



# A Novel Approach of Using Data Flipping for Efficient Energy on the Internet of Things

Ziyad Almudayni\*, Ben Soh , and Alice Li

Department of Computer Science and Information La Trobe University, Melbourne 3086, Australia

## Abstract

The Internet of Things (IoT) can be defined as the connection of devices, sensors, and actors via the Internet to a single network to provide services to end-users. Owing to the flexibility and simplicity of IoT devices, which impart convenience to end-users, the demand for these devices has increased significantly in the last decade. To make these systems more scalable, achieve a larger number of connected devices, and achieve greater economic success, it is vital to develop them by considering parameters such as security, cost, bandwidth, data rate, and power consumption. This study aims to improve energy efficiency and prolong the lifetime of IoT networks by proposing a new approach called the constrained application protocol CoAP45. This approach reduces the number of updates to the CoAP server using a centralized resource. The simulation results show that the proposed approach outperforms all existing protocols.

**Index Terms:** CoAP, Energy, Centralized resource, Node.js, MATLAB

## I. INTRODUCTION

In the Internet of Things (IoT) networks, devices, sensors, and actors communicate *via* the Internet to serve the end-users [1]. IoT services make users' lives more convenient and flexible; consequently, the demand for IoT services is increasing [2]. Therefore, developing and enhancing these systems with a focus on security, cost, bandwidth, data rates, and power consumption is vital. Furthermore, because developers' interests vary depending on their fields of specialization, they need to cooperate in a parallel manner to improve all IoT device parameters. Improvements in any of the four layers of the IoT architecture (the perception, middleware, network, and application layers) can positively affect the IoT system parameters. Therefore, developers must work more in-depth to select suitable parameters and target-specific IoT layers to achieve their objectives.

This study aims to improve one of the most important

parameters of connected devices, namely the energy efficiency of IoT devices. Enhancing the energy efficiency and prolonging the network lifetime of IoT networks can be achieved without directly reducing the watts or joules; it can also be achieved by improving the parameters of any of the four IoT architecture layers. Therefore, in this study, we specifically target the application layer to prolong the lifetime of an IoT network. There are various application protocols, such as Message Queuing Telemetry Transport (MQTT), Constrained Application Protocol (CoAP), Data Distribution Service (DDS), and Extensible Messaging and Presence Protocol (XMPP); however, for more in-depth work and to enhance precision, we chose only one application protocol, CoAP. The fundamental purpose of our design is to modify the time interval for the IoT nodes, such as temperature and humidity, to update the CoAP server with new values. This is discussed further in the following sections.

This paper consists of five sections. Section 1 presents an


Received 15 February 2022, Revised 29 March 2022, Accepted 29 March 2022

\*Corresponding Author Ziyad Almudayni (E-mail: 20167676@students.latrobe.edu.au)

Department of Computer Science and Information La Trobe University, Melbourne 3086, Australia

Open Access <https://doi.org/10.56977/jicce.2023.21.3.185>

print ISSN: 2234-8255 online ISSN: 2234-8883

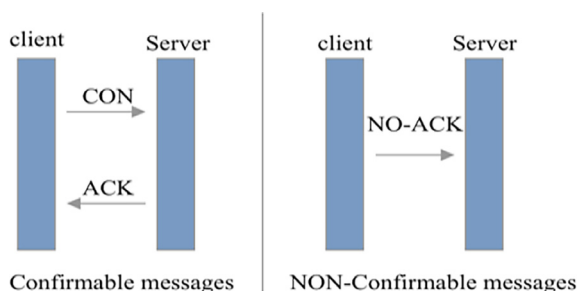
 This is an Open Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License (<http://creativecommons.org/licenses/by-nc/3.0/>) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.

Copyright © The Korea Institute of Information and Communication Engineering

overview of the CoAP and its message types. Section 2 describes the message format of CoAP and outlines the four main communication methods. Section 3 reviews previous studies on improving the energy efficiency of IoT networks using CoAP. Section 4 explains the work process of the new algorithm CoAP45 and fills in the gaps in existing protocols: the standard approach, tuning approach, update on different readings approach, and Fibonacci approach. Section 5 presents the tools used to implement CoAP45 and the method for calculating the battery level. The final section discusses the implementation results of CoAP45 and compares them with existing application protocols.

## II. CONSTRAINED APPLICATION PROTOCOL (COAP)

The IETF Constrained RESTful Environments (CoRE) working group developed a Constrained Application Protocol (CoAP) to enable machine-to-machine communication in a real environment. CoAP and HTTP are very similar in using a request/response communication structure and the same RESTful methods. The main difference between CoAP and HTTP is the type of communication protocol: CoAP runs over UDP, and HTTP runs over TCP. The UDP communication protocol helps the CoAP protocol to reduce the bandwidth size and simplify low-power communications for IoT applications. As UDP does not provide high-level message accuracy, CoAP successfully leverages some mechanisms to overcome the unreliability of messaging at the application layer to achieve greater accuracy and reliability. The following message types were used to ensure the reliability of the CoAP. **Confirmable (CON):** Like HTTP, both require an acknowledgment (ACK) of messages received between clients and servers. **NON-Confirmable (NON):** A message can be sent without acknowledgment; however, this is a low-accuracy message. **Acknowledgment (ACK):** The confirmation message that a CON uses to confirm the reception of messages. **Reset (RST):** Response when a message cannot be processed [3]. Fig. 1 shows the difference between confirmable and non-confirmable messages.



**Fig. 1.** Comparison between confirmable messages and NON-confirmable messages.

## III. COAP MESSAGE FORMAT AND METHODS

The message formats of the IoT application protocols can be distinguished from one another by their size and format. For instance, the message size of MQTT is 2 bytes, whereas that of CoAP is 4 bytes. The communication mechanism in application protocols can be deduced from the message format. This section discusses the format and methods of the CoAP application protocol.

The message format is divided into two main layers, containing five parts in total.

The first layer is responsible for controlling message packets. It has three parts:

1. **Version (2-bits):** which decelerates the CoAP version number.
2. **Type (2-bits):** This indicates whether the message type is confirmable (0), non-confirmable (1), acknowledgment (2), or reset (3).
3. **Token length (4-bits):** which indicates the length of a variable.

The second layer defines the message ID and indicates whether a message is a request or response to avoid issues such as packet arrival. It has two parts:

1. **Code (8-bits):** This has four different codes: request (0), success response (2), client error response (4), or server error response (5).
2. **Message-ID (16-bits):** which provides a unique ID for every message.

The CoAP protocol uses four methods to send and receive messages to and from the client and server. These methods are based on the following RESTful structures: GET, POST, PUT, and DELETE. **GET** can obtain data, such as room temperature, from URL requests. **POST** is used to create new data using the requested parent URL. **PUT** updates the data for the current readings. **DELETE** is used to delete data via URL requests [4].

## IV. RELATED WORKS

Centralized and distributed resource discovery (RD) are the two main mechanisms by which CoAP application resources frequently update the nodes' current reading. In centralized RD, devices use a single resource to process all queries. The nodes must be updated several times regarding the server status, which consumes considerable power. In a distributed RD, the devices find resources by sending direct queries. The server is usually updated with the current readings from the nodes within a standard time of 10s; however, this was modified by Yassein et al. (2020). The proposed method aims to minimize the time required for the CoAP server to be updated from the nodes to extend the network lifetime. To achieve this, a node's battery is split into four levels: Full

(100-95%) (10s for each update), High (95-50%) (40-60s for each update), Low (50-5%) (80-100s for each update). Thus, using the proposed approach, it is possible to obtain a 10% increase in the network lifetime [5].

The standard CoAP mechanism was modified by Mardini et al. [2018] to provide current readings to the CoAP server using a centralized resource. The CoAP server is updated every 10s from the nodes in standard CoAP. However, the proposed approach requires that data be sent to the server from IoT devices only when the reading environment changes after the most recent update. Compared with the standard CoAP protocol, the simulation results demonstrate that the proposed method improves the network lifetime by 33% and yields an 8% increase in the dynamic tuning approach [6].

To improve the energy efficiency of IoT systems, Jin et al. [2017] introduced a sleep-scheduling technique to regulate the sleep-wake status of the CoAP. Furthermore, the authors slightly modified the IoT middleware layer to achieve this objective. They updated the synchronicity of the IoT node concerning the sleep status using the message queue (MQ) and RD brokers. MQ uses the publish-subscribe method to connect the web client application to the IoT nodes, while RD searches for information for the IoT nodes registered in the CoAP server. After sending the data to the MQ, the CoAP node switches to sleep mode. The received data can then be transferred from the CoAP node to the MQ, allowing the CoAP to extract it before the status changes to sleep mode. However, the authors failed to perform any experiments comparing existing protocols with their proposed method [7].

Lai et al. [2020] made efforts to improve the process of transferring notifications from IoT applications to client devices by proposing group-based message management (GMM) for CoAP proxies. The following three modules form the basis of the proposed mechanism.

**Module 1:** The client demands are used to categorize their devices into groups.

**Module 2:** The cache memory capacity is modified to enable a proxy response to requests using notifications.

**Module 3:** Combination of all notifications from different IoT nodes requested by the client device. Based on simulation results, the proposed module enhances the energy efficiency of devices by minimizing the number of notifications [8].

To prevent delays between the gateway (CoAP client) and node (CoAP server), Vishakha et al. [2019] applied a time-synchronization method to the CoAP protocol. In their approach, the CoAP server was updated with the current time from the CoAP client, and a time delay occurred when there was a difference between the CoAP server time and the current time. To ensure no time delays, a time difference was added to the CoAP server to synchronize the server time with that of the CoAP client. In contrast to the standard

CoAP, the authors' modified CoAP protocol improves the energy efficiency of WSNs. The simulation tool used by the authors to implement their theory was NS2 [9].

Ludovici et al. [2012] adopted a novel approach to enhance the average delivery ratio, energy consumption, and delay performance of a CoAP application protocol. Using an e-health application, the proposed approach was applied to a real wireless sensor network (WSN). The proposed approach aims to effectively utilize priorities in the CoAP application protocol. There are two types of notifications (critical and non-critical) and four priority levels for order notifications. Server notifications to the observer were successfully reduced with this approach because neither type of notification was required at all levels, thus achieving the study's goal [10].

To resolve the issues of proxies in IoT devices that use CoAP, Randhawa et al. [2018] employed a security protocol known as OSCoAP (Object Security of CoAP). The scalability and flexibility of communication between servers and clients in the CoAP protocol increased when using proxies. In other words, the proxies helped improve device performance by linking servers to clients. In data exchange, OSCoAP encrypts and decrypts messages to protect these proxies and secure the transmission between servers and clients. The proposed approach increased the device's energy efficiency by 10, improved battery life, and boosted memory efficiency [11].

A new mechanism, AFT, was proposed by Albalas et al. [2017] to adjust the time nodes take to transmit the current reading in the centralized CoAP to resource discovery (RD). In the proposed approach, the time required to update the RD from the nodes, typically 10s, was altered to reduce the power consumption and extend the network lifetime. In the AFT method, the major criterion for updating the RD is the node's battery level. As calculated using the Fibonacci-based tuning algorithm, the time interval increased as the battery level decreased. The Fibonacci interval kept increasing until the battery level decreased to 5%, after which sleep mode was activated for the node. Regarding overall network lifetime, the simulation results illustrated that the AFT was 75% more effective than the standard CoAP [12].

Although all the above approaches have demonstrated that they perform better than the standard CoAP regarding energy efficiency, gaps still need to be addressed (see the following section).

## V. PROPOSED APPROACH

Previous modifications to the CoAP application protocol have reduced the power consumption of IoT systems compared to the standard CoAP application protocol. There are two main techniques for saving power in CoAP, and both are

linked to decreasing the frequency of CoAP server updates to save power. In [6], the nodes update the CoAP server only if they read new values. However, the proposed solution has some gaps that need to be addressed, such as if the CoAP server receives frequent updates in less than 10s, for example, if it is updated every 5s. In this case, the modified CoAP protocol consumes more power than the standard CoAP protocol because the nodes would read new with an interval of less than 10s, whereas the standard CoAP updates the server only every 10s. Another drawback is that when a node is dead, the CoAP discovers that after 300 s, which is too long to wait. In [5], the nodes update the CoAP server within a time interval based on the battery level. The maximum time interval required to update the CoAP server was 100 s when the battery level was 5% to 25%. The drawback of the previously modified CoAP protocol is the delay in updating the CoAP server with the current node reading, especially when the time interval is 100s, which is very long compared to 10s for the standard CoAP. In [12], the RD suffers from issues with the freshness of the data because the interval between updates can be too long. This is because the time interval is based on the Fibonacci calculation, and the Fibonacci number increases when the battery level decreases. For instance, when the Fibonacci number was 15, the interval was 610 min, which was longer than 10 min.

To address the gaps in the previously modified CoAP application protocols, we propose a new approach called CoAP45, which is an extension of the previously modified CoAP protocols to address the gaps in them. Two issues in previous works need to be fixed: the long wait for the CoAP to be updated about whether a node is dead and the long interval to update the server between readings. The proposed approach, CoAP45, addresses these gaps to prolong the network lifetime and improve the energy efficiency of IoT systems in the CoAP protocol by providing fresh updates at satisfactory intervals. Concerning the freshness of the data, the maximum interval between a node updating the CoAP server in the centralized directory (RD) with the current reading is 45s, and the maximum time that the RD would wait to specify whether a node is dead is also 45s.

Equation (1) calculates the best maximum time interval for nodes to update the CoAP server. In this equation, we consider the freshness of the data to keep the CoAP server updated as early as possible.

$$(A - B) / 2 \tag{1}$$

where A is the best current maximum time interval to update the CoAP server (100s), and B is the time interval of the standard CoAP protocol. The CoAP45 algorithm and Fig. 2 show how the proposed approach works, whereas Fig. 3 shows CoAP45-related components.

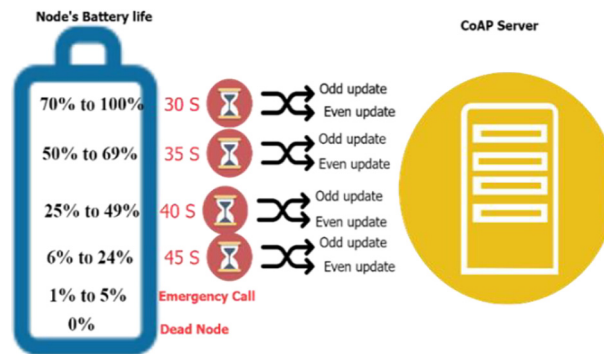


Fig. 2. How the proposed approach works.

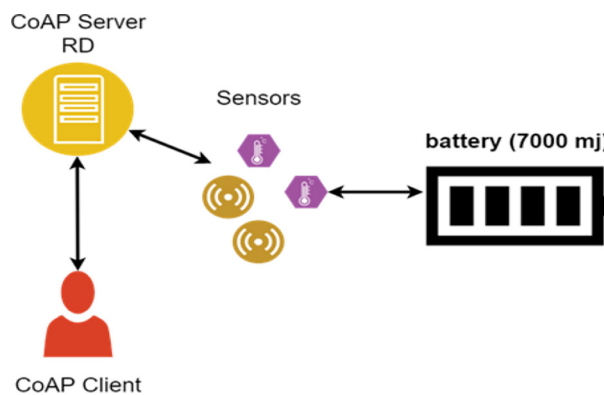


Fig. 3. CoAP45-related components.

**CoAP45 algorithm: If the battery life is between 70% to 100%**

Call the function (to update on different readings) in the first 30s. If there are no changes, do not update the CoAP server [odd update]; after another 30s, call the function (to update on different readings), and in the [even update], **force** nodes to update the CoAP server even if there are no changes in the collected data to let the server know whether or not the nodes are dead.

**If the battery life is between 50% to 69%**

Call the function (to update on different readings) in the first 35s. If there are no changes, do not update the CoAP server [odd update]; after another 35 s, call the function (to update on different readings), and in the [even update], **force** nodes to update the CoAP server even if there are no changes in the collected data to let the server know whether or not the nodes are dead.

**If the battery life is between 25% to 49%**

Call the function (to update on different readings) in the first 40s; if there are no changes, do not update the CoAP server [odd update]; after another 40 s, call the function (to update on different readings); and in the [even update], **force**

nodes to update the CoAP server even if there are no changes in the collected data to let the server know whether the nodes are dead.

**If the battery life is between 6% to 24%**

Call the function ( to update on different readings) in the first 45s; if there are no changes, do not update the CoAP server[ odd update]; after another 45s, call the function (to update on different readings); and in the [even update], **force** nodes to update the CoAP server even if there are no changes in the collected data to determine whether the nodes are dead.

**If the battery life is between 1% to 5%**

Emergency call

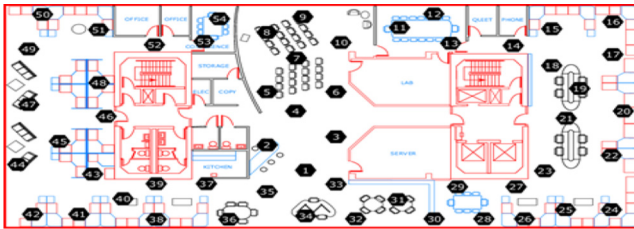
**If the battery life is 0%:** Dead nodes

**VI. PERFORMANCE EVALUATION SETUP**

**Dataset:** Mica2Dot sensors were deployed to collect data from the Intel Berkeley Research Lab between 28 February and 5 April 2004 (Fig. 4). Humidity, temperature, light, and voltage were varied in the experiment. These parameters were updated every 31s. TinyDB was used to extract information from a network of TinyOS sensors [13]. Table 1 lists the parameters used in this experiment.

**Table 1.** The parameters used in this experiment

Parameters	Value
Operating system	TinyOS
Database Type	TinyDB
Sensor Type	Mica2Dot
Value Types	Humidity, temperature, light and voltage
Number of Sensors	54
Transmission Range	40.5 m * 31 m
Implementation Tool	Nodejs
Full Battery	7000 mj



**Fig. 4.** Intel Berkeley Research Lab.

**Validation:** Node.js, a JavaScript runtime built on Chrome’s V8 JavaScript engine, validated the proposed approach.

**Voltage Average:** The average voltage was approximately 2.7 V as it ranged from 2-3 V.

**Battery Level Calculations:** The battery level was set to 7000 mJ for the full battery, as this is the full battery level mentioned in all papers in the literature review.

$$7000 \text{ milli Joule} = 7 \text{ Joule}$$

$$\text{mAh} \times \text{voltage} \times 3.6 = \text{Joule of energy}$$

$$\Rightarrow \text{mAh} = \frac{\text{Joule of energy}}{\text{Voltage} \times 3.6}$$

$$= 0.7 \text{ mAh}$$

The battery capacity is 0.7 mAh. However, because we designated one day of the experiment to update the CoAP server during the implementation, we extended the battery lifetime to 24 h. The battery lasts for 24 h if there is no additional cost, such as updating the CoAP server or additional degradation that affects lithium-ion batteries.

$$24 \text{ hour} \Rightarrow \text{the battery decreases } 0.7 \text{ mAh}$$

$$1 \text{ hour} \Rightarrow \frac{0.7}{24} \text{ mAh}$$

$$1 \text{ second} \Rightarrow \frac{0.7}{24 \times 3600}$$

Equation (2) is used to calculate the differences in the power consumption of the battery from beginning to end. The calculation was based on the time interval for nodes to update the CoAP server with the current reading plus 10% as a random additional cost (e.g., impact of temperature on the battery) and the cost of one reading update. The cost of one update was set as a constant to achieve accurate results when the time interval was changed.

$$EPC = IPC - (EP + (N \times \epsilon)) \tag{2}$$

where EPC denotes the estimated power consumption, IPC is the initial power consumption (which is frequently updated). N is the number of updates in one-time intervals, and ε (the cost for one update) is a constant number (2.1 mJ), which is about 0.03% when the battery is 7000 mJ. The EP is energy cost plus 10% of the additional cost to run the battery at a specific time.

Table 2 presents the energy costs in millijoules for running the battery at four different times: 10, 30, 35, 40, and 45 s. The second row in the table outlines the energy cost required to operate the battery for a time only without additional costs. The third row represents the previous energy cost plus the additional energy costs that impact the battery, such as the effect of temperature degree. It is counted as +10% of the energy cost.



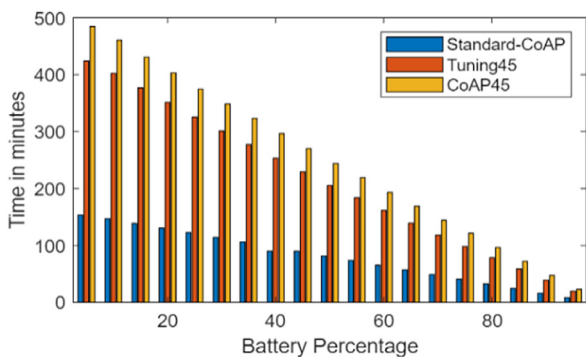
**Table 2.** The energy cost in mj to run the battery at four different times

Time in seconds	Energy cost (mj)	Energy cost plus 10% additional cost (mj) (EP)	Total Percentage
10 s	0.78	0.86	0.012%
30 s	2.36	2.59	0.037%
35 s	2.75	3.03	0.043%
40 s	3.15	3.46	0.049%
45 s	3.54	3.89	0.055%

## VII. RESULTS AND DISCUSSION

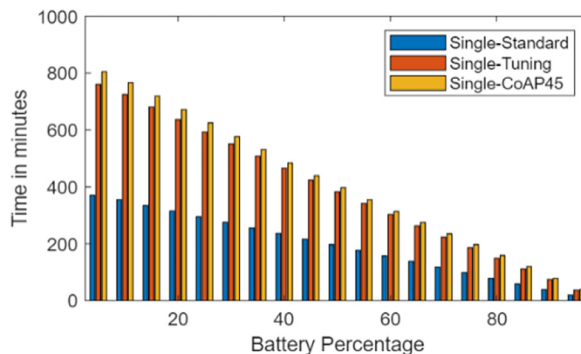
MATLAB was used to compare the energy efficiency performance and network lifetime of the standard CoAP, the tuning approach, and the proposed CoAP45.

Fig. 5 compares the network lifetime of the battery (7000 mJ fully charged) when updating the CoAP server with the current reading for the three different approaches. Different sensors obtain three types of updates: temperature, humidity, and light. The results in Fig. 5 prove that the proposed approach, CoAP45, outperforms the existing standard and tuning approaches regarding network lifetime and power consumption. In addition, the CoAP45 approach increased the network lifetime by approximately 14.3% compared with the tuning approach. Furthermore, 3500 mJ, approximately half of the battery capacity, was shown to last for 82, 205, and 244 min for the standard, tuning, and CoAP45 approaches, respectively.



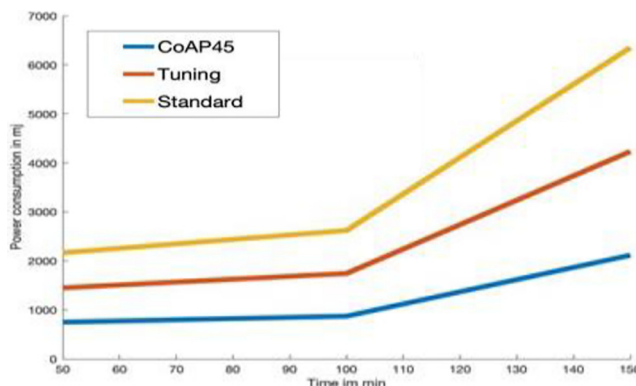
**Fig. 5.** Network lifetime for three types of sensor updates.

Fig. 6 shows the network lifetime of the battery (7000 mJ fully charged) when periodically refreshing the CoAP server with the current reading. Various sensors have been used to obtain temperature updates. The results in Fig. 6 prove that the proposed approach, CoAP45, surpasses the standard and tuning approaches regarding energy efficiency and network lifetime. CoAP45 succeeded in prolonging network lifetime by approximately 6% compared with the tuning approach when updating the temperature because of its frequent updates. Therefore, when the frequency of updates decreases, power savings increase.



**Fig. 6.** Network lifetime for temperature updates.

Fig. 7 shows the battery’s energy cost to update the CoAP server with the current readings for the first 150 min. The CoAP server receives updates from various sensors regarding temperature, humidity, and light. The figure presents a comparison of energy consumption using the three approaches. The results in Fig. 7 prove that the proposed approach, CoAP45, outperforms the existing approaches (the standard and tuning approaches) regarding power consumption. The energy costs in the first 150 min for the standard CoAP approach were approximately 6350 and 2620 mJ, and the CoAP45 approach was about 2166 mJ, respectively. These figures emphasize that the proposed approach succeeded in prolonging the network lifetime by approximately 65% compared with the standard approach and 17% compared with the tuning approach.



**Fig. 7.** Power consumption comparison.

The power savings and the network lifetime for the CoAP45 approach vary from one case to another depending on the number of updates the CoAP server receives. The number of updates in our scenarios was linked to changes in the read data when the sensors detected new values. Therefore, the network lifetime increases when the number of updates decreases. Fig. 8 shows the network life for two fully charged batteries (7000 mJ) when periodically updating

the CoAP server with current readings. The first battery was responsible for the temperature changes, and the second battery was responsible for the light changes. In our scenario, the temperature changes more frequently than under light. The results proved that network life increased when the number of updates decreased. The network lifetime of the battery responsible for temperature updates was approximately 805 min, compared with approximately 911 min for the battery responsible for light updates.

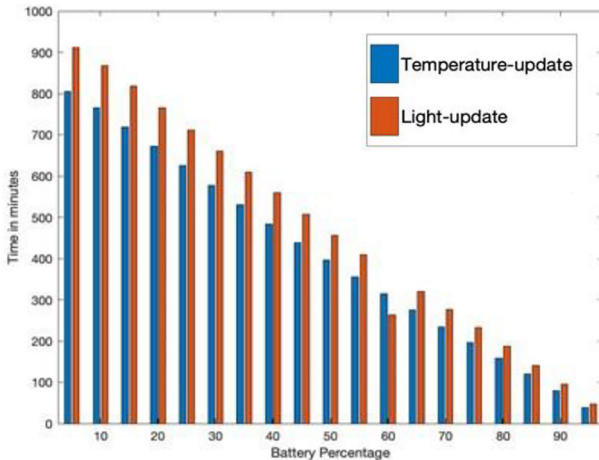


Fig. 8. Network lifetime for temperature and light updates.

## VIII. CONCLUSION

The proposed CoAP45 modifies the frequency of the standard CoAP application protocol to update the CoAP server regarding current reading in centralized resources, which prolongs the network lifetime of IoT devices. Using the Node.js platform, the simulation results showed that the proposed approach achieved better results than four approaches: the Fibonacci approach, the update on different reading approaches, the standard CoAP approach, and the battery-level approach in terms of the freshness of data and the lifespan of the IoT networks.

## REFERENCES

- [1] F. E. F. Samann, S. R. Zeebaree, and S. Askar, "IoT provisioning QoS based on cloud and fog computing," *Journal of Applied Science and Technology Trends*, vol. 2, no. 01, pp. 29-40, Mar. 2021. DOI: 10.38094/jastt20190.
- [2] S. Nižetić, P. Šolić, D. L.-I. González-de, and L. Patrono, "Internet of Things (IoT): Opportunities, issues and challenges towards a smart and sustainable future," *Journal of Cleaner Production*, vol. 274, p. 122877, Nov. 2020. DOI: 10.1016/j.jclepro.2020.122877.
- [3] M. Martí, C. Garcia-Rubio, and C. Campo, "Performance evaluation of CoAP and MQTT\_SN in an IoT environment," in *Multidisciplinary Digital Publishing Institute Proceedings*, vol. 31, no. 1, p. 49. Nov. 2019. DOI: 10.3390/proceedings2019031049.
- [4] R. A. Rahman and B. Shah, "Security analysis of IoT protocols: A focus in CoAP," in *2016 3rd MEC international conference on big data and smart city (ICBDSC)*, Muscat, Oman, pp. 1-7, 2016. DOI: 10.1109/ICBDSC.2016.7460363.
- [5] M. B. Yassein, I. Hmeidi, O. Meqdadi, F. Alghazo, B. Odat, O. AlZoubi, and A. Smairat, "Challenges and techniques of constrained application protocol (CoAP) for efficient energy consumption," in *2020 11th International Conference on Information and Communication Systems (ICICS)*, Irbid, Jordan, pp. 373-377, 2020. DOI: 10.1109/ICICS49469.2020.239564.
- [6] W. Mardini, M. B. Yassein, M. AlRashdan, A. Alsmadi, and A. B. Amer, "Application-based power saving approach for IoT CoAP protocol," in *Proceedings of the First International Conference on Data Science, E-learning and Information Systems*, Madrid, Spain, pp. 1-5, 2018. DOI: 10.1145/3279996.3280008.
- [7] W. Jin and D. Kim, "A sleep-awake scheme based on CoAP for energy-efficiency in internet of things," *JOIV: International Journal on Informatics Visualization*, vol. 1, no. 4, pp. 110-114, Nov. 2017. DOI: 10.30630/joiv.1.4.37.
- [8] W.-K. Lai, Y.-C. Wang, and S.-Y. Lin, "Efficient scheduling, caching, and merging of notifications to save message costs in IoT networks using CoAP," *IEEE Internet of Things Journal*, vol. 8, no. 2, pp. 1016-1029, Jan. 2021. DOI: 10.1109/JIOT.2020.3009332.
- [9] V. D. Khatade and M. A. Askhedkar, "Time synchronization for CoAP using NS2," in *2019 5th International Conference On Computing, Communication, Control And Automation (ICCUBEA)*, Pune, India, pp. 1-4, 2019. DOI: 10.1109/ICCUBEA47591.2019.9129316.
- [10] A. Ludovici, E. Garcia, X. Gimeno, and A. C. Augé, "Adding QoS support for timeliness to the observe extension of CoAP," in *2012 IEEE 8th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Barcelona, Spain, pp. 195-202, 2012. DOI: 10.1109/WiMOB.2012.6379074.
- [11] R. H. Randhawa, A. Hameed, and A. N. Mian, "Energy efficient cross-layer approach for object security of CoAP for IoT devices," *Ad Hoc Networks*, vol. 92, p. 101761, Sep. 2019. DOI: 10.1016/j.adhoc.2018.09.006.
- [12] F. Albalas, W. Mardini, and M. Al-Soud, "Aft: Adaptive fibonacci-based tuning protocol for service and resource discovery in the internet of things," in *2017 Second international conference on fog and mobile edge computing (FMEC)*, Valencia, Spain, pp. 177-182, 2017. DOI: 10.1109/FMEC.2017.7946427.
- [13] Intel Lab Data [Online] Available: <http://db.csail.mit.edu/labdata/labdata.html>.

### Ziyad Almudayni

Ziyad Almudayni is a PhD candidate at La Trobe University. I have done five publications in different conferences and journals. All publications are about the Internet of Things.