Article

Load Balancing Algorithm on the Immense Scale of Internet of Things in SDN for Smart Cities

Himanshi Babbar [1], Shalli Rani [1,*], Divya Gupta [1], Hani Moaiteq Aljahdali [2] and Aman Singh [3,*] and Fadi Al-Turjman [4,*]

[1] Chitkara University Institute of Engineering and Technology, Chitkara University, Punjab 140401, India; himanshi.babbar@chitkara.edu.in (H.B.); divya.gupta@chitkara.edu.in (D.G.)
[2] Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 37848, Saudi Arabia; Hmaljahdali@kau.edu.sa
[3] Department of Computer Science and Engineering, Lovely Professional University, Punjab 144411, India
[4] Research Center for AI and IoT, Artificial Intelligence Engineering Department, Near East University, Mersin 10, Turkey
* Correspondence: shalli.rani@chitkara.edu.in (S.R.); amansingh.x@gmail.com (A.S.); fadi.alturjman@neu.edu.tr (F.A.-T.)

Abstract: Since the worldwide Internet of Things (IoT) in smart cities is becoming increasingly popular among consumers and the business community, network traffic management is a crucial issue for optimizing the IoT 's performance in smart cities. Multiple controllers on a immense scale implement in Software Defined Networks (SDN) in integration with Internet of Things (IoT) as an emerging paradigm enhances the scalability, security, privacy, and flexibility of the centralized control plane for smart city applications. The distributed multiple controller implementation model in SDN-IoT cannot conform to the dramatic developments in network traffic which results in a load disparity between controllers, leading to higher packet drop rate, high response time, and other problems with network performance deterioration. This paper lays the foundation on the multiple distributed controller load balancing (MDCLB) algorithm on an immense-scale SDN-IoT for smart cities. A smart city is a residential street that uses information and communication technology (ICT) and the Internet of Things (IoT) to improve its citizens' quality of living.Researchers then propose the algorithm on the unbalancing of the load using the multiple controllers based on the parameter CPU Utilization in centralized control plane. The experimental results analysis is performed on the emulator named as mininet and validated the results in ryu controller over dynamic load balancing based on Nash bargaining, efficient switch migration load balancing algorithm, efficiency aware load balancing algorithm, and proposed algorithm (MDCLB) algorithm are executed and analyzed based on the parameter CPU Utilization which ensures that the Utilization of CPU with load balancing is 20% better than the Utilization of CPU without load balancing.

Keywords: Internet of Things (IoT); Software-Defined Networking (SDN); MDCLB algorithm; mininet; ryu controller

check for updates

Citation: Babbar, H.; Rani, S.; Gupta, D.; Aljahdali, H.M.; Singh, A.; Al-Turjman, F. Load Balancing Algorithm on the Immense Scale of Internet of Things in SDN for Smart Cities. *Sustainability* **2021**, *13*, 9587. https://doi.org/10.3390/su13179587

Academic Editors: Zubair Baig and Andrea Colantoni

Received: 28 July 2021
Accepted: 18 August 2021
Published: 26 August 2021

1. Introduction

With the rising development of smart cities, the needs for SDN resource distribution mechanisms are rapidly increasing. Many areas in smart cities, such as automobiles, aviation, smart buildings, and factory equipment, rely on SDN. Different networking system environments have been designed and implemented in smart cities over the last twenty years in order to realize so-called smart cities. Dedicated IoT devices, as well as ubiquitous but non-dedicated devices like smart phones and car sensors, are used in such networks. The IoT is an innovative concept that facilitates diverse networks to be leveraged by adaptive ecosystems. In [1] specific, RFID-equipped devices, actuators, wireless sensors, and wireless networking devices are hooked up to the web to build an IoT framework.

As per the rise of the web in the Economic Times report, in India in 2019 the number of connected devices was increased to 700 million. IoT is an extension of access to the Internet in daily objects and virtual devices. IoT is the description of a system/network in such a way that "things" including devices have sensors that are integrated by public or private networks. According to Cisco's recent study in [2], 7 billion devices are linked across the web; however, by 2022 this number will increase to 70 billion. With the massive intervention of integrated devices, carriers are now facing the complicated regulation of the elements and the congested systems. If the systems are not equipped, this surge of Internet-of-Things, in which objects are traffic creators, not just users in the system, can indeed paralyze the system. Therefore, these integrated devices generate a massive amount of data that includes the data that is fabricated in the present year. Instead of the unique capabilities and features, the different challenges in [3] are established that are scalability, security, privacy, and flexibility. Therefore, managing the networks in the traditional technology like IoT and cloud is not feasible; therefore, the need of Software-Defined Networking (SDN) came into existence for the novel approach used for efficient management of the resources of the network. Software-Defined Networking (SDN) is a novel and evolving concept that offers growing possibilities and opportunities for effective and scalable network design [4]. To satisfy the fulfillment of Quality of Service (QoS) and with the limited network availability, one of the keynotes that have been taken into consideration is the load balancing issue. It is quite a tedious job to handle the massive amount of load for a single server. Therefore, we can use many servers with the load balancers that behave as the front end, although load balancing is an important aspect in SDN. The essential principle underlying SDN is that forwarding plane and control plane are isolated. Within dedicated hardware, both the control plane and the forwarding plane are coupled in conventional networking. Networks are nowadays more programmable and thus the applications like load balancers do not rely on the dedicated hardware that can be executed with the association of controller, and OpenFlow switches in the network can act as a load balancer; for this we do not require dedicated hardware [5].

1.1. Motivation

Load balancing in SDN-IoT is the distribution of load amongst the different servers in order to refine the resources, degrade the response time, maximize the throughput, and enhance Quality of Service (QoS) [6]. In conventional load balancing networks, it facilitates the support of hardware which is very expensive and suffers from a lack of flexibility; with this it becomes a single point of failure easily. Availability of service is predominant in aligning the satisfaction of the end-users, which has a higher level of significance on the achievement of balancing the load amongst the process clusters [7]. The connection of most of the people with the internet affects the web traffic which causes the congestion of the network and the loss of packets. With this technique, load balancing refines the efficiency of the network. Load balancing can be either fixed or variable. In fixed load balancing, the conditions are predetermined, and the criteria are now to be followed to accomplish the conditions. However, this form of load balancing is not as effective as the users' conditions cannot be constant in an environment in real-time [8]. Then the requirement for handling variable load balancing arises. In variable load balancing, the conditions are not predetermined and based on the modified conditions the load is segregated. There are multiple issues associated with reliability, scalability, and flexibility in a single controller for the large scale SDN-IoT. Therefore, to overcome some of the issues mentioned above, the researchers have chosen multiple distributed controllers for load balancing on a immense-scale SDN-IoT using the Ryu Controller [9]. Some items are required to identify the load balancing in SDN-IoT:

1. In an efficient distributed load balancing approach in SDN-IoT, how the load can be balanced among the heterogeneous devices and how to adapt our distributed architecture to various security frameworks.

2. An efficient switch migration load balancing in SDN-IoT; the problem is how the framework can be executed in IoT for the large-scale environment and how switch migration can be done on the traffic demands.
3. As per the survey, the authors found the problem of implementation of Switch Migration Decision Making (SMDM) on an immense scale IoT with the massive amount of traffic needed to evaluate the performance.

To interpret the issues for the immense scale SDN-IoT, researchers have thoroughly undergone the issue of balancing the load of the forwarding plane in the immense scale SDN-IoT. In this paper, researchers have considered the basic flow of multiple distributed controllers for load balancing on an immense scale SDN-IoT, and the motive of the proposed algorithm is to interpret the issues of balancing the load in a distributed manner.

1.2. Problem Definition

The drawback of the existing algorithms is that they did not consider the large scale networks in multiple controllers which lead to the constraint of fault tolerance and reliability. It requires load balancing on all the servers rather then creation of one heavily loaded server which can be unresponsive during the huge traffic. However, some researchers have worked upon the combination of alternatives for its solution (TOPSIS) but in load balancing rather than assigning weights to the different alternatives, things are required to be handled logically for real-time applications. Consequently, in this article, intra and inter cluster showing the migration of load from heavily loaded server to the least loaded server is simulated and tested to make the network stable and reliable. The proposed scheme focuses on this approach only and functions on the basis of threshold value of the load for one server. If the load on any server in intra cluster is greater than threshold value then the load is migrated from the intra cluster to inter cluster which is selected with the help of the proposed algorithm. The proposed algorithm also considers the efficient use of resources and hence has shown the optimal results on the existing approaches in terms of CPU utilization.

1.3. Contributions

The main contributions of this paper are as follows:

- Facilitates the SDN-IoT architecture and survey of the load balancing techniques in SDN.
- Proposes a Multiple Distributed Controller Load Balancing (MDCLB) algorithm on an immense scale SDN-IoT. In this, variable load balancing for the controllers in the control plane is to minimize the network delay and restrict unbalancing the traffic load.
- For this, we have the threshold value to compare with the load on the servers. If the threshold value is more than the load then the particular switch on the server and the migration of packets from intra custer to inter cluster will be performed to balance the load.
- To solve the problem of scalability and reliability, the issue of balancing the load in the control plane on an immense scale shows the uniformity of the information between the controllers.
- Tests the proposed algorithm in mininet emulator with python language using Ryu controller.
- Validates the proposed algorithm based on the QoS parameters, including CPU Utilization.

The remaining paper is structured as follows: Section 2 studies the background of the papers, Section 3 focuses on the methodology of the proposed multiple distributed controllers algorithm and flowchart, Section 4 evaluates the implementation and performance of the proposed algorithm, and Section 5 concludes the study.

2. Background

2.1. Integration of SDN and IoT

The integration of SDN and IoT as shown in Figure 1 is benefited in various ways: firstly, performing the integration of SDN and IoT is a fundamental problem that can be solved when one can have intelligent routing decision making that can be deployed using SDN [10] and secondly, simplification of information collection, analysis, and decision making [11]. Thirdly, the visibility of network resources and management of the network is simplified based on user, device, and application-specific requirements; therefore, the visibility for network resources and simplification concerning these aspects can be done with the indication of SDN in IoT. Lastly, intelligent traffic pattern analysis and coordinated decisions are done with the help of SDN IoT [12]. In SDN technology, more intelligence is proved in the network and improves the efficiency of the network.

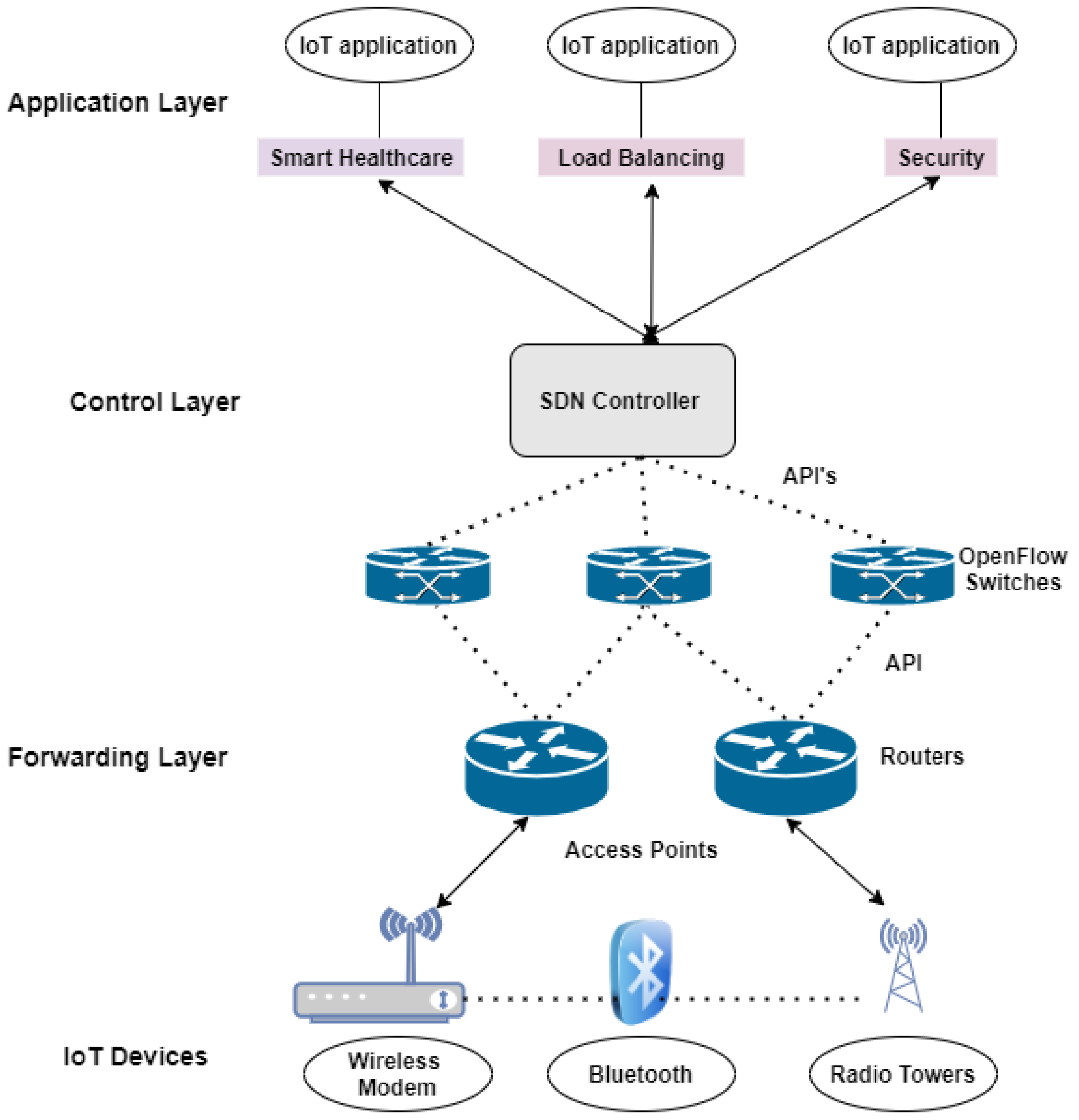

Figure 1. SDN Integration with IoT.

Therefore, this tends to increase the burden and diminishes system extensibility. With the emergence of SDN in telecommunications, the entire idea of network management has indeed been modified, thereby attempting to make it the inter-operable platform that enables the system administrators to modify the flow of traffic from one switch to the next with few lines of code [13,14]. The necessity for this system is to enable the connection of tens of millions of devices and fast content providers for traffic routing. The three-layered architecture of SDN-IoT is defined as the layer is known as the forwarding plane which consists of many network elements, which has Datapaths of SDN that have explored the capabilities of Control Data-Plane Interface (CDPI) Agent. The control layer, which is also coined as SDN Controller, converts the requirements and explores the level control over the data paths when giving the information to the applications of SDN. The application layer is the topmost layer that interacts for their requirements with Northbound Interface (NBI) Drivers that include load balancers, firewalls, IDS, etc.

2.2. Related Research

This section reviews recent research achievements of the published work in a study of the past five years from 2016 until mid-2020.

Wang et al. [15] proposed the switch migration decision making approach to focus on the load imbalance by a switch migration trigger factor which will utilize the efficiency while migrating the switches. To illustrate the feasibility of our idea, the authors introduce a validation of the concept and provide a quantitative analysis via a Mininet emulator. In the suggested reference Hai and Kim [16], methodology ensures load balancing to be done in the control plane by effective utilization of resources, improved system stability, and durability. Besides, communication latency is greatly decreased compared to existing methods by the use of a predetermined load threshold. The authors in this paper have eliminated the single point of failure problem in order to provide the reliable network and ensure the balancing of load in the control plane for each controller. Simulation results show that the methodology being proposed is successful in terms of overhead communication and load performance by using the threshold value. The authors of Babbar et al. [17] proposed a load balancing algorithms based on the multiple controllers. In this, by selecting the latency and multiple overheads, this proposed algorithm has improved the latency by 25%. The study of the proposed technique in the Madzharov and Nemkov [18] verified its quality and productivity in terms of acquiring an optimized solution for achieving a balanced load of connectivity and enforcing the network management security schemes needed for rapid re-routing in SDN. The dynamic mapping is done in Cello et al. [19], who framed the algorithmic approach intended to resolve and lessen load imbalances between SDN controllers by effective SDN switch migration. Modeling proves that BalCon is computationally compact and lessens the load disparity between SDN controllers (expressed as variance) by 35% by switching only minimal switching devices.

Memon et al. [20] presented a framework designed to detect a spoofing attack that investigates the probability distributions of obtained power established for the regions configured for mobile (moving) users. Consequently, we investigate the effect on the confidentiality scope of targeted customers in the absence and presence of observer. Authors of Deng and Wang [21] suggest Application-aware QoS routing algorithm (AQRA) SDN dependent IoT networking to ensure various QoS specifications for high priority IoT applications and to conform optimal navigation paths to existing network status. Cui et al. [22] proposed the SMCLBRT approach of various SDN controllers based on the response time which takes into consideration the modified characteristics of response time in the real environment in contrast to the load of the controller. Reference [23] adopts EASM to handle loads of controllers and enhance the capacity of the migration. For evaluating load balancing on controllers, authors implement the load difference matrix and trigger factor. Authors also implement the migration efficiency problem, which effectively addresses the load balancing rate and the migration cost to efficiently migrate switches. The authors of Eghbali et al. [24] suggested the distributed approach for the load balance between SDN and IoT based devices. The designed model enables the application of distinct distributed management approaches. Experimental findings demonstrate that the proposed method gives away tasks equally between devices and shows an average reduction turnaround time and average waiting time and increased performance processing. In this paper Li et al. [25] developed the Nash game bargaining method to reasonably enhance the two conflicting objectives of migration costs and load balance. An enhanced firefly technique is used to overcome the problem, and the optimum network configuration status is achieved. The test results suggest that this technique should maximize the cost of migration as well as the load balance simultaneously. Babu et al. [26] illustrates the need for Cloud and Internet of Things connectivity and agent-oriented and Cloud-assisted Cloud IoT model built on the multilayered design model. A Cloud-based IoT model applications paradigm is presented, as well as a reference architecture for agent-oriented vision and Cloud-assisted vision. Dashtipour et al. [27] proposed the multimodel persian dataset for more than 800 utterances, as a benchmark resource and the other novel is context aware proposed for the sentimental analysis. In comparison to unimodal information, experimental results show that contextual integration of multimodal features such as textual, audio, and visual features delivers greater performance (91.39 percent) (89.24 percent) respectively. Sahoo et al. [28] paper represents ESMLB mechanism which aims to efficiently allocate switches to an underutilized controller. Amongst many alternatives, a multi-criteria decision-making approach, i.e., for choosing a goal controller. In this mechanism, Technique for Order by Similarity to an Ideal Solution (TOPSIS) is used.

Our proposed approach focuses on resolving the issue of balancing the load based on the multiple controllers that interact with the switch of forwarding layer which thereby transfers the packet_in packets to the switch and controller responds to the message with the packet_out packets and acquires the switch information through the packet_in packets. In this proposed approach the main focus is on the migration of load from the heavily loaded server to the lightly loaded one by checking the threshold value. If the threshold value is more than the load assigned to the server then the cluster's load is balanced, or else we will migrate from intra cluster to inter cluster between the five SDN controllers.

3. Methodology

In this section, researchers have proposed the MDCLB algorithm for making the load balance among the various controllers [29]. For this, researchers have proposed the mathematical model for optimizing the multiple controllers in the forwarding layer to achieve the maximum CPU utilization amongst the various controllers.

3.1. System Model

The messages which are requested are transmitted by the switch on the control layer inside the multiple distributed controllers for load balancing because controllers are vast, which puts a huge workload on the following controller which results in the packet delay [23]. As a result, controller is failed and network collapse in the worst case scenario. The researcher demonstrated multiple distributed controllers load balancing strategy focused on a consistent delay for controllers to configure the identified issues in the control layer [24]. When a controller crashes or the one controller is overburdened then the algorithm is being used to encourage the overloaded or defective switches controlled by the controller. Thereby, optimizing load balancing of the control layer becomes effective and efficient. The network topology for the control plane shows the interaction between the switches and controllers. The SDN network in Figure 2 is comprised of 16 switches and 4 controllers. Switches and Controllers are represented as "P" and "S" respectively.

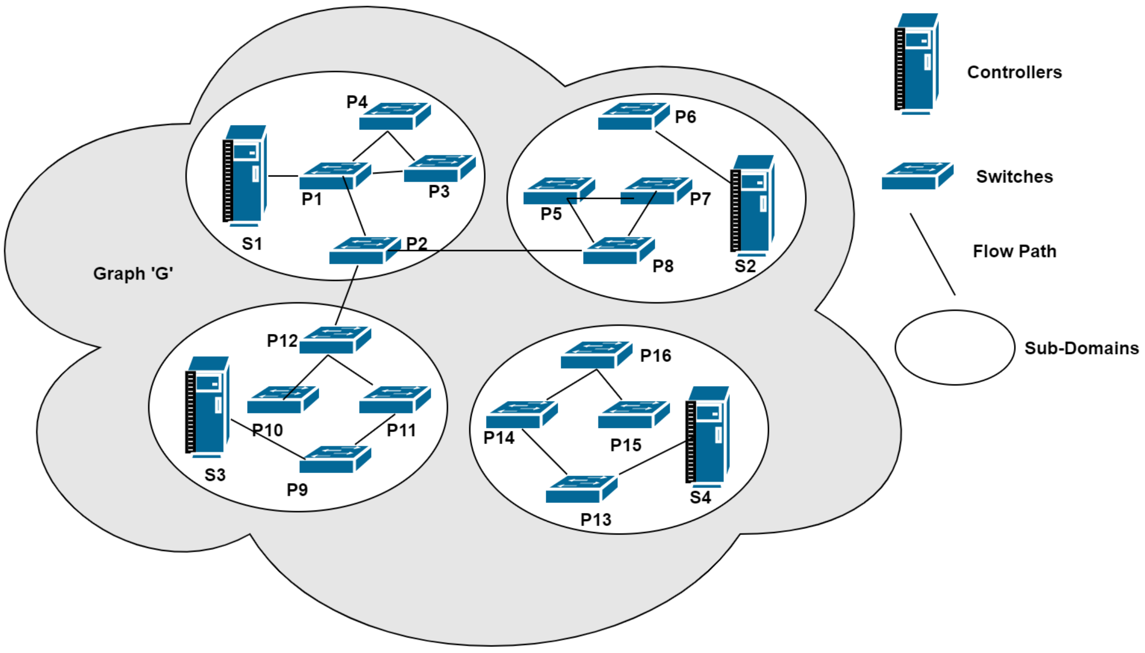

Figure 2. SDN Architecture.

In the control layer architecture explained in the above Figure 2, every switch is interconnected to one controller and one or more than one switches are connected to the controller. Let us say the quantity of SDN-IoT controllers is m; therefore, $S = S_a$ $a = 1, 2, 3, \ldots, m$ S is the collection of controllers, S_a represents the a^{th} controller. Let us say that the total of SDN-IoT switches in the forwarding layer is n; therefore, $P = P_b$ b = 1, 2, 3......, n, where P is the collection of switches and P_b represents the b^{th} switch. Representation of switches taken as "n" and controllers as "m" can create a connection of control plane; therefore, they are denoted by a X_{mn} matrix derived from [28] which is comprised of rows as m and columns as n, as explained in Equation (1):

$$X_{mn} = \begin{bmatrix} x_{11} & x_{12} & x_{13} & \ldots & x_{1n} \\ x_{21} & x_{22} & x_{23} & \ldots & x_{2n} \\ x_{31} & x_{32} & x_{33} & \ldots & x_{3n} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ x_{m1} & x_{m2} & x_{m3} & \ldots & x_{mn} \end{bmatrix} \quad (1)$$

where, X_{mn}

$$\begin{Bmatrix} 0 & ; & b^{th} \textit{ switch } P_b \textit{ is not controlled by the } a^{th} \textit{ controller } S_a. \\ 1 & ; & b^{th} \textit{ switch } P_b \textit{ is controlled by the } a^{th} \textit{ controller } S_a. \end{Bmatrix}$$

Let us assume that the packet_in packets speed are passed by the b^{th} switch P_b to the a^{th} controller S_a at time t1 is u_{ab}; therefore, Equation (2) is mentioned below:

$$u_{ab} = x_{ab} g_{ab}(t1) \quad (2)$$

where $g_{ab}(t1)$ represents the speed where b^{th} switch P_b passes the packet. In the course of interim $[1, T1]$ in Equation (3), number of packets passed by b^{th} switch P_b to a^{th} controller S_a is u_{ab},

$$\begin{aligned} U_{ab} &= \int_1^{T1} u_{ab} wt1 \\ &= \int_1^{T1} \frac{X_{ab} g_{ab}(t1) wt1}{totalnumberofpackets} \end{aligned} \tag{3}$$

Therefore, the totality of the speed of the incoming packets are processed by the a^{th} controller; U_{ab} from n switches $P_1, P_2, P_3 \ldots P_n$ at time $t1$ in Equation (4) is:

$$U_a = \sum_{b=1}^{n} (X_{ab} g_{ab}(t1)) \tag{4}$$

In the course of interval $[1, T1]$, the number of packet_in packets are refined by the a^{th} controller $U_a b$ from n switches that is $P_1, P_2, P_3, \ldots\ldots\ldots P_n$ is described in Equation (5) as:

$$U_a = \int_1^{T1} \sum_{b=1}^{n} (X_{ab} g_{ab}(t1) wt1) \tag{5}$$

Therefore, the number of packet_in packets is refined in Equation (6) as: by m controllers in the control layer from n switches in the forwarding layer is:

$$\begin{aligned} U &= \sum_{a=1}^{m} U_a \\ &= \sum_{a=1}^{m} (\int_1^{T1} \sum_{b=1}^{n} (x_{ab} g_{ab}(t1)) wt1) \\ &= \int_1^{T1} \sum_{a=1}^{m} (\sum_{b=1}^{n} (x_{ab} g_{ab}(t1))) wt1 \end{aligned} \tag{6}$$

In the multiple distributed controllers in the immense scale SDN-IoT referred from [17], on acquiring the new flow from the switch, the switches transfer the initial flow packet to the designated controller. Evaluation and decision of the path to transfer the flow are taken by the controller. Forwarding rules are installed on all the flow paths of the switches. The controller transmits the new flow request information of the flow path and evaluates them when the massive amount of flows are put in the IoT and then the controller immediately acquires. The peak forwarding rules installed on the flow path of the switches create the hinder/delay in the network, which results in the imbalance of load of the IoT.

The delay in the network consists of processing, queuing, transmission, and propagation delay. In particular, controllers and switches can be assigned as constant in the processing and propagation delay. Thus, the delay in the network is dependent on queuing delay. From the queuing model Q|Q|1, researchers have concluded that queuing delay $T1_{b,y}$ of the packet_in packets are transferred by the b^{th} switch P_b in the a^{th} controller S_a which can be represented in Equation (7) as:

$$T1_{b,y} = \frac{\beta_a}{\gamma_a(\gamma_a - \beta_a)} \tag{7}$$

where $\beta_a = U_a / T1$ is the entry speed of the packets, thus, the average value of incoming packets perforate at the a^{th} controller S_a in unit time, $\beta_a = U_a / T1_w$ indicates the average rate of the a^{th} controller S_a refines the incoming packets.

Suppose that the communication delay of packets transferred by the b^{th} switch P_a of the forwarding layer to the a^{th} controller S_a of the control layer is $T1_{a,c}$, can be achieved by evaluating the maximal delay of the effectual minimal path between the switch P_b and controller S_a; therefore, in Equation (8),

$$T1_{a,c} = w(P_b, S_a) \quad where\ max(P_b) \in P; min(S_a) \in S \tag{8}$$

The result obtained for a complete time K_a of the b^{th} switch that acquires a new flow, the a^{th} controller evaluates the flow path and switches on the path that installs the forwarding rules are calculated in Equation (9) as:

$$K_a = T1_p + T1_w + T1_{b,y} + T1_{a,c} \tag{9}$$

$T1_p$ and $T1_w$ represent the processing and transmission delay respectively. Therefore, the totality of the delay between the control layer and forwarding layer is explained in equation :

$$\begin{aligned} K &= \sum_{a=1}^{m} K_a \\ &= \sum_{a=1}^{m} (T1_p + T1_w + T1_{b,y} + T1_{a,c} \\ &= \sum_{a=1}^{m} (T1_p + T1_w + \frac{\beta_a}{\gamma_a(\gamma_a - \beta_a)} + w(P_b, S_a) \\ & where\ max(P_b) \in P; min(S_a) \in S \end{aligned} \tag{10}$$

In SDN-IoT, workload from the controllers can be estimated as the number of packet_in packets requests. Therefore, the system model of the controllers can be calculated in Equation (11) as:

$$D = \{X_{mn}, K, U\} \tag{11}$$

where X_{mn}, K, U are represented by (1), (6), (9) respectively. If the delay in the network can be decreased it is easy to accomplish the balancing of the load. Thus, the system model in Equation (12) can be transformed into the least delay model, named as:

$$D = \{X_{mn}, K\} \tag{12}$$

Whenever the requests for the controller by the switch are very massive or the controller loses, the switch is matched to a specific controller for balancing the delay [30]. Congestion in the network is induced by the expanded load of the controllers that can be lessened by diminishing the queuing and transmission delay.

3.2. Proposed Multiple Distributed Controllers Algorithm

This technique is based on the balanced delay which represents the multiple distributed controllers load balancing algorithm. For a sustainable network, load on the servers needs to be balanced, as per the related research the server in the IoT network can handle maximum 256 user requests per second and 1 terabyte (TB) of transferring the data per day. For the light weight requests of the users the normal server can handle 2000–2500 requests per second. It depends upon the requirement of the data transmission and for the different level of the application and under different scenarios, the value of threshold will vary. Therefore, in a new algorithm either the threshold of user requests or threshold of transmitted data can be considered to balance and migrate the load on the servers. However, in the present article we have assumed threshold of 40 user requests for a smaller network by keeping in mind the data generated in the smart cities which can be further enlarged as per the requirement of the application. Let us state that the threshold value for the packet_in packets that are refined by the controller in intra cluster at time is U^{th} can be named as the immense load on the controllers. The quantity of the incoming packets that the a^{th} controller S_a refines from the n switches is higher than or equivalent to the immense load U^{th} of the a^{th} controller S_a, and the requested quantity of incoming packets in the intra cluster that the b^{th} switch S_b to the a^{th} controller S_a is U_a and $U_a > U^{th}/\text{n}$. The addition of the queuing and transmission delay is higher than the nodal processing delay and they are thereby used to again transmit the incoming packets to the unused controllers at the minimum distance and the related controller transforms from start to end, i.e., x = 1. The fragment of the packet_in packet in intra cluster that goes beyond the controller is passed to the another controller in the inter cluster that has the shortest distance. Or else, the packets pursue to hold for the controller to address unless the iteration is achieved. x = 0 says that the controller is fizzled and the switch through the G(P, S) signifies the idle slave with the minimum distance as the master, i.e., x = 1, and the mentioned steps are rehashed. Algorithm 1 is explained by considering the Figure 2 topology, which shows the migration of switches overloaded load from intra cluster to inter cluster and flowchart proposed in Figure 3 is clarified beneath.

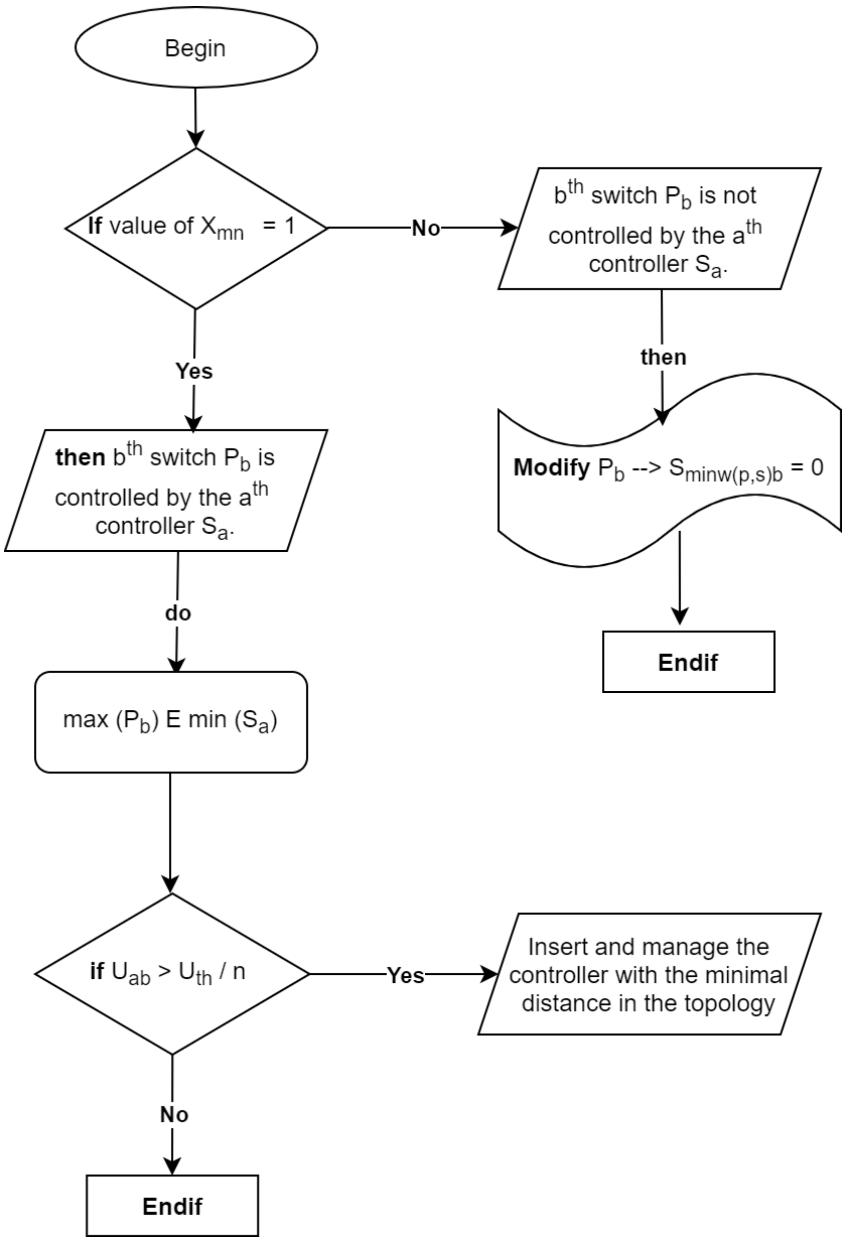
Begin
If value of X_{mn} = 1
No
b^{th} switch P_b is not controlled by the a^{th} controller S_a.
then
Modify P_b --> $S_{minw(p,s)b}$ = 0
Endif
Yes
then b^{th} switch P_b is controlled by the a^{th} controller S_a.
do
max (P_b) E min (S_a)
if U_{ab} > U_{th} / n
Yes
Insert and manage the controller with the minimal distance in the topology
No
Endif

Figure 3. Proposed Flowchart.

Algorithm 1 Load Balancing in SDN using multiple controllers (MDCLB).

```
Input: Initial value of $X_{mn}$ mentioned in equation-1
speed of the packets $U_{ab}$ mentioned in equation-2
Output: Updated the controllers load on CPU
Begin:
1    $b^{th}$ switch $P_b$ is controlled by the $a^{th}$ controller $S_a$ do
2        $max(P_b) \in \ min(S_a)$ do
3      if($S_a$ decreases) then
4            $X_{mn} = 0$, $b^{th}$ switch $P_b$ is not controlled by the $a^{th}$ controller $S_a$
5          modify $P_b \rightarrow S_{minw(p,s)}$
6            $X_{minw(p,s)_b} = 0$
7        endif
8            Packet speed if $U_ab >$ threshold_value $U^{th}/n$, addition of processing delay $T1_p$
9            and propagation delay $T1_{b,y}(U_{ab} - U^{th}/n) >$ addition of transmission delay
10               $T1_s(U_{ab} - U^{th}/n)$
11               if $U_{ab} > U^{th}/n$ and
12               $T1_p + T1_{b,y}\ (U_{ab} - U^{th}/n > T1_s(U_{ab} - U^{th}/n + T1_w$
13               Insert and manage the controller
14               with the minimal distance in the topology
15           $X_{minw(p,s)_b} = 1$
16           messages of incoming packets are higher than the threshold value
17           set by the controller is being processed to the newest controller.
18           endif
19     endfor
End
```

The above mentioned algorithm denotes that in this section, researchers have converted the balancing of load into network response time which circumvent by diminishing the queuing and transmission delay. Steps to generate the migrate action for each overloaded controller are as follows:

1. Starting value of X_{mn} given in Equation (1), the rate of packet U_{ab} given in Equation (2).
2. The value of X_{mn} is defined as:
   - if $X_{mn} = 0$; b^{th} switch P_b is not controlled by the a^{th} controller S_a.
   - if $X_{mn} = 1$; b^{th} switch P_b is controlled by the a^{th} controller S_a.
3. if $X_{mn} = 0$; modifying the b^{th} switch P_b at the shortest distance to the controller $S_{minw(p,s)}$.
4. if $S_{minw(p,s)} = 0$ then update the packet rate and addition of processing and propagation delay.
5. If both are greater then insert and manage the controller with the shortest distance in the topology for the switch.
6. if $S_{minw(p,s)} = 1$ then incoming message packets higher than the threshold value set by the controller is being processed to the newest controller.

4. Experimentation and Performance Evaluation

To assess the performance of our multiple distributed controllers, load balancing algorithm for the immense scale SDN-IoT, researchers organized the experimental platform. This proposed scheme is analyzed by network emulation by the topology of network using mininet [31] and ryu controller [32]. Ryu is chosen as the most appropriate controller amongst all the other controllers available which has the finer flexibility and reliability which is best suited for the immense scale SDN-IoT. The Ryu experimental and simulation environment comprised of mainly $8th$ generation $Intel^{@}\ Core^{TM}$ i7 Quad-Core processor, 16 GB RAM with 512 GB hard disk, Oracle Virtual Box 5.2, Processor having 4 CPU's, the system is bidirectional in Ubuntu 18.04 LTS. The interaction between the switches and controllers is performed by the use of the OpenFlow 1.3 protocol in which we have created the linear topology having the remote controller 127.0.0.1:6653.

Load Balancing is defined as the distribution of load amongst the different servers in order to refine the resources, degrade the response time, and maximize the throughput. Connection of most of the people with the internet affects the web traffic which causes the congestion of the network and the loss of packets. With this technique, load balancing refines the efficiency of the network. The dynamic load balancing based on Nash bargaining, efficient switch migration load balancing algorithm, efficiency aware load balancing algorithm, and proposed algorithm are executed and analyzed based on the parameter CPU Utilization. The iperf test tool is used to produce the load sent to the respective controllers. The number of controllers and time are displayed at x-axis and CPU Utilization by the controllers displayed at y-axis. The iperf test tool calculates the total number of controllers that took how much time to utilize the CPU without and with load balancing. The controllers load with load balancing is facilitating better results than controllers load without load balancing. The proposed algorithm has shown improvement in average-CPU Utilization with 60%, 62%, 61%, 64%, and 63% over dynamic load balancing based on Nash bargaining, efficient switch migration load balancing algorithm, and efficiency aware load balancing algorithm.

The performance can experiment with the proposed algorithm in the large scale SDN-IoT integration. Researchers explain the Algorithm 1 for the distributed controllers. The influence of CPU Utilizing for designing the controllers without balancing the load is represented in the Figure 4. Before balancing the load, the CPU utilization as per the time duration is assigned to each controller. With regard to 0, 5, 10, 15, 20 s the CPU utilization for controller 1 is 50%, 40%, 51%, 50%, 45% for DLBNB; 55%, 62%, 56%, 60%, 60% for ESMLB; 69%, 56%, 35%, 65%, 72% for EASM; 64%, 49%, 45%, 60%, 68% for proposed algorithm and so on for the different controllers.

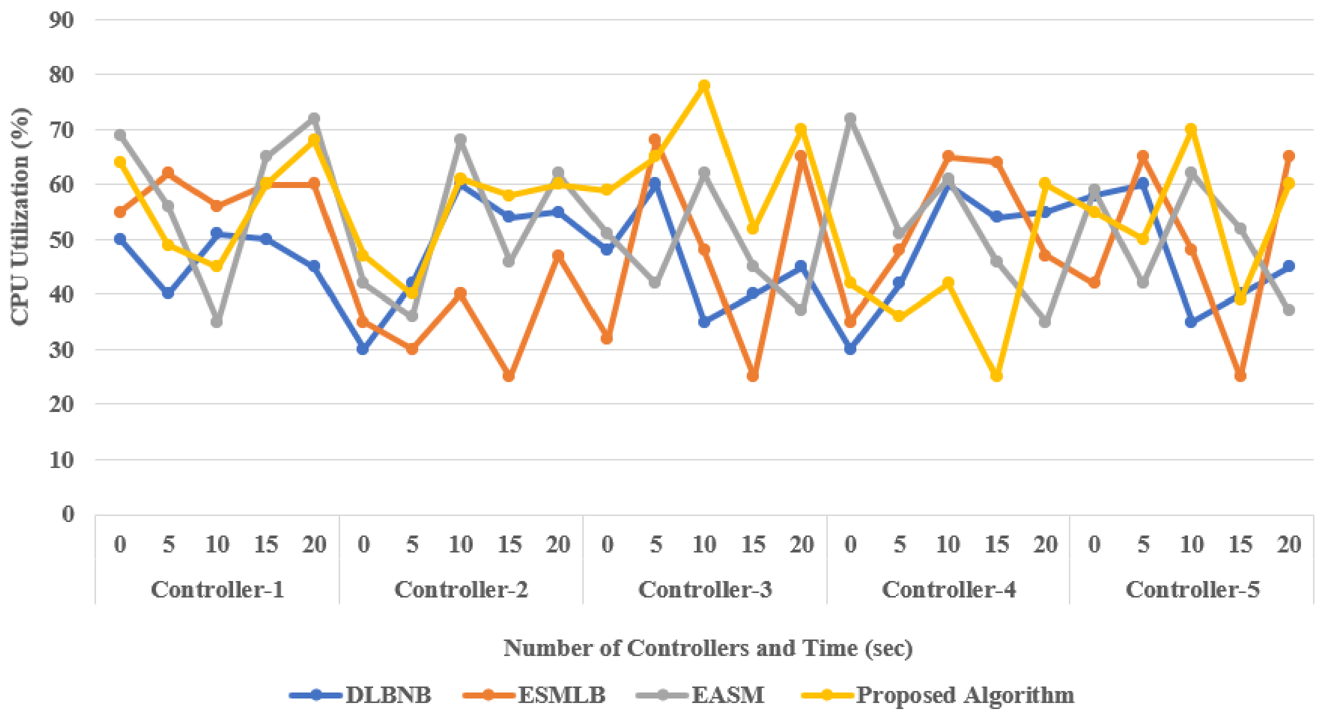

Figure 4. Controller load without load balancing.

After balancing the load among the different controllers, the CPU utilization at different time intervals is shown in Figure 5. The controllers are modifying drastically every time as the influence of configuring with the load balancing algorithm as shown in Figure 5. The below-represented figure shows the minimum change in the controllers' fluctuations. With regard to 0, 5, 10, 15, 20 s the CPU utilization for controller 1 is 66%, 66%, 64%, 66%, 64% for DLBNB; 67%, 65%, 63%, 65%, 62% for ESMLB; 64%, 68%, 66%, 64%, 68% for EASM; 69%, 70%, 68%, 68%, 69% for proposed algorithm and so on for the different controllers.

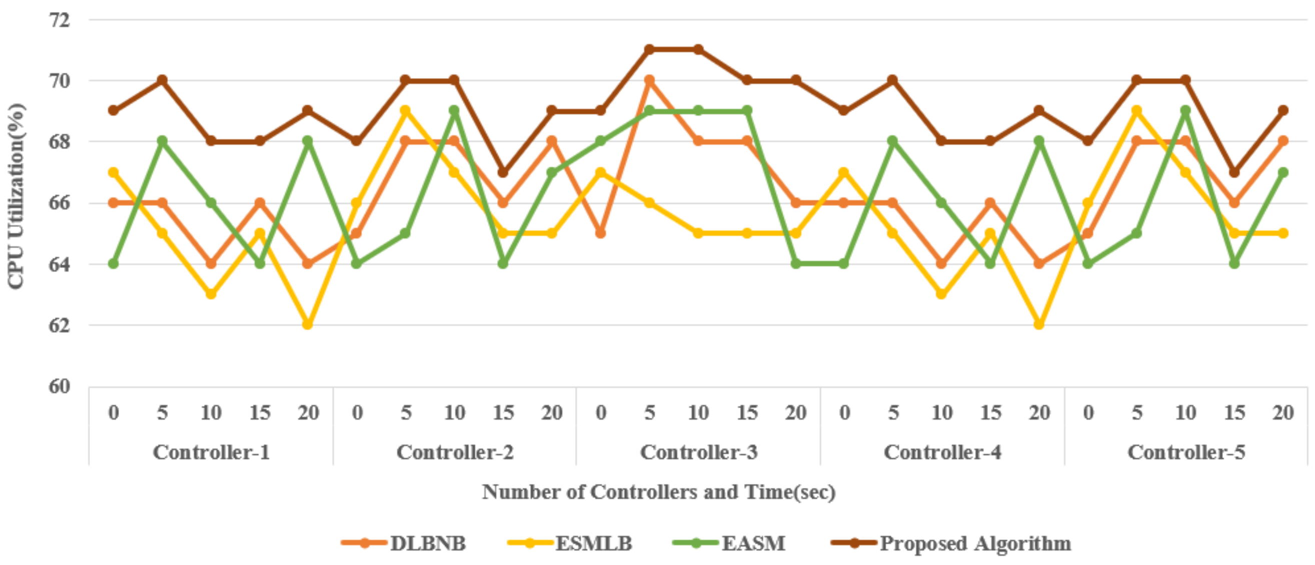

Figure 5. Controller load with load balancing.

The average CPU Utilization of the controllers in both the scenarios before and after load balancing as shown below: the average case of CPU Utilization of the controllers in both the scenarios are the same. In Figure 6 before balancing the load among the different controllers the average CPU utilization for controller 1 is 60%; Controller 2 is 40%, Controller 3 is 54%, Controller 4 is 45%, and for Controller 5 is 50%. Maximum CPU utilized is by Controller 1; therefore, the load is to be balanced for Controller 1 to show the migration.

In Figure 6 average CPU utilization after balancing the load for Controller 1 is 60%; Controller 2 is 62%; Controller 3 is 61%; Controller 4 is 64%, and for Controller 5 is 63%. Therefore, the average of the controllers without load balancing is 5% lesser than average with load balancing. The load is more balanced in Figure 6 as compared to Figure 6; this is dependent on the balancing delay of the proposed algorithm which chooses a controller and accomplishes the aim of balancing the load. Therefore, the aim of load balancing is fulfilled based on the balanced delay.

MDCLB algorithm has shown the improvement with load balancing on dynamic load balancing based on Nash bargaining, efficient switch migration load balancing algorithm, and efficiency aware load balancing algorithm. Figure 6 shows the utilization of memory by the algorithms implemented in this paper. MDCLB algorithm is again performing better over the comparative algorithms.

To accomplish the balancing of the load equally among all the controllers, MDCLB algorithm can be taken. Figure 5 displays the CPU utilization with load balancing of the dynamic load balancing based on Nash bargaining, efficient switch migration load balancing algorithm, and efficiency aware load balancing algorithm. CPU is considered to be the most essential factor for estimating the controllers' load on the controllers. Therefore, consideration of utilization of CPU is eminent to compute the load on controllers. CPU utilization is computed on already existing algorithms which ensures that the time is unbalanced. To balance the load on the controller, it is mandatory to make better use of the resources to avoid unbalancing the load on the controllers. Therefore, given comparison of Multiple Distributed Controllers, the load balancing algorithm has shown improvement for the utilization of resources.

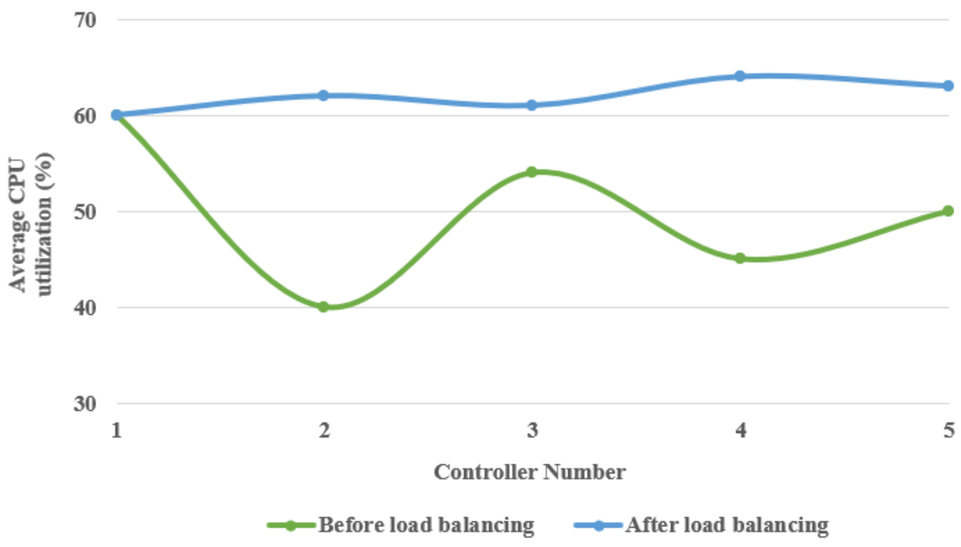

Figure 6. Controller load with average CPU utilization.

5. Conclusions

Software-defined networking in Internet of Things is defined as the latest and most enhancing technology is SDN and is to be considered as one of the most promising solutions to meet and cater to the demands. The objective of this paper is to resolve the issue of the unbalancing of the load on an immense scale SDN-IoT. A novel load balancing algorithm named an MDCLB Algorithm has been proposed to vanquish the load imbalance in the forwarding plane and control plane. In this paper, a Multiple Distributed Controller Load Balancing (MDCLB) algorithm is proposed on an immense scale SDN-IoT. In this, variable load balancing for the controllers in the control plane is to minimize the network delay and restrict unbalancing the traffic load. For this, we have the threshold value to compare with the load on the servers. If the threshold value is more than the load, the particular switch on the server then the migration of packets from intra custer to inter cluster will be performed to balance the load It is implemented in python using mininet emulator and iperf test tool. Comparison of the various algorithms for SDN-IoT based applications including dynamic load balancing based on Nash bargaining, efficient switch migration load balancing algorithm, and efficiency aware load balancing algorithm with proposed MDCLB algorithm for the parameter CPU utilization which has proved the validation of the proposed algorithm. Controllers' load with load balancing increased the CPU utilization compared to comparative algorithms for SDN-IoT based applications. The findings are preliminary and show that load balancing algorithms are successful in the forwarding plane. This research will facilitate and promote the road network and traffic congestion.

Author Contributions: Conceptualization, H.B., S.R., D.G. and H.M.A.; Methodology, H.B., S.R. and A.S.; Validation, A.S., F.A.-T. and H.M.A.; Formal Analysis, A.S. and H.M.A.; Investigation, H.M.A. and S.R.; Resources, H.M.A.; Data Curation, H.B. and S.R.; Writing—Original Draft, H.B., S.R. and H.M.A.; Writing—Review Editing, S.R. and A.S., Supervision, F.A.-T., D.G. and A.S.; Project Administration, H.M.A., D.G. and S.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: This project was supported by the Deanship of Scientific Research (DSR), King Abdulaziz University, Jeddah, under grant No. (KEP-11-611-42).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Khedkar, S.P.; Aroulcanessane, R. SDN enabled cloud, IoT and DCNs: A comprehensive Survey. In Proceedings of the 2019 5th International Conference on Computing, Communication Control and Automation, ICCUBEA 2019, Pune, India, 19–21 September 2019. [CrossRef]
2. Babbar, H.; Rani, S. Emerging Prospects and Trends in Software Defined Networking. *J. Comput. Theor. Nanosci.* **2019**, *16*, 4236–4241. [CrossRef]
3. Sood, K.; Yu, S.; Xiang, Y. Software-Defined Wireless Networking Opportunities and Challenges for Internet-of-Things: A Review. *IEEE Internet Things J.* **2016**, *3*, 453–463. [CrossRef]
4. Sondur, S. Software Defined Networking for Beginners. Technical Report. 2014. 3370.4640. Available online: https://www.researchgate.net/publication/335000866_Software_Defined_Networking_for_Beginners (accessed on 19 August 2021). [CrossRef]
5. Babbar, H.; Rani, S. Software-Defined Networking Framework Securing Internet of Things. In *Integration of WSN and IoT for Smart Cities*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 1–14.
6. Khan, S.; Ali, M.; Sher, N.; Asim, Y.; Naeem, W.; Kamran, M. Software-Defined Networks (SDNs) and Internet of Things (IoTs): A Qualitative Prediction for 2020. *Int. J. Adv. Comput. Sci. Appl.* **2016**, *7*, 385–404. [CrossRef]
7. Pourghebleh, B.; Hayyolalam, V. A comprehensive and systematic review of the load balancing mechanisms in the Internet of Things. *Clust. Comput.* **2020**, *23*, 641–661. [CrossRef]
8. Horvath, R.; Nedbal, D.; Stieninger, M. A Literature Review on Challenges and Effects of Software Defined Networking. *Procedia Comput. Sci.* **2015**, *64*, 552–561. [CrossRef]
9. Islam, M.T.; Islam, N.; Refat, M.A. Node to Node Performance Evaluation through RYU SDN Controller. *Wirel. Pers. Commun.* **2020**, *112*, 555–570. [CrossRef]
10. Tayyaba, S.K.; Shah, M.A.; Khan, O.A.; Ahmed, A.W. Software defined network (SDN) based Internet of Things (IoT): A road ahead. In Proceedings of the International Conference on Future Networks and Distributed Systems, Cambridge, UK, 19–20 July 2017; p. 15.
11. Kalkan, K.; Zeadally, S. Securing internet of things (iot) with software defined networking (sdn). *IEEE Commun. Mag.* **2017**, *56*, 186–192. [CrossRef]
12. Vandana, C. Security improvement in iot based on software defined networking (sdn). *Int. J. Sci. Eng. Technol. Res. (IJSETR)* **2016**, *5*, 2327–4662.
13. Mishra, S.; AlShehri, M.A.R. Software defined networking: Research issues, challenges and opportunities. *Indian J. Sci. Technol.* **2017**, *10*, 1–9. [CrossRef]
14. Jarraya, Y.; Madi, T.; Debbabi, M. A survey and a layered taxonomy of software-defined networking. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 1955–1980. [CrossRef]
15. Wang, C.; Hu, B.; Chen, S.; Li, D.; Liu, B. A Switch Migration-Based Decision-Making Scheme for Balancing Load in SDN. *IEEE Access* **2017**, *5*, 4537–4544. [CrossRef]
16. Hai, N.T.; Kim, D.S. Efficient load balancing for multi-controller in SDN-based mission-critical networks. In Proceedings of the 2016 IEEE 14th International Conference on Industrial Informatics (INDIN), Poitiers, France, 19–21 July 2016; pp. 420–425. [CrossRef]
17. Babbar, H.; Rani, S.; Masud, M.; Verma, S.; Anand, D.; Jhanjhi, N. Load balancing algorithm for migrating switches in software-defined vehicular networks. *Comput. Mater. Contin.* **2021**. [CrossRef]
18. Madzharov, N.D.; Nemkov, V.S. Smith predictor with sliding mode control for processes with large dead times. *J. Electr. Eng.* **2017**, *68*, 235–244. [CrossRef]
19. Cello, M.; Xu, Y.; Walid, A.; Wilfong, G.; Chao, H.J.; Marchese, M. BalCon: A distributed elastic SDN control via efficient switch migration. In Proceedings of the 2017 IEEE International Conference on Cloud Engineering, IC2E 2017, Vancouver, BC, Canada, 4–7 April 2017; pp. 40–50. [CrossRef]
20. Memon, I.; Shaikh, R.A.; Hasan, M.K.; Hassan, R.; Haq, A.U.; Zainol, K.A. Protect Mobile Travelers Information in Sensitive Region Based on Fuzzy Logic in IoT Technology. *Secur. Commun. Netw.* **2020**, *2020*, 8897098. [CrossRef]
21. Deng, G.C.; Wang, K. An Application-aware QoS Routing Algorithm for SDN-based IoT Networking. In Proceedings of the 2018 IEEE Symposium on Computers and Communications (ISCC), Natal, Brazil, 25–28 June 2018; pp. 186–191. [CrossRef]
22. Cui, J.; Lu, Q.; Zhong, H.; Tian, M.; Liu, L. A Load-Balancing Mechanism for Distributed SDN Control Plane Using Response Time. *IEEE Trans. Netw. Serv. Manag.* **2018**, *15*, 1197–1206. [CrossRef]
23. Hu, T.; Lan, J.; Zhang, J.; Zhao, W. EASM: Efficiency-aware switch migration for balancing controller loads in software-defined networking. *Peer-to-Peer Netw. Appl.* **2019**, *12*, 452–464. [CrossRef]
24. Eghbali, Z.; Lighvan, M.Z.; Beheshti, A. An Efficient Distributed Approach for Load Balancing in IoT Based on SDN Principles. In Proceedings of the 2019 10th International Conference on Computing, Communication and Networking Technologies, ICCCNT 2019, Kanpur, India, 6–8 July 2019; pp. 1–6. [CrossRef]
25. Li, G.; Li, K.; Liu, Y.; Pan, Y. An efficient dynamic load balancing scheme based on nash bargaining in SDN. *Future Internet* **2019**, *11*. [CrossRef]

26. Babu, S.M.; Lakshmi, A.J.; Rao, B.T. A study on cloud based Internet of Things: CloudIoT. In Proceedings of the 2015 Global Conference on Communication Technologies (GCCT), Thuckalay, India, 23–24 April 2015; pp. 60–65.
27. Dashtipour, K.; Gogate, M.; Cambria, E.; Hussain, A. A novel context-aware multimodal framework for persian sentiment analysis. *arXiv* **2021** arXiv:2103.02636.
28. Sahoo, K.S.; Puthal, D.; Tiwary, M.; Usman, M.; Sahoo, B.; Wen, Z.; Sahoo, B.P.S.; Ranjan, R. ESMLB: Efficient Switch Migration-based Load Balancing for Multi-Controller SDN in IoT. *IEEE Internet Things J.* **2019**, *7*, 5852–5860. [CrossRef]
29. Hamdan, M.; Hassan, E.; Abdelaziz, A.; Elhigazi, A.; Mohammed, B.; Khan, S.; Vasilakos, A.V.; Marsono, M.N. A comprehensive survey of load balancing techniques in software-defined network. *J. Netw. Comput. Appl.* **2020**, *174*, 102856. [CrossRef]
30. Valdivieso Caraguay, Á.L.; Benito Peral, A.; Barona López, L.I.; García Villalba, L.J. SDN: Evolution and opportunities in the development IoT applications. *Int. J. Distrib. Sens. Netw.* **2014**, *2014*. [CrossRef]
31. Babbar, H.; Rani, S. Software-Defined Networking Based on Load Balancing Using Mininet. In *Proceedings of the Second International Conference on Information Management and Machine Intelligence*; Springer: Singapore, 2021; pp. 69–76.
32. Babbar, H.; Rani, S. *Performance Evaluation of QoS Metrics in Software Defined Networking Using Ryu Controller*; IOP Conference Series: Materials Science and Engineering; IOP Publishing: Bristol, England, 2021; Volume 1022, p. 012024.