A New Back-off Algorithm with Priority Scheduling for MQTT Protocol and IoT Protocols

Marwa O Al Enany[1], Hany M. Harb[2]
Department of Systems and Computers
Faculty of Engineering, Al-Azhar University
Cairo, Egypt

Gamal Attiya[3]
Department of Computer Science and Engineering
Faculty of Engineering, Menoufia University
Menouf, Egypt

***Abstract*—The Internet of Things (IoT) protocols have encountered great challenges as the growth of technology has led to many limitations of the performance of the IoT protocols. Message Queuing Telemetry Transport protocol (MQTT) is one of the most dominant protocols in most fields of smart applications, so it has been chosen in this research to be a use case for implementing and evaluating a new proposed Back-off algorithm that is designed to eliminate suspicious and fake messages by calculating an initial frequent rate for each publisher connected to the MQTT broker. The proposed Back-off algorithm was designed to mitigate the traffic load of the uplink traffic by applying an exponential delay factor to suspicious publishers. Another priority scheduling algorithm was proposed to classify publishers as high priority or low priority depending on the new calculated frequent rate. The two algorithms were implemented on the Mosquitto broker and evaluated using a simulation environment by measuring specified performance metrics. The simulated results proved that the Back-off algorithm eliminated network load and introduced an acceptable range of CPU and RAM consumption. The results also concluded that the priority classification algorithm managed to reduce the latency of high-priority publishers.**

***Keywords*—Back-off algorithm; priority scheduling; MQTT protocol; average transmission frequency rate; IoT protocols**

I. INTRODUCTION

Binding the whole world became an easy mission by using new technologies with the aid of the internet. One of the most important and modern technologies that conquer all the fields of human life is the IoT (Internet of Things) in which any group of devices can be connected and communicate together without the need for any interference or intervention of humans. Accessing and controlling remote applications and remote information has become easier and faster because of IoT. IoT extended the capabilities of the internet to cut across ordinary computers by allowing smart devices or actuators to send and receive information remotely from different environments and networks.

The most important fields of human life that depend nowadays on IoT are Healthcare and wearable devices [1], where treatment and patient follow-up became easier for the medical board and for patients themselves without the need of the presence of doctors and patients in the same place, smart agriculture [2] depends on IoT as monitoring soil and factors affecting agriculture crops can help in increasing the quantity and quality of final products, smart homes[3] and smart cars where the dream of an automated life became true and modern houses are equipped with sensors which react with human to facilitate our life and help elderly [4] like door authentication scheme sensors [5], temperature sensors to adjust air conditioner [6] automatically, car sensors that allow car parking automatically [7] with only flipping a switch and other sensors that can be managed remotely with smartphones.

Due to the diversity of IoT applications in different fields of life, various types of IoT protocols are employed depending on the required function of the protocol. These protocols can gather data from sensing nodes or send data and manage communication between sensing nodes and the processing nodes depending on the function required from the protocol at every point in the network, a suitable protocol was employed [8].

One of the most widespread protocols is the Message Queuing Telemetry Transport protocol (MQTT) which is a small and lightweight messaging protocol suitable for resource-constrained and machine to machine (M2M) networks and relies on the TCP/IP protocol with a publish/subscribe model. The main function of the MQTT protocol is to gather data from sensing nodes, which are called publishers, and send them to a central intermediate device called a broker, which in turn sends this data to the required destination, which is called a subscriber.

MQTT [9] is primarily used for low bandwidth and high latency networks as it has a small fixed header of 2 bytes and depending on the publish/subscribe model which guarantees flexibility and simplicity of communication [10], it also used in loosely coupled networks as publisher and subscribers are not connected and they do not need to be available at the same time. On the contrary, publishers and subscribers do not know the availability or identification of each other. MQTT is considered to be a many-to-many protocol as many subscribers and publishers can be connected to the broker at the same time. UTF-8 string topics like mynewhouse/myroom1/temperature are used for message addressing in a hierarchal structure form that can use single-level wildcards by using + character or multilevel wildcards by using # character besides using SSL/TLS for security.

Publisher or subscriber can select one of three quality of service (QoS) levels defined in MQTT protocol depending on the employed system and network condition [11] so, message delivery assurance is performed depending on the selected level

of QoS. For QoS 0, the message is delivered at the best effort at most once without any message acknowledgment or reception assurance, so it is called "fire and forget". QoS 1 ensures message receiving at least one time. The sender keeps the message stored at its side until the reception of acknowledgment of the message from the receiver. Thus, if an acknowledgment was not received at a certain predefined time, the sender would send the same message again until receiving acknowledgment thus message could be sent several times to the receiver. QoS 2 uses 4-way handshaking for sending a message, so it is called "exactly once" because this level ensures the message receiving only one time without any duplication.

The main central device that is responsible for receiving and sending messages from publishers to subscribers is called the MQTT broker. The broker is responsible for message organization and distribution of messages among publishers and subscribers. It can handle thousands of connections simultaneously. It accepts messages from publishers then manages filters and distributes messages to the appropriate subscribers depending on the associated topics identified by subscribers. Many organizations have developed and implemented different brokers for the MQTT protocol that vary in features and programming language but all of them are made to operate with the MQTT protocol and to meet its specifications. The most famous brokers are Mosquitto which is an open-source written in C as a part of Eclipse Foundation and applicable for low powered devices because of being a very lightweight broker, RabbitMQ which is written in Erlang mainly to support AMQP protocol and MQTT protocol but it lakes some features of MQTT like QoS 2, HiveMQ is another famous MQTT broker written in Java with high and efficient performance and 100% compliance with MQTT protocol, and VerneMQ which is written in Erlang/OTP as a distributed MQTT message broker.

MQTT is implemented in different widespread applications such as Health care monitoring devices and sensors that rely on IoT technology [12], social media like Instagram, a Facebook messenger [13], energy monitoring in industrial applications [14], surveillance [15], smart farming and soil states monitoring [16], android application and smart homes[17].

II. MQTT Challenges

Despite having many advantages, such as its lightweight, simplicity of implementation, deploying it in most of the life applications, and consuming lower power than other available protocols, MQTT has many open issues and faces some challenges that may affect its performance in critical applications. Some security issues that need to be solved to enhance the performance of MQTT as mentioned in [18], data transit attacks, scalable key management, and the overload resulting from TLS are the major security problems that some new researchers are concerned with.

In [19], some security issues were discussed and some mechanisms were presented that can help to enhance data encryption, authentication, and confidentiality between clients and brokers. It also proposed a Value-to-HMAC that can be used to ensure message disclosure only by its specified client. High latency and high bandwidth consumption for constrained applications may be considered as a critical open issue of MQTT. As it relies on the TCP protocol, latency and bandwidth consumption are considered to be high because of the exchanged acknowledgments and using the triple handshake of TCP and QoS as mentioned in [20].

When comparing MQTT with Constrained Application Protocol (COAP), in the case of losses, the COAP protocol shows fewer delays than MQTT because of the TCP handshaking over heading that leads to more delays as results obtained in [21].

Another study to measure the performance of MQTT [22] was done using an NB-IoT system that provided simulation results that showed using TCP has a negative impact on the performance of the MQTT protocol when compared to the COAP protocol that uses UDP as a lightweight and cheap reliability confirmation process. This leads to the fact of adding TCP for reliability leads to less service availability than using UDP, especially when deployed with MQTT it affects the overall delay.

Most IoT applications and protocols are exposed to malicious hacking where a hacker can abuse a client or any IoT device to send fake messages only to keep the network busy and degrade the performance of the connected devices. Besides that, any sensor can be exposed to uncontrolled external factors that can affect the performance of the sensor itself, like sending the same message several times or accelerating the response and sending rate of a sensor. All that mentioned problems affect the communicating protocol and misbehave its performance, leading to more limitations that affect its performance.

MQTT-SN [23] is a new modified version of MQTT that mainly developed to operate with sensor networks was proposed to overcome the previously mentioned problem of TCP overhead work over UDP instead of TCP.

This paper contributes to proposing a new algorithm to help overcome some of the presented problems and limitations of IoT protocols, especially MQTT protocol that affects the communication delay of the overall traffic of the network. By proposing a new Back-off algorithm that organizes the communication between the broker and publishers to prevent overloading and, hence, broker failure due to unnecessary or fake messages. Besides proposing a second algorithm that can coordinate the publishing of messages between publishers depending on specified priority parameters that help critical messages to be delivered in time and reducing the latency of these messages.

III. Related Work

Because of the huge growth in the number of IoT devices and applications, the number of messages generated from IoT devices and sensors has increased. This increase leads to great congestion and packet loss in some cases, resulting in a great increase in latency, besides requiring high processing power and a high amount of consumed bandwidth. So, the whole world tends to solve these limitations by introducing new layers of computing like edge, fog, and cloud computing [24], where, cloud computing offers renting only the required amount of resources where gathered data that needs further processing can be transmitted to this layer. Transmitting data to this layer is suitable for data that needs high processing. However, it can

affect sensitive data, especially real time data, and increase its latency.

The need for intermediate processing layer leads to the fog layer where it refers to moving computers with sufficient storage and processing capabilities near to the sources of data for further processing without the need of transferring data to the cloud layer that will reduce the latency caused by transferring data to the cloud layer. So, it decreased the amount of data needed to be transferred to the cloud layer.

Due to the processing of data near to data sources, responsiveness and throughput of applications will be increased as processing data in this layer will be faster than processing it in the cloud layer. The need for edge computing will be raised as this layer will allow processing of data to be transferred near to the edge of the network, which suits the most sensitive and real time data, such as data generated from healthcare devices and sensors. Because of this advancement, not all IoT application layer protocols can operate in these modern processing layers [25]. Only special protocols have the capability of transferring data between these layers. One of them is the MQTT protocol, which can operate on constrained devices and even with cloud processing servers because of its simplicity and flexibility. To cope with these new layers of processing, it has become critical to modify MQTT and add new features to its broker.

One of these MQTT enhancements was [26], where the authors proposed a new model for MQTT edge and fog communication. That model was called the multi-tier edge computing model, in which a broker was added in the fog layer besides the primary broker in the cloud layer, where users could communicate directly with the fog layer broker rather than communicate with the cloud broker. That led to reducing the overall latency. The simulation environment was created to test the proposed model as three levels of devices were created, which are IoT devices, fog instances with the introduced intermediate broker, and cloud components with the primary broker. The simulation results were compared with the original MQTT IoT-based broker and proved that the overall latency was decreased and performance outperformed the original model.

MQTT has many new features, and modern research is concerned with enhancement of this protocol not only to upgrade its performance in IoT but also to serve the edge and fog layers. [27] Proposed a novel authentication mechanism for ensuring data privacy and integrity where the authors presented security threats to the IoT layer and MQTT attacks. When a broker is installed on all edge hubs to use the MQTT protocol in edge computing, the authority's complexity grows, and the challenge of dealing with a large number of brokers develops. Generally, IoT devices transmit a certain message to maintain availability with the broker, and this operation might generate a bottleneck due to several brokers installed on the edge hub. As a result, a system is required to supervise brokers installed on all edge hubs and to exchange data between many edge systems without further affiliation with brokers by using cryptography calculations of RSA and AES to encrypt the payload in order to make the correspondence more secure.

Another new feature of MQTT was presented in [28], where integration between blockchain and IoT systems has been done and deployed in the edge layer to obtain the advantages of blockchain decentralization in securing the IoT systems using the MQTT protocol, which in turn will increase the overall performance and security of the MQTT protocol. To control the transmission of data, the authors utilized the MQTT protocol and a central edge server as a broker. The IoT network will send and receive data via a secure link provided by blockchain.

In the field of machine learning, the MQTT protocol has attracted a great deal of attention. Some research has been concerned with attacking MQTT to overcome its security limitations by using a random forest algorithm for detecting attacks, as in [29], and other research has been concerned with generating new datasets like [30] that can help models in training to detect more attacks on the MQTT protocol.

All of the mentioned new research and new features of MQTT were concerned with security and decreasing latency, with an overall increase in the performance of the MQTT protocol. This research, on the other hand, is concerned with reducing a network's overall traffic in the event of congestion, which can result in significant packet loss. The concept of the Back-off algorithm was introduced to the MQTT broker to decrease suspicious traffic and a new mechanism of assigning priority was proposed to filter and categorize received messages.

A. Back-off Algorithm in IoT

Exponential Back-off is a prominent algorithm mainly used in networks to efficiently separate the repeated retransmission of messages or data by a random delay time depending on the slot time to eliminate network congestion. This algorithm is the organizer of retransmitted packets in the CSMA/CD after a collision is detected as it identifies the waiting interval for collisional stations after collision depending on the number of collisions and the slot time. Each collide station picks a random integer that can be presented by k from the contention window to wait a period = k * slot time. If the collision occurs in for the same packet, the contention window will be doubled. For example, if the first collision occurs, a contention window will be between 0, 1 and each station choose a random integer of it and the probability of collision will be decreased to 50%. If a collision occurs again the contention window will be doubled and become {0, 1, 2, 3} and each station will choose a random integer then the probability of collision will be decreased to 25% and so on. The contention window is doubled for each collision and the waiting time increases exponentially.

Due to the great revolution in communication systems, wireless systems need new solutions that can control congestion and delay of messages. One of these new solutions was proposed in [31] as a new algorithm that is based on the concepts of the Back-off algorithm to help in improving the MAC-layer performance by queuing the packets based on their delay. This algorithm was evaluated in a dynamic wireless sensor network where the network consists of several mobile nodes by defining a new parameter called delay timer used for reducing the number of dropped packets. Based on this parameter, packets are queued and served with a minimum

delay timer first with a reduction in energy processing and a high delivery ratio of packets.

The Back-off algorithm also has a vital role in Wireless Body Area Networks, where [32] proposed a channel switching procedure by a rescheduling algorithm based on optimal Back-off time. By identifying the neighboring list, current channels can be switched to one of its neighboring lists in the case of performance degradation.

B. Priority Scheduling in IoT

Scheduling messages based on certain criteria is a severe issue in assigning tasks to CPUs, hence the IoT extended that concept to schedule tasks and received messages in a variety of IoT applications. Some applications use the ordinary contending priority algorithms such as Frist Come First Served, Round Robin or Shortest Job First or any of other primary scheduling algorithms. Other applications imposed a modified scheme of scheduling based on the procedure applied to that application or technology.

In the transportation field [33], an application was designed to overcome the problem of traffic congestion using the IoT environment by proposing a traffic monitoring system that in turn controls the passing of vehicles depending on an assigned level of priority to each lane where high priority passing vehicles lane is assigned to the lanes with high traffic.

Smart homes have gained great attention for achieving priority scheduling among their huge number of deployed sensors and applications. The author in [34] introduced a new technique for evaluating contextual priorities that is concerned with non-functional requirements based on the end user's preferences, and context awareness. According to the current context, a context-aware system can adapt itself with the aid of a developed web platform that asks users to classify their preferences then users validate the assigned priority scheduling. As a result, users were satisfied with the tested scenarios that coped with their choice of contextual factors.

For healthcare, [35] has classified received data from healthcare sensors into two categories as emerging data that has a higher priority level or vital data that has a lower priority level to save the battery life of wearable devices as much as possible. An efficient routing protocol based on these two categories of priority classification was proposed to deliver high-priority data with direct communication. In contrast, low priority data will be delivered using multi-hop communication.

Priority scheduling is one of the big open issues of the MQTT protocol because it does not have any priority algorithm of its common brokers. The author in [36] proposed a priority algorithm in which messages were classified into three categories. Based on the category, messages were classified into three queues as normal or critical or urgent queues inside the broker itself. Messages in the urgent queue have the highest priority to be served first, which causes the latency of these messages to become smaller and the message loss rate is decreased. However, this algorithm was concerned only with the latency and the loss rate of urgent messages and ignored the latency of the overall network and the consumed memory assigned to each queue.

As mentioned before, MQTT has no priority algorithms for message scheduling. Even the proposed [36] algorithm is concerned only with the urgent messages, not the overall performance of the protocol. Besides, the priority level was assigned by the client itself that allows any message to be urgent without any constraints or predetermined specifications.

IV. Integrated Back-off with Priority Scheduling Algorithm

A. Proposed Back-off Algorithm

The proposed exponential Back-off algorithm aims to reduce network congestion by slowing down the transmission rate of suspicious devices. For each client connected to the broker, the broker will record the arrival time of the first message then repeat that for the next N messages. The time interval length between every two consecutive messages will be calculated to obtain an average frequent rate for each client. After a chosen N messages, the broker will have a saved average of the publishing frequent rate for each client. When the publisher asks the broker to send a new message, the broker compares the current publishing frequent rate with the initial average frequent rate of that publisher. If the current rate is higher than the initial rate, then that publisher may have a problem or be under attack. So the broker activates an exponential Back-off algorithm to hold on receiving from that client until a specified waiting time depending on a calculated delay factor based on the current frequent rate of that publisher. The exponential delay continues with a publisher whenever the current publishing rate became until the publisher reaches its original frequent publishing rate. That delay will reduce communication between these publishers and reduce network overload. Fig. 1 shows the flow chart of the proposed Back-off algorithm steps.

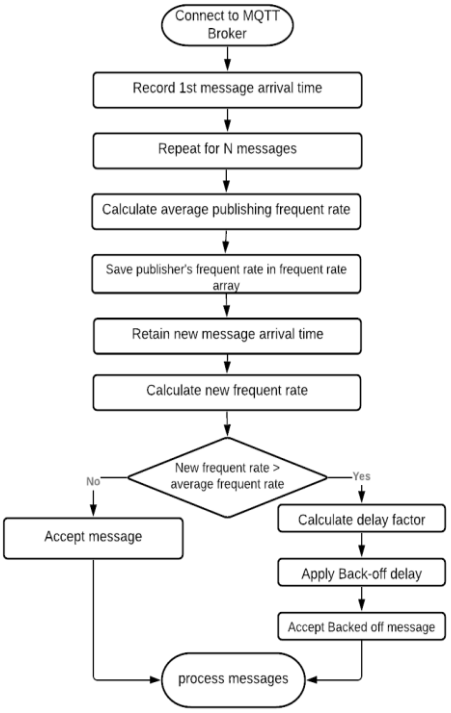


Fig. 1. Flowchart of Proposed Back-off Algorithm Steps.

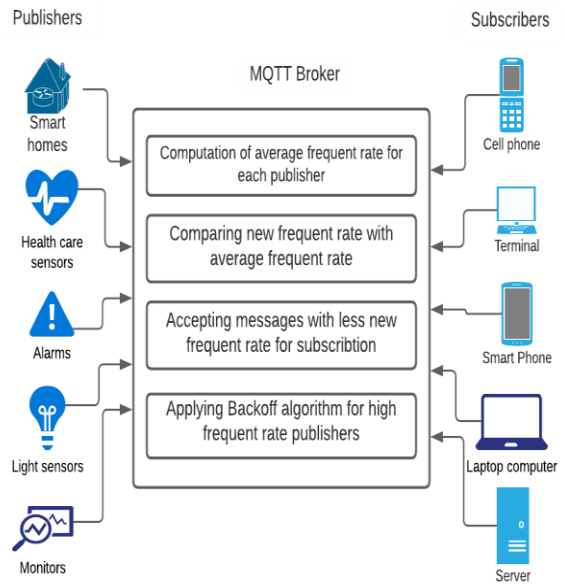


Fig. 2. The Proposed MQTT Back-off based Broker Structure.

The proposed Back-off algorithm can be deployed into the Mosquitto broker for the MQTT protocol to test the performance of the network under the new Back-off algorithm. The Mosquitto broker, the most common broker for the MQTT protocol was chosen because of its simplicity and its lightweight as it is written in C language. Fig. 2 shows the proposed MQTT Back-off based broker structure as it is modified according to the proposed Back-off algorithm. The detailed steps of the proposed Back-off algorithm were presented sequentially in Algorithm1 under the name of enhanced Back-off algorithm as it differs from the original CSMA/CD Back-off algorithm in the deployed layer, function, and execution.

Algorithm1: Proposed Back-off algorithm.

INPUT: Messages received from publisher M_1, M_2, M_3, .M_K, Number of messages selected for calculating average frequent rate N.

OUTPUT: Delay factor D_f, Average frequent rate FR_{avg}, New frequent rate FR_{new}

00 For each publisher P_j
01 For $M_{i=1}$ to $M_{i=N}$
02 Save T_{Mi}
03 Calculate Time interval $I_{(Mi, Mi+1)} += T_{M(i+1)} - T_{Mi}$
04 End for
05 $FR_{avg} = 1/[I_{(Mi, Mi+1)} / (N-1)]$
06 Save FR_{avg} in Array FR_{Pj} [ ]
07 End for
08 For each new message i = N +1 to i = k
09 $FR_{new} = (1/ I_{(Mi, Mi+1)})$
10 If $FR_{new} > FR_{avg}$
11 Calculate D_f for the current message M_i
12 Set waiting time for $M_i = D_f$
13 Accept Backed off message for subscribing
14 Else
15 Accept message for subscribing
16 process accepted messages
17 End If
18 Return D_f, FR_{avg}, FR_{new}

B. Back-off Delay Factor Calculations

The Back-off delay factor depends on the current frequent rate as it is an exponential function of the current frequent rate where DF represents the Back-off delay factor. Whenever the current frequent rate increases, DF for this publisher will increase until the publishing rate degrades to the original frequent rate. Hence, the Back-off delay factor reaches its threshold or its maximum value then terminates the Back-off algorithm and begins again if the new frequent rate exceeds the average frequent rate.

Suppose a publisher P sends a number of messages K and M represents a sequence of published messages [M1, M2, M3,….MK]. Each message arrives at Time T where T_{Mi} is the arrival time saved for message Mi. For the first N messages, an interval of time I between every two consecutive messages was calculated in seconds to obtain the average frequent rate FRavg of publisher P where I can be calculated from (1).

$$I_{Mi,(Mi+1)} = T_{M(i+1)} - T_{Mi} \quad (1)$$

Let N =4, where the number of messages selected by the user to calculate the average frequent rate for each publisher. Then three intervals of time I1, 2 , I2,3 , I3,4 will be calculated to get the average frequent rate in messages per one second from (2).

$$FRavg = 1/ ((\sum_{i=1}^{N} I_{Mi,(Mi+1)})/(N-1)) \quad (2)$$

After calculating the average frequent rate, new messages from P will be received. To calculate its current frequent rate, let FRnew is the new current rate that can be calculated from (3) where $I_{MN,(MN+1)} = T_{(M(i=N+1))} - T_{(M(i=N))}$ that represents the new time interval between the new message M_{N+1} and the previous message M_N.

$$FRnew = (1/ I_{MN,(MN+1)}) \quad (3)$$

After calculating FRnew for the new message, it will be compared with FRavg. If FRnew exceeds FRavg then a delay factor Df can be calculated as an exponential function of FRnew in (4) will be added to that publisher's message.

$$Df = e\ (FRnew) \quad (4)$$

For example, if publisher P sends 5 messages per 30 seconds at a regular rate, by using the mentioned equations FRavg will be 0.16 messages per second. If the publisher continued with the same rate or less than that rate, the Back-off algorithm will be inactive. If FRnew > FRavg, the Back-off delay factor will be calculated and applied to the message in turn.

Table I shows the effect of changing frequent rate on the delay factor with different increased frequent rates for the same publisher until reaching the maximum value of delay. Fig. 3 shows the exponential increase of delay factor based on the new calculated frequent rate until reaching the maximum allowed delay value.

C. Proposed Priority Scheduling Algorithm

Previously mentioned that MQTT protocol has no methodology for priority scheduling messages as any message received will be forward directly irrespective of its priority level. So, if 2 messages arrived at the same time which one will

be processed first, this decision never exists in the MQTT broker as there is no priority scheduling.

Based on the calculated average frequent rate recorded previously in the broker of the MQTT protocol, K number of publishers can be classified into levels based on the average frequent rate as the higher frequent rate is assigned the lower priority level donated by PRL as mentioned in (5) and the lower frequent rate is assigned a higher priority level donated by PRH as mentioned in (6) where Pi $_{FRavg}$ is the average frequent rate for publisher Pj.

$$PRL = MIN \{P1_{FRavg}, P2_{FRavg}, P3_{FRavg},\ldots PK_{FRavg}\} \quad (5)$$

$$PRH = MAX \{P1_{FRavg}, P2_{FRavg}, P3_{FRavg},\ldots PK_{FRavg}\} \quad (6)$$

For example, a sensor that sends one message every 24 hours has a higher priority than a sensor that sends a message every one second. As in the case of congestion, the first sensor's data may be lost and cannot be retrieved or resent unless the next 24 hours be over. Depending on the factor of original publishing frequent rate, the arrived messages are arranged in a queue for processing based on the assigned priority level.

This algorithm can be implemented with the Back-off algorithm to organize the overall network communication, where a network administrator can control the activation of this algorithm depending on the nature of connected devices, as the main goal of this algorithm is to assign priority to the connected devices from the broker's side, not from the client's side, where any hacker cannot assign himself a high priority.

TABLE I. DELAY FACTOR VARIATION ACCORDING TO INCREASING IN NEW FREQUENT RATE

Number of messages per seconds	$I_{Mi, (Mi+1)}$	FR_{new}	D_f
10 messages/30 seconds	I_1= 3 seconds	0.3	1.3 seconds
20 messages/30 seconds	I_2= 1.5 seconds	0.6	1.82 seconds
30 messages/30 seconds	I_3= 1 seconds	1	2.7 seconds
40 messages/30 seconds	I_4 = 0.75 seconds	1.3	3.66 seconds
50 messages/30 seconds	I_5 = 0.6 seconds	1.6	4.95 seconds
60 messages/30 seconds	I_6 = 0.5 seconds	2	7.38 seconds

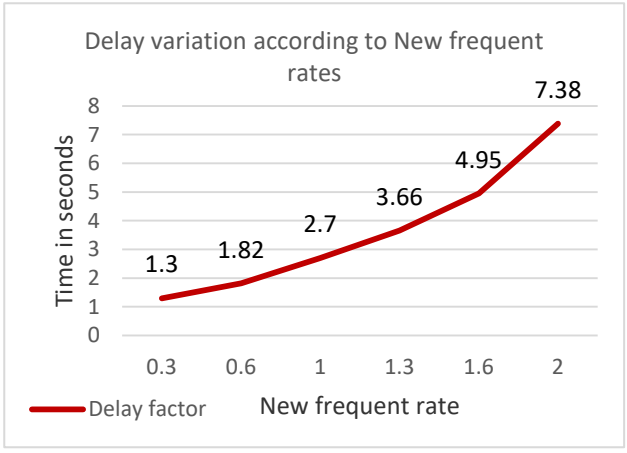


Fig. 3. The Exponential Growth of Delay Factor According to Increased New Frequent Rate.

Algorithm2: Priority scheduling algorithm based on frequent rate

INPUT: Average frequent rate Array for publishers {P1, P2, P3…P_x} FR_{Pj} [ ]

OUTPUT: priority level of publishers PR_{pj}

Recall FR_{Pj} [ ] from Back-off algorithm
For each publisher p_j
Get Max of FR_{Pj} [ ]
Get Min of FR_{Pj} [ ]
If P_j FR_{avg}> Max of all items of Array FR_{Pj} [ ]
  Set priority level of P_j = PR_L
  Set location of P_j FR_{avg} = last item of Array FR_{Pj} [ ]
Else if P_j FR_{avg} < Min of all items of Array FR_{Pj} [ ]
  Set PR_{Pj} = PR_H
  Set location of P_j FR_{avg} = 1st item of Array FR_{Pj} [ ]
Else
  Sort Array FR_{Pj} [ ]
  Set PR_{Pj} = location of P_j FR_{avg} in Array FR_{Pj} [ ]
End if
Increment of j
Return PR_{pj}

V. EVALUATION AND RESULTS

The open source Mosquitto version 1.6.12 broker was chosen for evaluation and implementation of the new proposed algorithm as it offers free, simple, and open-source libraries written in C language that helped in modifying the broker source to deploy the new algorithm in it. The Mosquitto Broker was implemented on windows 10 pro, Intel® core ™ i7 machine with 16 GB RAM and 64-bit operating system. With the aid of open-source libraries supported by the Eclipse Paho project, the publishers and subscribers were implemented on the same machine.

Simulation experiments were done on a variable number of publishers and subscribers in each experiment. Starting from only 2 publishers reaching 100 publishers, the performance metrics were measured for each experiment respectively. The Wireshark which is a network tracer program was used to capture network traffic and consumed bandwidth. Consumed CPU and RAM were measured and captured on the same workstation using the jconsole application.

A. Network Traffic Load

The most important metric to be traced and measured was the network traffic, especially from the publisher's side, which is uplink traffic, because it is the main issue that this paper is concerned with to eliminate suspicious and undesired traffic from the publisher side that affects the performance of the MQTT broker. Fig. 4 illustrates the uplink traffic for discrete experiments using individual 2, 5, 10, 20, 30, 50, 100 publishers. Each experiment was traced for 10 minutes resulting in the number of bytes transferred in this specified period. For the Back-off MQTT broker, publishers were set up to publish the first 4 messages regularly at constant frequent rates then random intervals of time between messages were inserted to create suspicious publishers and force the Back-off algorithm to

be activated. Compared with the original MQTT broker, the occupied traffic was decreased in the Back-off broker by filtering out the fast rate messages from suspicious publishers. As shown in Fig. 4 for the original MQTT broker, whenever the number of publishers becomes larger the load on the broker becomes heavier and the network becomes exposed to congestion. In contrast with Back-off MQTT, the network is not exposed to congestion and still can serve a larger number of publishers than the original MQTT broker.

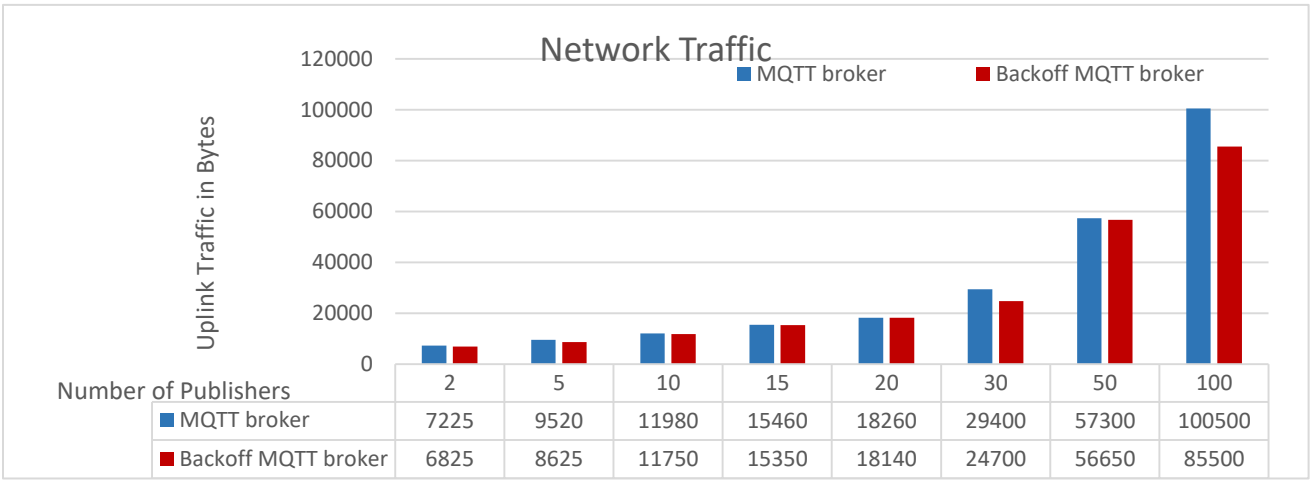
According to Fig. 4, as the load becomes heavier on the broker due to the increasing number of connected publishers, the Back-off broker can manage traffic and accept a higher number of publishers. However, the original MQTT broker suffers from congestion and a high load of unwanted messages that raise the network traffic. Taking 30 publishers as a use case experiment for evaluating the new algorithm, the MQTT broker consumed 29400 bytes in 10 minutes, whereas Back-off MQTT consumed 24700 bytes in 10 minutes.

B. CPU Load

The second metric to be measured is the used CPU during four whole minutes. It is expected that the processing power for calculating the Back-off algorithm will be increased because of the sophisticated calculation of temporal frequent rate for each new message. However, the simulation results in Fig. 4 show that a slight increase in processing power can be equal to less than 0.75 % percent, which emphasizes that the Back-off broker can be implemented in IoT applications and resource-constrained devices. Fig. 5 shows that the maximum consumed processing power for the original MQTT broker was 2.53% whereas the Back-off MQTT broker consumed 3.53% where the difference is less than 1% that IoT devices can handle.

C. Consumed RAM

The third metric to be measured is the consumed RAM for the Back-off MQTT broker and the original MQTT broker. Fig. 6 compares the consumed RAM for 30 connected publishers in both cases with a table that shows the exact value of RAM consumption. The figure shows that the consumed RAM is approximately equal in both cases despite using the calculations of delay factors and saving the arrival time of N messages to calculate and save frequent rates. This proves that the Back-off broker utilizes small RAM like the original broker and can be applied to IoT devices easily.



Number of Publishers	2	5	10	15	20	30	50	100
MQTT broker	7225	9520	11980	15460	18260	29400	57300	100500
Backoff MQTT broker	6825	8625	11750	15350	18140	24700	56650	85500

Fig. 4. Uplink Traffic for MQTT broker and Back-off MQTT Broker.

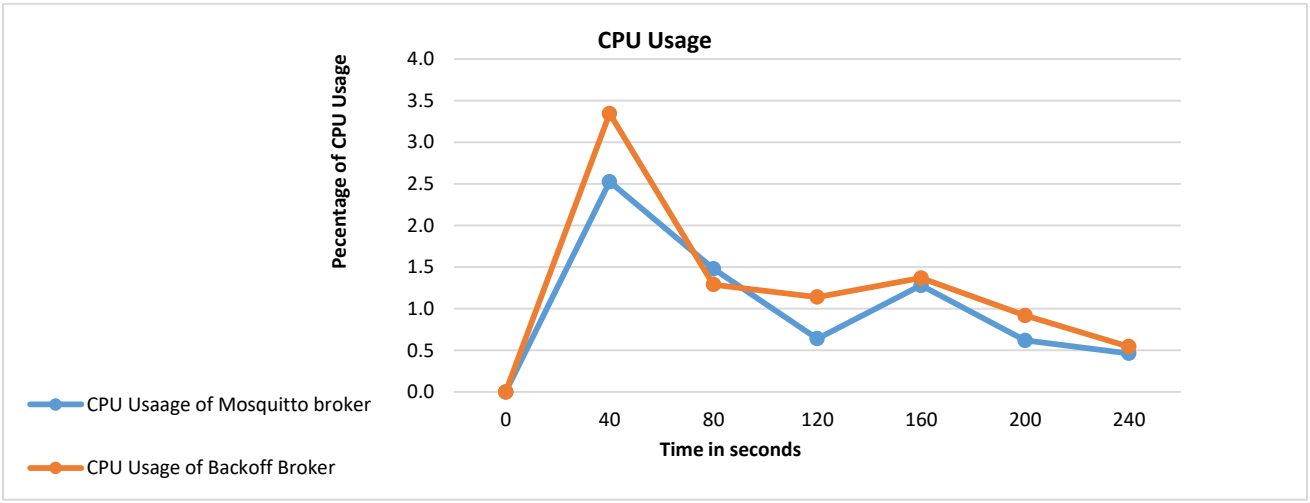


Fig. 5. CPU Consumption for MQTT Broker and Back-off based Broker.

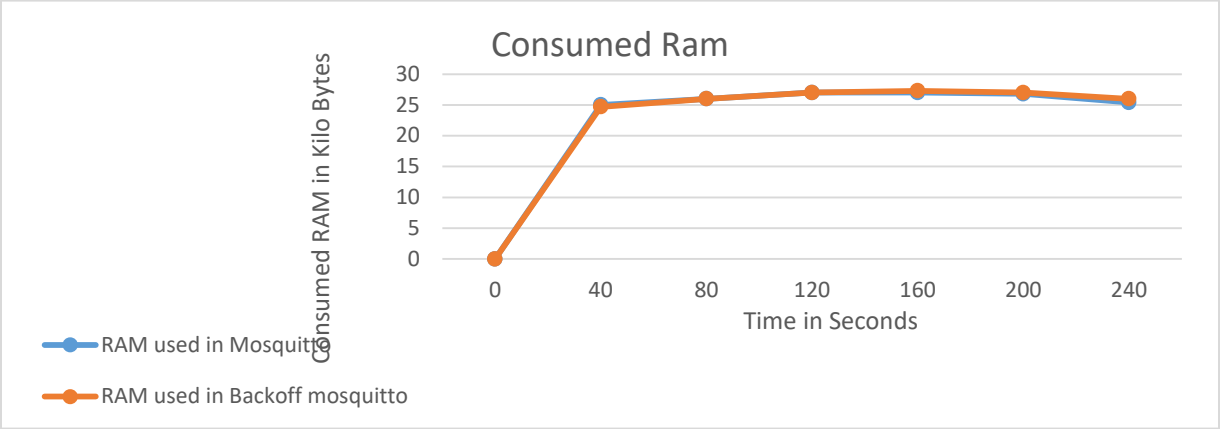


Fig. 6. RAM Consumption for MQTT Broker and Back-off Broker.

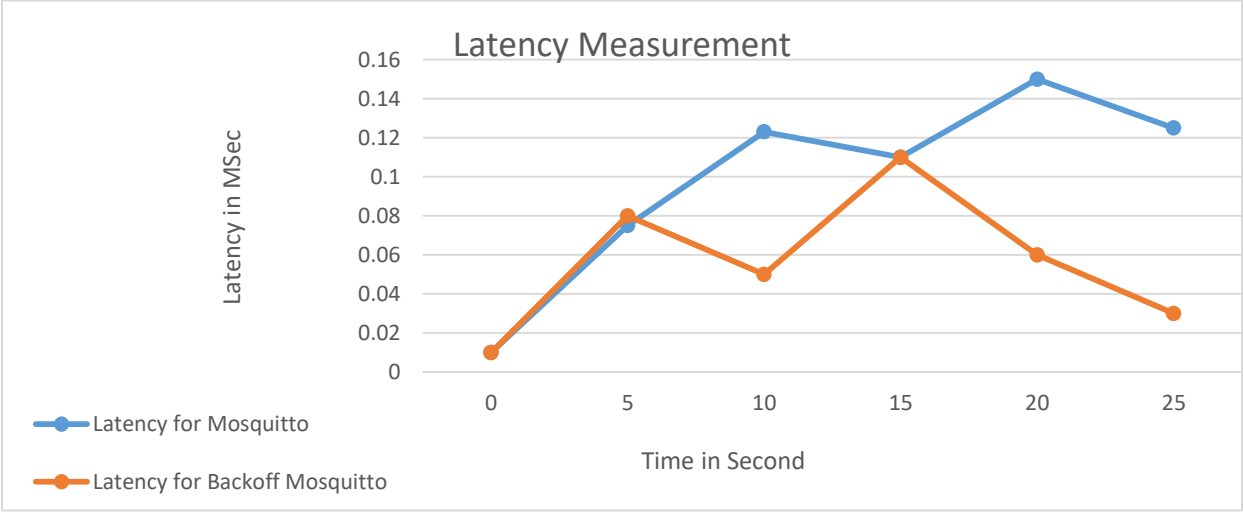


Fig. 7. The Latency Variation for High Priority Publisher in MQTT Broker and Back-off Broker.

D. Latency

The latency was the most important metric to evaluate the proposed priority scheduling mechanism as the high-priority publishers should have fewer latency measurements. The experiment was done to measure the latency of messages for one publisher with a high priority level in the MQTT broker and was repeated for the Back-off MQTT broker. Fig. 7 shows that the same publisher was exposed to less latency in Back-off Mosquitto with priority scheduling than the latency measured by the original MQTT broker. As shown, the maximum latency that was recorded for the Back-off broker was equal to the minimum latency recorded by the original broker. As a result, high-priority publishers can publish with less latency than other low-priority publishers, as they have the priority of publishing their message immediately.

VI. Conclusion

This research paper proposed a new Back-off algorithm that was designed to eliminate the effect of network and traffic congestion in IoT protocols. The problem was driven by suspicious clients or clients with undetected errors that affect the performance of the network. The MQTT protocol was chosen to be tested and evaluated under the new Back-off algorithm. Some performance metrics such as uplink traffic showed better traffic performance and less congestion than the original broker. Also, the CPU and RAM consumption were measured to record approximate results as the original broker that proved the ability to deploy that algorithm in resource-constrained devices. Another algorithm for priority scheduling was designed specially to cope with the new Back-off algorithm and the MQTT broker as it does not possess any priority scheduling algorithms. The experimental results recorded less latency for the high-priority publisher in the Back-off broker than the original broker.

Generally, the proposed Back-off and priority scheduling algorithms showed an acceptable result for RAM and CPU consumption with a minimum traffic load that leads to the ability to be employed in constrained resource devices.

VII. Future Work

The MQTT protocol was chosen to be a use case for evaluating the new algorithm because of its simplicity and its prevalence. However, the new proposed algorithm can be employed in another IoT protocol to increase its performance. AMQP has a similar structure to the MQTT protocol and it relies on distributing messages in queues that can help in

deploying the priority based on frequent rate algorithm for this protocol easily. Besides AMQP, the COAP protocol was designed for resource-constrained devices that can afford the implementation of the new proposed algorithm with higher performance metrics.

REFERENCES

[1] Arefin, ASM Shamsul, KM Talha Nahiyan, and Mamun Rabbani. "The basics of healthcare IoT: Data acquisition, medical devices, instrumentations and measurements." A Handbook of Internet of Things in Biomedical and Cyber Physical System. Springer, Cham, 2020. 1-37.

[2] Farooq, Muhammad Shoaib, et al. "Role of IoT technology in agriculture: A systematic literature review." Electronics 9.2 (2020): 319.

[3] Zaidan, A. A., and B. B. Zaidan. "A review on intelligent process for smart home applications based on IoT: coherent taxonomy, motivation, open challenges, and recommendations." Artificial Intelligence Review 53.1 (2020): 141-165.

[4] Sokullu, Radosveta, Mustafa Alper Akkaş, and Eren Demir. "IoT supported smart home for the elderly." Internet of Things 11 (2020): 100239.

[5] Kumar, Pankaj, and Lokesh Chouhan. "A secure authentication scheme for IoT application in smart home." Peer-To-Peer Networking And Applications 14.1 (2021): 420-438.

[6] Ramschie, Ali AS, Johan F. Makal, and Veny V. Ponggawa. "Implementation of the IoT Concept in Air Conditioning Control System Base on Android." International Journal of Computer Applications 975: 8887.

[7] Sarangi, Manisha, et al. "IoT aware automatic smart parking system for smart city." Cognitive Informatics and Soft Computing. Springer, Singapore, 2020. 469-481.

[8] Pandya, Hetal B., and Tushar A. Champaneria. "Notice of Removal: Internet of things: Survey and case studies." 2015 international conference on electrical, electronics, signals, communication and optimization (EESCO). IEEE, 2015.

[9] Standard, O. A. S. I. S. "MQTT version 3.1. 1." URL http://docs. oasis-open. org/mqtt/mqtt/v3 1 (2014).

[10] Soni, Dipa, and Ashwin Makwana. "A survey on mqtt: a protocol of internet of things (iot)." International Conference On Telecommunication, Power Analysis And Computing Techniques (ICTPACT-2017). Vol. 20. 2017.

[11] Archana, E., et al. "A formal modeling approach for QOS in MQTT protocol." Data Communication and Networks. Springer, Singapore, 2020. 39-57.

[12] Kadhim, Kadhim Takleef, et al. "Monitor human vital signs based on IoT technolgy using MQTT protocol." AIP Conference Proceedings. Vol. 2290. No. 1. AIP Publishing LLC, 2020.

[13] Detti, Andrea, Ludovico Funari, and Nicola Blefari-Melazzi. "Sub-linear scalability of mqtt clusters in topic-based publish-subscribe applications." IEEE Transactions on Network and Service Management 17.3 (2020): 1954-1968.

[14] Ramelan, A., et al. "IoT Based Building Energy Monitoring and Controlling System Using LoRa Modulation and MQTT Protocol." IOP Conference Series: Materials Science and Engineering. Vol. 1096. No. 1. IOP Publishing, 2021.

[15] Norrdine, Abdelmoumen, et al. "MQTT-Based Surveillance System of IoT Using UWB Real Time Location System." 2020 International Conferences on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics). IEEE, 2020.

[16] Mandal, Santanu, Imran Ali, and Sujoy Saha. "IoT in Agriculture: Smart Farming Using MQTT Protocol Through Cost-Effective Heterogeneous Sensors." Proceedings of International Conference on Frontiers in Computing and Systems. Springer, Singapore, 2021.

[17] Eleyan, Amna, and Joshua Fallon. "IoT-based Home Automation Using Android Application." 2020 International Symposium on Networks, Computers and Communications (ISNCC). IEEE, 2020.

[18] Frustaci, Mario, et al. "Evaluating critical security issues of the IoT world: Present and future challenges." IEEE Internet of things journal 5.4 (2017): 2483-2495.

[19] Dinculeană, Dan, and Xiaochun Cheng. "Vulnerabilities and limitations of MQTT protocol used between IoT devices." Applied Sciences 9.5 (2019): 848.

[20] Hamdani, Samer, and Hassan Sbeyti. "A Comparative study of CoAP and MQTT communication protocols." 2019 7th International Symposium on Digital Forensics and Security (ISDFS). IEEE, 2019.

[21] Prabhu Kumar, P. C., and G. Geetha. "Web - cloud architecture levels and optimized MQTT and COAP protocol suites for web of things." Concurrency and Computation: Practice and Experience 31.12 (2019): e4867.

[22] Larmo, Anna, Antti Ratilainen, and Juha Saarinen. "Impact of coap and mqtt on nb-iot system performance." Sensors 19.1 (2019): 7.

[23] Stanford-Clark, Andy, and Hong Linh Truong. "Mqtt for sensor networks (mqtt-sn) protocol specification." International business machines (IBM) Corporation version 1.2 (2013).

[24] Bierzynski, Kay, Antonio Escobar, and Matthias Eberl. "Cloud, fog and edge: Cooperation for the future?." 2017 Second International Conference on Fog and Mobile Edge Computing (FMEC). IEEE, 2017.

[25] Dizdarević, Jasenka, et al. "A survey of communication protocols for internet of things and related challenges of fog and cloud computing integration." ACM Computing Surveys (CSUR) 51.6 (2019): 1-29.

[26] Veeramanikandan, M., and Suresh Sankaranarayanan. "Publish/subscribe based multi-tier edge computational model in Internet of Things for latency reduction." Journal of parallel and distributed computing 127 (2019): 18-27.

[27] Pandya, Sharnil, et al. "A Novel Multicast Secure MQTT Messaging Protocol Framework for IoT-Related Issues." Proceedings of Second International Conference on Computing, Communications, and Cyber-Security. Springer, Singapore, 2021.

[28] Abdullah, Maha A., and Omar H. Alhazmi. "A Triumvirate Approach of Blockchain MQTT and Edge Computing Toward Efficient and Secure IoT." Proceedings of International Conference on Communication and Computational Technologies. Springer, Singapore, 2021.

[29] Chunduri, Naga Venkata Hrushikesh, and Ashok Kumar Mohan. "A Forensic Analysis on the Availability of MQTT Network Traffic." International Symposium on Security in Computing and Communication. Springer, Singapore, 2020.

[30] Vaccari, Ivan, et al. "MQTTset, a new dataset for machine learning techniques on MQTT." Sensors 20.22 (2020): 6578.

[31] Babu, Palamakula Ramesh, et al. "An enhanced virtual backoff algorithm for wireless sensor networks." International Journal of Wireless and Mobile Computing 13.3 (2017): 179-187.

[32] Xie, Zhijun, et al. "An Optimal Backoff Time-Based Internetwork Interference Mitigation Method in Wireless Body Area Network." Journal of Sensors 2020 (2020).

[33] Nagmode, Varsha Sahadev, and S. M. Rajbhoj. "An IoT platform for vehicle traffic monitoring system and controlling system based on priority." 2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA). IEEE, 2017.

[34] Serral, Estefanía, et al. "Contextual requirements prioritization and its application to smart homes." European conference on ambient intelligence. Springer, Cham, 2017.

[35] Bahattab, Abdullah Ali, Abdelbasset Trad, and Habib Youssef. "PEERP: An Priority-Based Energy-Efficient Routing Protocol for Reliable Data Transmission in Healthcare using the IoT." Procedia Computer Science 175 (2020): 373-378.

[36] Kim, Yong-Seong, et al. "Message queue telemetry transport broker with priority support for emergency events in Internet of Things." Sensors and Materials 30.8 (2018): 1715-1721.