VOL. 20, NO. 1, 2021, 65-75
www.elektrika.utm.my
ISSN 0128-4428

ELEKTRIKA
Journal of Electrical Engineering

Quality of Service Evaluation of Software Defined Internet of Things Network

Paulson Eberechukwu Numan[1,3*], Kamaludin Mohamad Yusof[2*], Jafri Bin Din[1], Muhammad Nadzir Bin Marsono[2], Umar Suleiman Dauda[3], Salawu Nathaniel[3] and **Fapohunda Kofoworola O[2]**

[1]Wireless Communication Center, Universiti Teknologi Malaysia, 81310 UTM Skudai, Johor, Malaysia.
[2]School of Electrical Engineering, Universiti Teknologi Malaysia, 81310 UTM Skudai, Johor, Malaysia.
[3]School of Electrical Engineering & Technology, Federal University of Technology, PMB 65 Minna, Nigeria.

[*]Corresponding author: enpaulson2@gmail.com, kamalmy@utm.my

Abstract: With the exponential growth of the Internet of Things (IoT) devices connected to the internet, resource provisioning for such the heterogeneous network is a challenging task for the traditional network architecture. In this context, the Software-Defined Networking (SDN) introduces many opportunities and provides the potential to overcome challenges associated with traditional network architecture. This work presents a Software-Defined IoT (SDIoT) architecture. The main focus of this research is to design a control plane (CP) for the SDIoT. The scope of this work is limited to the introduction of an overlay SDN CP in the traditional IoT network architecture. The proposed architecture focuses on resource provisioning while ensuring the quality of service (QoS) satisfaction for the network. A comparative analysis between the traditional and the SDN network approach was done in terms of Jitter, Latency and Throughput. From the latency, delay and throughput performance results, the SDN-based IoT network improves network efficiency by reducing network overheads generated from frequent communication between the nodes and the controllers. Precisely, the average latency and average jitter percentile improvement from the traditional IoT network to the SDIoT for all the nodes is 574% and 600% respectively. Also, an overall throughput improvement is recorded for the SDIoT when compared to traditional IoT network for all the nodes.

Keywords: Internet of Things, Software-Defined Networking, SDIoT, QoS, Jitter, Latency and Throughput.

Article History: received 28 June 2020; accepted 27 April 2021; published 30 April 2021.

1. INTRODUCTION

The introduction of the IoT and SDN technologies refines every precedent and preconceived notion of networking. The IEEE defined IoT as a system that deals with the interconnection of "Things." The word "Things" refers to any physical object that is relevant from a user or application perspective [1]. IoT envisions a self-configuring, adaptive, complex network that interconnects 'things' to the internet through the use of standard communication protocols. The things offer services which are made available anywhere, anytime, and for anything, with or without human intervention [2, 3]. The evolution of IoT, as shown in Figure 1, dates back to the pre-internet era when there is only human to human communication.

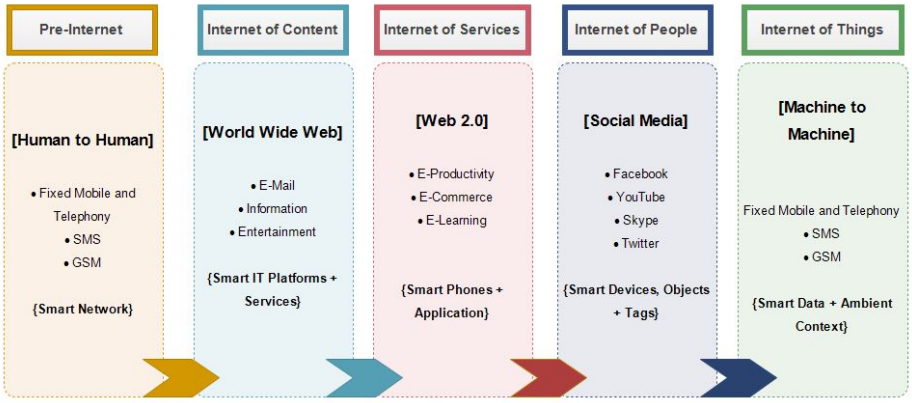

Figure 1 Evolution to IoT [4]

The era of the internet of content with the evolution of the worldwide web (www) followed afterwards. After that, we had the era of the internet of services with the evolution of "web 2.0", the first game-changer of the modern internet. In this era, we started using the internet more frequently for communication and other purposes. Then

came in the era of the internet of people where humans are connected in various ways and in real-time not only via phone and computers. The IoT is the result of many different enabling technologies such as embedded systems, wireless sensor networks, cloud computing, and big-data used to gather, process, and transmit data [5]. It is expected in the coming era, IoT devices will be part of the environment around us which will generate an enormous amount of data. Processing is required on the generated data, which is then presented in an understandable form to the requester. There are many different application domains where IoT plays a crucial role like manufacturing, health-care, transport, administration, insurance, public safety, local community, metering, road safety, traffic management and tracking. These application scenarios typically require a geographical distribution of sensors and actuators that aid in fine-grained data collection and actuation. Also, in these scenarios, resource provisioning is critical to ensure network users are serviced adequately. One of the fundamental challenges in supporting such distributed application scenarios is the need for high bandwidth, low latency wireless backplane that can seamlessly interconnect all these sensors and actuators especially when the sensors and actuators could have high mobility. These application scenarios continuously generate large volumes of "tiny data", and the existing network infrastructure provides minimal functionality for empowering such applications, primarily due to the low latency, Throughput, high Jitter and other QoS requirements.

Moreover, existing networks infrastructure are vertically integrated. Vertical integration is the process of integrating a network's subsystems according to their functionality, usually by creating information silos. Usually, networking infrastructure consists of different networking devices such as switches, routers, and intermediate devices, in which application-specific integrated circuits are installed to perform dedicated tasks [6]. Therefore, the devices are vertically integrated and pre-programmed with different complex rules which cannot be modified in real-time, to perform the dedicated tasks. The vertical integration makes it incredibly difficult for operators to specify high-level network-wide policies using new technologies. Also, proprietary software and closed development in traditional network devices by a handful of vendors make it extremely difficult to introduce and deploy new protocol. Likewise, a clean-slate approach to change the internet architecture (for example, replacing routing table), is regarded as a daunting task in practice [7, 8].

Furthermore, in the traditional network, the CP and the data plane (DP) are bundled inside the same networking devices, as illustrated in Figure 2. The CP decides how to handle the network traffic while the DP forwards traffic according to the decisions made by the CP. As a result, the flexibility of the network is reduced, and the innovation and evolution of the networking infrastructure are also hindered [6].

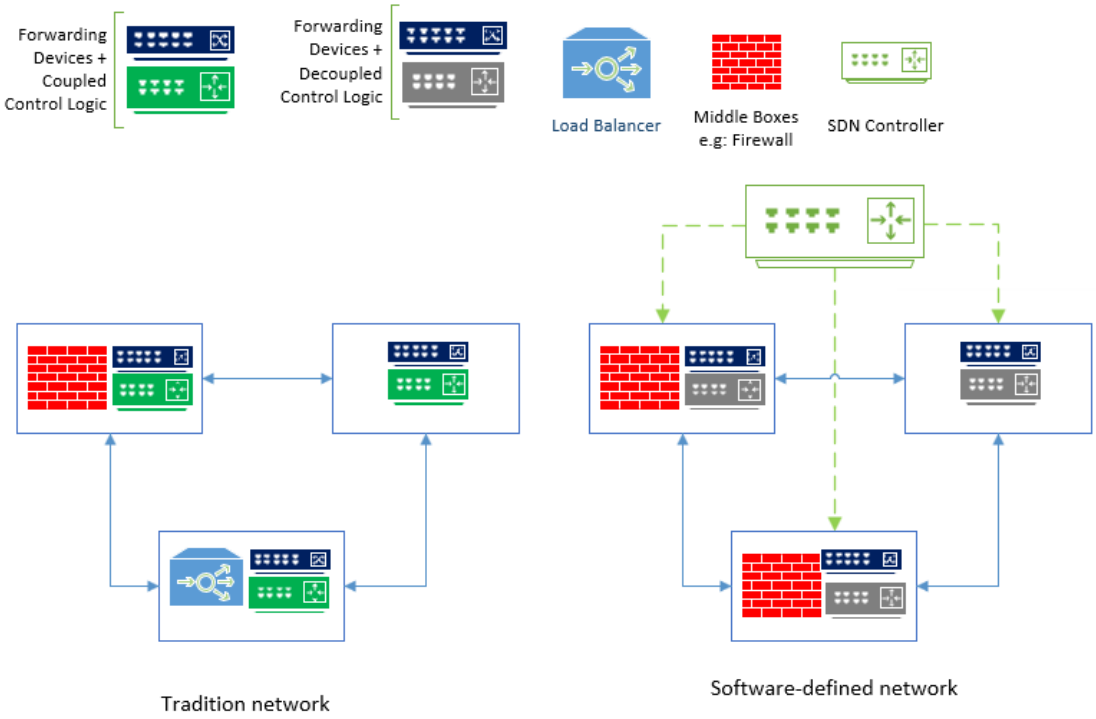

Figure 2. Tradition network vs Software-defined network

To mitigate the challenges associated with traditional networks, the SDN paradigm offers an attractive solution to manage IoT resources which have been lately under focus. The primary objective of the SDN is to separate the CP from the DP involving the forwarding devices, as shown in Figure 2. This separation of CP and DP in SDN can be achieved using a well-defined programming interface between the switches and the SDN controller. Experts predict that IoT could comprise a whopping 21 billion devices by 2020, making SDN a critical technology for IoT [9]. SDN will become important as IoT matures, and the expected result is that SDN will help drive the

expansion of IoT-enabled devices, enable a dynamic and more efficient network resource sharing and improve IoT network. SDN is significantly advantageous to IoT in several ways; (i) SDN simplifies the creation, deployment and ongoing management of the IoT devices and the applications that benefit from them. (ii) SDN solutions can help provide a more agile optimised user experience while allowing the agility increase across the entire network, decrease deployment time, and reduce costs. (iii) SDN ensures network admin can deal with varied bandwidth demands in a flexible manner, providing the bandwidth to the entity that needs it most at a specific moment. The IoT and SDN are two technologies that very much depend on each other. SDN technology can better prepare a network for a successful and robust IoT. It provides the agility and elasticity, which IoT demands. Moreover, it provides an open environment for application developers to develop innovative tools and software connecting the IoT more effectively.

Both of these technologies (SDN and IoT) complement each other in bringing us a better and vastly more connectedworld. The goal of this work is to design and evaluate the SDIoT network architecture in contrast with the traditional network architecture. We have presented results in terms of network latency, Jitter and Throughput to show the strength of the SDIoT architecture over the traditional network architecture. The remainder of this paper is organised as follows. Section 2 gives a brief introduction to the IoT paradigm and its challenges. Section 3 presents a review of the existing SDN based IoT solution. In Section 5, we present the measurement results and analysis and conclude the paper in Section 5.

2. IOT NETWORK PARADIGM

We are living in the era of connected objects where devices can communicate with the physical world and capable of taking decisions due to the data analytics. The IoT is simply the point in time when more things or objects are connected to the internet than people [10-13]. As the boundaries of connected objects are not limited to a specific technology, diverse ranges of objects connect and communicate with each other using a different communication protocol, resulting in the heterogeneous network. IoT devices are used to sense, collect, process, infer, transmit, notify, manage, and store data. IoT architecture follows layered architecture, and a layered architecture ensures flexibility and capability of invocation of new services in the network. Most IoT architecture models are modelled, as shown in Figure 3.

(i) **Sensing layer**: Sensing layer is physical object layer consisting of sensors, actuator, RFIDs, mobile devices, motes, blue tooth etc. This layer collects the data from the environment and transmits on the edge of the network, i.e. gateway or sink.

(ii) **Network layer**: This layer is responsible for transmitting data from physical objects to the gateway/edge of the network for further processing on the collected information. Different transmission technologies contribute to the heterogeneity of IoT such as ZigBee, blue tooth, Wi-Fi etc.

(iii) **Data Processing Layer:** The data processing layer consists of the central data processing unit of IoT devices. The data processing layer takes data collected in the sensing layer and analyses the data to make decisions based on the result. In some IoT devices (e.g., smartwatch, smart home hub, etc.), the data processing layer also saves the result of the previous analysis to improve the user experience. This layer may share the result of data processing with other connected devices via the network layer.

(iv) **Application layer**: This layer deal with the application/services of the user demand by manipulating the information collected from the perception layer and processed in the processing system. The application layer is a user-centric layer which executes various tasks for the users. There exist diverse IoT applications, which include smart transportation, smart home, personal care, health-care, etc.

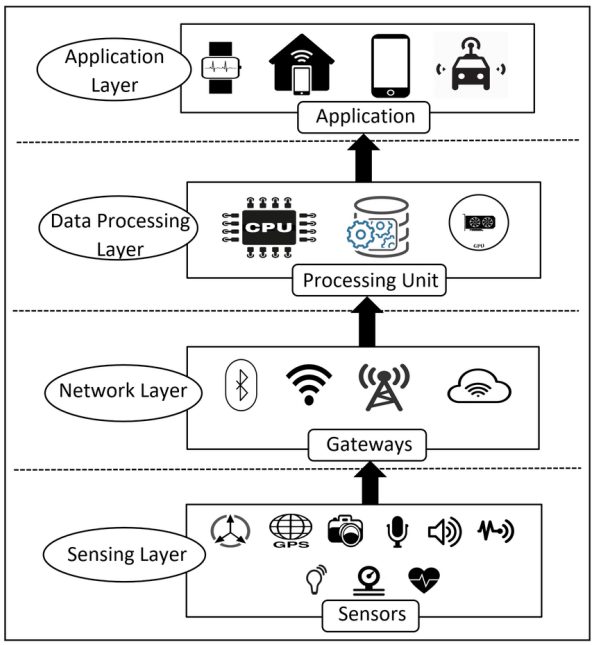

Figure 3. The layered IoT architecture [10]

2.1 IoT Architectural Challenges

Initially, the internet was distinctly established over TCP/IP suite and provided support for a large number of the connected computer. However, TCP/IP does not support heterogeneous network. Therefore, the TCP/IP is not suitable for IoTs. Hence, the heterogeneity of the connected device in the IoT environment is creating unprecedented complexity and functional diversity. Conventional network management techniques are inapplicable in IoT due to distinctive challenges. IoT devices are connected to the internet via a gateway. Usually, in the IoT network, high fault rates are experienced due to shortfall in energy and connectivity interruptions. A typical management solution should provide various management functions integrating configuration, security operation, administration of devices and services of IoT. The following set of functions, as illustrated in Figure 4, should be provided by a management solution for IoT.

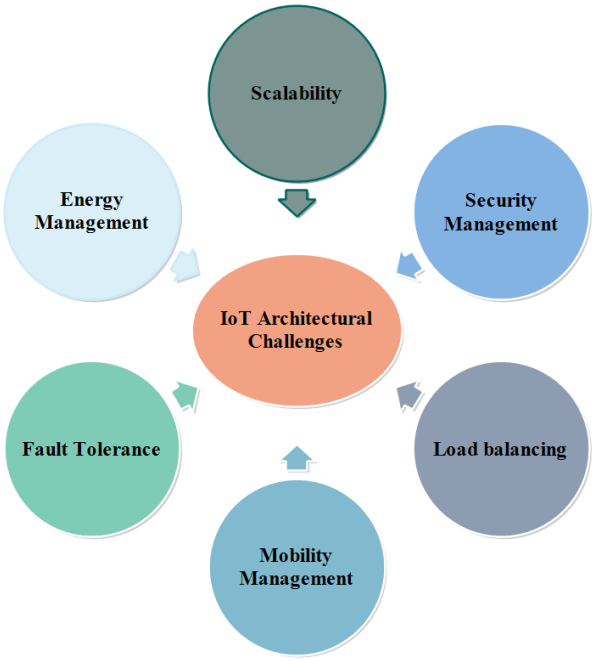

Figure 4. The set of functions as that should be provided by management solution for IoT.

3. SDN BASED IOT ARCHITECTURES

The introduction of SDN to the IoT framework could leverage the network efficiency and attain the programmability and flexibility of networks. The SDN based IoT will help drive the expansion of IoT-enabled devices, enable a dynamic and more efficient network resource sharing and improve IoT network. There are still open research questions regarding the integration of SDN in current trends of IoT. In this regard, many studies have been conducted on the campuses and on the industrial level to get full advantage of programmability from SDN. In this section, we present a review of the existing SDIoT solution. Many IoT platforms have also been proposed [14-16]. We focus on the platforms utilising SDN. The early efforts for employing SDN to support policies to manage wireless sensor networks include Flow-Sensor [17], Sensor OpenFlow [18], and Software-Defined Wireless Network (SDWN) [19].

Moreover, Yiakoumis et al., [9] reported a home network slicing mechanism, which allows multiple service providers to share a common infrastructure in a smart home. However, many of these efforts are isolated to specific application domains. Yue *et al*. suggested the concept of IoT community [15]. Users with the same interests construct a community to facilitate data maintenance, retrieval, and distribution. However, the suggested architecture is neither open to users nor convenient for introducing new IoT applications and services. One recent effort is to design a software-defined approach for IoT environments to dynamically achieve differentiated quality levels to different IoT tasks [20]. Still, here the focus was on developing a layered SDN controller in IoT setting and flow QoS performance. The developers of one M2M [21] made efforts on horizontal IoT through standardising IoT protocols and data models. It addresses the need for a standard M2M service layer that can be readily embedded within various hardware and software. We share the same idea with oneM2M by using a shared service layer.

In [22, 23], Qin *et al*., enhanced the idea of Multi-network controller architecture for heterogeneous IoT network based on SDN controller for a multi-network environment and another cellular network at the campus level and evaluated the performance by measuring delay, Jitter and Throughput. Wu *et al*., in [24] present UbiFlow framework, which provides the integration of the SDN and the IoT. UbiFlow proposed an efficient flow control and mobility management in urban multi-networks using SDN distributed controllers. In UbiFlow architecture, IoT network is partitioned into small network chunks/cluster in which each partition is controlled by a physically distributed SDN controller. The IoT devices in each partition may be connected to the different access point for different data requests. These distributed controllers coordinate to provide flow scheduling, mobility management, optimised access point selection in a consistent, reliable and scalable control order, and provide fault tolerance and load balancing for multi-network IoT.

In this paper, we propose an IoT architecture based on SDN technologies for the horizontal IoT services. This architecture enables sharing various resources, including devices, data, and software, among various applications at different levels. Data interoperability can be supported interiorly in the network, and new services and applications in different domains can be supported rapidly and efficiently. We also present a prototype of the proposed architecture using OpenFlow and Open vSwitch. The architecture tends to describe a general model to integrate SDN and IoT by introducing an SDN controller in the current IoT architecture that will serve as the CP so that solutions to the current scalability challenges associated with the traditional networks are achieved. The architecture will improve the scalability of IoT networks using the advantages of SDN. The overview of SDN based IoT (SDIoT) integration architecture is shown in Figure 5.

4. PERFORMANCE OF THE SDIOT NETWORK ARCHITECTURE

This section presents the simulation results on the performance evaluation of the SDIoT network architecture. To study the advantages of the architecture, we compared the SDIoT architecture with existing IoT network based on the traditional network architecture. In the traditional network, devices include two fundamental elements; the DP and CP which are coupled in a device. Traditional networks include the various number of devices such as router and switches among different vendors which are responsible for data transfer through the network. While in the SDN architecture, these DP and CP are decoupled. The CP being the control logic is implemented in a logically centralised controller. The traditional network was emulated in GNS3. GNS3 is a network emulating software used to design networks in lab environments for testing and study purpose.

VOL. 20, NO. 1, 2021, 65-75
www.elektrika.utm.my
ISSN 0128-4428

ELEKTRIKA
Journal of Electrical Engineering

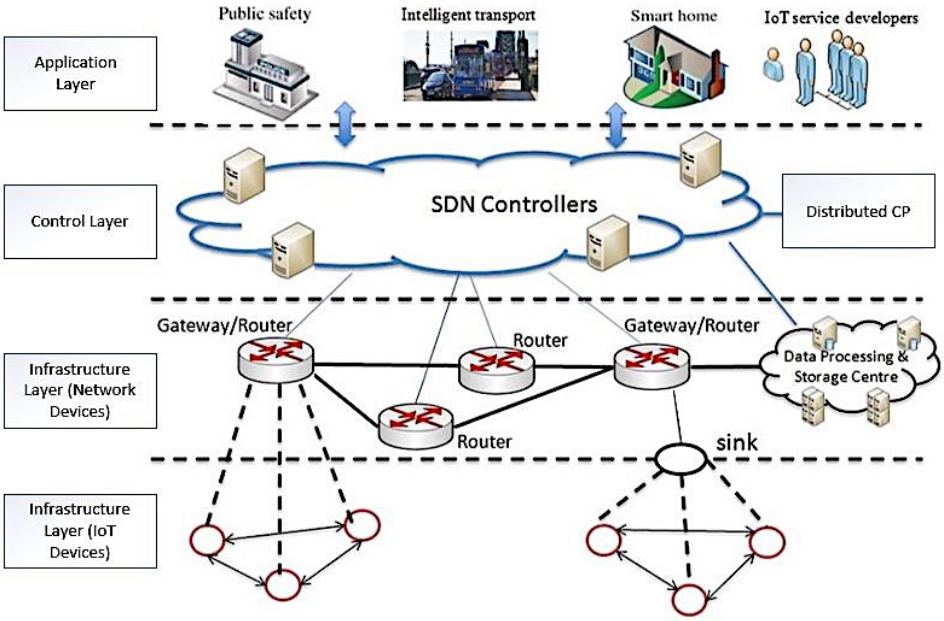

Figure 5. Overview of the SDIoT integrated architecture

On the other hand, the SDN topology was created in the Mininet network emulator. Mininet provides a comprehensive environment for prototyping SDNs. It consists of a controller, for making routing decisions, an OpenFlow enabled switch for forwarding the packets based on the controllers routing decisions and end Hosts. Once the network is up and running, the latency of the packets can be determined by conducting a ping test between the hosts on the network. On arrival of a packet to a switch, an SDN switch undergoes the process shown in Figure 6. From Figure 6, the external packet arrives at the switch and the switch is connected to a controller. The switch block and controller block are modelled as a queue. There are three essential phases an SDN model must capture. Phase (1), the first packet of a flow arrives at the switch, and there are no matching Flow table Entries (FTE) for the packet. Phase (2), the packet without a matching flow entry is forwarded to the controller or a packet with the matching FTE is serviced by the switch and forwarded to the destination. Finally, Phase (3), the controller feeds the forwarding information back to the switch and updates the flow table in the switch. Note that the controller is not a substantial part of the switch, but plays a critical role in switch forwarding and network performance and cannot be neglected.

4.1 Custom Topology

One of the benefits of the SDN is that the forwarding device is free from any computation related tasks. SDN supports two types of controller models: centralised and distributed CP model. Each model has its different speciality to be considered based on the infrastructure and requirements. Moreover, the central controller has certain restrictions as limitations in terms of scalability and reliability issues. An alternate approach is to make CP in a distributed fashion to execute applications of a centralised controller. In a network with more than one controller, it is difficult but possible to make the control action to be centrally controlled. Such an architecture is also known as logically centralised [25-28] architecture. An illustration of the custom network model of SDN used in this work is shown in Figure 7.

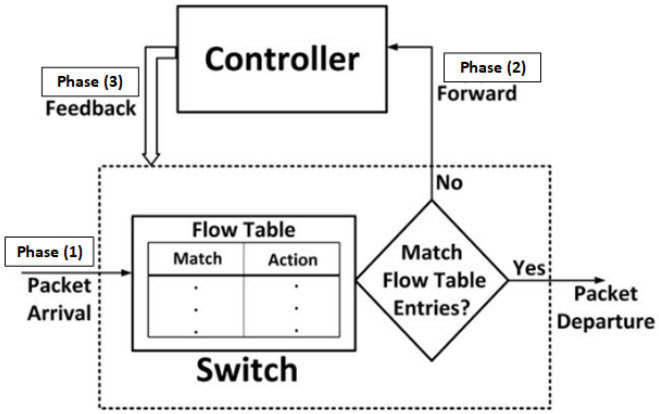

Figure 6. Block diagram of a data flow SD-based network

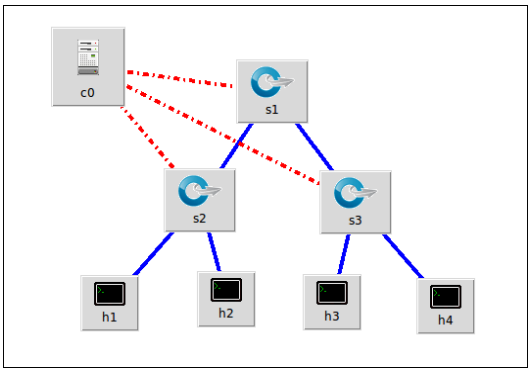

Figure 7. Custom network model of SDN as emulated in Mininet

Also, the number of controllers needed and their respective location must be taken into consideration concerning the QoS constraints. There is also the hybrid controller model which combines the benefits of both these models. The controller must be able to visualise a load of a switch globally across the routing path. All the routing-related information such as statistics, errors, and faults are collected and communicated to the controller with the help of OpenFlow. In this work, the distributed control model has been adopted to proof the concept. The distributed control model entails using a multi-controller to manage and supervise the entire network. In this model, the intelligence does not rely on a single decision unit, unlike the centralised model. From Figure 7, every flow decision in the network is taken by the controller. This decision is based on the initial packet of every flow that is sent to the controller. The flow table is updated after creating a path on a hop-by-hop basis. With the increase in network traffic, the numbers of flow tables states are increased [25, 29]**.** As shown in Figure 4.2, the source nodes send a packet to the forwarding device, and then the forwarding device sends it to the controller. The forwarding device receives the packet from the source and sends it to the controller for processing. The controller only processes the first packet of flow from any forwarding device and update the flow table accordingly. Depending on this flow table entry, all other packets of the same source get similar treatment to reach the desired destination. This flow rule is set by the controller, and the forwarding device cannot update them without any instructions from the controller.

4.2 Performance Metrics

To evaluate the performance of the SDIoT architecture over the traditional networking architecture, the system has been emulated, and the results are analysed. The traditional way of measuring network performance is network latency, Jitter and Throughput. The emulation for the SDN environment was done using Mininet. On the other hand, GNS3 emulator was used for emulating the traditional network environment. It's worth noting that the performance evaluation is independent of the software used. It mainly depends on the hardware components like the number of Processors, Memory, Hard disk and a network connection between the nodes. Details of the performance evaluation metrics are given as follows.

a. **Network Latency**: Network latency is the term used to indicate any kind of delay that happens in data communication over a network. Latency is generally defined as the time it takes for a source to send a packet of data to a receiver, as illustrated in the expressions below.

$$L = P_d + S_d \tag{1}$$

$$P_d = \frac{D}{S} \tag{2}$$

$$S_d = \frac{P_S}{T_R} \tag{3}$$

Where L is network latency, Pd is propagation delay, and Sd is the serialisation delay in Equation 4.6 and D is the distance between links in meters and S is the speed in meters per second. For wireless communication, we have used the speed of light in Equation 2 and PS is the Packet size (bits)and TR is the Transmission rate (bps)in Equation 3. High network latency creates bottlenecks in any network communication. It prevents the data from taking full advantage of the network pipe and effectively decreases the communication bandwidth. The impact of latency on network bandwidth can be temporary or persistent based on the source of the delays.

b. **Jitter:** Jitter is defined as the amount of variation time of received packets in delay, latency or response. Jitter can be approximated as the difference between the maximum packet latency and minimum packet latency over a given period. Let Tj represent the delay experienced by the jth packet going through a queue. The difference of transit time between two consecutive packets of the tagged flow can be written as;

$$J_i = T_{i+1} - T_i \tag{4}$$

Equation 4 can be positive or negative. Where N is the number of samples, the average end-to-end delay jitter is then given by the expected absolute value of the random variable in Equation 5

$$J_i = \frac{[|\, T_{i+1} - T_i \,|]}{Nth - 1} \tag{5}$$

From Equation 4, to measure Jitter, we take the difference between samples, then divide by the (Nth 1) number of samples. If there is a change in network conditions (for instance, there is a sudden burst on the network and queues start to build on an interface), then this Jitter will increase. The transit time between two consecutive packets will not be the same anymore: the second packet will have to go through a long queue, spending more time, and generating positive Jitter. Once this burst is over, the queue will progressively reduce, reversing the situation. Out of two consecutive packets, the second one will spend less time in the queues, and will, therefore, generate negative Jitter.

c. **Throughput**: Throughput is the actual rate that information is transferred in a computer network; Throughput is defined as the actual number of bits that flows through a network connection in a given period. In mobile networks, the end-user throughput is the amount of information received in bits /second. Throughput is measured at the physical layer, data link layer or even at the application layer. In a network, often cell throughput is calculated, which is the throughput of all simultaneous users in the cell. The throughput can be affected by many factors such as (i) Network congestion due to heavy network usage. (ii) Too many users are accessing the same server. (iii) Low bandwidth allocation between network devices. (iv) Medium loss of a computer network. (v) Resources (CPU, RAM) of network devices. Throughput [220] is calculated as requests/unit of time. The time is calculated from the start of the first sample to the end of the last sample. This includes any intervals between

samples, as it is supposed to represent the load on the server. The throughput is always less than or equal to bandwidth but can never exceed the bandwidth, and it is given by the expression in Equation 6:

$$T_R = \frac{M_S}{R_T} * \frac{1.2}{p^{0.5}} \quad (6)$$

Where TR is the network throughput, p is the packet loss, MS is the packet size, and RT is the round-trip time (RTT). Sometimes the term Round trip latency or RTT is also used to define latency. This is the same as ping time.

4.3 Results and Discussions

For the performance evaluation network architectures, the performance metrics for comparison are in terms of latency, Jitter and Throughput of the two network architectures. The simulation parameters and results of the evaluation of the SDIoT and the Traditional IoT network architecture are presented in Table 1. The subsequent section presents discussions of the results for simulation performance based on the latency, Jitter and Throughput of the two network architectures.

a. Average Network Latency

Latency has been defined as the term used to indicate any kind of delay that happens in data communication over a network. Latency is generally defined as the time it takes for a source to send a packet of data to a receiver. This section presents the results for the average network latency performance of the SDIoT in comparison with the traditional network IoT. A useful tool in measuring a connection's latency is the "ping" tool. Ping sends a packet to the destination, and waits for a response, measuring the round-trip time and calculating connection latency. The results of several ping commands were averaged over time. The latency results were tested multiple times (not just a single series of pings), and at different times and different custom network topology having the specified number of nodes as detailed in Table 1. The result of the average latency of the traditional IoT network in comparison with the SDIoT network is illustrated in Figure 8.

Table 1. Simulation Parameters and Results of the evaluation of the SDIoT and the Traditional IoT Network Architecture.

Simulation Parameters	Traditional	SDN
Total number of host nodes	70	70
Total number of packets	450	450
ICMP Packet Size	64 bytes	64 bytes
Simulation Platform	GNS3	Mininet
Simulation Instances	100	100
Network Topology	Custom Topology	
Number of routers/controllers	1	
Performance Metrics	Latency, Jitter and Throughput	

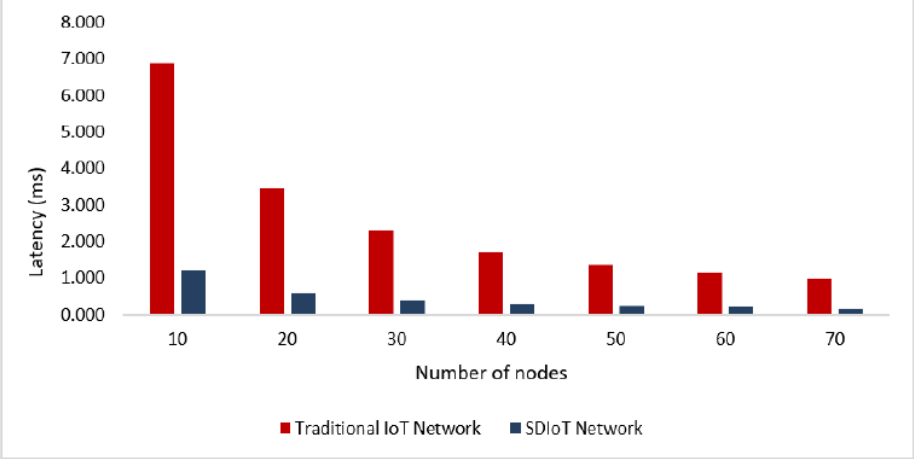

Figure 8. Average Latency of the Traditional and SDIoT Network.

From individual results observed for the traditional network, the latency results of the packets sent from several hosts are either same or more for the traditional network architecture. When the first packet arrives at the switch, the switch checks its MAC table for the corresponding destination address in the MAC table. Since this is the first packet and there is no entry for the address, the switch forwards the packet to the router for the routing decisions. The router makes the routing decisions and sends the routing decision entry to switch. The switch forwards the packet accordingly and flushes the entry. This process is repeated for all packets that arrive at the switch. Therefore, this account for the high latency experienced by all the packets. While in SDN, the time taken by the first packets is more compared to the consecutive packets.

The reason for the high latency of the first packet in comparison to other packets is that the routing decision happens only for the first packet. Once the controller updates itself with the flow rule for the first packet, the switch buffers the flow rule in its flow table after thirty seconds [8, 30]. SDN switch buffers the flow entry inserted by the controller into the flow table of the switch. This

eliminates the necessity of contacting the controller for routing decision for every packet, thus reducing the latency of the consecutive packets after the first packet. The consecutive packets are forwarded by the switch without contacting the controller for the routing decision. After thirty seconds, the buffer is timed out, and the flow table is cleared, and the same procedure repeats. There are different scenarios to get an idea of how network load can and will affect latencies.

The high network latency experienced in the traditional network set-up can be attributed to a range of issues. Still, generally, it comes down to the state of routers and the distance between your network devices. These include (i) propagation (the amount of time it takes for a packet to travel from one source to another). (ii) routers take time to analyse the header information of a packet as well as, in some cases, add additional information. Each hop a packet takes from router to router increases the latency time. The more routers a packet has to travel through the more latency there is because each router has to process the packet. In this simulation, the distance that a packet travelled had a significant influence on the amount of latency within a network. When packets travel across a network to their destination, they rarely travel to the node in a straight line. As such, the amount of latency is dependent on the route that the packet takes.

In SDN domain, latency is an important measurement to express the expected responsiveness of a network connection. From the result of the average latency of the traditional IoT network in comparison with the SDIoT network illustrated Figure 8, the SDIoT network architecture has a significantly better average latency performance when compared to the traditional IoT network architecture for all number of nodes considered. The average latency percentage improvement from the traditional IoT network to the SDIoT for all the nodes is 574%. In SDN, services like loading a web page are less sensitive to latency than others, such as real-time gaming or VoIP calling, which results in a noticeable delay in both gameplay and conversation, respectively. Hence, the IoT network built on SDN architecture will perform better when compared to traditional network architecture.

b. Average Network Jitter

Jitter has been defined as the time variation in latency/response time in milliseconds. Jitter depends on the packet routes and is caused by multiplexing several flows in the node queues. There are several definitions of Jitter that try to capture the delay variation of packets. In this work, we adopt the IETF [31] definition of Jitter. The IETF definition is based on the transit delay between the entry and the exit nodes. The results were computed using the parameters in Table 1 and the equation for jitter measurement, as explained in [31]. The result of the average Jitter of the traditional IoT network in comparison with the SDIoT network is illustrated in Figure 7.

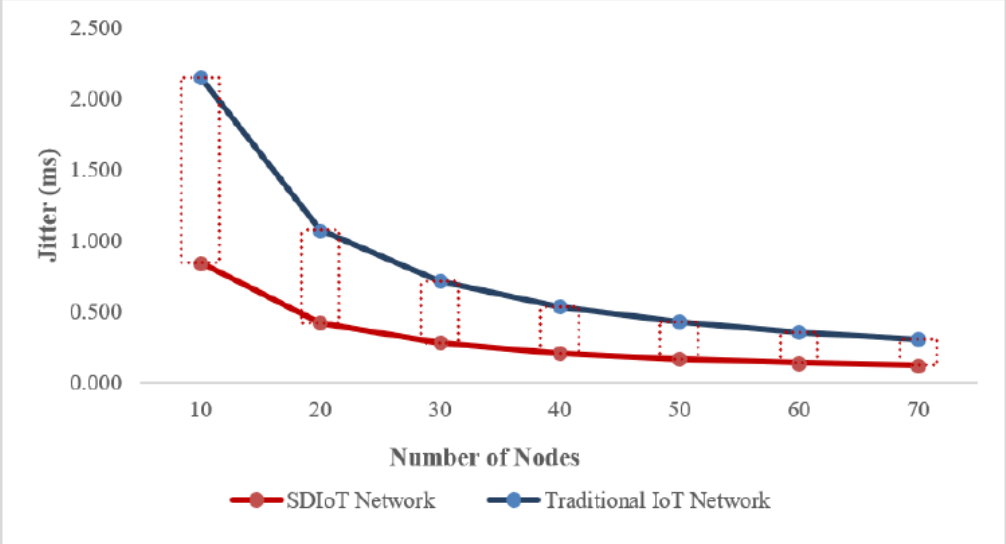

Figure 9. Average Jitter of the Traditional and SDIoT Network.

There are several ways to measure Jitter. The approach adopted in work for the measurement of Jitter involves comparing send and receive timestamps on pairs of packets transmitted as such: (a) transmit and receive time of the first packet in the pair and (b) transmit and receive time of the second packet in the pair. On observing Figure 9, the average Jitter for the traditional network is higher when compared with SDIoT network architecture. This high Jitter makes the service unusable by negatively impacting service quality. This is not acceptable, especially for a large-scale SDIoT network where packets will travel through multi-hops. For the SDN, the Jitter is consistently low for every packet after the first packet. This is the direct opposite of the situation in the traditional network set-up.

The Jitter is usually introduced by network devices themselves and can be slightly more complicated to measure than latency. Packets being buffered, queued, and switched around a network all introduce small delays that tend to add up, and are measured as the Jitter. Estimating and controlling the delay variation is essential for the operator to avoid both a buffer overflow, which causes packet losses, and buffer underflow when the application does not receive packets for some time.

The main challenge of the traditional architecture will be in the integration and support of a wide variety of applications and services combining voice, data, streaming, and Video on Demand in IoT. The different media types exchanged by these applications have different requirements in terms of Jitter. The popularity of these applications has highlighted the limitations of the traditional network infrastructure for IoT. For real-time and interactive applications, delay jitter remains one of the essential parameters of QoS. For most of the applications, the variation in the arrival time of packets at the terminal must be compensated by using a playback buffer to provide a regular packet stream to the application.

Comparatively, from the results shown in Figure 9, the traditional network architecture cannot support the jitter requirement of the IoT network. The SDN, on the other hand, offers a better performance in terms of network jitter to the traditional network architecture. The average jitter percentage improvement from the traditional IoT network to the SDIoT for all the nodes is over 600%. This makes SDN ideal for the deployment of IoT network.

c. Network Throughput

Although it's essential to measure latency and Jitter, it is also essential to know the throughput performance of the network because it gives information about how the network behaves under different circumstances. Network throughput is the term used to refer to the quantity of data being sent that a system can process within a specific period. Throughput in this work has been measured in bits per second (bps). Throughput is an excellent way to measure the performance of the network connection because it tells you how many messages are arriving at their destination successfully. If the majority of messages are delivered successfully, then throughput will be considered high. In contrast, a low rate of successful delivery will result in lower throughput. The throughput performance of the SDIoT and the traditional network is shown in Figure 10.

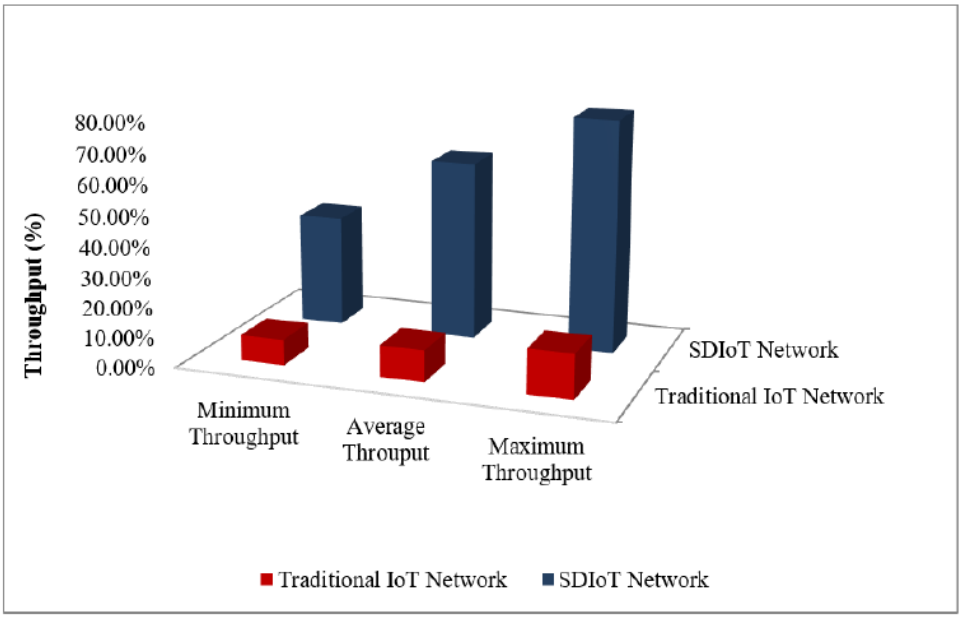

Figure 10: The Throughput of the Traditional and SDIoT Network.

From the results of Figure 10, the traditional network architecture performs poorly when compared with this SDN architecture for all cases. The performance of IoT devices rely on successful packet delivery to communicate with each other, so if packets aren't reaching their destination, the end result is going to be poor service quality. Generally, poor network throughput can be caused by several factors. One of the main factors is poor hardware performance. If devices like routers are experiencing performance degradation, faults, or are simply outdated, then you can end up with low throughput. Likewise, if networks are congested with lots of traffic, then packet loss will occur. Packet loss is where packets are lost in transit. Low network throughput is often caused when packets are lost in transit. In this case, the later and the architecture of the traditional network contributed to the low throughput experienced.

5. CONCLUSION

In this paper, we have presented an SDN controller design in IoT networks whose central, novel feature is the layered architecture that enables flexible, effective, and efficient management on task, flow, network, and resources. From the results presented for the latency, delay and throughput performance of the SDIoT network, it can be concluded that the SDIoT improves network efficiency by reducing network overheads generated from frequent communication attempt between the CP and DP every time a packet is received. Thus, accounting for the higher throughput observed from the result. The study has been conducted with the use of one CP architecture. This may not guarantee optimal results and also represent a single point of failure. These challenges, among other issues, are related to the scalability of the CP. When the network scales up in the number of nodes, switches, services or traffic, it can drastically increment the load on the

controllers which can become a potential bottleneck to the network resources such as bandwidth, memory and processor of controllers. We are currently in the process of integrating this layered controller design with a new controller placement algorithm. The algorithm has been designed to find the number of controllers and their placement in any kind of network

REFERENCES

[1] R. Minerva, A. Biru, and D. Rotondi, "Towards a definition of the Internet of Things (IoT)," *IEEE Internet Initiative,* vol. 1, pp. 1-86, 2015.

[2] C. Associati, "The evolution of internet of things," *Focus. Milão, fev,* 2011.

[3] J. Chase, "The evolution of the internet of things," *Texas Instruments,* p. 1, 2013.

[4] M. Fischer, S. Lyon, and D. Zeitlyn, "The Internet and the future of social science research," in *The Sage handbook of online research methods.*, N. Fielding, R. M. Lee, and G. Blank, Eds. London: Sage, 2008, pp. 519-536.

[5] A. Hakiri, P. Berthou, A. Gokhale, and S. Abdellatif, "Publish/subscribe-enabled software defined networking for efficient and scalable IoT communications," *IEEE communications magazine,* vol. 53, no. 9, pp. 48-54, 2015.

[6] H. Farhady, H. Lee, and A. Nakao, "Software-defined networking: A survey," *Computer Networks,* vol. 81, pp. 79-95, 2015.

[7] B. Raghavan, M. Casado, T. Koponen, S. Ratnasamy, A. Ghodsi, and S. Shenker, "Software-defined internet architecture: decoupling architecture from infrastructure," in *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*, 2012, pp. 43-48: ACM.

[8] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE,* vol. 103, no. 1, pp. 14-76, 2015.

[9] S. Singh and N. Singh, "Internet of Things (IoT): Security challenges, business opportunities & reference architecture for E-commerce," in *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)*, 2015, pp. 1577-1581: IEEE.

[10] K. K. Patel and S. M. Patel, "Internet of things-IOT: definition, characteristics, architecture, enabling technologies, application & future challenges," *International journal of engineering science and computing,* vol. 6, no. 5, 2016.

[11] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future generation computer systems,* vol. 29, no. 7, pp. 1645-1660, 2013.

[12] B. Dorsemaine, J.-P. Gaulier, J.-P. Wary, N. Kheir, and P. Urien, "Internet of things: a definition & taxonomy," in *2015 9th International Conference on Next Generation Mobile Applications, Services and Technologies*, 2015, pp. 72-77: IEEE.

[13] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks,* vol. 54, no. 15, pp. 2787-2805, 2010.

[14] M. Floeck, A. Papageorgiou, A. Schuelke, and J. Song, "Horizontal M2M platforms boost vertical industry: Effectiveness study for building energy management systems," in *Internet of Things (WF-IoT), 2014 IEEE World Forum on*, 2014, pp. 15-20: IEEE.

[15] H. Yue, L. Guo, R. Li, H. Asaeda, and Y. Fang, "DataClouds: Enabling community-based data-centric services over the Internet of Things," *IEEE Internet of Things Journal,* vol. 1, no. 5, pp. 472-482, 2014.

[16] Y. Li, X. Su, J. Riekki, T. Kanter, and R. Rahmani, "A SDN-based architecture for horizontal Internet of Things services," in *Communications (ICC), 2016 IEEE International Conference on*, 2016, pp. 1-7: IEEE.

[17] A. Mahmud and R. Rahmani, "Exploitation of OpenFlow in wireless sensor networks," in *Proceedings of 2011 International Conference on Computer Science and Network Technology*, 2011, vol. 1, pp. 594-600: IEEE.

[18] T. Luo, H.-P. Tan, and T. Q. Quek, "Sensor OpenFlow: Enabling software-defined wireless sensor networks," *IEEE Communications letters,* vol. 16, no. 11, pp. 1896-1899, 2012.

[19] S. Costanzo, L. Galluccio, G. Morabito, and S. Palazzo, "Software defined wireless networks: Unbridling SDNs," in *2012 European Workshop on Software Defined Networking*, 2012, pp. 1-6: IEEE.

[20] Z. Qin, G. Denker, C. Giannelli, P. Bellavista, and N. Venkatasubramanian, "A software defined networking architecture for the internet-of-things," in *Network Operations and Management Symposium (NOMS), 2014 IEEE*, 2014, pp. 1-9: IEEE.

[21] O. M. Alliance, "Onem2m: Standards for m2m and the internet of things," ed, 2014.

[22] Z. Qin, L. Iannario, C. Giannelli, P. Bellavista, G. Denker, and N. Venkatasubramanian, "MINA: A reflective middleware for managing dynamic multi-network environments," in *2014 IEEE Network Operations and Management Symposium (NOMS)*, 2014, pp. 1-4: IEEE.

[23] Z. Qin, G. Denker, C. Giannelli, P. Bellavista, and N. Venkatasubramanian, "A software defined networking architecture for the internet-of-things," in *2014 IEEE network operations and management symposium (NOMS)*, 2014, pp. 1-9: IEEE.

[24] D. Wu, D. I. Arkhipov, E. Asmare, Z. Qin, and J. A. McCann, "UbiFlow: Mobility management in urban-scale software defined IoT," in *Computer Communications (INFOCOM), 2015 IEEE Conference on*, 2015, pp. 208-216: IEEE.

[25] S. Rout, S. S. Patra, and B. Sahoo, "Performance Evaluation of the Controller in Software-Defined Networking," in *Computational Intelligence in Data Mining*: Springer, 2017, pp. 543-551.

[26] C.-S. Li and W. Liao, "Software defined networks," *IEEE Communications Magazine,* vol. 51, no. 2, pp. 113-113, 2013.

[27] M. Casado, T. Koponen, S. Shenker, and A. Tootoonchian, "Fabric: a retrospective on evolving

SDN," in *Proceedings of the first workshop on Hot topics in software defined networks*, 2012, pp. 85-90: ACM.

[28] Y. Kanaumi, S. Saito, and E. Kawai, "Toward large-scale programmable networks: Lessons learned through the operation and management of a wide-area openflow-based network," in *Network and Service Management (CNSM), 2010 International Conference on*, 2010, pp. 330-333: IEEE.

[29] R. Raghavendra, J. Lobo, and K.-W. Lee, "Dynamic graph query primitives for sdn-based cloudnetwork management," in *Proceedings of the first workshop on Hot topics in software defined networks*, 2012, pp. 97-102: ACM.

[30] F. Souad and M. Moughit, "Evaluation of MTCP over POX Controller."

[31] C. Demichelis and P. Chimento, "IP packet delay variation metric for IP performance metrics (IPPM)," 2002.