sustainability

Article

Cloud Based Smart City Services for Industrial Internet of Things in Software-Defined Networking

Himanshi Babbar [1], Shalli Rani [1,*], Aman Singh [2], Mohammed Abd-Elnaby [3] and Bong Jun Choi [4,*]

[1] Chitkara University Institute of Engineering and Technology, Chitkara University, Rajpura 140401, Punjab, India; himanshi.babbar@chitkara.edu.in
[2] Department of Computer Science and Engineering, Lovely Professional University, Kapurthala 144411, Punjab, India; amansingh.x@gmail.com
[3] Department of Computer Engineering, College of Computers and Information Technology, Taif University, P.O. Box 11099, Taif 21944, Saudi Arabia; maahmed@tu.edu.sa
[4] School of Computer Science & Engineering, and School of Electronic Engineering, Soongsil University, Seoul 06978, Korea
* Correspondence: shalli.rani@chitkara.edu.in (S.R.); davidchoi@soongsil.ac.kr (B.J.C.)

Abstract: The network session constraints for Industrial Internet of Things (IIoT) applications are different and challenging. These constraints necessitates a high level of reconfigurability, so that the system can assess the impact of an event and adjust the network effectively. Software Defined Networking (SDN) in contrast to existing networks segregates the control and data plane to support network configuration which is programmable with smart cities requirement that shows the highest impact on the system but faces the problem of reliability. To address this issue, the SDN-IIoT based load balancing algorithm is proposed in this article and it is not application specific.Quality of service (QoS) aware architecture i.e., SDN-IIoT load balancing scheme is proposed and it deals with load on the servers. Huge load on the servers, makes them vulnerable to halt the system and hence leads to faults which creates the reliability problem for real time applications. In this article, load is migrated from one server to another server, if load on one server is more than threshold value. Load distribution has made the proposed scheme more reliable than already existing schemes. Further, the topology used for the implementation has been designed using POX controller and the results has been evaluated using Mininet emulator with its support in python programming. Lastly, the performance is evaluated based on the various Quality of Service (QoS) metrics; data transmission, response time and CPU utilization which shows that the proposed algorithm has shown 10% improvement over the existing LBBSRT, Random, Round-robin, Heuristic algorithms.

Keywords: software defined networking; industrial internet of things (IIoT); cloud; smart cities; POX; QoS metrics

check for updates

Citation: Babbar, H.; Rani, S.; Singh, A.; Abd-Elnaby, M.; Choi, B.J. Cloud Based Smart City Services for Industrial Internet of Things in Software-Defined Networking. *Sustainability* **2021**, *13*, 8910. https://doi.org/10.3390/su13168910

Academic Editor: Fadi Al-Turjman

Received: 12 July 2021
Accepted: 4 August 2021
Published: 9 August 2021

1. Introduction

New technologies including the IIoT, autonomous vehicular networks, and intelligent medical systems are time-critical applications [1]. Time-critical control loops that demand a strict delay bound have begun to be implemented in industrial automation machines. The novel difficulty of handling time-critical traffic has developed with the emergence of Industry 4.0 and the IIoT. Nowadays, with many physical objects linked to the internet, IIoT has been evolved gradually. Today, IIoT technology connects everyone and everything around the globe via sensor networks, it can detect or track environments and can recognize objects by scanning a Radio Frequency Identification (RFID) tag [2]. Smart homes, automation systems, medical care, smart grid, smart cities, etc. are some current applications where IoT technology is being used. As the volume of linked objects on the internet grows with each day, IoT management and control is becoming a very challenging task. To support better management and handling in current scenario without changing the structure of existing implementations, SDN arose to provide the consistency and programmability of the IoT network [3]. In addition, it will simplify network operations, minimize the costs and speed up the delivery of services by segregating the SDN controllers from the forwarding devices.

Any delays in response time during data transmission would have a negative effect on overall efficiency of the whole network system, particularly in real-time transaction situations. IoT applications are used by SDN to decrease the response time while enhancing the communication approach. Multiple controllers have recently been utilized in SDN rather than using a centralized controller which thereby tends to disperse whole IoT congested load amongst these controllers. The use for SDN in IoT is essential to understand the complexity and difficulty of traditional networks. The SDN in IoT will provide an easy to operate network architectures with no further complications [4,5]. These networks are managed by the routers and switches for packet forwarding in each direction. Therefore, it is hard to be customized for higher-level management policies. The higher-level policies, for example, Second Generation (2G) traffic should be given lesser priority than Third Generation (3G) demand routers by having two queues where one queue is for 2G and the other for 3G packets and allows the 3G packets to be processed in higher proportion concerning 2G packets [5,6]. Another aspect of traditional networking is that network equipment is traditionally proprietary specifying the vendors that provides the integrated solution in terms of operating system. Moreover, software is also composite of networking devices, configuration, protocol implementation and this whole comes from the major vendors. SDN in the cloud is required because different users of cloud need to innovate new kinds of rules and methods to configure the network and run the services as per their requirements [7,8].

The main constraints of SDN and IoT are fault tolerance, reliability, scalability, heterogeneous communication technologies, dynamic network conditions, massive amount of data, application specific quality of service requirements etc. To overcome these constraints we have proposed the SDN-IoT based load balancing algorithm which handles all the network management related issues with the proposal of IoT for smart cities with SDN as it is not application specific therefore, it can be implemented in any kind of application in IoT for smart cities. This paper demonstrates the interaction of the switches and hosts for data transmission on request of users for different applications in smart city domain. The following contributions have been made towards this approach.

Contributions of the Paper

1. This study considers multiple distributed controllers for the implementation of SDN in IIoT for smart city. The use of multiple controllers will help in managing excessive traffic load which otherwise resides on single centralized controller in traditional networks.
2. In the proposed architecture, authors are working on multiple controllers for the load handling in which the network communication is done by the user in intra and inter cluster which thereafter forwards the request to any one of the cluster.
3. The SDN POX controller is keeping the check on the number of requests and the threshold value. If threshold value is more than the assigned number of requests then the requests will be transferred from the intra cluster to inter cluster.
4. The clustering approach is used for the wide scale IoT network configuration which eventually will support scalable and reliable communication with reduced power consumption. The proposed architecture design will also improve the interoperability and versatility of the whole network design.
5. The comparison of LBBSRT, Random, Round robin and heuristic algorithms are done based on the various QoS metrics to evaluate the maximum throughput, minimum latency and maximum CPU utilization.
6. The proposed architecture has been evaluated over Mininet emulator with POX as experimental controller. The results have proved efficiency of our proposed work is 10% accurate than the existing work.

The rest of the paper is organised in Figure 1:

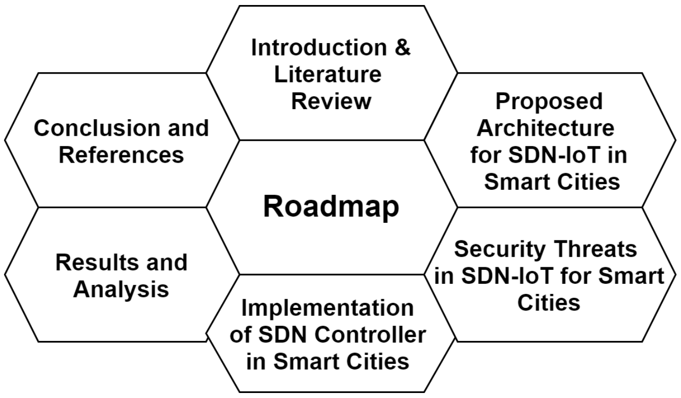
Introduction & Literature Review
Proposed Architecture for SDN-IoT in Smart Cities
Conclusion and References
Roadmap
Security Threats in SDN-IoT for Smart Cities
Results and Analysis
Implementation of SDN Controller in Smart Cities

Figure 1. Roadmap of the paper.

2. Literature Review

In this section, authors are focusing on the recent publications that are attempting to work on the smart city ecosystem based on IoT with the leverage of the SDN paradigm. To the best of our knowledge based on the recent review done on the secure IoT paradigm with SDN, various researchers have performed the integration of IoT based smart cities with SDN.

Balasubramanian et al. [9] developed an algorithm that uses an SDN controller with a broad perspective of the network and is focused on basic online techniques. This article, more especially, in the framework of IEEE Time Sensitive Networking (TSN) norms: creates the TSNu control policy framework, which ensures Scheduled Traffic data transmission distributions while reducing congestion issues. Romero-Gázquez and Bueno-Delgado [10] overcomes the challenges faced by the open source architecture solution based on the OpenDaylight (ODL), an SDN controller for the IIoT scenarios orchestration. Son and Buyya [3] proposed methodology to represent different benefits of cloud computing allowed by SDN but also to describe every feature in description. Thorough surveys of cloud computing research using SDN are described concerning power optimization in data centres, engineering of the traffic and reliability. Kang and Choo [11] implemented the cloud-based network inflow is distributed by the SDN-enhanced InterCloud Manager (S-ICM). For the evaluation, S-ICM uses SDN control packet information and data collection, and the decision-making process is focused on network congestion calculated for packets. Yen and Su [12] presented IaaS and network security issues and issues of the federation presently dealt with this by emerging technologies and innovative software-defined networking ideas that tackle a few of the problems and could be used as effective solutions in the huge impact. Chen et al. [2] proposed a wildcard mask to introduce the load balance method directly on switches and routers and incorporate a user forecast function to dynamically modify the wildcard mask. Accordingly, the load balancing mechanism can be applied following the actual service situation. Vishnu Priya and Radhika [13] evaluated the efficiency of most common OpenFlow controllers, such as NOX, POX, Ryu, FloodLight and OpenFlow reference controller depending on their managing of the efficiency of the packet, by varying the size of a packet, number of packets and pattern of arrival in the direction of IP-traffic.

2.1. IoT in Smart Cities

Mehmood et al. [14] developed a taxanomy based on IoT for smart cities which relies on the various wireless technologies namely IEEE 802.11p, Wave, 6LowPan, etc. For the benefit of researchers, we recently explored big open IoT platforms. In fact, to demonstrate a growing pattern of IoT deployments, a range of published case studies of some of the latest IoT deployments and research projects have been introduced. Arasteh et al. [15] discussed the technologies, various features and components used in smart cities. Shown some light on the practical experiences over the world which will enhance the day to day activities that can be developed and increased by the use of it. Zhao et al. [16] authors implemented the two schemes EOERA and CHERA, to efficiently and dynamically allocate all the edge resources for various applications. EOERA embraces an apportionment approach for all various applications to consider an appropriate case for allocation of resources. Urbieta et al. [17] introduced a structure for adaptive service composition that facilitates such dynamic reasoning. The architecture based on wEASEL, an abstract service model that describes the identity, description (i.e. context-aware pre-conditions, post-conditions and effects) and discussions of services and user tasks (i.e., behaviour with related data-flow and context-flow constraints).

2.2. SDN in Smart Cities

Chen et al. [18] emphasised on locating the path and suggest a statistics-based trace back strategy by using benefits of the SDN architecture to better protect against distributed denial of service (DDoS) attacks. To evaluate the eigenvalue and create the anomaly tree, we examine the flow changes via the Base Station (BS) nodes. Then trim it to obtain the attack route with the DDoS detection algorithm. Xu et al. [19] proposed the strategy for DDOS attacks based on defense classification of traffic to enhance the security of management of data in SDN-enabled smart cities. Bi et al. [20] developed a dynamic architecture based on SDN technology to enable smart city services. Instead, under the proposed system, we examine the time-constrained big data transfer scheduling (TBTS) problem and introduce an intelligent strategy to resolve the TBTS problem by using the SDN controller to perform dynamic flow control and multi-path transfer scheduling.

2.3. IoT-SDN in Smart Cities

In order to provide a more scalable and agile network, we support IoT-based smart city environments with the SDN paradigm. On top of the configured city, Gheisari et al. [21], authors introduced a privacy-preserving technique. This is accomplished by the fact that the data packets of all IoT devices are handled by the SDN controller and their data is separated depending on the context. Ghosh et al. [22] presented our SDN-IoT focused smart city framework, which is optimised, regulated, and controlled by a global control centre. The proposed architecture embraces heterogeneous networks and includes multiple networks, namely ZigBee, MANETs, sensor networks, and Bluetooth. Ouhab et al. [23] proposed model utilizes flow routing, a network routing technique that takes into consideration changes in data flow to improve routing efficiency and modify the duration of packet forwarding between the various nodes from each instance of the Routing Protocol for Low-Power and Lossy Networks (RPL). These two network control levels, called SD- multi-hop clustering technique (MHC)-RPL, deliver better results in respect of network efficiency. Gheisari et al. [21] implemented to test the SDN network infrastructure integrated IoT management framework, and also the interaction of IoT traffic with SDN switches. We conducted numerous IoT traffic tests to perform management system and network infrastructure behavioral investigations, which were generated based on the one M2M specification of the productive resources. Ogrodowczyk et al. [24] discussed the use case of Poznan Smart City, illustrating how a single unifying SDN-based platform can be used to “split” a city into multiple smart spaces functioning over a shared network and cloud infrastructure, as well as how the network architecture allowed by OpenFlow can be used to automate the development of IoT devices to be used in cloud-based multi-tenant implementations. Rego et al. [25] proposed a new control system based on SDN and IoT integration in smart city environments. When an emergency situation occurs, this control system works and automatically alters the paths of regular and emergency traffic congestion to diminish the amount of time that emergency services need to access the emergency area [26] proposed the LBBSRT load balancing method, which is based on server response times and takes benefit of SDN adaptability. We execute user requests by achieving equally balanced server loads using real-time response time of each server determined by the controller for load balancing. Our approach has a greater load balancing impact and processes requests with the shortest average server response times, according to simulations. The authors of [27] suggested to use POX controller to transform openflow devices into hubs, switches, load balancers, and firewalls. The POX controller makes running OpenFlow/SDN tests simple. Various QoS can be supplied to POX based on real or experimental topologies, enabling you to execute experiments on real hardware, testbeds, or in a mininet emulator. The first half of this paper will offer an introduction to POX, OpenFlow, and SDN, followed by a description of the link between POX and Mininet [28] represent a new software-defined networking (SDN) architecture for meeting the QoS needs of different IoT services and balancing traffic among IoT servers at the same time. The issue is first expressed as an NP-hard integer linear programming (ILP) paradigm. Then, using time-series analysis and fuzzy logic, a predictive and proactive heuristic technique is implemented.

The proposed methodology creates the different clusters in IoT which will balance the load of all the clusters depending on the number of requests received on each cluster. If the number of requests are greater than the threshold value assigned then the extra requests received will be switched to the inter cluster from intra cluster so as to balance the load on all the clusters in IoT. The SDN POX controller is responsible for managing the incoming requests and forward to the IoT nodes in the respective clusters.

3. Proposed Architecture for Smart Cities in SDN-IoT

The grounds of the proposed work relies on creating an IoT network supporting smart city application requirement [29]. The network uses SDN technology to create a complex, programmatically optimized IoT network. The design of any physical networks should be as simple as possible while improving the interoperability and versatility of the network. Basically, the structure of this whole network is divided into three layers which comprises:

3.1. Data/Forwarding Layer

To implement optimized and effective decisions, IoT networks are composed of nodes that facilitate the network with the cloud data. On the other hand, virtual devices especially Openflow Switches which perform the role of Cluster Head (CH), are composed of SDN. For the better network management with available resources, this study suggested a clustering approach to arrange the IoT nodes in an effective way [24]. A cluster may contain multiple nodes of the network say $(I_1, I_2, I_3, \dots, I_m)$. The sensing capability of these IoT nodes is to obtain data from their environment. Each cluster is controlled by CH. We have placed the SDN POX Controller with every CH. The key objective of POX controller is to manage and control the cluster region along with providing security from all external and internal threats [25]. The inter cluster communication is achieved through Gateway Nodes (GN).

The Figure 2 depicts virtual devices composed of various routers, switches, load balancers, firewall, etc. The interaction between CH and SDN-IoT gateways controls the SDN controller and control plane transfers the traffic through SDN OpenFlow protocol by completing the whole routing on the data layer.

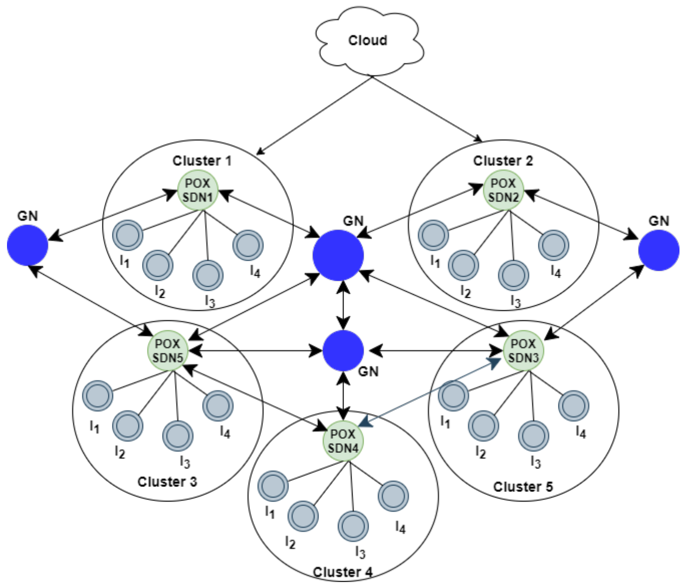

Figure 2. Clustering in IoT for SDN POX Controller.

3.2. Control Layer

Group of multi-functional controllers and virtualized resources are formed by the control layer. It offers guidance of the behaviour of transmitting packets and virtualized services for smart city applications. It is essential to delegate the tasks of the controller to minimize congestion problems. The proposed system framework as shown in Figure 3, the authors suggested several controllers. The researchers use three main categories of network controllers: application, packet, and security controller.

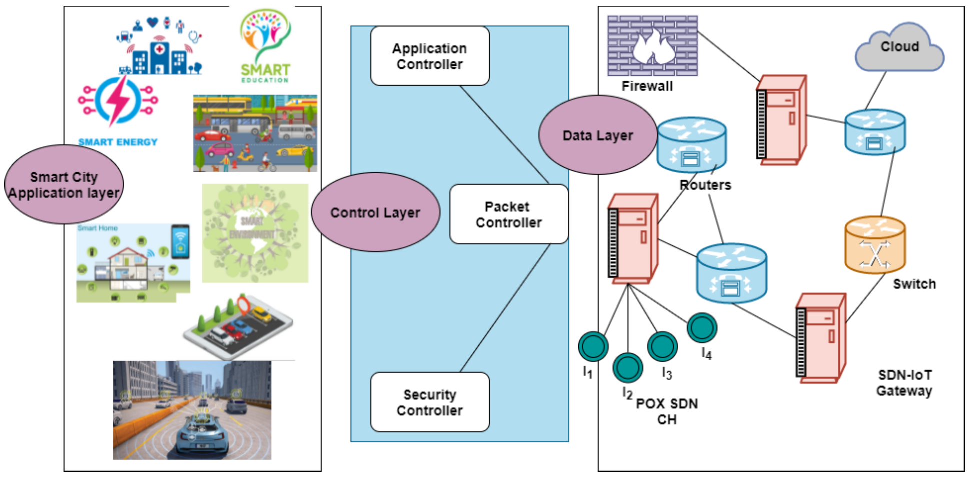

Figure 3. Proposed Framework for SDN-IoT in Smart Cities.

The application controller is configured for the tracking of unauthorized network applications. For balancing the load and monitoring of packets, the packet controller is responsible. The three extended controllers, such as the main, intrusion and crypto controllers, are added by the security controller. Throughout the entire network service, these controllers are used to preserve honesty, anonymity, and confidentiality and are referred as high-level applications.

3.3. Application Layer

This layer is considered as the uppermost layer of the system's architecture in which the built network's application fields are deployed. This system architecture has been proposed by the researchers, specifically for smart city applications as shown in Figure 4.

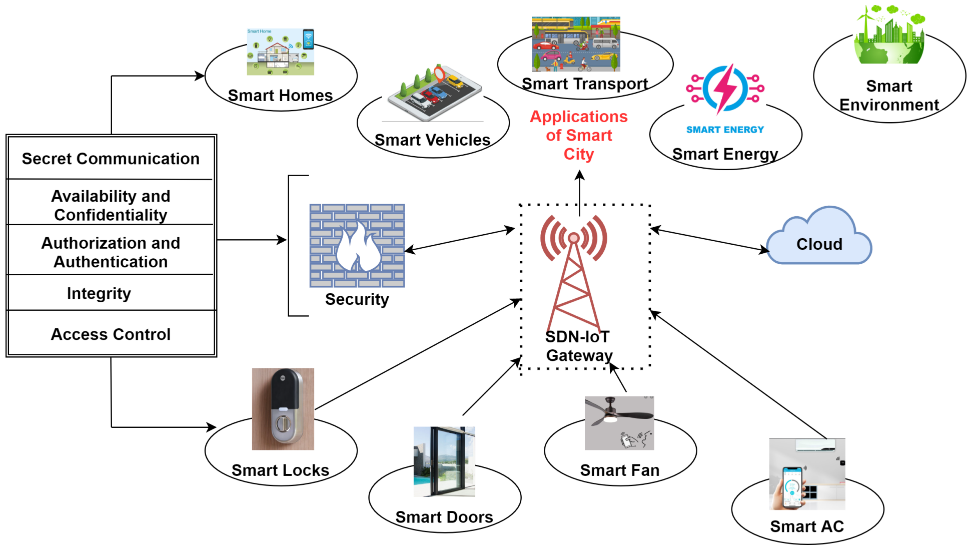

Figure 4. Applications of Smart City.

This layer includes a range of applications for smart cities such as smart homes, smart energy, smart cars, smart health, smart transport, etc. [30]. In addition, it involves server and cloud infrastructures that exchange content and facilitate the user with real-time services. The most important tasks of this layer are both data processing and supplying services.

3.4. System Model

The communication in the network gets initiated by an user while sending request towards any one of the cluster. Considering a network scenario as shown in Table 1, each cluster say $cluster_i$ is supposed to have predefined threshold value TH_i which defines the increased number of requests which can be served by this cluster. All the nodes in each cluster say I where $(I \rightarrow I_1, I_2, I_3, \ldots)$ are connected via a local SDN POX controller. This controller is responsible for maintaining check on the number of requests currently being served within its own cluster. At particular instance, when the threshold value matches with the current number of processed requests, the POX controller does not let any new request to get served in its own cluster and hence, it transmits this request towards another cluster via GN. This request again has to be checked at newly arriving cluster for its processing. In case of availability for request processing, the POX controller allows this request to come in and transmits it to the node in its cluster based on remaining request processing availability of that node and is checked at each node using Equation (1) :

$$current - req_{I1,cluster_{i+1}} < req_{I1,cluster_{i+1}} \tag{1}$$

Here, $current - req_{I1,cluster_{i+1}}$ is the current number of requests being served by node $I1$ in $cluster_{i+1}$, $req_{I1,cluster_{i+1}}$ is the number of requests that can be processed by node $I1$ corresponding to $cluster_{i+1}$.

In case of multiple nodes available for processing this request, it can be sent to any one of them by after increment of 1 to its $current - req$ count. The number of requests that can be processed by any node $I1$ in its cluster say $cluster_{i+1}$ is based on its threshold value TH_{i+1} and is computed using Equation (2):

$$req_{I1,cluster_{i+1}} = \frac{TH_{i+1}}{I} \tag{2}$$

where TH_{i+1} is the predefined threshold of $cluster_{i+1}$ and I is the total number of nodes in $cluster_{i+1}$.

The process goes on until all the requests received from the users are not satisfied by any one of the available clusters. The complete working of the proposed architecture is represented in Table 1.

Table 1. Algorithm for Cluster IoT nodes mechanism.

Algorithm 1: Cluster-IoT nodes for smart cities

```
Variables: $cluster_i$, $req_i$, $TH_i$
Procedure input: PacketIn Packets accepted from the controller
Begin: $Cluster_i$
1    If $TH_i = current_{req_i}$
2        send request to $cluster_{i+1}$
3    else $cluster_i$ processes the request
4        request received by $node_j$
5        If $current_{req}\ node_j < Max_{req_{node\,j}}$
6        $node_j$ process the request
7    else
8        send request to $node_{j+1}$
9            endif
10   endif
End
```

4. Implementation of SDN Controller in Smart Cities

SDN Controller is the program which serves as a technical control plane within a software-defined network. Generally, this is the network's "brains" [31]. To deploy efficient networks, an controller of SDN establishes flow access to switches/routers (via southbound APIs) and applications and business logic (via northbound APIs). The POX controller is used for the implementation of SDN.

POX Controller

SDN-based IoT network application that sits on top of the SDN controller provides the solution's primary logic. Using the SDN infrastructure, it rapidly generates and controls end-to-end channels of communication from IoT devices to the cloud using TCP/UDP, the program orchestrates cloud resources and controls IoT traffic between different clusters and cloud services. OpenFlow SDN Controllers are included in this Python-based open source framework SDN control program. It enables rapid prototyping and growth [27]. Features that are offered in POX are: Discovery of topology, follows similar visualization tools as NOX, it can run anywhere and is a pythonic open flow interface with support from platform like Windows, MAC OS, Linux.

Main Functions and Classes of POX Controller

- **ofp_match:** This class describes packet header fields and an input port to match on. All fields are optional and these fields are not specified. Therefore,this will be treated as wildcard and will match on anything. Some of the fields are:
  - **dl_type:** It is used to specify whether the packet is arp (0x806) or IP(0x800) type.
  - **dl_src,dl_dst:** It is for specifying layer 2 source and destination MAC address.
  - **in_port:** The port through which packet came
  - **tp_dst:** This is to specify TCP/UDP destination port
- **ofp_flow_mod:** It is OpenFlow message(instruction) sent by the controller to the switch to install flow entries into the flow table. Incoming packets are matched against these flow entries and action is performed on these packets as specified inflow entries. The main fields of ofp_flow_mod message are:
  - **hard_timeout:** After how many seconds the flow entry will be removed. The default is no timeout.

  - **idle_timeout:** After how many seconds the idle flow entries will be removed. The default is no timeout.
  - **priority:** relative importance of flow entries.
  - **buffer_id:** The buffer id of the packet.
- **ofp_packet_out:** It is an OpenFlow message sent by the controller to the switch to send the packet out. The packet sent out could be the one that was received by the switch and delivered to the controller after buffering or the packet created by the controller itself. The main fields are:
  - **data:** raw data that you want to send. No need if sending buffered data.
  - **buffer_id:** The buffer id of the packet
  - **action:** list of actions in_port: the port on which the packet arrived. Specify OFPP_NONE for the packet created at the controller.

5. System Analysis and Performance Evaluation

Authors are going to create a switch application using POX Controller as switches are intelligent devices therefore, they maintain a dictionary of MAC addresses and ports. Firstly, the logic of our application will be explained

Our switch topology consists of 4 hosts, 1 switch and 1 POX controller as shown in Figure 5 . When our switch application is executed the "switch application-efficient" on top of the POX controller, 2 tables will be maintained. One table will be maintained at the controller and another table (flow table) will be maintained at the switch. The table maintained at the controller will be "MAC to port table". The table maintained at the switch will contain flow entries. Initially, both tables will be empty.

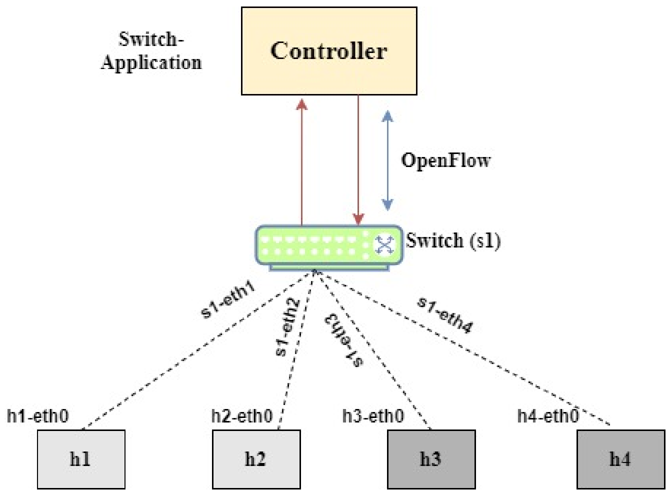

Figure 5. POX Controller Using Switch Application.

Initially, the "MAC to port" at the controller will be empty. Now host h1 wants to ping to host h4. What will happen in the MAC to port table ?

First, h1 will send ARP requests. Since the flow table at switch "s1" does not contain any flow entry therefore, the packet is transferred to the controller. The controller looks at SRC MAC address and the port on which packet came to the switch and makes 1 entry in the "MAC to port" table maintained at the controller. Since in our case, the packet's SRC MAC address was "00:00:00:00:00:01" and came to the switch through port 1. Therefore, the controller made the entry in the controller table as shown in Table 2. The controller will look for the DST MAC address in the "MAC to port" table. If entry is found, the packet will be sent out from the corresponding port, otherwise, it will be flooded. As can be seen, the DST MAC address "00:00:00:00:00:04" is not in the table, so the packet is getting flooded and no entry is made at the switch flow table. Now, h4 will send an ARP reply. Since the flow table at switch "s1" does not contain any flow entry, therefore, the packet is transferred to the controller. The controller looks at SRC MAC address and the port on which packet came to the switch and makes 1 entry in the "MAC to port" table maintained at the controller.

Table 2. Source MAC to port 1 table.

MAC Address	Port
00:00:00:00:00:01	1

Therefore, the packet's SRC MAC address was "00:00:00:00:00:04" and came to the switch through port 4. So the controller made the entry in the "MAC to port" table as shown in Table 3. Now the controller will look at the DST MAC address which is "00:00:00:00:00:01" in our case. The controller will look for the DST MAC address in the "MAC to port" table. If entry is found, the packet will be sent out from the corresponding port, otherwise, it will be flooded. The entry added tells the switch if further packets with DST MAC address "00:00:00:00:00:01" enters the switch, do not send these packets to the controller. Handle these packets at the switch itself and sent the packet out from port 1. The next packet will be ICMP request from host h1 (00:00:00:00:00:01) to host h4 (00:00:00:00:00:04). The switch has got no entry regarding how to handle traffic going to DST MAC "00:00:00:00:00:04", so the packet will be sent to the controller. The controller will consult its table. Since now the controller has information about DST MAC so it will instruct the switch to send the packet out from port 4. After some time the "MAC to port" table at controller and flow table at the switch.

Table 3. Destination MAC to port 4 table.

MAC Address	Port
00:00:00:00:00:01	1
00:00:00:00:00:04	4

5.1. Creation of Topology and Testing of Connectivity in Mininet Emulator

Mininet is an emulator that works over many networks having a limited number of resources. It is an emulator in which topologies can be created of very small size to large. This emulator is used to run the gathering of end-hosts, switches, routers and maintain a link by using the Linux kernel. To work with Mininet, require hosts, switches and wires/cables to have a connection between controllers and switches. The creation of topology comprises of 1 OpenFlow switch (s1) linked to 4 hosts (h1, h2, h3, h4) and a POX controller. The syntax for single topology is: mn−−topo single, n. It consists of n hosts and 1 OpenFlow switch. For e.g., h4 will ping h1. The successful pings mean all the links in the network are active.

5.2. Performance Evaluation

In this section authors have framed three topologies of network with various QoS metrics to calculate the flow of traffic of load and compared with the LBBSRT [26], Random [27], Round Robin [27], Heuristic [28] algorithms. The whole simulations are done in Mininet emulator with OpenFlow switches that imitates the real networking scenarios and have captured the packets from the Wireshark for packets to analyze and calculates the throughput, response time and CPU Utilization (TCP packets captured from Wireshark) of the different topologies. Authors found that proposed algorithms have better efficient results than the existing algorithms on the basis of various QoS metrics namely data transmission, response time and CPU utilization.

1. **Comparison of Data Transmission:** The data transmission is the number of incoming requests received on the traffic at different time intervals. In Figure 6 the transmission of data is achieved highest in the proposed algorithm as compared to the existing

algorithms, with the load of 100 Mbps the data transmission is 200 Mbps; and with 900 Mbps of load the data transmission is 800 Mbps and so on. As the load of traffic is increasing the data transmission is rising accordingly. More load on clusters gives rise to more data transmission. Proposed algorithm has shown the 65%, 70%, 70%, 72% improvement over LBBSRT [26], Random [27], Round Robin [27], Heuristic [28] algorithms respectively in data transmission.

2. **Comparison of Average Response Time:** The response time signifies the minimum number of incoming requests received on the clusters. Figure 7 gives the low response time for the proposed algorithm which has been able to provide the unique flows for the IoT nodes. As per the result, with 100 Mbps of load the response time is 0 ms; 300 Mbps of load the response time is 10 ms and so on. Proposed algorithm has shown the 90%, 290%, 257%, 161% improvement over LBBSRT [26], Random [27], Round Robin [27], Heuristic [28] algorithms respectively in response time.
3. **Comparison of CPU Utilization:** The balancing of load on the different servers i.e., server1, server2, server3 by various algorithms on different load on the clusters. As seen, the CPU utilization of existing algorithms in Figure 8 is higher in all the servers as compared to the proposed algorithm and the load imbalance occurs between the IoT nodes if the resources are properly utilized. Proposed algorithm has shown the 64%, 79%, 72%, 63% improvement over LBBSRT [26], Random [27], Round Robin [27], Heuristic [28] algorithms respectively in CPU utilization.

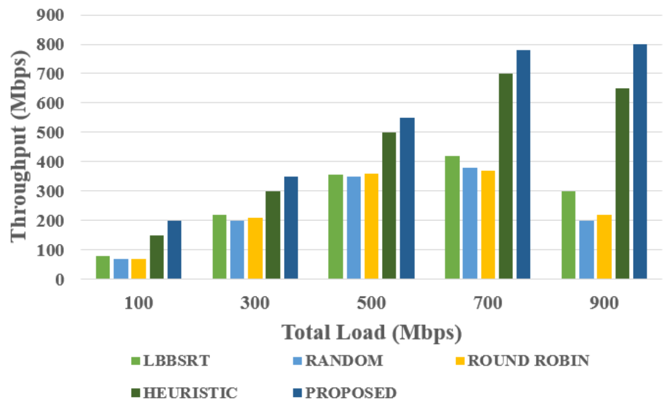

Figure 6. Data Transmission Rate.

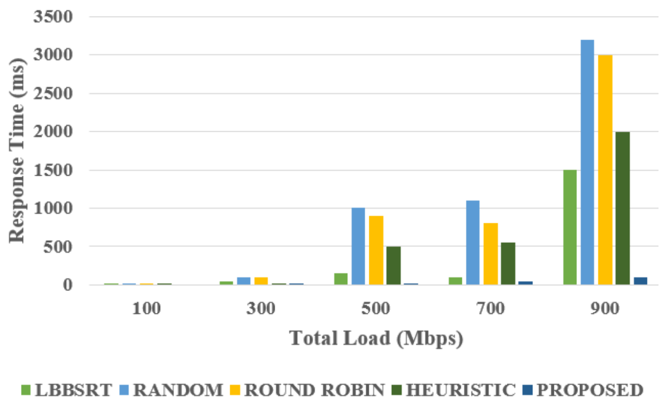

Figure 7. Response Time.

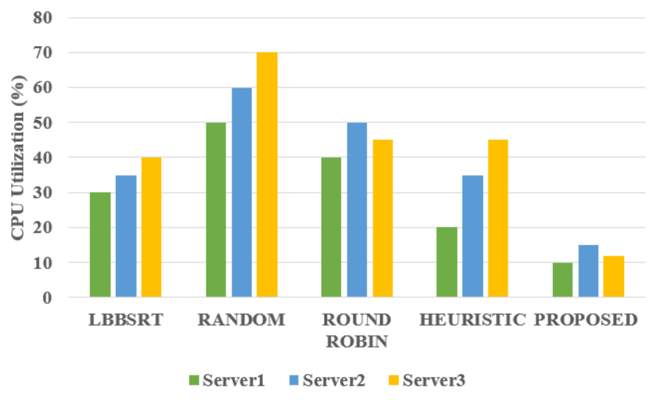

Figure 8. CPU Utilization.

6. Conclusions

Software-Defined Networking will help cloud computing service providers to build faster network with simple design configuration along with easy handling of huge number of devices. This paper has presented the implementation of the SDN controller for IIoT based 'cloud services. SDN in the cloud has provided enormous opportunities to network which facilitated changing adaptation and reorganization with its control layer separation from the controlled forwarding system by the central server. This paper proposed a switch application using the POX controller with its implementation in Mininet emulator using python package and libraries. The comparison of LBBSRT, Random, Round robin and heuristic algorithms are done based on the various QoS metrics to evaluate the maximum throughput, minimum latency and maximum CPU utilization and it has shown the 10%, 12%, 7% improvement in data transmission, response time, CPU utilization respectively over proposed algorithm. The future SDN can further be extended with more efficient technique such as machine learning for real time applications.

Author Contributions: Conceptualization, H.B. and S.R.; methodology, S.R. and H.B.; validation, A.S., S.R. and H.B.; formal analysis, A.S., S.R. and H.B.; investigation, S.R. and M.A.-E.; resources, S.R.; data curation, B.J.C. and A.S.; writing—original draft preparation, S.R. and H.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by Taif University Researchers Supporting Project Number (TURSP-2020/147), Taif University, Taif, Saudi Arabia; and the research was also supported by the National Research Foundation (NRF) Korea (2019R1C1C1007277) funded by the Ministry of Science and ICT (MSIT) Korea and by the Korea Institute for Advancement of Technology (KIAT) grant funded by the Korea Government (MOTIE) (P0017123, The Competency Development Program for Industry Specialist).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: This research was supported by Taif University Researchers Supporting Project Number (TURSP-2020/147), Taif University, Taif, Saudi Arabia; and the research was also supported by the National Research Foundation (NRF) Korea (2019R1C1C1007277) funded by the Ministry of Science and ICT (MSIT) Korea and by the Korea Institute for Advancement of Technology (KIAT) grant funded by the Korea Government (MOTIE) (P0017123, The Competency Development Program for Industry Specialist).

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

References

1. Al-Turjman, F.M. Information-centric sensor networks for cognitive IoT: an overview. *Ann. Telecommun.* **2017**, *72*, 3–18. [CrossRef]
2. Yang, C.T.; Chen, S.T.; Liu, J.C.; Su, Y.W.; Puthal, D.; Ranjan, R. A predictive load balancing technique for software defined networked cloud services. *Computing* **2018**, *101*, 211–235. [CrossRef]
3. Son, J.; Buyya, R. A taxonomy of software-defined networking (SDN)-enabled cloud computing. *ACM Comput. Surv.* **2018**, *51*, 1–36. [CrossRef]
4. Xia, W.; Wen, Y.; Foh, C.H.; Niyato, D.; Xie, H. A Survey on Software-Defined Networking. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 27–51. [CrossRef]
5. Astuto, B.N.; Mendonça, M.; Nguyen, X.N.; Obraczka, K.; Astuto, B.N.; Mendonça, M.; Nguyen, X.N.; Obraczka, K.; Sur, T.T.A.; Nunes, B.A.A.; et al. A Survey of Software-Defined Networking : Past , Present , and Future of Programmable Networks. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 1617–1634.
6. Azodolmolky, S.; Wieder, P.; Yahyapour, R. SDN-based cloud computing networking. In Proceedings of the 2013 15th International Conference on Transparent Optical Networks (ICTON), Cartagena, Spain, 23–27 June 2013; [CrossRef]
7. Lins, T.; Oliveira, R.A.R. Energy efficiency in industry 4.0 using SDN. In Proceedings of the 2017 IEEE 15th International Conference on Industrial Informatics, INDIN 2017, Emden, Germany, 24–26 July 2017; pp. 609–614. [CrossRef]
8. Al-Turjman, F.; Alturjman, S. Context-sensitive access in industrial internet of things (IIoT) healthcare applications. *IEEE Trans. Ind. Inform.* **2018**, *14*, 2736–2744. [CrossRef]
9. Balasubramanian, V.; Aloqaily, M.; Reisslein, M. An SDN architecture for time sensitive industrial IoT. *Comput. Netw.* **2021**, *186*, 107739. [CrossRef]
10. Romero-Gázquez, J.L.; Bueno-Delgado, M. Software architecture solution based on SDN for an industrial IoT scenario. *Wirel. Commun. Mob. Comput.* **2018**, *2018*, 2946575. [CrossRef]
11. Kang, B.; Choo, H. An SDN-enhanced load-balancing technique in the cloud system. *J. Supercomput.* **2016**, *74*, 5706–5729. [CrossRef]
12. Yen, T.C.; Su, C.S. An SDN-based cloud computing architecture and its mathematical model. In Proceedings ogf the 2014 International Conference on Information Science, Electronics and Electrical Engineering, ISEEE 2014, Sapporo, Japan, 26–28 April 2014; Volume 3, pp. 1728–1731. [CrossRef]
13. Vishnu Priya, A.; Radhika, N. Performance comparison of SDN OpenFlow controllers. *Int. J. Comput. Aided Eng. Technol.* **2019**, *11*, 467–479. [CrossRef]
14. Mehmood, Y.; Ahmad, F.; Yaqoob, I.; Adnane, A.; Imran, M.; Guizani, S. Internet-of-Things-Based Smart Cities: Recent Advances and Challenges. *IEEE Commun. Mag.* **2017**, *55*, 16–24. [CrossRef]
15. Arasteh, H.; Hosseinnezhad, V.; Loia, V.; Tommasetti, A.; Troisi, O.; Shafie-khah, M.; Siano, P. Iot-based smart cities: A survey. In Proceedings of the 2016 IEEE 16th International Conference on Environment and Electrical Engineering (EEEIC), Florence, Italy, 7–10 June 2016; pp. 1–6. [CrossRef]
16. Zhao, L.; Wang, J.; Liu, J.; Kato, N. Optimal Edge Resource Allocation in IoT-Based Smart Cities. *IEEE Netw.* **2019**, *33*, 30–35. [CrossRef]
17. Urbieta, A.; González-Beltrán, A.; Ben Mokhtar, S.; Anwar Hossain, M.; Capra, L. Adaptive and context-aware service composition for IoT-based smart cities. *Future Gener. Comput. Syst.* **2017**, *76*, 262–274. [CrossRef]
18. Chen, W.; Xiao, S.; Liu, L.; Jiang, X.; Tang, Z. A DDoS attacks traceback scheme for SDN-based smart city. *Comput. Electr. Eng.* **2020**, *81*, 106503. [CrossRef]
19. Xu, C.; Lin, H.; Wu, Y.; Guo, X.; Lin, W. An SDNFV-Based DDoS Defense Technology for Smart Cities. *IEEE Access* **2019**, *7*, 137856–137874. [CrossRef]
20. Bi, Y.; Lin, C.; Zhou, H.; Yang, P.; Shen, X.; Zhao, H. Time-Constrained Big Data Transfer for SDN-Enabled Smart City. *IEEE Commun. Mag.* **2017**, *55*, 44–50. [CrossRef]
21. Gheisari, M.; Wang, G.; Khan, W.Z.; Fernández-Campusano, C. A context-aware privacy-preserving method for IoT-based smart city using Software Defined Networking. *Comput. Secur.* **2019**, *87*, 101470. [CrossRef]
22. Ghosh, U.; Chatterjee, P.; Shetty, S.; Datta, R. An SDN-IoT-based Framework for Future Smart Cities: Addressing Perspective. *arXiv* **2020**, arXiv:2007.11536.
23. Ouhab, A.; Abreu, T.; Slimani, H.; Mellouk, A. Energy-efficient clustering and routing algorithm for large-scale SDN-based IoT monitoring. In Proceedings of the ICC 2020-2020 IEEE International Conference on Communications (ICC), Dublin, Ireland, 7–11 June 2020; pp. 1–6. [CrossRef]
24. Ogrodowczyk, .; Belter, B.; LeClerc, M. IoT Ecosystem over Programmable SDN Infrastructure for Smart City Applications. In Proceedings of the 2016 Fifth European Workshop on Software-Defined Networks (EWSDN), Den Haag, The Netherlands, 10–11 October 2016; pp. 49–51. [CrossRef]
25. Rego, A.; Garcia, L.; Sendra, S.; Lloret, J. Software Defined Network-based control system for an efficient traffic management for emergency situations in smart cities. *Future Gener. Comput. Syst.* **2018**, *88*, 243–253. [CrossRef]
26. Zhong, H.; Fang, Y.; Cui, J. Reprint of "LBBSRT: An efficient SDN load balancing scheme based on server response time". *Future Gener. Comput. Syst.* **2018**, *80*, 409–416. [CrossRef]

27. Kaur, S.; Singh, J.; Ghumman, N.S. Network programmability using POX controller. In Proceedings of the ICCCS International Conference on Communication, Computing & Systems, Macau, China, 19–21 November 2014; IEEE: Piscataway, NJ, USA , 2014; Volume 138.
28. Moghaddam, Y.; Hossein, M. Load-Balanced and QoS-Aware Software-Defined Internet of Things. *IEEE Internet Things J.* **2020**, *7*, 3323–3337.
29. Al-Turjman, F.; Malekloo, A. Smart parking in IoT-enabled cities: A survey. *Sustain. Cities Soc.* **2019**, *49*, 101608. [CrossRef]
30. Ullah, Z.; Al-Turjman, F.; Mostarda, L.; Gagliardi, R. Applications of artificial intelligence and machine learning in smart cities. *Comput. Commun.* **2020**, *154*, 313–323. [CrossRef]
31. Paliwal, M.; Shrimankar, D.; Tembhurne, O. Controllers in SDN: A review report. *IEEE Access* **2018**, *6*, 36256–36270. [CrossRef]