

Integration of MQTT-SN and CoAP protocol for enhanced data communications and resource management in WSNs

Emmanuel Nwankwo, Michael David, Elizabeth Nonye Onwuka

Department of Telecommunications Engineering, School of Electrical Engineering and Technology, Federal University of Technology, Minna, Nigeria

Article Info

Article history:

Received Nov 6, 2022

Revised Sep 6, 2023

Accepted Nov 14, 2023

Keywords:

Constrained application protocol
Hybrid protocol
Machine to machine communication
Message queue telemetry transport protocol for sensor network
Wireless sensor network

ABSTRACT

Lightweight communication protocols for wireless sensor networks (WSNs) are unfolding for machine to machine (M2M) communications and thus there is always going to be a possible conflict of interest on which protocol is best suited for any particular application. The two protocols of interest in this study are the message queue telemetry transport protocol for sensor network (MQTT-SN), a variant of message queue telemetry transport (MQTT) protocol and the constrained application protocol (CoAP). There have been studies that reveal that these protocols perform differently based on the underlying network conditions. CoAP experience lower delays than MQTT for higher packet loss and higher delays for lower packet loss. MQTT default communication via a broker is easier to scale compared to CoAP direct request-response paradigm. Although this is a huge advantage over CoAP, it presents the single point-of-failure problem. In this paper we propose an integration of MQTT-CoAP protocol using an abstraction layer that enables both MQTT-SN and CoAP protocol to be used in the same sensor node. Resources are managed by directly modifying sensor node configuration using CoAP protocol. Performance evaluation of these protocols under the integrated scenario shows acceptable levels of latency and energy consumption for internet of thing (IoT) operations.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Emmanuel Nwankwo

Department of Telecommunications Engineering, School of Electrical Engineering and Technology

Federal University of Technology

Minna, Nigeria

Email: emmanueln_nike@hotmail.com

1. INTRODUCTION

The internet of things (IoT) being an emerging new technology is formed by a large number of devices with capacity for sensing, processing, communication and actuation therefore new protocols are being developed in the protocol stack to enhance communications in IoT environments [1]. It is also evident that as the number of IoT applications increase, the need to modify or introduce new protocols also increase. These new protocols must address issues like dynamic adoption to network condition and interoperability [2]. In the layered architecture of IoT, the application layer provides various communication protocols to act as an interface between desired application and end-users [3], [4]. In the IoT application layer, several protocols have been designed specifically for constrained devices but this work is focusing on the two most popular protocols which are the constrained application protocol (CoAP) and the message queue telemetry transport protocol for sensor network (MQTT-SN).

The CoAP was designed for use in devices with limited processing capability by the constrained RESTful environment (CoRE) working group of IETF [5]. It is a lightweight version of hypertext transfer

protocol (HTTP) that uses a subset of HTTP methods to operate on the client-server architecture, which makes it interoperable with HTTP [6]. CoAP compresses and sends messages over UDP unlike HTTP which communicates over TCP. Message queue telemetry transport (MQTT) protocol operates under the publish/subscribe architecture and communicates over TCP just like HTTP. MQTT-SN [7] is a lightweight version of MQTT which is designed to adapt specifically to the wireless communication network. MQTT-SN communicates using UDP and functions in such a way that it is independent of the underlying network services. Table 1 shows the major differences between MQTT-SN and CoAP protocol.

Table 1. Major differences between MQTT-SN and CoAP

	MQTT-SN	CoAP
Application layer	Single layer	Single layer with 2 conceptual sublayers (message layer and request response layer)
Reliability	3 quality of service (QoS) levels	Confirmable/non-confirmable messages, acknowledgments and retransmissions
Architecture	Publish/subscribe	Request/response, resource/observe
Header size	7 bytes	5 bytes

The MQTT publish/subscribe (pub/sub) messaging systems [8] are well-known examples of data-centric communication and are widely used in enterprise networks. This is mainly because they are scalable and support dynamic application topology. The essence of data-centric communication necessitates the presence of a broker responsible for managing the flow of information between publishers and subscribers. This broker offers a high-level abstraction for individual nodes, users, or actors. Regardless of the numerous benefits and ease of this approach in wireless sensor network (WSN), this presents a single point-of-failure problem meaning that if the broker is down, then there is no communication within the network. Conversely, message-centric communication, supported by CoAP, often relies on interactions between individual network units, thereby eliminating the possibility of a single point-of-failure. The system engineer therefore must decide between these two approaches the one that is best suited for the application requirements and environment.

Since CoAP and MQTT-SN utilize the UDP protocol, congestion control is a problem that needs to be considered. There is currently very scarce information on the effects of congestion due to UDP in MQTT-SN system. CoAP handles congestion using simple binary exponential backoff (BEB) but faces problems of significant increase in retransmission delays [2]. This congestion leads to network retransmission, increasing energy consumption, packet loss, latency and reduces packet delivery ratio (PDR) [9]. There has been recent studies [9], [10], on improved congestion control algorithms for CoAP and are very promising even for resource constrained device.

There are complementary advantages of communicating with either of the request-response or publish-subscribe based protocols [11]. Request-response communication models are more reliable and timely compared to publish-subscribe models and publish-subscribe models are more adaptable and support mobility compared to request-response model [11]. The objective of this research is to integrate MQTT-SN and CoAP protocols in a single resource constrained device. The approach adopted to achieve this is by developing an abstraction layer that enables the sensor node to support both the MQTT-SN protocol and the CoAP protocol concurrently.

Sensor nodes are often pre-designed for specific purposes that are not adjusted after deployment. In this research the resources we are focused on managing are the energy and communication reliability. Unpredictable scenarios that require same sensor node to tradeoff reliable communication for lower energy footprint and vice-versa calls for a design architecture that enables remote reconfiguration of the sensor nodes. Although request-response function is being added to various versions of MQTT, this remote configuration is better done with CoAP. Integrating the two protocols in a sensor node will bring a lot of flexibility in device management. The proposal in this study does not involve remote reprogramming using FPGAs as proposed by Castellani *et al.* [6] mainly because FPGAs consume much more power than regular microcontrollers [12].

The rest of this paper is organized as follows: methodology (section 2) provides a detailed description of the developed abstraction layer and the application programming interfaces (API) it provides, the experimental setup and the simulation environment. The resource management procedure implemented with CoAP is also described in section 2. Results and discussion (section 3) presents the results of the implementation and a discussion of the results. The paper is then concluded in section 4 concludes the paper.

2. METHOD

The MQTT-SN and CoAP protocols were integrated in an abstraction layer where the complexity of managing both protocols were abstracted. The following sections provide information on the design of the abstraction layer and the experimental setup.

2.1. Abstraction layer

The design and implementation of the abstraction layer is structured to provide APIs that offer abstractions of the complexity in managing the two protocols separately. In addition it provides APIs suited to advanced users seeking to access the core functionalities of each individual protocol. The Figure 1 is a block diagram that illustrates the abstraction layer.

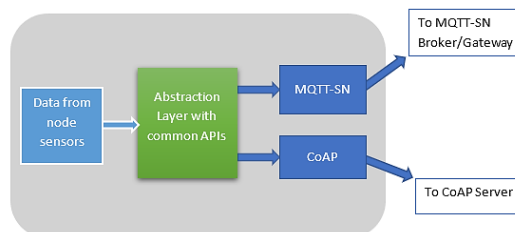


Figure 1. Block diagram of how the abstraction layer interfaces the protocols

The abstraction layer provides the following APIs; *Setup_Coap_Resource()*, *Init_Mqtt_Coap()*, *Mqtt_Sn_Sub()*, *Mqtt_Sn_Pub()*. The *Setup_Coap_Resource()* API is called for the purpose of creating the CoAP resource handlers and methods i.e., GET, POST, PUT or DELETE, endpoint URL. The *Init_MQTT_Coap()* API creates an MQTT connection to the broker and register any topics that would be used. These topics are then converted into numeric topics for MQTT-SN. It is also used to initialize the CoAP resources that were previously created. MQTT-SN nodes start sending keep-alive pings to the broker immediately after they are connected. The *Mqtt_Sn_Sub()* is used to subscribe to a topic on the broker and is specific to the MQTT-SN. The API takes two arguments; the topic for subscription and the QoS level. The *Mqtt_Sn_Pub()* is also specific to the MQTT-SN protocol and is used to publish a message on the broker on a previously registered topic. It takes four arguments; the topic, the message, the QoS and a retain flag. The retain flag just tells the broker whether to retain the message for new subscribers or to discard it. When both MQTT-SN and CoAP are used simultaneously in the same constrained device, the APIs in the abstraction layer are used to determine the communication latency.

2.2. Experiment setup

The performance of the integrated protocol system was evaluated when the abstraction layer was implemented on the constrained device. The latency per message size in bytes transmitted for different QoS levels was the metric we used for performance evaluation. For CoAP the latency was evaluated as the difference in time between when a request was sent to the node and when a response was received. For MQTT, the latency was measured as the difference in time between when a message was published and when the subscriber received the message.

2.2.1. Simulation environment and setup

The simulator used in this experiment was the Contiki OS based Cooja network simulator [13]–[15]. This firmware level simulator uses hardware emulation to execute deployable applications and OS code compiled for the target platform. Timing-sensitive software can be tested using this simulator [16].

Two types of sensor nodes were used, the Skymote and the Zolertia Z1 mote. An IPV6 router for low-power and lossy networks (RPL) Border router was setup on a Zolertia Z1 mote and using a tool called tunslip utility to interface the motes and the really small message broker (RSMB). A serial line interface protocol (SLIP) bridge was created between the RPL network and the local network. The interface for simulation and other parts of the simulator is shown in Figure 2.

The window labelled A is called the network map. The nodes added to the simulator are shown here as numbered circles in different colors. It also shows the IPV6 address of the node; 10×10 m² background grid for measuring radio coverage area, transmission distance and other relevant information. The window labelled B shows the connection status of the border router to the local network via an open socket connection. The simulation controls is shown in the window labelled C and the mote debug output in different color shades for different nodes is labelled as D. The power tracker and radio duty cycle for each mote is shown in the window labelled E.

Energy consumed by each node was estimated using the energest module in the Cooja network simulator. Energest is a software-based online energy estimation mechanism that measures the accumulated time the sensor node is in different states such as IRQ, CPU, LPM, Tx and Rx [17]. The average energy

consumed in any state is calculated using (1), therefore the total energy consumed on average by the node in the chosen interval is calculated using (2):

$$E_{avg}^{state} = (t_{end}^{state} - t_{start}^{state}) \times I_{state} \times V \tag{1}$$

$$E_{avg} = E_{avg}^{CPU} + E_{avg}^{Tx} + E_{avg}^{Rx} + E_{avg}^{LPM} \tag{2}$$

For the calculation to be done correctly, the value of current (amperes) consumptions of each state was taken from the Sky mote datasheet [18].

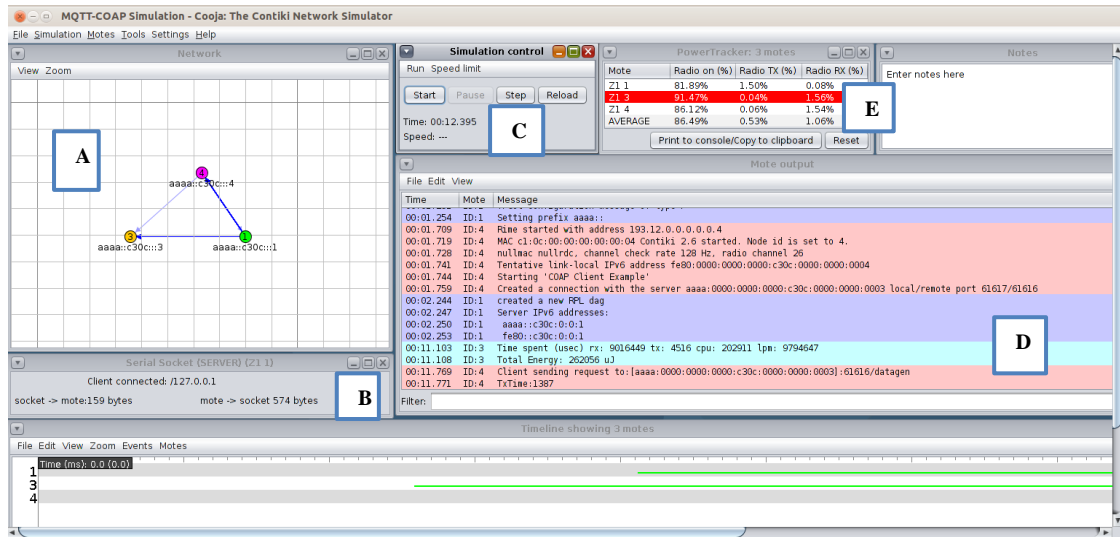


Figure 2. Cooja simulation environment

2.2.2. Software setup

Open-source implementations of CoAP and MQTT-SN were MQTT-SN for Contiki OS included in the example implementations and also the CoAP for Contiki [19]. In order to incorporate and develop the abstraction layer, the code-base for these implementations were modified. This was compiled for both the Skymote and the Zolertia Z1 mote and placed according to the network topology in Figure 3. After organizing the nodes, the RSMB was started then the tunslip utility was run to bridge the RPL network and local network thereby enabling the nodes to communicate with the RSMB running on the local machine and listening for connections on port 1883. The periodic interval of 10s was chosen within which the nodes sent packets and the packet size was increased by 10 bytes after each transmission starting from a packet size of 20 bytes.

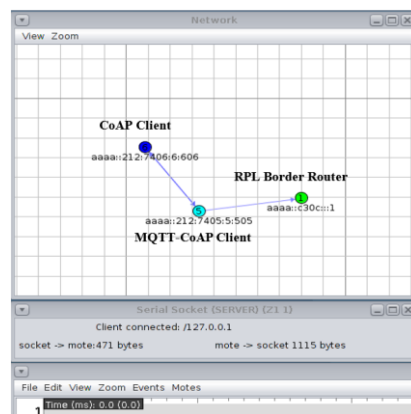


Figure 3. Network topology for sensor nodes

2.3. Resource management architecture

The resources of consideration are the energy and communication reliability. In the experimental scenario, MQTT protocol was used for periodic publishing of data to the broker while CoAP was setup for resource management using configuration parameters. The configuration parameters are MQTT QoS level (0, 1, 2), sensor probe and publish interval in seconds and sensors array toggle ON/OFF (1/0). The QoS level is stored in 1 byte memory, the sensor probe and publish interval is stored in 2 bytes memory (max of 65535 seconds) and the sensor array is stored in 2 bytes memory (max of 16 sensors). The sensor array is a binary representation of the 2 bytes (16 bits) where each bit toggle represents the ON/OFF toggle of a single sensor. All configuration parameters are accessed and adjusted via CoAP protocol. The Table 2 shows the request, endpoint and response that were made for each configuration.

Table 2. CoAP requests made for each configuration parameter

Function	Request	Parameters	Response
Retrieve QoS level	GET/QoS		Integer value representing QoS
Change QoS level	POST/QoS	QoS value	
Retrieve interval	GET/interval		Integer value representing interval
Change interval	POST/interval	Interval value	
Retrieve sensor toggle array	GET/sensors		Integer value representing 16 bits of sensors array
Change sensor toggle array	POST/sensors	Sensors array value	

3. RESULTS AND DISCUSSION

Since the experimental setup was done in a controlled environment and has just one broker, one publisher and one CoAP client, we experienced zero loss in message delivery. Our experimentation showed that the maximum packet size that could be transmitted from the node is 87 bytes. Further research proved that this is because the motes ran on Zigbee and this follows the IEEE 802.15.4 standard that has a limit on the data size allocated to user application [20].

As the packet size was increased, the latency values were observed to slightly increase as well. The lowest latency was observed in MQTT-SN QoS 0 while similar latency values were obtained for the CoAP and MQTT-SN QoS 1. The average latency was observed to be 163.2 ms, 188.5 ms and 191.5 ms for MQTT-SN QoS 0, MQTT-SN QoS 1 and CoAP respectively. The graph of latency against packet size for MQTT-SN and CoAP is shown in Figure 4.

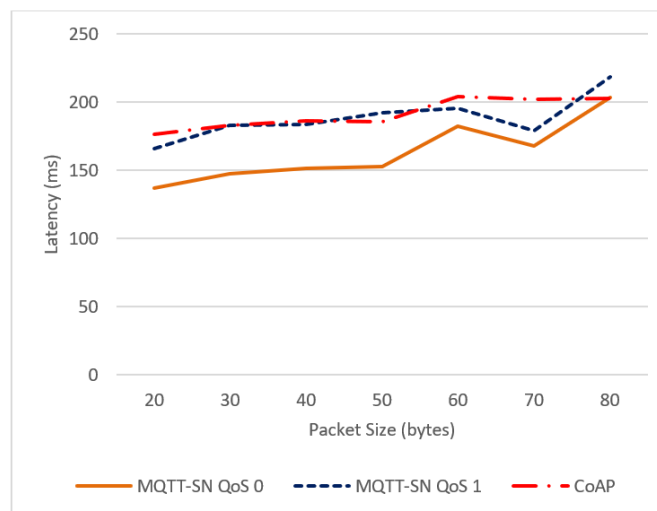


Figure 4. CoAP and MQTT-SN latency at different packet sizes

The total energy consumption of the node when using MQTT-SN for a single Tx/Rx operation within an interval of 10 s showed an average of 261.6 mJ for both QoS 0 and QoS 1 and an average of 261.3 mJ for CoAP. The energy consumption for the different protocols when transmitting and receiving different packet sizes is shown in Figure 5. Adjusting sensor toggle and publish interval resources led to

significant changes in power consumption of the sensor node and adjusting QoS level affects the reliable publishing of sensed data to the broker.

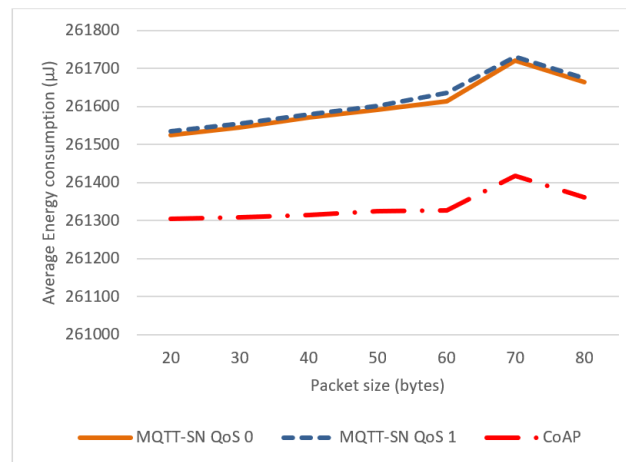


Figure 5. Average energy consumption of CoAP and MQTT-SN transmitting and receiving different packet sizes

Integration of publish-subscribe and request-response paradigms using an abstract communication model for ubiquitous system has been proposed in another study [11] to enable software solutions that combine synchronous and asynchronous communication. They further implemented a hybrid broker capable of both request-response and publish-subscribe paradigms. Unlike the current study, their model is independent of any existing protocol and does not evaluate the performance on resource constrained devices.

A similar study has been done to evaluate the performance of MQTT and CoAP via a common middleware [21] but it was implemented on devices with no resource constraints for running the protocols also classified as high-end IoT devices [22]. Their experimental results showed that the performance of these different protocols are dependent on different network conditions. MQTT messages experienced lower delays than CoAP for lower packet loss and higher delays than CoAP for higher packet loss.

Recent advances in improving CoAP such as the work done on congestion control [10], [23], security [24], does not give it the flexibility in QoS like MQTT-SN and recent advances in MQTT such as Adaptive QoS [25], security [26] does not give it better resource discovery or make communication possible without a broker. The importance of this study is to lay the foundation for multi-protocol integration in single resource constrained devices. This will undoubtedly add additional flexibility in communication between devices and reduce the time taken to evaluate the tradeoffs between choosing either protocol especially in current dynamic communication environments.

4. CONCLUSION

In this paper, we proposed an integration of MQTT-SN and CoAP on constrained devices, this was done by introducing an abstraction layer the architecture. This was done to take benefit from the advantages of CoAP and MQTT-SN while avoiding their shortcomings. The performance of the integrated system was evaluated with a firmware level simulation and the result was quite satisfactory.

Our results revealed that when operating with MQTT-SN protocol, average latency at an average data size of 50 bytes is 191.5 ms for CoAP, 163.2 ms for QoS 0 and 188.5 ms for QoS 1. Also, the energy consumption for a single transmit/receive operation within a 10s interval was an average of 261.6 mJ, 261.3 mJ for MQTT-SN and CoAP respectively. The integration of the two protocols on a constrained device does not have a negative impact on the system. The abstraction layer provides the ability for both MQTT-SN and CoAP protocols to be used simultaneously on constrained IoT devices therefore further studies can be done to leverage this technique for adaptively switching these protocols in situations of power saving, broker failures or varying network conditions.





ACKNOWLEDGEMENTS

We gratefully acknowledge the support of the Nigerian Tertiary Education Trust Fund (TETFund) via the Institutional Based Research Intervention (IBRI) grant 2019.





REFERENCES

- [1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: a survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys and Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015, doi: 10.1109/COMST.2015.2444095.
- [2] P. K. Donta, S. N. Srirama, T. Amgoth, and C. S. R. Annavarapu, "Survey on recent advances in IoT application layer protocols and machine learning scope for research directions," *Digital Communications and Networks*, vol. 8, no. 5, pp. 727–744, Oct. 2022, doi: 10.1016/j.dcan.2021.10.004.
- [3] T. Salman and R. Jain, "A Survey of Protocols and Standards for Internet of Things," *Advanced Computing and Communications*, vol. 1, no. 1, 2019, doi: 10.34048/2017.1.f3.
- [4] P. Sethi and S. R. Sarangi, "Internet of things: architectures, protocols, and applications," *Journal of Electrical and Computer Engineering*, pp. 1–25, 2017, doi: 10.1155/2017/9324035.
- [5] Z. Shelby, K. Hartke, and C. Bormann, "Constrained application protocol (CoAP) draft-ietf-core-coap-17," 2013. available: <http://tools.ietf.org/html/draft-ietf-core-coap-17> (accessed: Nov. 06, 2022).
- [6] A. P. Castellani, M. Gheda, N. Bui, M. Rossi, and M. Zorzi, "Web services for the Internet of things through CoAP and EXI," in *IEEE International Conference on Communications*, IEEE, Jun. 2011, pp. 1–6, doi: 10.1109/iccw.2011.5963563.
- [7] A. Stanford-Clark and H. L. Truong, "MQTT for sensor networks (MQTT-SN) protocol specification," *OASIS Draft*, pp. 1–28, 2013.
- [8] P. T. Eugster, P. A. Felber, R. Guerraoui, and A. M. Kermarrec, "The many faces of publish/subscribe," *ACM Computing Surveys*, vol. 35, no. 2, pp. 114–131, Jun. 2003, doi: 10.1145/857076.857078.
- [9] P. K. Donta, T. Amgoth, and C. S. R. Annavarapu, "Congestion-aware data acquisition with q-learning for wireless sensor networks," in *IEMTRONICS 2020 - International IOT, Electronics and Mechatronics Conference, Proceedings*, IEEE, Sep. 2020, pp. 1–6, doi: 10.1109/IEMTRONICS51293.2020.9216379.
- [10] N. Makarem, W. B. Diab, I. Mougharbel, and N. Malouch, "On the design of efficient congestion control for the Constrained Application Protocol in IoT," *Computer Networks*, vol. 207, Apr. 2022, doi: 10.1016/j.comnet.2022.108824.
- [11] C. Rodríguez-Domínguez, K. Benghazi, M. Noguera, J. L. Garrido, M. L. Rodríguez, and T. Ruiz-López, "A Communication model to integrate the Request-Response and the publish-subscribe paradigms into ubiquitous systems," *Sensors (Switzerland)*, vol. 12, no. 6, pp. 7648–7668, Jun. 2012, doi: 10.3390/s120607648.
- [12] Y. E. Krasteva, J. Portilla, J. M. Carnicer, E. De La Torre, and T. Riesgo, "Remote HW-SW reconfigurable wireless sensor nodes," in *IECON Proceedings (Industrial Electronics Conference)*, IEEE, Nov. 2008, pp. 2483–2488, doi: 10.1109/IECON.2008.4758346.
- [13] M. Jevtić, N. Zogović, and G. Dimić, "Evaluation of wireless sensor network simulators," *Proceedings of the 17th Telecommunications Forum TELFOR 2009 Belgrade Serbia*, pp. 1303–1306, 2009.
- [14] T. Mehmood, "COOJA Network Simulator: Exploring the Infinite Possible Ways to Compute the Performance Metrics of IOT Based Smart Devices to Understand the Working of IOT Based Compression & Routing Protocols," *Electrical Engineering and Systems Science > Signal Processing*, 2017.
- [15] K. Roussel, Y.-Q. Song, and O. Zendra, "Using Cooja for WSN simulations : some new uses and limits to cite this version : using cooja for wsn simulations : some new uses and limits," *EWSN 2016-NextMote workshop*, pp. 319–324, 2016.
- [16] J. Eriksson, "Detailed simulation of heterogeneous wireless sensor networks," 2009.
- [17] J. Schandy, L. Steinfeld, and F. Silveira, "Average power consumption breakdown of wireless sensor network nodes using IPv6 over LLNs," in *Proceedings - IEEE International Conference on Distributed Computing in Sensor Systems, DCOSS 2015*, IEEE, Jun. 2015, pp. 242–247, doi: 10.1109/DCOSS.2015.37.
- [18] Moteiv Corporation, "Moteiv: tmote sky low power wireless sensor module," *Product Data Sheet*, pp. 1–28, 2006.
- [19] Á. I. da Silva, "MQTT-SN for Contiki," *Github*. https://github.com/aiguacio/mqtt-sn-contiki_example (accessed Feb. 29, 2020).
- [20] T. R. Burchfield, S. Venkatesan, and D. Weiner, "Maximizing Throughput in ZigBee Wireless Networks through Analysis, Simulations and Implementations," *Proceedings of the International Workshop on Localized Algorithms and Protocols for Wireless Sensor Networks*, pp. 15–29, 2007.
- [21] D. Thangavel, X. Ma, A. Valera, H. X. Tan, and C. K. Y. Tan, "Performance evaluation of MQTT and CoAP via a common middleware," in *IEEE ISSNIP 2014 - 2014 IEEE 9th International Conference on Intelligent Sensors, Sensor Networks and Information Processing, Conference Proceedings*, IEEE, Apr. 2014, pp. 1–6, doi: 10.1109/ISSNIP.2014.6827678.
- [22] M. O. Ojo, S. Giordano, G. Procissi, and I. N. Seitanidis, "A Review of Low-End, Middle-End, and High-End IoT Devices," *IEEE Access*, vol. 6, pp. 70528–70554, 2018, doi: 10.1109/ACCESS.2018.2879615.
- [23] S. Bolettieri, G. Tanganelli, C. Vallati, and E. Mingozzi, "pCoCoA: A precise congestion control algorithm for CoAP," *Ad Hoc Networks*, vol. 80, pp. 116–129, Nov. 2018, doi: 10.1016/j.adhoc.2018.06.015.
- [24] S. Arvind and V. A. Narayanan, "An Overview of Security in CoAP: Attack and Analysis," in *2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS)*, IEEE, Mar. 2019, pp. 655–660, doi: 10.1109/ICACCS.2019.8728533.
- [25] F. Palmese, A. E. C. Redondi, and M. Cesana, "Adaptive quality of service control for MQTT-SN," *Sensors*, vol. 22, no. 22, pp. 1–19, Nov. 2022, doi: 10.3390/s22228852.
- [26] J. Roldán-Gómez, J. Carrillo-Mondéjar, J. M. C. Gómez, and S. Ruiz-Villafranca, "Security analysis of the MQTT-SN protocol for the internet of things," *Applied Sciences (Switzerland)*, vol. 12, no. 21, pp. 1–24, Oct. 2022, doi: 10.3390/app122110991.





BIOGRAPHIES OF AUTHORS

Emmanuel Nwankwo     holds an M.Eng degree in Telecommunications Engineering from Federal University of Technology, Minna, Nigeria. He is also a software engineer with proficiency in backend development and databases. His research interests include cloud computing, machine learning, internet of things, software defined networking, and microprocessors. He can be contacted at email: emmanueln_nike@hotmail.com.



Michael David     received his 1st and 2nd degree in Engineering from Federal University of Technology, Minna, Nigeria in 2004 and 2010, respectively. The Ph.D. degree in Electrical Engineering was conferred on him by Universiti Teknologi Malaysia (UTM), Skudai Johor, Malaysia in 2016 for his work on visible absorption based ozone sensors. He is currently a Senior Lecturer with Federal University of Technology Minna, Nigeria. He is a registered Engineer with the Council for the regulation of Engineering in Nigeria (COREN). He is also a Certified Fiber Optic Technologist (CFOT) and a member of Fiber Optic Association, inc. USA. He can be contacted at email: mikeforheaven@futminna.edu.ng.



Elizabeth Nonye Onwuka     is a Professor of Telecommunications Engineering at Federal University of Technology, Minna. She holds a Ph.D in Communications and Information Systems Engineering, from Tsinghua University, Beijing, People's Republic of China (2004); a Master of Engineering degree, in Telecommunications (1998); and a Bachelor of Engineering degree from Electrical and Computer Engineering Department, Federal University of Technology (FUT) Minna, Niger State, Nigeria (1992). Her research interest includes mobile communications networks, mobile IP networks, handoff management, network integration, resource management in wireless networks, spectrum management, and IoT applications. She can be contacted at email: onwukaliz@futminna.edu.ng.