RESEARCH ARTICLE

The Downside of Software-Defined Networking in Wireless Network

Zahraa Zakariya Saleh[1], Qahhar Muhammad Qadir [2,3*]

[1]*Department of English, College of Education and Language, Lebanese French University, Erbil, Kurdistan Region- F. R. Iraq*

[2]*Department of Electrical Engineering, College of Engineering, Salahaddin University-Erbil, Erbil, Kurdistan Region- F. R. Iraq*

[3]*Department of Computer Science and Engineering, School of Science and Engineering, University of Kurdistan Hewler, Erbil, Kurdistan Region- F. R. Iraq*

***Corresponding author's email:** zahraa.saleh@lfu.edu.krd

Received: 07-05-2020 **Accepted:** 23-07-2020 **Available online:** 31-12-2020

ABSTRACT

Mobile traffic volumes have grown exponentially because of the increase in services and applications. Traditional networks are complex to manage because the forwarding, control, and management planes are all bundled together and, thus, administrators are supposed to deploy high-level policies, as each vendor has its own configuration methods. Software-Defined Networking (SDN) is considered the future paradigm of communication networks. It decouples control logic from its underlying hardware, thereby promoting logically centralized network control and making the network more programmable and easy to configure. Low-power wireless technologies are moving toward a multitenant and multiapplication Internet of Things (IoT), which requires an architecture with scalable, reliable, and configured solutions. However, employing an SDN-based centralized architecture in the environment of a low-power wireless IoT network introduces significant challenges, such as difficult-to-control traffic, unreliable links, network contention, and high associated overheads that can significantly affect the performance of the network. This paper is a contribution toward a performance evaluation for the use of SDN in wireless networking by evaluating the latency, packet drop ratio (PDR), data extraction rate (DER), and overheads. The results show that SDN adds a high percentage of overheads to the network, which is about 43% of the 57% user packets, and the DER drops when the number of mesh nodes are increased, in addition to the high loss that was observed for packets that traveled over more hops.

Keywords: SDN, CSMA, WSN, IoT

1. INTRODUCTION

Network programming started around the 1980s (Feamster et al., 2014). The emergence of megatrend increases in the domain of information and communication technologies (ICT) is increasing the challenges for future networks (Xia et al., 2015). The legacy networks involve various components (routers and switches) running on dis tributed protocols and require manual configuration, long implementation times, and difficult to manage proprietary networks, which make it difficult for the customer to choose the hardware and software. With the major evolution of Internet of Things (IoT), mobile networks will need to handle a big influx in data, massive amounts of network traffic, and new types of connected devices such as industrial machines, smart cars, wearable sensors, actuators, and smart appliances (Nikoukar et al., 2018). One of the major building blocks of IoT devices is the low power and lossy networks (LLNs), a set of interconnected embedded devices such as sensor enabled devices. LLNs have been used widely in various fields such as modern

Access this article online

DOI: 10.25079/ukhjse.v4n2y2020.pp147-156 **E-ISSN:** 2520-7792

networking, traffic monitoring, home monitoring, process monitoring, medical monitoring, and environmental monitoring. The LLNs were introduced by different standardization bodies such as the Institute of Electrical and Electronics Engineers (IEEE) 802.15.4 and the IETF 6TiSCH standards (Ghaleb et al., 2019). The IEEE 802.15.4 standards form the basis for many low-power IoT protocols such as 6LowPAN, ZigBee, and WirelessHART. The main weakness of low-power wireless mesh networks is related to the limitations of the sensor resources and the underlying communication technologies. The constrained devices are restricted by their processing power, memory capacity, speed, energy, transmission rate, high variability of lossy links, and location.

These devices, however, are expected to operate for months or years with low power consumption. SDN is a well-defined approach and a promising solution for other networking areas. However, employing an SDN-based centralized architecture in the environment of a low-power wireless IoT network introduces important challenges, such as the difficulty to control traffic, unreliability of links, network contention, and high associated overheads, which can significantly affect the performance of the network (Baddeley et al., 2018). This paper evaluates the overhead cost of SDN traffic network performance, delay, DER, and PDR. We illustrate the results by showing how SDN-based carrier-sense multiple access (CSMA) can enhance the Quality of Service (QoS) and achieve a considerable reduction in the delay. The rest of this paper is organized as follows: section 2 discusses the previous work related to the use of SDN in low-power IoT networks, whereas section 3 explains the evaluation environment. The results are presented and discussed in section 4. Finally, the conclusion is presented in section 5.

2. OVERVIEW AND RELATED WORK

2.1. SDN: the need, architecture, and deployment

SDN is embodied by a separation of the network, thus, moving the control logic from the node to the centralized controller. This brings potential benefits such as a globally improved network performance, enhanced network manage ment and configuration, and encouraged innovation. In terms of network configuration and management, one of the key objectives is to achieve the possibility of reconfiguring network devices from a single point, automatically and dynamically, through software-controlled optimization based on the network status. SDN encourages innovation by providing a sufficient testing environment with isolation, easy software implementation for new applications, and quick deployment of new applications by using a software upgrade. Another benefit of SDN is that the dynamic global control can be improved with cross-layer consideration. Specifically, SDN allows for a centralized control with a global view of the network and feedback control with the information that is exchanged among different layers in the architecture of the network (Xia et al., 2015). Moreover, SDN can be easy to maintain because new services or network upgrades can be performed without affecting the whole network.

2.1.1. SDN architecture

The Open Networking Foundation (ONF) is a non-profit industry consortium aimed at the development, standardization, and commercialization of SDN architecture elements such as OpenFlow protocols and SDN controllers. The ONF introduced an SDN reference model that consists of a 3-layer model which ranges from the infrastructure layer to the control layer, and to an application layer, all stacking over each other. The infrastructure layer consists of the physical network components (e.g., ethernet switches, routers, etc.) and this forms the data plane. The main functions of the SDN switching device model are classified into 2 categories. First, they simply are responsible for collecting and reporting the network status by storing data temporarily in local devices before forwarding these to the controllers. Second, they are responsible for processing packets based on the applied forwarding rules (Ominike et al., 2016). The control layer is the most important component in the SDN architecture. It bridges the infrastructure layer and the application layer through its 2 interfaces. The controller infrastructure interface (southbound interface), which interacts with the infrastructure layer, allows the controller access to the functions that are provided by the switching devices. The functions include collecting the network status and updating the packet forwarding rules to the switching devices at the infrastructure layer. The controller communicates with the switching devices through an OpenFlow protocol. The application-controller interface (northbound interface), which handles the transactions with the application layer, provides a

variety of service access points such as an application programming interface (API). The policies received, described in high-level languages by SDN applications and network status synchronization are utilized to build the global network view (Xia et al., 2015). At the highest layer of the SDN architecture, the application layer includes the SDN applications. SDN applications are designed to fulfill the user requirements, such as the ability to access and manage the switch devices in the data plane, seamless mobility and migration, server load balancing, and network virtualization.

2.2. SDN for wireless networks

The recent evolutions in the wireless domain with the goal of integrating SDN and IoT are discussed in a number of previous studies (El-Mougy et al., 2015; Lasso et al., 2018; Jian et al., 2017; Anadiotis et al., 2019). However, there are many fundamental issues of what SDN indicates when it comes to low-power sensor networks such as IEEE 802.15.4, which is allowed to serve key enablers for the IoT in the near future. Similar to OpenFlow, Sensor OpenFlow (Luo et al., 2012) was the first attempt at integrating SDN in Wireless Sensor Networks (WSNs). The authors introduced a customized, low-power protocol built on the legacy southbound communications for SDN rather than using OpenFlow directly because of the complexity in the implementation of the Out-Of-Band (OOB) control-plane connection model within a sensor network.

They developed an algorithm called Control Message Quenching (CMQ) for OpenFlow to reduce the SDN control overhead. In a study by De Oliveira et al. (2015), the authors of TinySDN attempted to utilize an SDN to establish a flexible solution for WSN and IoT deployment, because an SDN-based centralized controller could achieve node retasking and routing and enable a better resource sharing and management platform. They examined the TinySDN and IPv6 routing protocol for LLNs (RPL) in terms of their routing features, interoperability, and ability to support traditional networks. The study only presented solutions to RPL shortages in the context of SDN. Costanzo et al. (2012) proposed SDNWN, an architectural framework that highlights the impact of SDN in low-power WSN. They presented the concept of utilizing protocol oblivious forwarding (POF) as a key enabler for a highly flexible and programmable SDN. It was demonstrated to minimize the memory footprint and allow the flowtable to match on bytes arrays and a packet index inside the packet rather than being included in multiple flows for specific packet types. Another SDWSN that seeked to improve the traffic routing and WSN sensor programmability was implemented and tested for IEEE 802.15.4 in the study by Galluccio et al. (2015). The aim of an SDN solution for Wireless Sensor networks (SDN-WISE) is to reduce the number of packets exchanged between the SDN controller and the sensor nodes, as well as to enable sensor nodes to be programmed as Finite State Machine (FSM) for running different domains. The SDN-WISE attempts to produce APIs that allow the developers to use the programming languages of their preference when they build SDN controllers.

The prototype of SDN-WISE was developed using a real SDN controller and an Objective Modular Network Testbed in C++ (OMNET++) simulator. The aim of their system is to increase the elasticity of the network and provide realization of network programmability. Lasso et al. (2018) proposed a software-defined wireless sensor network architecture based on 6LoWPAN networks (SD-WSN6LO). Two main components were introduced in the framework, namely an SDN sensor node and SDN controller node. They demonstrated the result of power consumption for their implementation in Contiki OS, however, no details about the architecture and implementation were presented. The work of Galluccio et al. (2015) demonstrated how logical WSNs can coexist by exploiting the same set of sensor nodes and how easy it is to program the behavior of sensor nodes with a few lines of code. Their system was compared with the state-of-art SDN-WISE system in terms of reducing the number of messages exchanged between the sensors and controllers.

Furthermore, the study provided a new method of network virtualization called SDN-Visor, which allows the creation of several virtual WSNs under different controllers. The challenge of including SDN architecture with a high associated cost into low-power sensor networks is addressed in the study by Theodorou and Mamatas (2017). The authors proposed to minimize the amount of RPL control messages in SDN for an Internet Protocol version 6 (IPv6)-based IEEE 802.15.4 network through fine tuning the timer setting in RPL. The aim was to provide the scalability and management for an SDN protocol.

2.2.1. SDN controllers

The most important component in SDN is the controller, which is the cornerstone of the architecture of SDN. The main concept behind the controller is to manage the traffic in underlying network devices by using a set of instructions. A number of previous studies conducted a partial performance evaluation for controllers (Zhao et al., 2015; Rowshanrad et al., 2016; Asadollahi et al., 2017; Asadollahi et al., 2018). The performance of 5 open course controllers, namely Ryu (RYU), POX (POX), NOX (NOX), Floodlight (Floodlight), and Beacon (Beacon) was investigated in a study by Zhao et al. (2015) using optimized configurations for each of the controllers. Beacon was found to outperform the others in terms of latency and throughput by having a low latency (0.1 ms) and high throughput (1750 ms).

It also increased fairness. Rowshanrad et al. (2016) evaluated the performance of controllers such as Floodlight and OpenDaylight. They showed that OpenDaylight performed better than Floodlight for low and medium network loads in terms of latency, loss of packets, and throughput. However, Floodlight performed well with heavy network loads such as multimedia. Previous studies recognized issues with the simulation and emulation of SDN.

Asadollahi et al. (2017) introduced a linear topology to evaluate the scalability and performance of a network by emulating an Open Flow Network (OFNet) (OFNet) over the Floodlight controller. The aim was to define the performance metrics for the Floodlight controller. However, Asadollahi et al. (2018) proposed a mesh topology to evaluate the performance and scalability of a Ryu controller. They performed various experiments using the simulation tools Mininet, Ryu controller, and iPerf (iperf). The objective of the study was to test the scalability feature of the Ryu controller in the SDN environment.

3. EVALUATION ENVIRONMENT

This paper evaluated the performance of SDN for wireless communication using a simulation. A number of different types of software, packages, and tools were used for this purpose. The main components of our evaluation platform are described in detail below.

3.1. Operating systems

Linux is a full open-source, UNIX-based system with a large support community. It has immediate advantages for developers and programmers who develop their own tools, packages, and customized applications. Being an open-source system, Linux has attracted the academic community and researchers whose concerns in terms of the ability to access and have full control over the hardware and system libraries are best met by this system. In this paper, Ubuntu, a flavor of Linux 12.04 LTS (64-bit), was used as the operation system.

It was installed on a Lenovo-IdeaPad-Y510P Laptop with an Intel Core i7-4700MQ processor with 7.7 gigabytes of random access memory. In this environment, it was unnecessary to install the Contiki platform because it was included as part of the µSDN (Baddeley et al., 2018), a low overhead SDN stack, and embedded in the SDN controller for Contiki OS. However, it was necessary to install compilers such as the 20-bit mspgcc compiler (20-bit) and the precompiled MSP430-GCC version 4.7.3 (msp430). The reason for using the 20-bit mspgcc compiler was to support up to 1 MB of memory. Platforms such as Cooja and WiSMote are based on the MSP430X series central processing unit (CPU) and support more memory than the 64K address space.

3.2. Simulator

A simulator could be used as an alternative to simplify the research environment. Cooja is an open-source simulator that aids in the testing of protocols or applications on emulated motes based on operating systems such as TinyOS or Contiki OS (Dunkels et al., 2004). The main feature of the Cooja network simulator is the ability to simulate any number of platform sensor nodes (Hendrawan & Arsa, 2017). It supports a set of standards such as TR 1100, TI CC2420, Contiki RPL, IEEE 802.15.4, uIPv6 stack, and uIPv4 stack (Helkey et al., 2016). All simulations in this work were tested in Cooja using a Unit Disk Graph Medium (UDGM) distance loss. The reason for using a simulated UDGM distance loss radio environment is that it allows implementation and testing of the new directional property of nodes. A node can receive a packet from a sender only if it is within its radius, which is defined by the transmission range.

3.3. SDN framework

An SDN standard for low-power wireless networks called µSDN (Baddeley et al., 2018) was used for simulation in this study. µSDN is a lightweight SDN architecture for Contiki OS, which supports both IPv6 and interoperability with distributed routing protocols such as RPL, as well as optimizes the combination of a number of overhead reduction functions to enhance the scalability and mitigate the cost of the SDN within a low-power IoT environment.

3.4. Simulation setup

We evaluated the performance of an implemented SDN in a wireless network through simulation, presenting a use-case scenario in which the SDN can be used within low power, multihop wireless networks in order to programmatically improve the QoS and show how a CSMA-SDN can achieve significant reductions in delay. The simulations were performed on an emulated EXP5438 platform with a TI's MSP430F5438 CPU and CC2420 radio, with evaluation in the Cooja simulator for the Contiki OS environment using a UDGM distance loss model with the configuration parameters listed in Table 1.

Table 1: Cooja Simulators Parameters Setup

Cooja simulation parameters	Setting
Simulation period	1 h
Radio environment	UDGM with Distance Loss Model
Node transmission range	100 m
MAC layer	CSMA
Transmitting nodes	All
Receiving node	controller
Number of nodes	20, 30, 40
Link quality	50%, 70%, 90%
Transmission data period	60–75 s
RPL mode	Non-storing
RPL route lifetime	10 min
RPL default route lifetime	∞
µSDN flowtable lifetime	300 s
µSDN update period	180 s

A total of 30 random realizations of the SDN deployment was run. Data from the Contiki logs were collected and the characteristics of the network entries were analyzed using Matlab.

The performance metrics included the end-to-end application flow delay, PDR, DER, and ratio of network traffic. All the performance metrics are described in Table 2.

Table 2: Performance Metrics

Metrics	Description
End-to-end application delay	It determines how the SDN overhead affects the application traffic latency.
Packet Drop Ratio (PDR)	The ratio of the number of lost application packets to the total number of sent application packets.
Data Extraction Rate (DER)	The ratio of received application messages to transmitted application messages over a period of time.
Ratio of network traffic	Ratio of application traffic, and SDN traffic in µSDN.

4. RESULTS AND DISCUSSION

The performance metrics of the SDN were investigated for wireless networks in the following scenarios:

(1) End-to-End Delay: In this scenario, we measured the overheads incurred by application messages and the end-to-end latency. In this experiment, the network consisted of 30 nodes with the maximum of 6 hops to the controller, a transmission ratio (Tx) of 100%, and a reception (Rx) ratio of 90% for each mesh node. In addition, the SDN controller collected information from all the nodes every 60 seconds, which included node energy, node state, and buffer congestion. Each simulation that was ran collected data from the mesh node flowtable entities, which have a 300 second lifetime. The transmitting nodes sent data to the sink every 60 to 75 seconds. It is clearly seen that there is an increase in the delay with an increase in the number of the hops. This is obviously because of the fact that packets travel longer when increasing the number of hops and every single node along the path needs to perform a flowtable check for incoming packets, which substantially contributes to the delay. This trend can be observed in Figure 1, which shows the average of the end-to-end application flow latency vs. the number of hops.

The results of the delay in this paper is corroborated by the results of the delay in other papers (Baddeley et al., 2018).

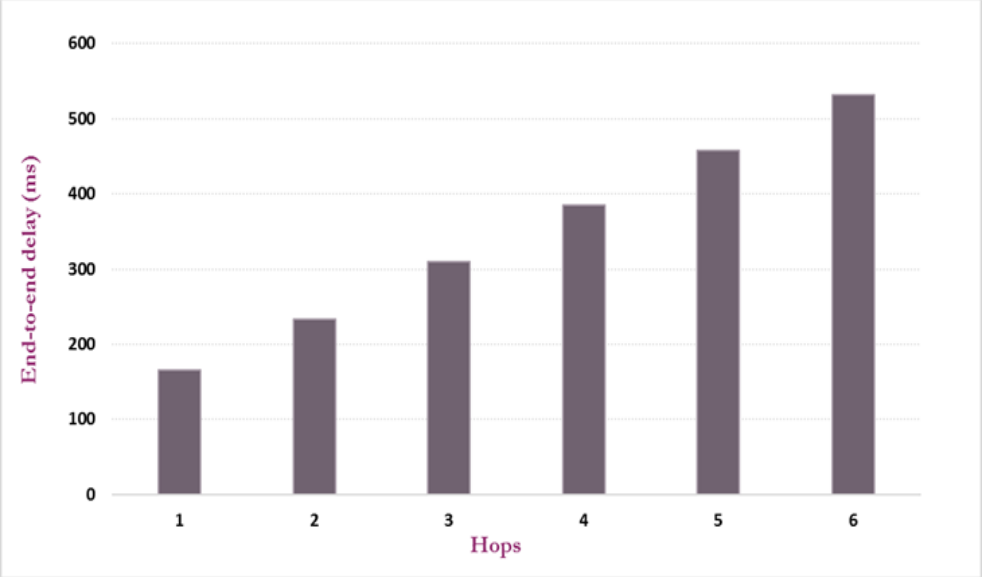

Figure 1. Average latency of application flow vs. hops for a 30-node network

(2) PDR: It refers to the ratio of the number of lost application packets to the total number of sent application packets. The PDR is computed with the help of the formula presented below:

$$PDR = \frac{\text{Total sent packets} - \text{Total received packets}}{\text{Total sent packets}} \quad (1)$$

A total of 30 mesh nodes with a maximum of 6 hops in which all the nodes need to participate in the SDN controller, were used to evaluate the SDN reliability. Figure 2 shows the PDR percentage for various hop numbers in a 30-node network. The overall trend indicated a higher percentage loss for packets that traveled over more hops. Because packets are forwarded by hops, there is high probability that packet loss will occur because of congestion and MAC layer fails shortly after initialization. In addition, because each node forwards packets through an SRHI, they require a source routing header, which needs to be received from the controller. The reason for the high network activity is because the FTQ/FTS messages are occasionally dropped and, therefore, the application messages are lost. However, this is not always the case as can be noticed in the Figure 2 in which the PDR for 3 and 4 hops are less than the PDRs for 2 hops.

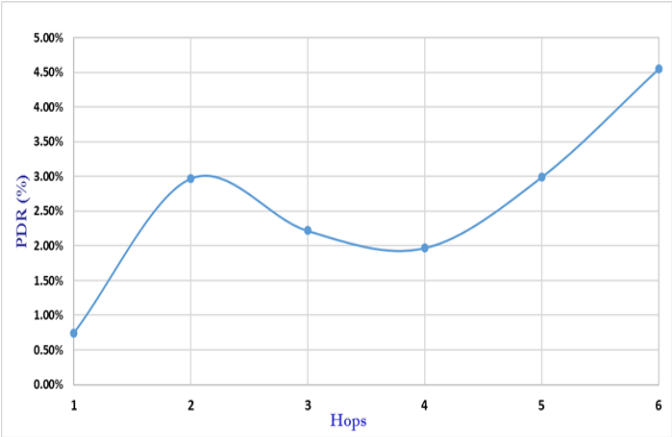

Figure 2. Percentage of PDR vs. hops for a 30-node network

(3) DER: It is 1 of the performance metrics determined in this study. It is defined as the ratio of the received packets to the total number of packets transmitted by a mesh node over a period of time. The formula for measuring the DER is as follows:

$$DER = \frac{Total\ received\ packets}{Total\ transmit\ packets} \quad (2)$$

We evaluated this by running a simulation of the network topology with a 20-, 30-, and 40-node mesh network over 1 to 15 hops with a 50%, 70%, and 90% link quality and a transmission range of 100 m. The simulation was ran for approximately 1 hour. DER is a value between 0 to 1: the closer the value is to 1, the more effective the deployment is. With an increase in the number of mesh nodes, the DER drops, as can be seen in Figure 3. For example, for a link quality of 50%, the DER is 0.45, 0.39, and 0.32 for a network with 20, 30, and 40 nodes, respectively. The DER, however, increases with a better-quality link. The DER was calculated to be 0.45, 0.5, and 0.611 for a link quality of 50%, 70%, and 90%, respective ly.

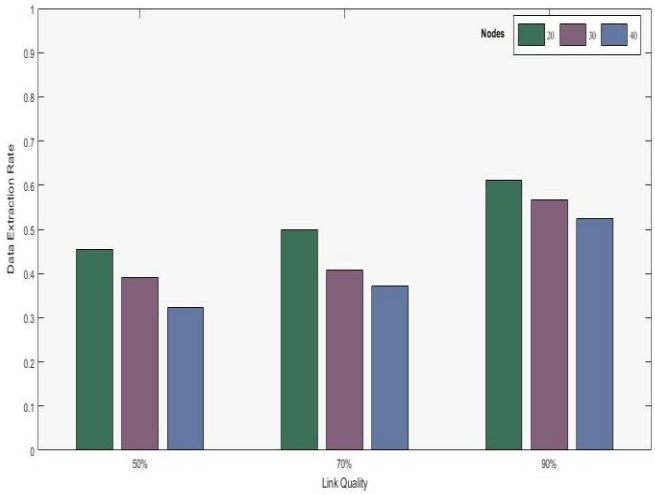

Figure 3. The DER for 20-, 30-, and 40-node topologies with a link quality of 50%, 70%, and 90%

(4) Analysis of the network traffic (user data and overhead): one objective of this paper was to evaluate the overheads introduced by using an SDN with overhead reduction techniques to show the effect of mitigating the cost of an SDN within a low power, multihop mesh framework on the network performance. The network traffic ratio can be determined by using the following formula:

$$SDN\ Overhead\ Ratio = \frac{Total\ Overhead\ App\ Traffic}{Total\ Network\ Traffic} \quad (3)$$

$$User\ Traffic = \frac{Total\ Network\ Traffic - Total\ Overhead\ App\ Traffic}{Total\ Network\ Traffic} \quad (4)$$

The application ratio and the SDN traffic are shown in Figure 4. The figure shows the network traffic for the user packets and network overheads at different numbers of nodes. The user packets refer to the application traffic, whereas the network overheads refer to the type of SDN packets such as CONF, FTQ, FTS, and NSU, which are described in the SDN framework. The figure clearly demonstrates higher traffic percentages for the user packets when compared with that of the network overheads for all considered network topology scenarios (20, 30, and 40 nodes). It also shows high traffic percentages for network overheads that are generated by the SDN packets. This high traffic percentage for network overheads places SDN in a challenging position, which requires further study. For instance, the user packets and network overhead percentages were found to be 57.31% and 42.68%, respectively, for a 30-node scenario. However, similar studies (Baddeley et al., 2018) reported approximately 25% for user packets and 75% for network overheads for the same network size.

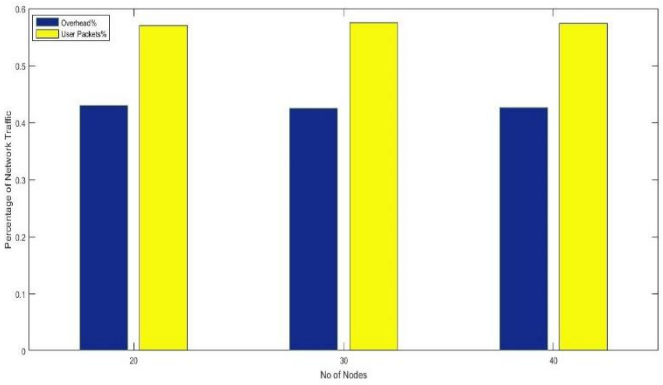

Figure 4. Percentage of network traffic in an SDN network

5. CONCLUSION

In this paper, we applied the SDN concept to a wireless network and evaluated its performance in terms of end-to-end delay, PDR, DER, and the SDN overhead. We used a lightweight SDN architecture designed for low-power wireless communication, called µSDN, to implement the SDN in the wireless environment in order to programmatically improve the QoS. In this study, the performance was evaluated using a Cooja simulator for Contiki OS. In particular, we considered the end-to-end delay, PDR, DER, and percentage of network traffic as evaluation metrics of the performance of the SDN-based wireless network. Our results indicated that increasing the number of nodes causes a drop in the DER of about 0.45, 0.5, and 0.6 for a link quality of 50%, 70%, and 90%, respectively. Finally, SDN simplifies the network management and configuration, however, it adds a high percentage of overhead to the network of about 43% in comparison with 57% for the user packets. Further investigation on the power consumption of the network is required.

REFERENCES

20-bit, M. (n.d.). Retrieved from https://github.com/contiki-os/contiki/wiki/MSP430X.

Anadiotis, A.-C., Galluccio, L., Milardo, S., Morabito, G. & Palazzo, S. (2019). SD-WISE: A Software-Defined WIreless SEnsor network. Computer Networks. doi: https://doi.org/10.1016/j.comnet.2019.04.029

Asadollahi, S., Goswami, B. & Sameer, M. (2018, 2). Ryu controller's scalability experiment on software defined networks. 2018 IEEE International Conference on Current Trends in Advanced Computing (ICCTAC), (pp. 1–5). doi:10.1109/ICCTAC.2018.8370397

Asadollahi, S., Goswami, B., Raoufy, A. S. & Domingos, H. G. (2017, 12). Scalability of software defined network on floodlight controller using OFNet. 2017 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT), (pp. 1–5). doi:10.1109/ICEECCOT.2017.8284567

Baddeley, M., Nejabati, R., Oikonomou, G., Sooriyabandara, M. & Simeonidou, D. (2018, 6). Evolving SDN for Low-Power IoT Networks. 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft), (pp. 71–79). doi:10.1109/NETSOFT.2018.8460125

Baddeley, M., Raza, U., Stanoev, A., Oikonomou, G. C., Nejabati, R., Sooriyabandara, M., & Simeonidou, D. (2019). Atomic-SDN: Is Synchronous Flooding the Solution to Software-Defined Networking in IoT? IEEE Access, 7, 96019-96034.

Beacon. (n.d.). Retrieved from https://openflow.stanford.edu/display/Beacon

Costanzo, S., Galluccio, L., Morabito, G., & Palazzo, S. (2012, 10). Software Defined Wireless Networks: Unbridling SDNs. 2012 European Workshop on Software Defined Networking, (pp. 1–6). doi:10.1109/EWSDN.2012.12

de Oliveira, B. T., Alves, R. C., & Margi, C. B. (2015, 10). Software-defined Wireless Sensor Networks and Internet of Things standardization synergism. 2015 IEEE Conference on Standards for Communications and Networking (CSCN), (pp. 60–65). doi:10.1109/CSCN.2015.7390421

Dunkels, A., Gronvall, B. & Voigt, T. (2004, 11). Contiki - a lightweight and flexible operating system for tiny networked sensors. 29th Annual IEEE International Conference on Local Computer Networks, (pp. 455–462). doi:10.1109/LCN.2004.38

El-Mougy, A., Ibnkahla, M., & Hegazy, L. (2015, 10). Software-defined wireless network architectures for the Internet-of-Things. 2015 IEEE 40th Local Computer Networks Conference Workshops (LCN Workshops), (pp. 804–811). doi:10.1109/LCNW.2015.7365931

Feamster, N., Rexford, J., & Zegura, E. (2014, 4). The Road to SDN: An Intellectual History of Programmable Networks. SIGCOMM Comput. Commun. Rev., 44, 87–98. doi:10.1145/2602204.2602219

Floodlight. (n.d.). Retrieved from http://floodlight.openflowhub.org/

Galluccio, L., Milardo, S., Morabito, G. & Palazzo, S. (2015, 4). Reprogramming Wireless Sensor Networks by using SDN-WISE:

A hands-on demo. 2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), (pp. 19–20). doi:10.1109/INFCOMW.2015.7179322

Galluccio, L., Milardo, S., Morabito, G. & Palazzo, S. (2015, 4). SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks. 2015 IEEE Conference on Computer Communications (INFOCOM), (pp. 513–521). doi:10.1109/INFOCOM.2015.7218418

Ghaleb, B., Al-Dubai, A.Y., Ekonomou, E., Alsarhan, A., Nasser, Y., Mackenzie, L.M. & Boukerche, A. (2019). A Survey of Limitations and Enhancements of the IPv6 Routing Protocol for Low-Power and Lossy Networks: A Focus on Core Operations. IEEE Communications Surveys Tutorials, 21, 1607–1635. doi:10.1109/COMST.2018.2874356

Helkey, J., Holder, L., & Shirazi, B. (2016). Comparison of simulators for assessing the ability to sustain wireless sensor networks using dynamic network reconfiguration. Sustainable Computing: Informatics and Systems, 9, 1–7. doi:https://doi.org/10.1016/j.suscom.2016.01.003

Hendrawan, I. N. & Arsa, I. G. (2017, 11). Zolertia Z1 energy usage simulation with Cooja simulator. 2017 1st International Conference on Informatics and Computational Sciences (ICICoS), (pp. 147–152). doi:10.1109/ICICOS.2017.8276353

iperf. (n.d.). Retrieved from https://iperf.fr/.

Jian, D., Chunxiu, X., Muqing, W., & Wenxing, L. (2017, 12). Design and implementation of a novel software-defined wireless sensor network. 2017 3rd IEEE International Conference on Computer and Communications (ICCC), (pp. 729–733). doi:10.1109/CompComm.2017.8322639

Lasso, F. F., Clarke, K., & Nirmalathas, A. (2018, 4). A software-defined networking framework for IoT based on 6LoWPAN. 2018 Wireless Telecommunications Symposium (WTS), (pp. 1–7). doi:10.1109/WTS.2018.8363938

Luo, T., Tan, H., & Quek, T. Q. (2012, 11). Sensor OpenFlow: Enabling Software-Defined Wireless Sensor Networks. IEEE Communications Letters, 16, 1896–1899. doi:10.1109/LCOMM.2012.092812.121712

Miguel, M., Jamhour, E., Pellenz, M., & Penna, M. (2018, 11). SDN architecture for 6LoWPAN wireless sensor networks. Sensors, 18, 3738. doi:10.3390/s18113738

msp430. (n.d.). Retrieved from https://github.com/pksec/msp430-gcc-4.7.3

Nikoukar, A., Raza, S., Poole, A., Güneş, M., & Dezfouli, B. (2018). Low-Power Wireless for the Internet of Things: Standards and Applications. IEEE Access, 6, 67893–67926. doi:10.1109/ACCESS.2018.2879189

NOX. (n.d.). Retrieved from https://github.com/noxrepo/nox.

OFNet. (n.d.). Retrieved from http://sdninsights.org/.

Ominike, A., Seun, E., A. O., A., & Osisanwo, F. (2016, 12). Introduction to Software Defined Networks (SDN). International Journal of Applied Information Systems, 11, 10–14. doi:10.5120/ijais2016451623

POX. (n.d.). Retrieved from http://www.noxrepo.org/pox/about-pox/.

Rowshanrad, S., Abdi, V. & Keshtgari, M. (2016, 11). Performance evaluation OF SDN controllers: Floodlight and Openday Light. *IIUM Engineering Journal, 17*, 47–57. doi:10.31436/iiumej.v17i2.615

Ryu. (n.d.). Retrieved from http://sdnhub.org/tutorials/ryu/

Theodorou, T., & Mamatas, L. (2017, 11). CORAL-SDN: A software-defined networking solution for the Internet of Things. *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, (pp. 1–2). doi:10.1109/NFV-SDN.2017.8169870

Xia, W., Wen, Y., Foh, C., Niyato, D., & Xie, H. (2015, 1). A Survey on Software-Defined Networking. *Communications Surveys & Tutorials, IEEE, 17*, 27–51. doi:10.1109/COMST.2014.2330903

Zhao, Y., Iannone, L., & Riguidel, M. (2015, 11). On the performance of SDN controllers: A reality check. *2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*, (pp. 79–85). doi:10.1109/NFV-SDN.2015.7387