International Journal of Future Generation Communication and Networking
Vol. 13, No. 3, (2020), pp. 2757-2767

Improving QoS using Software-Defined Networking for Smart City (SDSC)

Mohammadreza Moslehi*[1], Hossein Ebrahimpour-Komleh[2], Salman Goli-Bidgoli[3]
1- *University of Kashan, Department of Computer Engineering, Kashan, Iran.*
2- University of Kashan, Department of Computer Engineering, Kashan, Iran.
3- University of Kashan, Department of Computer Engineering, Kashan, Iran.
moslehi@acecr.ac.ir (Corresponding author), ebrahimpuor.kashanu@gmail.com, salmangoli@gmail.com

Abstract

Recently IoT, as one of the emerging and developing concepts, is rapidly evolving into other aspects such as smart cities and smart buildings, by enabling physical things and devices, including sensors, smartphones, and cameras, to interact .The exponential increase in the number of connected things has accelerated the development of new applications and services. These applications and services have new requirements, including dynamic management and control, and QoS that conventional network platforms are unable to meet. In other words, Software-Defined Networking (SDN) and Multi-access Edge Computing (MEC) can be a promising solution in IoT. SDN separates the control plane from the forwarding plane to enable more automated provisioning and policy-based management of network resources. With edge computing solutions we can enhance the data delivery ratio and processing power, decrease delays. So in this article, we propose and evaluate an SDN-based architecture for the smart city applications using the SDN and MEC concepts. Simulation results show that the proposed SDN-based architecture outperforms conventional architecture.

Keywords: *Internet of things, software-defined networking(SDN), MiniNet, QoS, OpenFlow*

1. Introduction

With the rapid growth of the internet and computer networking technologies, including IoT, we are witnessing the emergence of new concepts (e.g. smart cities). The consequence of this growth is the increase in the number of connected devices such as smartphones and the increase in users. According to Cisco annual internet report white paper, the number of connected devices will be 14.7 billion and over 70 percent of the global population will have mobile connectivity and the total number of global mobile subscribers will grow from 5.1 billion (66 percent of the population) in 2018 to 5.7 billion (71 percent of the population) by 2023 [1]. This poses major challenges in their respective networks. The most important challenges of this rapid growth are the complexity of network management and the lack of scalability. Other problems are the increase in network traffic, delay, and congestion. Smart city services and applications resource requests often requested on demand from many IoT devices and end-user devices at different locations. These applications have stringent requirements, such as low latency. Some smart city applications and services such as e-health and even security and energy are not only useful but also critical to serve at an expected time with higher QoS. With current networking technologies, those challenges and requirements cannot meet. To meet these strict requirements it is essential that networking technologies adapt to future needs by applying new concepts such as Software Defined Networking or SDN and Multi-access Edge Computing or MEC. Open Networking Foundation or ONF defined SDN as an emerging architecture and perfect for the high-bandwidth, dynamic nature of today's applications. This is because of the dynamic, manageable, cost-effective, and adaptable characteristics of SDN [2]. SDN architecture separates the network control from forwarding functions. With the separation of the network control plane and data plane or forwarding functions we can make the network control to become directly programmable and the underlying infrastructure to be abstracted for applications and network services. SDN can centrally control and manage the entire network and this can take by dynamic network programming.

[1] Corresponding Author: Mohammadreza Moslehi

A centralized SDN controller as the heart of the network can optimize network operation dynamically. Multi-access Edge Computing or MEC as described by ETS offers application developers and content providers cloud-computing capabilities and an IT service environment at the edge of the network [3]. The edge of the network is characterized by ultra-low latency and high bandwidth as well as real-time access to radio network information that can be leveraged by applications [3]. By using MEC real-time and low latency applications can serve efficiently. ETSI in collaboration with OpenFog Consortium is currently producing standards for MEC API. ETSI formed an industry specification group (ISG) that is tasked with creating MEC standards that optimize the benefits for all players.

The main goal focuses on cooperation between network operators, applications, and content providers to boost the overall user experience. The IoT environments such as smart cities consist of a massive number of heterogeneous sensors, actuators, smartphones and so on that connect together with network infrastructure, so it needs flexible architecture to dynamically manage and control devices and networks. In this paper first, we propose a simple model for smart city networks that leverage SDN and then evaluate and compare network performances with or without SDN technology. In fact, the main objective is to evaluate the performances of the network under different types of traffics and study the role of the SDN controller. The remainder of the paper is organized as follows: Section II reviews some related works. In section III a brief overview of SDN is explained. Section IV proposes the SDSC model. Section V analyzes results and the paper concluded in section VI.

2. Related Works

Muhammet et al. [4] to deal with big data problem that generated by physical objects in IoT environments, propose an IoT architecture based on Software Defined Networking (SDN). To overcome the big data problem of IoT, they evaluated the usefulness of the sensed values in the lower layer, especially in the gateway layer, instead of the application layer. Therefore the number of packets that being sent to the internet is reduced. Tryfon et al. [5] introduce an SDN solution for WSNs called CORAL-SDN. Their proposal try to solve problems that arise with integrating SDN with WSN such as an increased amount of control packets that SDN produces and impairs the low quality of radio communication. Do Sinh et al. [6] propose an SDN/NFV architecture to meet IoT requirements for deploying IoT framework. Their proposed architecture is capable of orchestrating the whole network by SDN controller applications. They studied the roles of SDN/NFV in deploying IoT services. Intidhar et al. [7] analyze and compare the performances of the SDN-based networking architecture with the traditional networking architecture. Then they use different SDN controllers deploying to three topologies and send traffic from different protocols. After measurements of some of the network's criteria, they compare performances of the underlying networks. Raül et al. [8] propose IoT-aware multilayer transport software-defined networking and edge/cloud orchestration architecture. Their architecture deploys an IoT traffic control and congestion avoidance mechanism that considers the dynamic distribution of IoT processing to the edge of the network in terms of actual network resource state. Chuan Feng Xu et al. [9] according to Traffic Classification (DDTC) for smart cities, propose a DDoS attack Defense strategy. To increase flexibility and reduce the load of SDN against DDoS attacks, they use software-defined network function virtualization or SDNFV with a mechanism for traffic classification. Chuan Lin et al. [10] to schedule the transfer of delay-sensitive traffic, propose a TE engine DTE-SDN by using SDN. DTE-SDN monitor QoS metrics such as throughput and delay of each network link by grabbing an overall view of the network in real-time that is possible by using OpenFlow protocol.

Shen Wang et al. [11] by using transfer learning and IEC 61850 standards, propose a mechanism called SSDS or smart software-defined security for one of the most important components for smart cities called vehicle-to-grid or V2G which provide novel energy storage and scheduling approach. Ji-Young Kwak et al. [12] by functions of the flexibility of control and analytics based on SDN, propose the SDN controller for intelligent inter-datacenter (inter-DC) cloud networking. Their proposal enables optimal traffic loads distribution across the distributed datacenters with the global inter-DC view of link states and flows statistics by applying the network-aware flow scheduling on a policy-driven inter-DC traffic control layer.

Djabir Abdeldjalil Chekired et al. [13] by using a decentralized cloud computing architecture based on SDN and NFV, present a pricing model that is real-time and dynamic for EVs charging and discharging service and building energy management, in order to reduce the peak loads. Zhiyong Zhang et al. [14] with a practical SDN-based network architecture for IoT propose an Optimal Control Channel (OCC) policy. In order to reduce

the performance loss caused by unstable IoT nodes, OCC selects the minimum number of nodes to establish a stable OpenFlow channel.

Albert Rego et al. [15] for managing and controlling the actions of emergency services and the evacuation plans, propose an SDN based architecture. This work focuses on SDN based networks that deploy in smart cities. Samir Kumar Tarai et al. [16] propose a well-defined strategy of Master-Equal-Slave (M-E-S) controllers combination that takes care of controller fault tolerance. This work optimizes the flow setup time for dynamically changing network conditions by focuses on the dual problem of optimal and secure controller placement in an SDN based network. Jean-Charles Grégoire [17] presents a solution to the problem of integrating networks of sensors and actuator devices for smart city applications. The author proposes architectural elements and discusses the use of the implementation of this framework from modern hybrid networking/computing technology, such as SDN/NFV. Igor Miladinovic et al. [18] for the optimal allocation of smart city applications between MEC and cloud, propose an IoT architecture based on SDN. They use machine learning methods to taking decisions.

Shirajum Munir et al. [19] concentrate on a large volume of multidimensional smart city network data challenges. They propose a distributed model for smart city network, based on the SDN IoT network to solve that challenge. Their proposal introduce a fog-based SDN controller with three modules: an intelligent agent, fog unit and virtual mesh topology. They use a reinforcement learning (RL) algorithm for IoT network that accomplishes city services. Jahidul Islam et al. [20] propose a distributed secure SDN-IoT architecture with NFV implementation for smart cities called Black SDN-IoT. Their proposal improves security, reliability, and privacy by using SDN, for both metadata and payload within each layer. With Black SDN-IoT, they tried to present a more effective approach for building clusters.

3. SDN overview

This section presents an overview of the Open Networking Foundation SDN architecture and then apply SDN architecture to very common network architecture for a smart city. IoT and smart city networks have time-sensitive and heterogeneous traffic and it is important to reduce the bandwidth consumption and other timing-related needs such as throughput, packet loss, delay, and jitter. Traditional networks are hardware-centric. In a large traditional network, it is very hard for network administrators to reconfigure network devices such as routers and switches, and hence network management and control are very complicated tasks. SDN architecture as an evolution of network technology has emerged and aimed to create networks with more flexibility and better management with lower complexity. The main idea of SDN is decoupling the control plane from the data plane. SDN enables better programmability, agility, and flexibility in the network and allows to manage network centrally that keeps a global view of the network [2]. With a centralized SDN controller, network administrators can address quickly changes in the network.

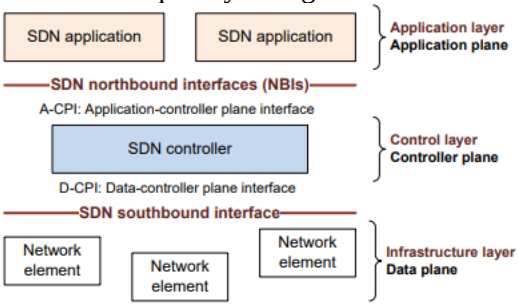

Figure 1 – Basic SDN components [2]

Based on the Open Networking Foundation definition, SDN has three planes (layers),

i) Application plane (layer)

This plane is at the top of the SDN architecture, which consists of one or more end-user applications (traffic engineering, security, visualization, etc. [21]) that interact with controller (s) to utilize an abstract view of the network for their internal decision-making process [22]. The task is to perform the definition of the requirements and behavior of the network.

ii) Controller plane (layer)

The control plane (layer) is between applications and data plane and acts as an intermediary plane (layer) between applications by northbound Interface (NBI) and the data plane (layer) by the southbound interface (SBI) [21]. The task of this plane is an interpretation of the requirements from the application plane to the data plane. It consists of SDN controllers which control data plane devices. An SDN controller has two main components called functional components and control logic. The functional components are responsible to manage controller behavior and control logic is responsible for issues instructions for network resources to furnish with the needs of SDN applications [2].

iii) Data plane (Infrastructure layer)

The Data plane is the lowest plane (layer) of the SDN architecture. This plane contains physical and virtual devices (such as routers, switches, and access points) that carry out forwarding and routing tasks. Network devices can be connected to the central SDN controller through a secure connection. SDN controller specifying rules and send out them to the network devices to create forwarding tables. The most popular control protocol used to connect data plane devices and controllers is OpenFlow [22].

4. The Proposed SDN-based architecture for Smart City network

In the IoT and smart city networks, a huge number of devices (sensors, actuators, smartphones, etc.) can connect to the network and there are many IoT and smart city applications with different QoS requirements such as bandwidth, throughput, and delay and so on. On the other hand, these very large numbers of devices will generate massive amounts of data.

The massive amount of data is a big challenge that influences satisfying QoS requirements. With the static nature of conventional networks and the fact that IoT environments are dynamics, it is a challenging task to manage and allocate network resources with good enough QoS. Conventional layer2 and layer 3 switches do the same task for every input packet. In that devices the forwarding operations take place for each packet individually that is data plane is packet-base. In the SDN-based networks, forwarding operations take place for flows. Flows are sets of streams of packets for particular end-to-end-connections. With SDN and OpenFlow, it is possible to define forwarding behavior for each of those end-to-end connections in the switches and hence SDN switches operation called flow-based forwarding. Recently, to overcome the challenges, SDN-based techniques are being deployed to IoT and smart city networks.

In this section, we propose an SDN-based IoT architecture for the smart city use case to get better network performance in terms of bandwidth and latency as important factors of QoS metrics. With the centralized SDN controller, we suppose a very common and real scenario that consists of connectivity between end-user devices (clients) to some resources (servers) with the centralized SDN controller (Fig 2). With this scenario, we are looking to evaluate SDN technology compared to conventional network technologies to highlight SDN advantages.

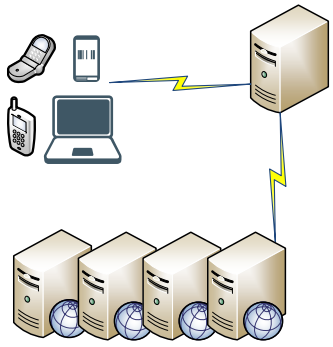

Figure 2. Centralized SDN controller for client-server connectivity.

To cover most used protocols we generate TCP and UDP traffics as general traffics for many applications (the MQTT as an M2M connectivity protocol for IoT environments uses TCP). For the central controller, we have used a python-based controller called Ryu [23]. Ryu is an open-source network operating system and is a lightweight and flexible framework for developing SDN applications. It provides software components with well-defined APIs that enable developers to easily construct new network management and control applications.

Ryu controller supports several protocols such as OpenFlow, OF-config, and Netconf for managing network devices. Ryu fully supports OpenFlow including OpenFlow versions 1.0 to 1.5. All of Ryu's codes in python available under the Apache 2.0 license[24]. OpenFlow version 1.3 was used for the interaction between the controller and the switches as forwarding plane devices. The results obtained from the performance evaluation of the SDN-based scenario will be compared to the performance evaluation with the conventional network scenario.

5. Experimental results

For the simulation environment, we use Mininet [25] as it widely used for SDN test environments. Mininet, as a network emulator is able to create different topologies with hosts, Openflow enabled switches (OVSwitches), and links. Figure 3 shows the topology of the SDN-based scenario.

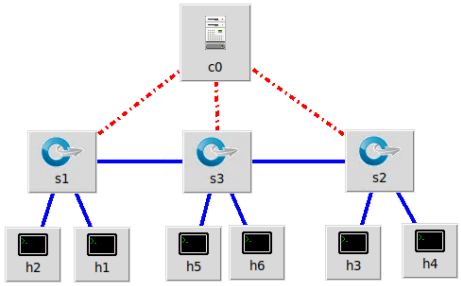

Figure 3. The topology of the SDN-based scenario

Virtual machines that used for the implementation of scenarios have been installed on a computer that uses Intel Core i7- 4510 CPU with 2.6 GHz and 12 GB of RAM. The SDN-based network scenario consists of three Openflow enabled switches that are implemented as Open VSwitche or OVS in mininet and two hosts (client and server) connected to each of them that controlled by an SDN controller (python code of figure 4.). We used a Ryu controller [23] for the central controller of the SDN-based scenario implementation.

In order to implement Ryu controller "ryu.base.app_manager.RyuApp" used and also to use OpenFlow 1.3 version, "simple_switch_13.py" was applied. The conventional network scenario is composed of six hosts and three ordinary switches (clearly without any controller). Both scenarios are implemented on a VM. After the implementation of two scenarios, the performance measurement is done by using TCP and UDP (and also MQTT protocols as a TCP protocol). After generating the traffic we measure latency and throughput as the average of ten iterations for each scenario. For two scenarios, SDN-based and conventional network architecture the same procedure can be followed and the measured parameters are recorded during the execution of the scenarios.

	Pseudocode of the scenario
1.	*Add controller: net.addController;* *Set controller name to C0;* *Set controller ip address and port number;* *Set controller as remote with TCP connection;*
2.	*Add OpenFlow switches: net.addSwitch;* *Set S1, S2 ,S3 as OVSKernelSWitches;*
3.	*Add Hosts: Net.addHost(h1, h2, h3, h4);* *Set ip address for h1, h2, h3;*
4.	*Add links:* *Net.addLink(h1, h2 to s1);* *Net.addLink(h3, h4 to s2);* *Net.addLink(h5, h6 to s3);*
5.	*Starting Controller;*
6.	*Starting network with host and OVSSwitches;*
7.	*Establish connection between hosts;*

8. *Establish client-server connection between h2 and h4;*
9. *Evaluate and record QoS parameters from the network;*

Figure 4.

For performance evaluation, the throughput and latency are two important factors. The throughput is the actual speed of data transmission on the network and latency is the time taken by a packet to reach the destination from a source. To generate traffic and measure the bandwidth and throughput, we use the Iperf tool.

Iperf is a client-server tool that is able to generate TCP and UDP traffic and measure the throughput and quality of a network link between two hosts (such as client and server). First, we run the python script of the proposed scenario and then open the terminal emulation of the hosts by running "*xterm"*. To establish a connection between client and server by Iperf we have to run Iperf with specific arguments on each client and server terminal emulation opened by xterm. For TCP protocol, on the server-side (host2), Iperf with "*–s*" argument was issued which determines the server-side of the connection. The argument *"-p*" specifies the listening port of the server. The complete issued command is "*Iperf -s -p 6633 -i 1*". Similarly, in the client-side, Iperf command with the "*-c*" argument was run which determines the client side of the connections. Arguments, "*-p*" determines the port number, *"<server_ip_address>*" determines server IP address and "*-t*" determines Iperf running times in seconds. The complete issued command is "*Iperf –c <server_IP_address> -p 6633 –t 120*".

In order to establish a UDP connection we have to use the "*-u*" argument in both client and server-side. Although Iperf supports both TCP and UDP as the transport protocol, there is a difference between TCP and UDP tests in the bandwidth and the results that the output of the Iperf returned. In the TCP test, the sender generates as much as data as enduring by the network, while in the UDP test we have to define the rate of transmission by the "*-b*" argument, otherwise it limits the rate of transmission on one megabit per second.

In the experiment, we set the transmission rate on 40 megabits per second by the "*-b*" argument. By running the Iperf on the server-side (host-2), the server starts listening to a port that specified on the command. and also by running Iperf on the client-side (host-4) the client generates traffic on that port with determined parameters (specific transmission rate, transmission duration, and TCP window size). With the ICMP protocol by using the ping command, we can measure the latency between two hosts.

Table 1: Configuration setting

OS	Ubuntu 18
Environment	Mininet 2.3.0d6
SDN-controller	OpenFlow 1.3
Simulation duration	120 seconds
TCP window size	85.3 Kbytes (standard)
UDP buffer size	208 Kbytes
ICMP (ping) packet size	1500 bytes (1472+20+8, data, IP header, ICMP header)
Number of Controllers	1
Number of OpenFlow switches	3
Number of hosts	6
Time (seconds) to listen for new traffic connections	1
Interpacket gap (IPG)	280 us

MQTT is an IoT or M2M data transmission protocol that runs over the TCP/IP protocol. It is a lightweight publish-subscribe messaging protocol [26]. This protocol consists of MQTT server or broker and devices that want to send data to the broker (publish operation) and devices that want to receive data from the broker (subscriber operation). We can use Mosquito broker which is widely used among many brokers that implement MQTT protocol.

Figure 5 shows the throughput comparison obtained by the SDN-based network and conventional network approach TCP flow during 120 seconds. As it observed average throughput achieved by the SDN-controller is about 34.3 Gbits/s while the average throughput achieved by the conventional network is about 33.4 Gbit/s. The results for UDP protocol are very close (as it is shown in figure 6). Figure 7 shows that the jitter is lower under the SDN-based network. It is showed that the SDN-based network achieved higher throughput under TCP and UDP protocol and also better jitter under UDP protocol.

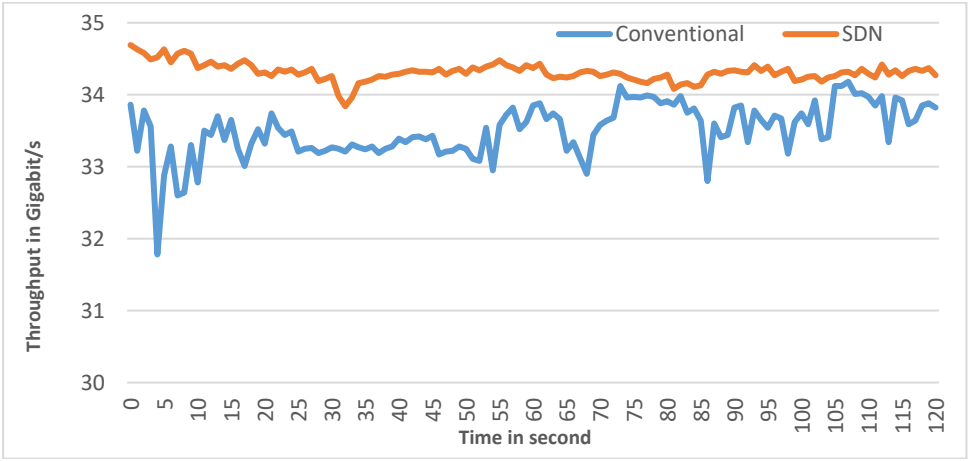

Figure 5. TCP protocol: Throughput consumption in SDN-based compare to conventional networking technology

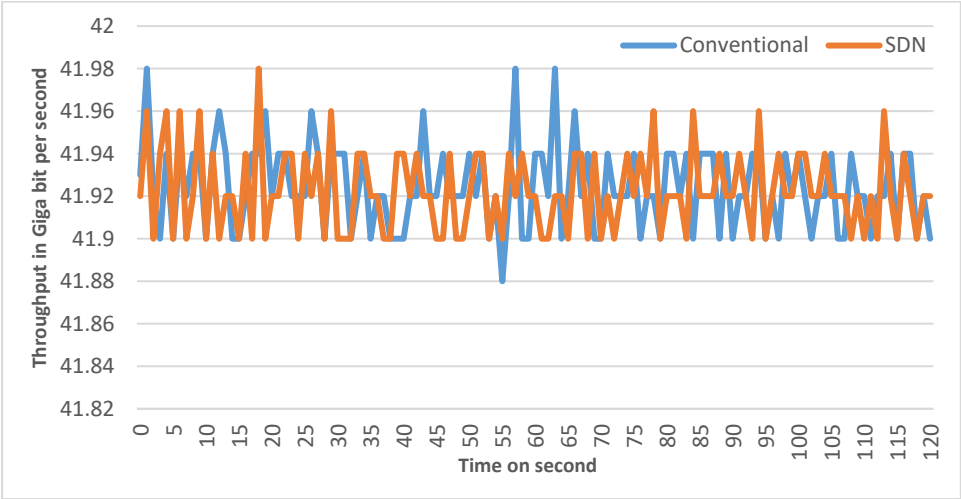

Figure 6. UDP protocol: Throughput consumption in SDN-based compare to conventional networking technology

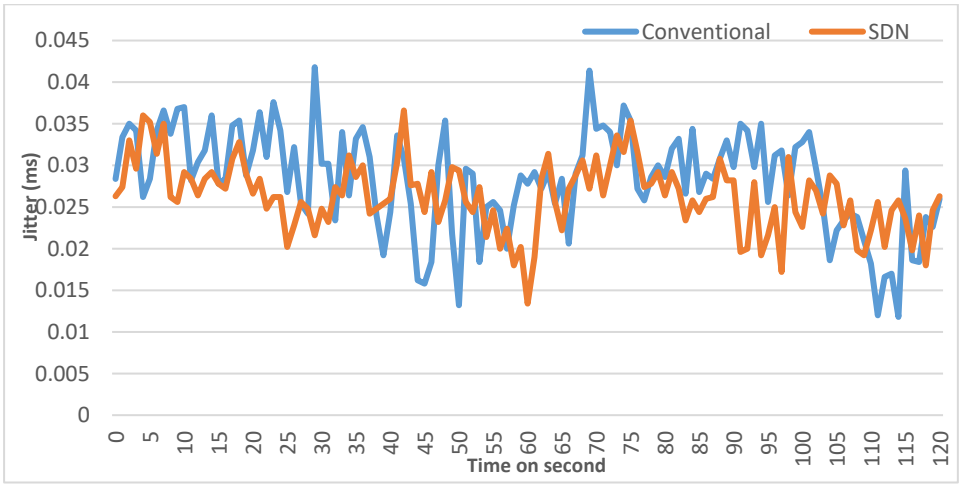

Figure 7. Jitter comparison between SDN-based and conventional networks.

The latency comparisons between SDN-based and conventional networks shown in figure 8. The minimum, average, and maximum network latency is measured by standard ICMP packet size. It is observed that the SDN-based network has lower latency in comparison to the conventional network.

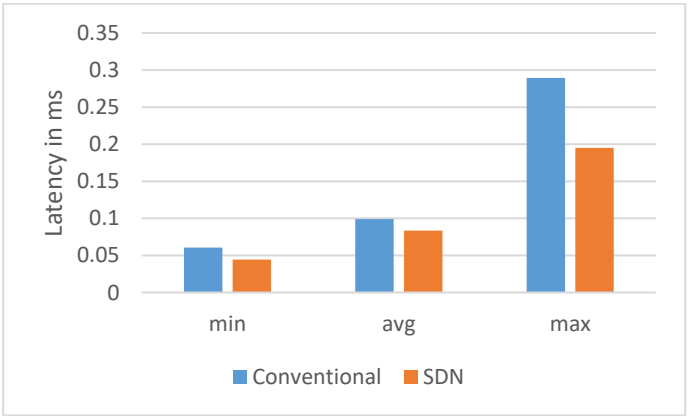

Figure 8. Latency comparison between SDN-based and conventional network

The experiment was repeated with one hundred parallel TCP connections from the client to the server in both scenarios. The results are shown in figure 9. As it observed average throughput achieved by SDN-based network is 49.62 Gbits/s and average throughput achieved by the conventional network is 43 Gbits/s.

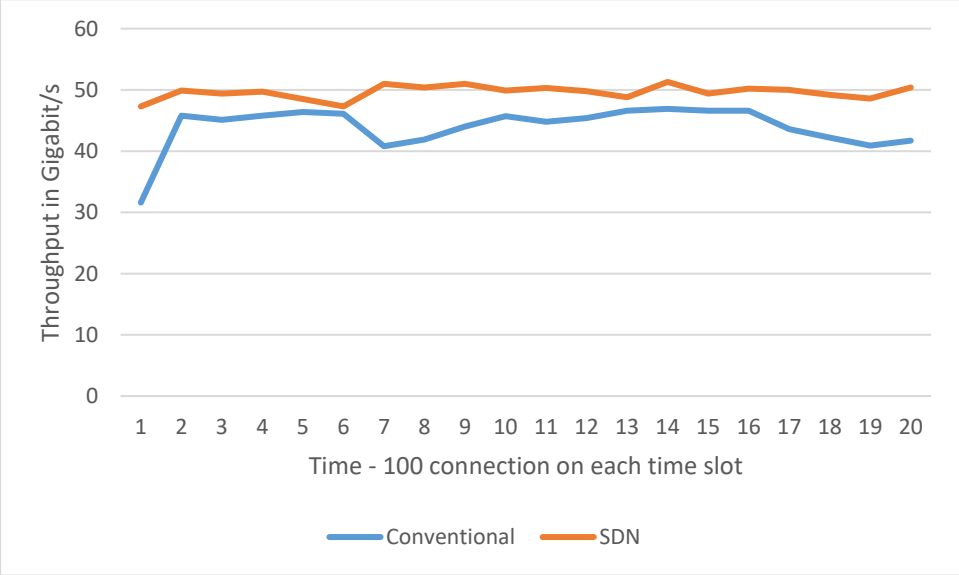

Figure 9. Throughput comparison between conventional and SDN-based network – with 100 TCP connection.

As shown in figure 10, the average volume of data transferred by the SDN-based network, during the test with 100 connection is 5.8 Gbytes in 20 seconds while the average volume of data transferred by the conventional network is 5.1 Gbytes.

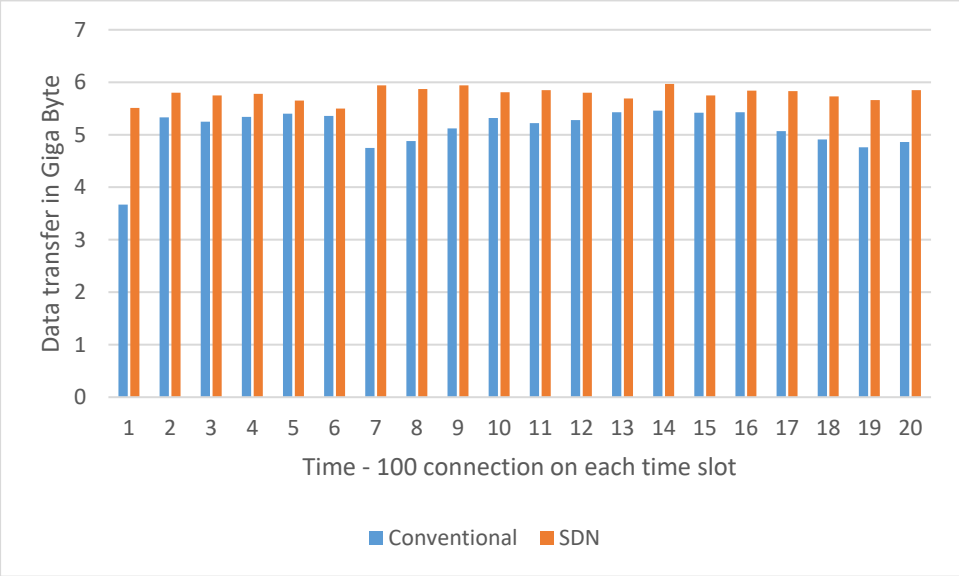

Figure 10. Data transfer comparison between SDN-based and conventional network – with 100 connection on the client

According to the graphs, it was observed that the SDN-based network can achieve better performances in comparison to conventional network technologies. It is important to note that we have evaluated the scenarios with a limited number of hosts and switches and the results achieved by SDN may be better when network scaled up. Also, we use the Ryu controller in the SDN scenario, but there are many controllers that are needed to study and compare.

6. Conclusion

SDN is an emerging technology that creates new and impactful aspects for networking and hence it is a promising technology. According to its programmability, flexibility and central management of the network enables us to do better network management, monitor, control, and improve QoS parameters.

In this paper, we have presented an SDN-Based network architecture for IoT and Smart city environments. Also, we have tried to study and analysis of the effect of using the SDN approach to the networks by comparing performances of the SDN and conventional approaches. The experiments for the evaluations of throughput, latency, and jitter were done in two scenarios, SDN-based network architecture and conventional network architecture.

TCP and UDP flows generated and performances of each protocol were obtained. In addition, 100 parallel TCP connections generated and performances of the network were obtained. With regard to the performances obtained by the tests, we can conclude that the SDN is achieved better performance in throughput and latency in comparison to the conventional networks. The experimentation has limited to the hardware characteristics such as CPU, RAM and etc, and clearly, by improving some of the limitations, larger networks with different configurations can be taken into consideration. Also, in the SDN-based scenario use of various protocols for the central controller may achieve different results that need to study and analyze.

References

[1] "Cisco Annual Internet Report - Cisco Annual Internet Report (2018–2023) White Paper - Cisco." [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html. [Accessed: 23-Feb-2020].

[2] "Software-Defined Networking (SDN) Definition - Open Networking Foundation." [Online]. Available: https://www.opennetworking.org/sdn-definition/?nab=1&utm_referrer=https%3A%2F%2Fsdn.itrc.ac.ir%2F%3Fq%3Dfa%2Fcontent%2Fsdn. [Accessed: 08-Jan-2020].

[3] "ETSI - Multi-access Edge Computing - Standards for MEC." [Online]. Available: https://www.etsi.org/technologies/multi-access-edge-computing. [Accessed: 28-Feb-2020].

[4] M. T. Kakiz, E. Öztürk, and T. Çavdar, "A novel SDN-based IoT architecture for big data," in *IDAP 2017 - International Artificial Intelligence and Data Processing Symposium*, 2017.

[5] T. Theodorou and L. Mamatas, "CORAL-SDN: A software-defined networking solution for the internet of things," in *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks, NFV-SDN 2017*, 2017, vol. 2017-January, pp. 1–2.

[6] D. Sinh, L. V. Le, B. S. P. Lin, and L. P. Tung, "SDN/NFV - A new approach of deploying network infrastructure for IoT," in *2018 27th Wireless and Optical Communication Conference, WOCC 2018*, 2018, pp. 1–5.

[7] I. Bedhief, M. Kassar, and T. Aguili, "From Evaluating to Enabling SDN for the Internet of Things," in *Proceedings of IEEE/ACS International Conference on Computer Systems and Applications, AICCSA*, 2019, vol. 2018-November.

[8] R. Muñoz *et al.*, "Integration of IoT, Transport SDN, and Edge/Cloud Computing for Dynamic Distribution of IoT Analytics and Efficient Use of Network Resources," *J. Light. Technol.*, vol. 36, no. 7, pp. 1420–1428, Apr. 2018.

[9] C. Xu, H. Lin, Y. Wu, X. Guo, and W. Lin, "An SDNFV-Based DDoS Defense Technology for Smart Cities," *IEEE Access*, vol. 7, pp. 137856–137874, 2019.

[10] C. Lin, Y. Bi, H. Zhao, Z. Liu, S. Jia, and J. Zhu, "DTE-SDN: A Dynamic Traffic Engineering Engine for Delay-Sensitive Transfer," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 5240–5253, Dec. 2018.

[11] S. Wang, J. Wu, S. Zhang, and K. Wang, "SSDS: A smart software-defined security mechanism for vehicle-to-grid using transfer learning," *IEEE Access*, vol. 6, pp. 63967–63975, Sep. 2018.

[12] J. Y. Kwak, C. Cho, Y. Shin, and S. Yang, "IntelliTC: Intelligent inter-DC traffic controller for the Internet of everything service based on fog computing," *IET Commun.*, vol. 14, no. 2, pp. 193–205, Jan. 2020.

[13] D. A. Chekired, L. Khoukhi, and H. T. Mouftah, "Decentralized Cloud-SDN Architecture in Smart Grid: A Dynamic Pricing Model," *IEEE Trans. Ind. Informatics*, vol. 14, no. 3, pp. 1220–1231, Mar. 2018.

[14] Z. Zhang, R. Wang, X. Cai, and Z. Jia, "An SDN-based Network Architecture for Internet of Things," in *Proceedings - 20th International Conference on High Performance Computing and Communications, 16th International Conference on Smart City and 4th International Conference on Data Science and Systems, HPCC/SmartCity/DSS 2018*, 2019, pp. 980–985.

[15] A. Rego, L. Garcia, S. Sendra, and J. Lloret, "Software defined networks for traffic management in emergency situations," in *2018 5th International Conference on Software Defined Systems, SDS 2018*, 2018, pp. 45–51.

[16] S. K. Tarai and S. Shailendra, "Optimal and secure controller placement in SDN based smart city network," in *International Conference on Information Networking*, 2019, vol. 2019-January, pp. 254–261.

[17] J. C. Gregoire, "A data flow architecture for smart city applications," in *21st Conference on Innovation in Clouds, Internet and Networks, ICIN 2018*, 2018, pp. 1–5.
[18] I. Miladinovic, S. Schefer-Wenzl, and H. Hirner, "IoT Architecture for Smart Cities Leveraging Machine Learning and SDN," in *27th Telecommunications Forum, TELFOR 2019*, 2019.
[19] M. S. Munir, S. F. Abedin, M. G. R. Alam, N. H. Tran, and C. S. Hong, "Intelligent service fulfillment for software defined networks in smart city," in *International Conference on Information Networking*, 2018, vol. 2018-January, pp. 516–521.
[20] M. J. Islam, M. Mahin, S. Roy, B. C. Debnath, and A. Khatun, "DistBlackNet: A Distributed Secure Black SDN-IoT Architecture with NFV Implementation for Smart Cities," in *2nd International Conference on Electrical, Computer and Communication Engineering, ECCE 2019*, 2019.
[21] R. Masoudi and A. Ghaffari, "Software defined networks: A survey," *Journal of Network and Computer Applications*, vol. 67. Academic Press, pp. 1–25, 01-May-2016.
[22] M. Karakus and A. Durresi, "A survey: Control plane scalability issues and approaches in Software-Defined Networking (SDN)," *Computer Networks*, vol. 112. Elsevier B.V., pp. 279–293, 15-Jan-2017.
[23] "What is Ryu Controller? — SDxCentral .com." [Online]. Available: https://www.sdxcentral.com/networking/sdn/definitions/what-is-ryu-controller/. [Accessed: 02-Apr-2020].
[24] "Apache License, Version 2.0." [Online]. Available: https://www.apache.org/licenses/LICENSE-2.0.html. [Accessed: 16-Apr-2020].
[25] "Mininet: An Instant Virtual Network on your Laptop (or other PC) - Mininet." [Online]. Available: http://mininet.org/. [Accessed: 22-Mar-2020].
[26] "MQTT." [Online]. Available: https://mqtt.org/. [Accessed: 23-Mar-2020].