZx} <=$ threshold value; $A_m$ is added to DST_C.

We obtain the interval of time $Z_x$, we evaluate the set SRC_C and DST_C to acquire the latency time of all the controllers. In this section, we conclude the load balancing algorithm by facilitating the flow of detection of load balancing as mentioned in Algorithm 1 in Tab. 1, and the flowchart for the algorithm proposed is shown below in Fig. 5.

**Table 1:** Controller's load detection

**Algorithm 1:** Load balancing detection based on the controller's load in software-defined vehicular networks

```
Input: D = {$z_{A1Zx}$, $z_{A2Zx}$, $z_{A3Zx}$}
Output: SRC_C and DST_C
Begin:
1. Set the controller and initialise SRC_C = empty and DST_C = empty
2. Assume j to be the number of the controller compared
3. for j = 1 → 3 do
4.     choose $z_{A1Zx}$ from the set
5.     if $z_{AmZx}$ > threshold value
6.         $A_m$ is added to SRC_C
7. else
8.         $A_m$ is added to DST_C
9.         endif
10. endfor
11. return SRC_C and DST_C
End
```

### *4.5 Switches Migration*

When there are multiple overloaded controllers, we suggest the technique based on migrating the switches in a single controller load balancing to boost the performance of the load balancing algorithm. It can do the shifting of a load of the multiple controllers too, reducing load-balancing time considerably. Balancing network traffic among controllers, is the key factor to consider for a congestion-free network, and to have that, migration protocol is one where switch migration occurs if an imbalance network resides [26]. It is the main SDN controller function that aims to provide an improved system environment. To improve network efficiency and prevent the crashing

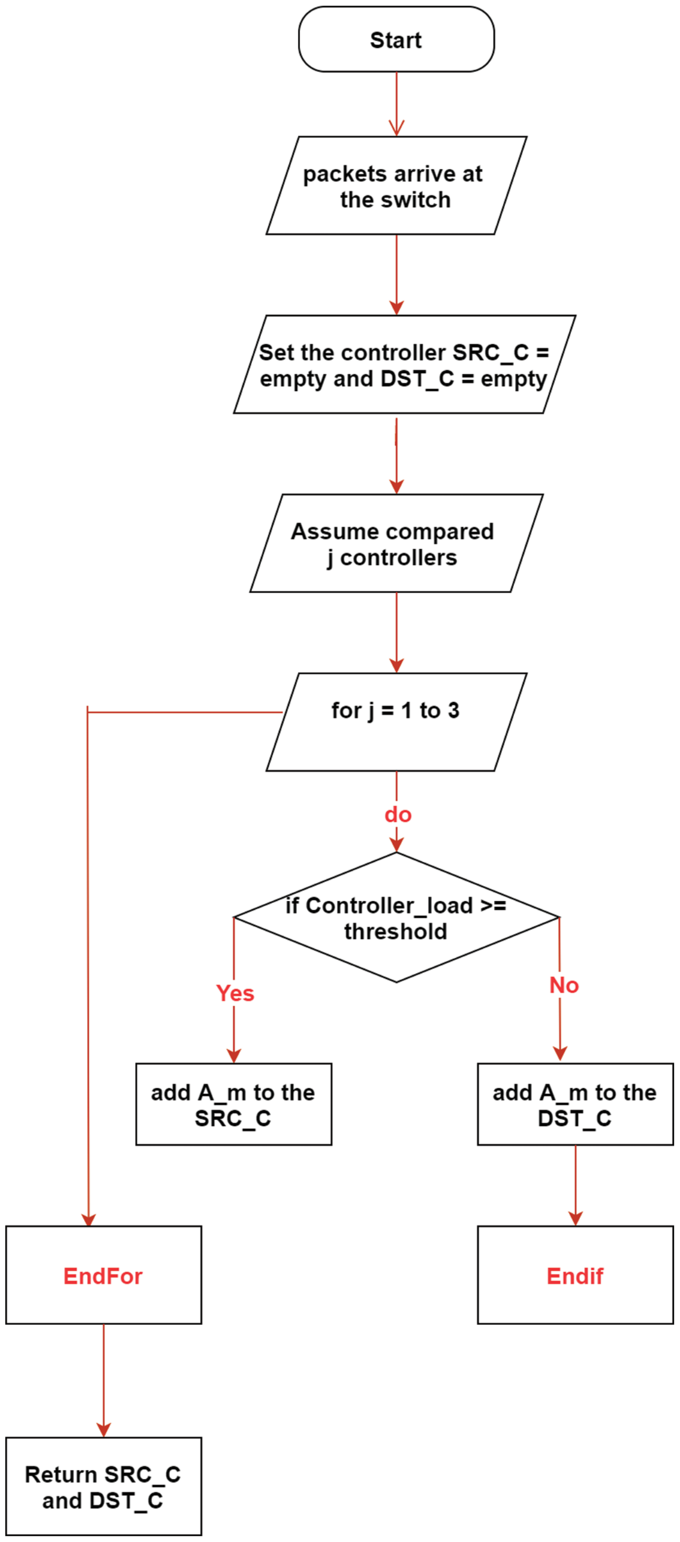

Start
packets arrive at the switch
Set the controller SRC_C = empty and DST_C = empty
Assume compared j controllers
for j = 1 to 3
do
if Controller_load >= threshold
Yes
No
add A_m to the SRC_C
add A_m to the DST_C
EndFor
Endif
Return SRC_C and DST_C

**Figure 5:** Flowchart of proposed model

of controllers, packets must be evenly distributed amongst controllers. By moving the transition from the master controller to the slave controller, load balancing is accomplished [27]. When trying to migrate a switch, two principal concepts must be kept in mind:

a. It should not be overloaded if the slave controller is called to become a new master controller.
b. After migration, the controller must have a balanced load.

To determine how and where to move the switches, we need to see whether the controller is overloaded and test the viability of each linked switch. If yes, switch to the lightly loaded controllers to remove the gap among the controller and the switch, then continue with the process of packet transmission.

Algorithm 2 in Tab. 2 states that the migration of switches is evaluated as a set of migration actions followed by an ($A_u$, $B_e$, $A_v$).

**Table 2:** Migrating the switches

**Algorithm 2:** Load balancing algorithm based on migrating switches in software-defined vehicular networks

```
Input: Switches load information in SRC_C and DST_C
Output: SM: Set of migration of switches
Begin:
1 Set the controller migration set SM
2 if (SRC_C ∩ DST_C! = NULL) then
3 for each controller set in $A_u$ = largest workload
4       $A_m \in$ SRC_C managed by $Load_{AmZx}$
5 for each controller set in $B_e$ = largest
6       $B_y \in R_{Au}$ managed by $Load_{ByZx}$ then
7 Calculate the Av = smallest workload of
8       $A_m \in$ DST_C controlled by $Load_{AmZx}$
9          insert ($A_u$, $B_e$, $A_v$) to SM
10         eliminate $A_u$ from SRC_C
11         eliminate $A_v$ from DST_C
12 endif
13 return SM
End
```

The steps followed are:

1. If the condition SRC_C ∩ DST_C! = NULL which says controller overloaded must be balanced and the least loaded controller required to accept the shifting of load. Therefore, the migration of switches in balancing the load is needed.
2. For each controller set in $A_u$ of $A_m \in$ SRC_C managed by $Load_{AmZx}$, indicates the largest load on the controller.
3. Switch $B_e$ has the highest load on the switch is chosen.

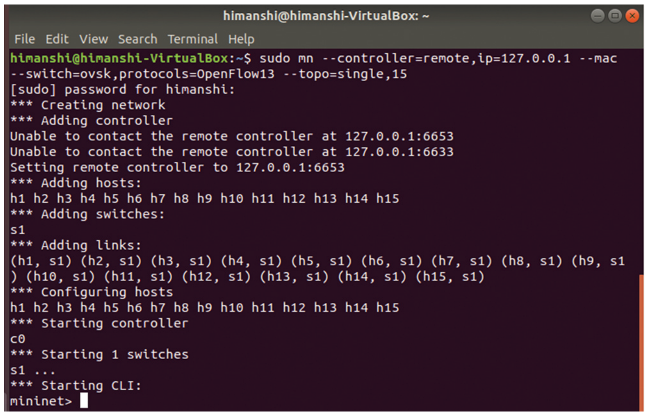
4. In the destination controllers, i.e., DST_C comparison of load can be considered according to the latency. Therefore, we select the $A_v$ with the smallest load is chosen for migration of load.
5. Insert ($A_u$, $B_e$, $A_v$) to Switch Migration set SM. Then later eliminate $A_u$ from SRC_C and $A_v$ from DST_C.

Lastly, the shifting of load and migration of switches is accomplished and we have decided the migrations for the overloaded multiple controllers. By these steps of implementation, our proposed algorithm is optimizing the process of packets' transmission.

## 5 Implementations and Performance Evaluation

We have discussed the load balancing algorithm based on migrating switches in Software-Defined Vehicular Networks. We present an analysis that can efficiently manage the allocation of loads between multiple controllers. This demonstrates what controller will encounter when there is increased delay which is responsible for bottleneck in performance. The detection of load balancing of all the controller's average latency to compare it with the threshold value. If the load is not balanced, the algorithm will migrate the switches to the under-loaded controllers (DST_C) to make it balanced.

We are implementing Ryu as a controller for the SDN. Throughout simulations the average setting time, T has been estimated to be 10 s. A brief period would then vastly enhance monitoring and overhead computation and can lead to regular migration. Several other important network state information may be missing in a long-time-interval because the default flow time in the Ryu controller is 10 s. In this paper, we are emulating the real-time application on the internet with the Zookeeper having OpenFlow switches (OVS) and 15 hosts shown in Fig. 6 and Tab. 3, all the switches/hosts are connected to the controllers but the whole network will be managed by the master controller.


himanshi@himanshi-VirtualBox: ~
File Edit View Search Terminal Help

```
himanshi@himanshi-VirtualBox:~$ sudo mn --controller=remote,ip=127.0.0.1 --mac
--switch=ovsk,protocols=OpenFlow13 --topo=single,15
[sudo] password for himanshi:
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Unable to contact the remote controller at 127.0.0.1:6633
Setting remote controller to 127.0.0.1:6653
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1) (h5, s1) (h6, s1) (h7, s1) (h8, s1) (h9, s1
) (h10, s1) (h11, s1) (h12, s1) (h13, s1) (h14, s1) (h15, s1)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```

**Figure 6:** Customized creation of single topology

The performance of the topology can be evaluated by the traffic load on the planes. It enhances the availability and scalability while transmitting the packets through the planes. Therefore, we choose a lightweight emulator for analysis of the tool, mininet, and edit 7 OVS to inject the packet inflows to the respective controllers. There are three controllers required to set up the

7 hosts/OVS: Controller 1 to Controller 3. We have bifurcated all the 7 OVS into 3 controllers to balance the load of the different controllers.

**Table 3:** Parameters for experimentation

| Parameters | Description |
|---|---|
| Architecture of controller | Centralized SDN controller |
| Controller | Ryu |
| Total controllers used | 3 |
| Switch used | Open virtual switch |
| Protocol version | OpenFlow v1.3 |
| Load testing tool | iperf, TCP/UDP |
| Number of servers | 15 |
| Number of users/clients | 250 |
| Quality of Service (QoS) metrics | Throughput, response time, average utilization of load. |

In this section, we compared the four techniques with our proposed technique: Distributed Hoping Algorithm (DHA), switch migration-based decision-making (SMDM), online controller load balancing (OCLB), traffic pattern based load balancing algorithm (TPLB), and the proposed algorithm. The distribution of load technique is applied by enhancing the requested messages through the packet in for 20 s. Various strategies can be applied balance the load and to get the stable state before 150 s. Assume the ratio indicating the distribution of load as given in the Eq. (4):

$$\text{Ratio} = \frac{\text{Total Load of all controllers (Ax)}}{\text{Average load of 3 controllers (A)}} \tag{4}$$

We compared the five techniques of load balancing with our proposed algorithm in Figs. 7 and 8.

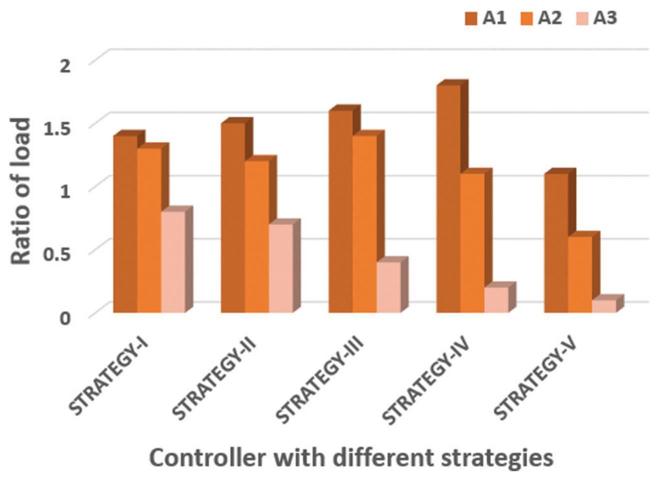


**Figure 7:** Balanced ratio of different strategies

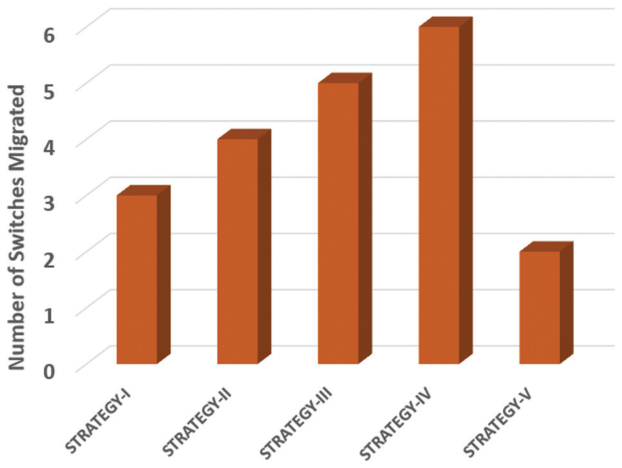


**Figure 8:** Number of switches migrated

The latency threshold selection will help us to identify and work with overloaded controllers which also helps the proposed approach to outperform the other load balancing strategies. In the ongoing messages requested by the switches, our strategy migrated the switches on time to the least loaded controllers, A2 and A3. In A2 and A3 load, the balancing phenomenon begins after every 20 s. Therefore, we have diminished the load of the controller (A1) in Fig. 7. In Fig. 8 the five load balancing strategies were compared and our strategy-V proved to have the lesser number of switches migrated after balancing the load.

## 6 Conclusion

Throughout this study we examined the issues of overloaded multiple controllers. Firstly, we presented some recent studies on latency features *vs.* load controller. The required latency threshold has to compromise between the load balancing effect and the cost of migration. Finding an efficient migration strategy for load balancing in SDN controllers is the prime purpose. We proposed highly scalable Load Balancing Algorithm based on migrating switches and thus achieved the optimized results. Our strategy facilitated better load balancing for controllers and migrated the load of overloaded switches to the other controllers in a short amount of time. Therefore, the proposed algorithm achieves the load balancing of multiple controllers in SDN efficiently and fastly. In the future, we plan to study the migration cost and distance between controllers and switches in the multiple overloaded controllers.

**Acknowledgement:** The authors are thankful for the support of Taif University Researchers Supporting Project No. (TURSP-2020/10), Taif University, Taif, Saudi Arabia.

**Funding Statement:** Taif University Researchers Supporting Project No. (TURSP-2020/10), Taif University, Taif, Saudi Arabia.

**Conflicts of Interest:** The authors declare that they have no conflicts of interest to report regarding the present study.

## References

[1] K. Benzekki, A. El Fergougui and A. Elbelrhiti Elalaoui, "Software-defined networking (SDN): A survey," *Security and Communication Networks*, vol. 9, no. 18, pp. 5803–5833, 2016.

[2] Y. Fan and N. Zhang, "A survey on software-defined vehicular networks," *Journal of Computers (Taiwan)*, vol. 28, no. 4, pp. 236–244, 2017.

[3] M. Chahal, S. Harit, K. K. Mishra, A. K. Sangaiah and Z. Zheng, "A survey on software-defined networking in vehicular adhoc networks: Challenges, applications and use cases," *Sustainable Cities and Society*, vol. 35, pp. 830–840, 2017.

[4] J. Zhang, H. Guo, J. Liu and Y. Zhang, "Task offloading in vehicular edge computing networks: A load-balancing solution," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, pp. 2092–2104, 2020.

[5] D. S. Rana, S. A. Dhondiyal and S. K. Chamoli, "Software-defined networking (SDN) challenges, issues and solution," *International Journal of Computer Sciences and Engineering Open Access*, vol. 7, no. 1, pp. 884–889, 2019.

[6] M. R. Belgaum, S. Musa, M. M. Alam and M. M. Su'ud, "A systematic review of load balancing techniques in software-defined networking," *IEEE Access*, vol. 8, pp. 612–636, 2020.

[7] G. Li, X. Wang and Z. Zhang, "SDN-based load balancing scheme for multi-controller deployment," *IEEE Access*, vol. 7, pp. 39612–39622, 2019.

[8] T. Hu, Z. Guo, P. Yi, T. Baker and J. Lan, "Multi-controller based software-defined networking: A survey," *IEEE Access*, vol. 6, pp. 15980–15996, 2018.

[9] G. Cheng, H. Chen, Z. Wang and S. Chen, "DHA: Distributed decisions on the switch migration toward a scalable SDN control plane," in *2015 IFIP Networking Conf. (IFIP Networking)*, Toulouse, pp. 1–9, 2015.

[10] Y. Zhou, K. Zheng, W. Ni and R. P. Liu, "Elastic switch migration for control plane load balancing in SDN," *IEEE Access*, vol. 6, pp. 3919–3909, 2018.

[11] X. He, Z. Ren, C. Shi and J. Fang, "Cloud/fog networking on the internet of vehicles," *China Communications*, vol. 13, pp. 140–149, 2018.

[12] Y. Zhou, Y. Wang, J. Yu, J. Ba and S. Zhang, "Load balancing for multiple controllers in SDN based on switches group," pp. 227–230, 2017.

[13] R. Trestian, K. Katrinis and G. M. Muntean, "OFLoad: An openflow-based dynamic load balancing strategy for data center networks," *IEEE Transactions on Network and Service Management*, vol. 14, no. 4, pp. 792–803, 2017.

[14] T. Hu, P. Yi, J. Zhang and J. Lan, "Reliable and load balance-aware multi-controller deployment in SDN," *China Communications*, vol. 15, no. 11, pp. 184–198, 2018.

[15] L. Nkenyereye, S. M. Riazu Islam, Y. H. Choi, M. Bilal and J. W. Jang, "Software-defined network-based vehicular networks: A position paper on their modeling and implementation," *Sensors (Switzerland)*, vol. 19, no. 17, pp. 1–14, 2019.

[16] H. Sufiev and Y. Haddad, "A dynamic load balancing architecture for SDN," in *2016 IEEE Int. Conf. on the Science of Electrical Engineering*, Eilat, pp. 1–3, 2016.

[17] S. Askar, "Adaptive load balancing scheme for data center networks using software defined network," *Science Journal of the University of Zakho*, vol. 4, no. 2, pp. 275–286, 2016.

[18] L. Ochoa-Aday, C. Cervello-Pastor, I. Member and L. Ochoa-Aday, "Discovering the network topology: An efficient approach for SDN," *Advances in Distributed Computing and Artificial Intelligence Journal*, vol. 5, pp. 1–8, 2016.

[19] F. Al-Tam and N. Correia, "On load balancing via switch migration in software-defined networking," *IEEE Access*, vol. 7, pp. 95998–96010, 2019.

[20] D. Tennakoon, S. Karunarathna and B. Udugama, "Q-learning approach for load-balancing in software defined networks," in *2018 Moratuwa Engineering Research Conf. (MERCon)*, Moratuwa, pp. 1–6, 2018.

[21] T. Hu, J. Lan, J. Zhang and W. Zhao, "EASM: Efficiency-aware switch migration for balancing controller loads in software-defined networking," *Peer-to-Peer Networking and Applications*, vol. 12, no. 2, pp. 452–464, 2019.

[22] C. Wang, B. Hu, S. Chen, D. Li and B. Liu, "A switch migration-based decision-making scheme for balancing load in SDN," *IEEE Access*, vol. 5, pp. 4537–4544, 2017.

[23] R. Chaudhary and N. Kumar, "LOADS: Load optimization and anomaly detection scheme for software-defined networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 12, pp. 12329–12344, 2019.

[24] A. Pietrabissa, L. Ricciardi Celsi, F. Cimorelli, V. Suraci, F. Delli Priscoli *et al.,* "Lyapunov based design of a distributed wardrop load-balancing algorithm with application to software-defined networking," *IEEE Transactions on Control Systems Technology*, vol. 27, no. 5, pp. 1924–1936, 2019.

[25] A. A. Abdelaziz, E. Ahmed, A. T. Fong, A. Gani and M. Imran, "SDN based load balancing service for cloud servers," *IEEE Communications Magazine*, vol. 56, no. 8, pp. 106–111, 2018.

[26] H. Babbar and S. Rani, "Emerging prospects and trends in software-defined networking," *Journal of Computational and Theoretical Nanoscience*, vol. 16, pp. 4236–4241, 2019.

[27] H. Babbar and S. Rani, "Software-defined networking framework securing Internet of Things," in *Integration of WSN and IoT for Smart Cities, EAI/Springer Innovations in Communication and Computing*, Cham, Springer, pp. 1–14, 2020.