International Journal of Web Engineering and Technology

ISSN online: 1741-9212 - ISSN print: 1476-1289
https://www.inderscience.com/ijwet

PR-MQTT: a novel approach for traffic reduction and message prioritisation in IoT applications

Jiby J. Puthiyidam, Shelbi Joseph

DOI: 10.1504/IJWET.2024.10061282

Article History:

Received:	13 June 2023
Last revised:	17 October 2023
Accepted:	28 October 2023
Published online:	29 April 2024

PR-MQTT: a novel approach for traffic reduction and message prioritisation in IoT applications

Jiby J. Puthiyidam* and Shelbi Joseph

School of Engineering,
Cochin University of Science and Technology,
Kochi, Kerala, India
Email: jibyjp@cusat.ac.in
Email: shelbi@cusat.ac.in
*Corresponding author

Abstract: IoT applications often involve devices with limited processing power, memory capacity, and low resource consumption. The vast number of devices connected to IoT networks generates massive amounts of data, making effective data management crucial for IoT applications. IoT applications prefer lightweight messaging protocols over the standard internet protocol, hyper text transfer protocol (HTTP). Message queue telemetry transport (MQTT) has emerged as a popular communication protocol for IoT applications. In some instances, specific messages may hold greater importance than others. However, most standard IoT protocols lack inherent mechanisms to prioritise incoming messages. This paper presents a new approach to reducing network traffic in IoT applications by selectively transmitting messages while prioritising the processing of urgent messages. The proposed method is integrated with HBMQTT, an MQTT broker. The experimental evaluation indicates that with the proposed PR-MQTT broker, the latency of the priority messages remains nearly constant, irrespective of the message's index position. Priority messages are consistently delivered within 10–15 milliseconds, resulting in a speed improvement of over 90% compared to regular messages. Additionally, the proposed approach reduces CPU resource utilisation and network traffic by 25% and the transmission delay of normal messages by 50%.

Keywords: internet of things; IoT; MQTT protocol; message priority; IoT networks; network traffic.

Reference to this paper should be made as follows: Puthiyidam, J.J. and Joseph, S. (2024) 'PR-MQTT: a novel approach for traffic reduction and message prioritisation in IoT applications', *Int. J. Web Engineering and Technology*, Vol. 19, No. 1, pp.44–66.

Biographical notes: Jiby J. Puthiyidam is a research scholar at the Division of Information Technology, School of Engineering, Cochin University of Science and Technology (CUSAT), Kochi, India. He has completed his Post-Graduation (MTech in Computer and Information Science) from the Cochin University of Science and Technology (CUSAT). He is an Assistant Professor at Government Model Engineering College, Ernakulam, India. His research interests include internet of things, IoT network security, lightweight cryptography, data mining and machine learning.

Shelbi Joseph is a Professor at the Division of Information Technology at School of Engineering, Cochin University of Science and Technology, Kochi, India. He has completed his Post-Graduation (MTech in Computer Science and Engineering) from the National Institute of Technology, Tiruchirappalli, India and PhD in Software Reliability from the Cochin University of Science and Technology. His areas of interest are software engineering, software reliability, open source software, big data, data mining, and IoT. He has several publications in national and international journals and conference proceedings to his credit.

This paper is a revised and expanded version of a paper entitled 'Prioritization of MQTT messages: a novel approach' presented at International Conference on Communication, Networks and Computing, ITM University, Gwalior, 8–10 December 2022.

1 Introduction

Advancements in computing and communication technology have facilitated the connection of an increasing number of devices to the internet, giving rise to the revolutionary concept of the internet of things (IoT). In an IoT environment, various components such as physical devices, software components, and cloud-based services communicate over a network to provide specific services or functions (Wazid et al., 2020). An IoT application comprises three main parts:

1 IoT devices

2 an IoT communication protocol

3 systems for storing and processing IoT data (Bayılmış et al., 2022).

The IoT serves as a platform where everyday devices become smarter, processing becomes intelligent, and communication becomes informative (Houimli et al., 2021; Ray, 2018). The application domains of IoT are extensive and encompass healthcare, manufacturing, industrial operations, transportation, smart homes, and agriculture (Ashima et al., 2022).

The success of IoT technology heavily relies on effective machine-to-machine (M2M) (Lawton, 2004) communication over the internet. IoT applications involve devices with limited storage, processing, and networking capabilities, often running on batteries (Yugha and Chithra, 2020). Limitations inherent in smart devices often lead to congestion in IoT networks, resulting in reduced performance and data loss. Therefore, it becomes imperative to implement a congestion control mechanism to enable efficient data transmission within IoT networks (Jain et al., 2022; Anitha et al., 2023). The commonly used internet communication protocol, hyper text transfer protocol (HTTP) (Bressoud et al., 2020), is unsuitable for such resource-constrained devices. Consequently, specialised communication protocols like constrained application protocol (CoAP)(Alhaidari and Alqahtani, 2020), message queue telemetry transport (MQTT) (OASIS Standard Incorporating Approved Errata, 2015), and advanced message queuing protocol (AMQP) (Uy and Nam, 2019) are employed in IoT applications to manage these constraints effectively.

Numerous sensors and devices are involved in various real-life IoT applications. The substantial volume of data generated and transferred by these devices can impact the performance of the IoT network, leading to reduced throughput and increased latency (Singh et al., 2022). Furthermore, not all input data generated by IoT devices hold equal importance. Certain messages or messages from specific sources may be more critical and necessitate immediate attention and handling. For instance, messages from a fire alarm or gas leak sensor carry higher significance than messages from an atmospheric temperature or pressure sensor. However, the standard MQTT protocol lacks a built-in mechanism for message prioritisation, which poses a challenge (Tatyasaheb and Kumar, 2021). The published contents have limited time for residing on the server before it is discarded or replaced (Ali and Zafar, 2023). To address this issue, we propose a novel message priority algorithm that introduces minimal computation overhead while prioritising input messages and reducing network traffic. The algorithm identifies a normal range for each message topic, allowing differentiation between priority and non-priority messages. Leveraging the concept of trivial intervals, the algorithm identifies identical or irrelevant messages and prevents their forwarding to the subscriber, thereby reducing network traffic. The proposed PR- MQTT broker identifies priority messages and forwards them ahead of normal messages. The algorithm is implemented within the HBMQTT broker, which is the only broker written in Python. This integration eliminates the need for a separate server to handle priority data. Experimental results demonstrate that by avoiding the transmission of identical or irrelevant data, the proposed algorithm improves the transmission time of normal messages as well.

Our major contributions include:

- identified the necessity of prioritising incoming messages in IoT protocols
- proposed a novel method to identify priority messages in IoT communications.
- proposed a method based on trivial intervals to reduce the data traffic in IoT networks.
- the MQTT broker, HBMQTT, is modified to incorporate the new priority algorithm, PR-MQTT.

The outline of this paper is as follows. Section 2 presents related works on prioritising MQTT messages. A brief description of the working of the MQTT protocol is given in Section 3. The proposed system framework is discussed in Section 4. Section 5 discusses the implementation and evaluation details and the results. Section 6 concludes our work and discusses future research directions.

2 Related works

The proliferation of IoT devices and applications has resulted in a rapid growth of data generated and transmitted across IoT networks. This exponential increase in data traffic can lead to various challenges, such as network congestion, packet loss, increased latency, and higher bandwidth consumption. To tackle these issues, researchers have proposed various approaches in the literature to enhance network performance in IoT environments. These approaches include assigning priorities to incoming messages and

employing data compression and aggregation methods to reduce network traffic. In this session, we will delve into a review of the literature covering these topics.

Kim et al. (2018a) proposed an approach that modifies the structure of standard MQTT message fixed header format for assigning message priority. This method uses two bits in byte 3 of the MQTT message header to set four priority levels. Each priority level maintains a separate message queue. The work by Kim and Oh (2017) also uses two bits in byte 2 of the message header to set the priority flag. The higher the number of the priority flag, the higher the priority. Both approaches increase the MQTT message minimum header size from two to three bytes.

Kim et al. (2018b) suggested a method that uses reserved message types (message type 0 and message type 15) of the MQTT standard packet types to denote the priority messages. The remaining 14 packet types (1–14) are used for their usual purpose. Message 0 indicates urgent messages with the highest priority, and message type 15 denotes critical messages with the second-highest priority. This approach maintains three separate queues for each priority level. The work discussed in Kim et al. (2018a) uses a message scheduling method based on weighted round robin (WRR) to prevent excessive processing delay of a queue with low priority. Methods that use a plurality of queues require more CPU resources than a single queue method. As the above-mentioned approaches modify the MQTT standard specification, these methods may be incompatible with other high-level application programmes.

Oh and Kim (2019) proposed a message priority approach that does not modify the message header. In this method, the first character of the message topic determines the message priority. For example, if the message topic begins with a predefined rarely used character such as ^, it is assumed to have higher priority than a message topic beginning with other characters. Hence the priority of a message is sent with the 'PUBLISH' message, and it does not affect the existing MQTT packet structure. The authors claim that the MQTT protocol can run with relatively low CPU resources. Chen et al. (2022) proposed a priority scheduling algorithm, which considers the popularity of message topics, as a solution to the challenge of effectively distributing messages with diverse topics in microgrids. Priority permissions are determined by the number of clients subscribing to a particular topic. Topics with more subscribers receive the highest priority and are transmitted ahead of messages from other topics. A time period is established to prevent low-priority data from experiencing long delays, and a low-priority transmission mechanism is implemented to ensure that these messages are forwarded before they reach their timeout period.

Hwang et al. (2022a) modified the Mosquitto broker to handle urgent messages efficiently. The improved Mosquitto broker, U-Mosquitto, maintains an urgent message list in addition to the subscription list of normal messages and processes the urgent message list before the subscription list.

AlEnany et al. (2021) used the back-off algorithm to calculate the average frequency rate of messages published by each publisher and assign priority to publishers based on the average publishing rate. The publisher with the highest average frequent rate (maximum delay between successive messages) is assigned the highest priority, placing its messages in the front of the queue. Publishers are sorted based on their average frequency and placed correctly in the array. This method assumes that a publisher sending messages less frequently is more important than a publisher publishing messages frequently.

Table 1 Summary of related works

Paper	*Priority method*	*Evaluation tools*	*Evaluation parameters*	*Evaluation result*	*Limitations*
AlEnany et al. (2021)	Frequent rate of messages	Mosquitto broker	Network traffic, CPU load, consumed RAM, latency	Less network traffic, improved latency	Increased CPU load, frequent rate not a good priority parameter
Kim and Oh (2017)	Set 2-bit priority flag at byte 3 of message header	Mosquitto broker, Eclipse Paho client library	End-to-end delay, latency	Lowest delay with QoS 0, improved latency	Increases header size from 2 to 3 bytes
Chen et al. (2022)	Topics with more number of subscriptions	IEDA compiler, MQTT broker	Data packet delay, distribution time period	Fast processing of priority messages, low CPU resources usage	Urgent messages from low-priority topics has no priority. A feedback scheduler is required
Oh and Kim (2019)	First character of message topic	Mosquitto broker, Raspberry Pi, Paho client	Latency with different message sizes	Fast processing of priority messages, low CPU resources usage	Separate queue for each priority level. A priority scheduler required
Jung (2020)	Threshold and tolerance values	Raspberry Pi, Arduino board	Energy usage, packet loss rate, bandwidth, transmission delay	Reduced packet loss, energy consumption and bandwidth utilisation. Improved latency	Threshold value computation not clear
Kim et al. (2018b)	Use reserved message types to set priority	Mosquitto broker, Eclipse Paho	Latency, message loss rate	Latency of urgent messages lowered by 35%	Use of reserved message types not recommended, higher loss rate
Kim et al. (2018a)	2-bit priority flag at byte 3 of message header	Mosquitto broker	Latency, message throughput	Reduced latency, improved throughput	Increases the fixed header size, maintain separate priority queues
Hwang et al. (2022a)	Priority messages stored in urgent list	Mosquitto broker, Raspberry Pi	Message delivery time	Fast delivery rate for urgent messages	need to maintain additional message list
Park et al. (2018)	Messages are classified based on characteristics	Mosquitto broker, Paho client library	Message scheduling time	Better performance than MQL algorithm	Message characteristic details are not given

Jung (2020) proposed a priority assignment scheme to reduce network traffic in a resource-constrained network. Two threshold values are identified, and if a new message external to the threshold value arrives, it is assigned the highest priority. Suppose the difference between the previously and newly measured data is within a pre-specified tolerance range. In that case, the latter is considered the same as the former and is assigned the lowest priority. The keep-alive message notification scheme is applied to transmit the lowest priority packets.

Park et al. (2018) suggested an algorithm that classifies messages based on their characteristics into three groups, namely, unconditional messages (UNC), real-time (R.T.) messages and delay-tolerant (D.T.) messages. Each message class has its message queue. UNC messages should be forwarded immediately. Whenever an R.T. message is sent, the R.T. message queue priority is decreased, and the priority of the D.T. message queue is increased. Similarly, whenever a D.T. message is sent, the priority of the D.T. message queue is decreased, and the priority of the R.T. message queue is increased. This algorithm ensures that the transmission of D.T. messages is allowed for a while.

Park et al. (2017) proposed a multiclass Q-learning algorithm (MQL) for remote monitoring of patients at home. When a new message with higher priority arrives, older messages in the queue are pushed one position backwards. An ageing technique prevents the indefinite postponement of a lower-priority message. In this work, the message priority is set at the sensor level. Oyewobi et al. (2021) have discussed a priority queuing technique to control congestion in IoT networks. This approach follows a preemptive/non-preemptive discipline where node packets are grouped and transmitted based on the real-time requirements of their IoT applications.

Donta et al. (2022) comprehensively reviewed how traditional IoT application layer protocols have been enhanced and refined. They also examined real-time applications and the corresponding adapted application layer protocols aimed at enhancing performance. The research delved into the importance of request-response and Pub-Sub protocols within various use cases. Furthermore, this paper proposes the integration of machine learning to permeate these protocols with intelligence, enabling them to adapt dynamically to varying application conditions without human intervention. An intelligent congestion control algorithm called iCoCoA, based on the CoAP protocol is introduced by Donta et al. (2023). This work employs a deep reinforcement learning approach. This algorithm is designed to effectively predict and manage congestion in dynamic environments, particularly on constrained devices. iCoCoA leverages insights from various network characteristics to make informed decisions regarding the optimal Retransmission Timeout, thereby alleviating congestion in dynamic scenarios. Furthermore, it enhances throughput, conserves energy, and reduces the occurrence of needless retransmissions when compared to existing congestion control models. Congestion-aware data acquisition (CADA), a resource control-based mechanism for wireless sensor networks, is proposed in Donta et al. (2020). CADA effectively detects congested nodes within the network and employs alternative routing paths to the base station for efficient data acquisition.

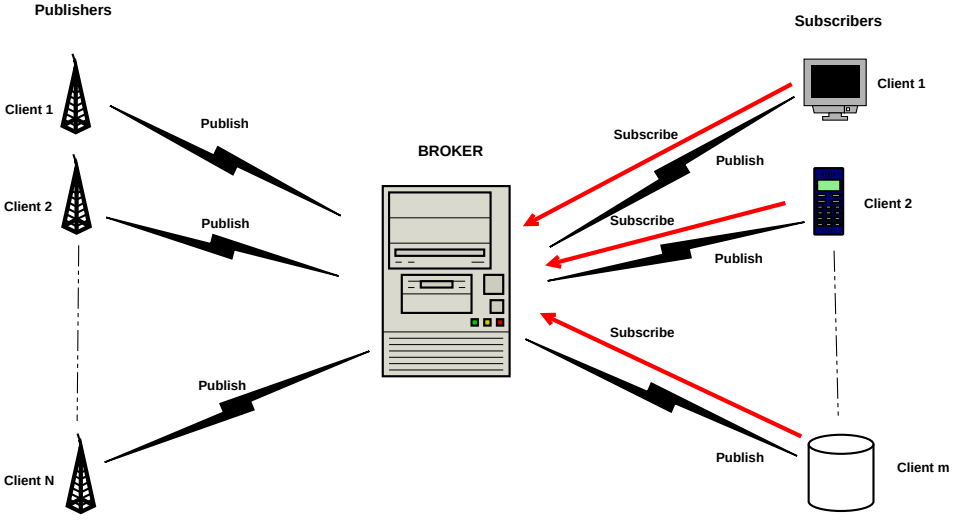
The analysis of related works in the literature reveals that prioritisation of incoming messages is an unexplored field in IoT communication protocols, with only a limited number of related works available. Most of the existing literature comprises only proposals and lacks implementation details. Most attempts in literature either assign the responsibility of identifying priority messages to the constrained client nodes or

needs the modification of standard MQTT packet structure or message header size. Such approaches are not desirable in IoT networks.

3 MQTT protocol

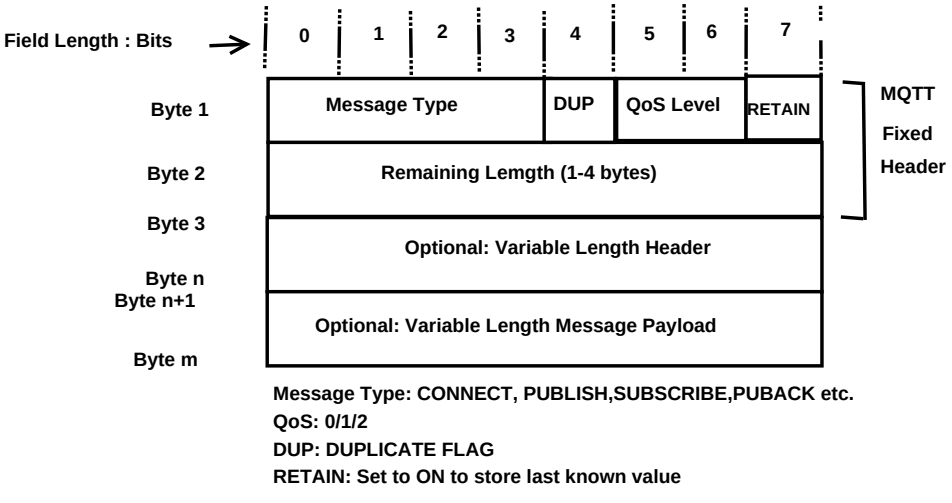
The IoT community widely adopts the MQTT protocol due to its lightweight nature, small message header size, and low bandwidth consumption (Yudidharma et al., 2023). It utilises a publish/subscribe pattern for data transportation. Built on top of the TCP/IP protocol, MQTT is well-suited for unreliable communication networks (Akshatha et al., 2022). IoT applications seeking a secure communication environment and can broadcast messages to multiple subscribers concurrently might favour the MQTT protocol (Bayılmış et al., 2022). Major public cloud platforms like Amazon Web Services, Microsoft Azure, and Google Cloud Platform leverage the capabilities of MQTT (Naik, 2017). Therefore, in this work, the MQTT protocol is chosen to implement and analyse message priority in IoT networks. The MQTT architecture consists of several key components, including a central broker server and clients acting as publishers and subscribers. The MQTT protocol facilitates the decoupling of publisher and subscriber clients (Lazidis et al., 2022). Clients communicate with the MQTT broker using message topics (Hwang et al., 2022b). Initially, clients establish a connection with the broker. Subscribing clients indicate their topics of interest to the broker when requesting a connection. Publishing clients send messages to the broker, specifying a topic. The broker filters messages based on the topic name and forwards them to the subscribed clients. Figure 1 illustrates the basic architecture of the MQTT protocol. Prominent MQTT brokers include Mosquitto broker (Mosquitto MQTT Broker, 2023), RabbitMQ (Johansson and Dossot, 2020), Hive MQ (Koziolek et al., 2020), VerneMQ (Gruener et al., 2021) and HBMQTT (Broker API Reference – HBMQTT 0.6 Documentation, 2023), among others.

Figure 1 MQTT protocol architecture (see online version for colours)



MQTT has a two-byte fixed message header, the smallest among the IoT communication protocols. The message header provides information such as the message type, various associated flags, and details of the optional fields. The structure of the MQTT message header is given in Figure 2.

Figure 2 MQTT message header format



Source: Abdul Ameer and Hasan (2020)

MQTT clients can use different quality of service (QoS) levels when sending or subscribing to messages. The QoS level determines the reliability and guarantee of message delivery that the client requires (Liu and A-Masri, 2021). The MQTT protocol offers three and levels QoS for message delivery.

1 QoS 0: at most once delivery

2 QoS 1: at least once delivery

3 QoS 2: exactly once delivery.

QoS 0 is the fastest and least reliable level, as it provides no guarantees of message delivery, and the sender does not receive an acknowledgement from the receiver. This level is suitable for scenarios where message loss is acceptable, and the communication network is reliable. QoS 1 provides guaranteed message delivery but may result in duplicate messages. The receiver acknowledges each message, and the sender will retry sending it until it receives the acknowledgement. This level is appropriate for applications that can handle duplicate messages and require guaranteed delivery. QoS 2 provides the most robust level of service, ensuring that each message is delivered only once, without loss or duplication. This level incurs higher processing and network resource costs. It is typically used in mission-critical scenarios where message loss or duplication is unacceptable.

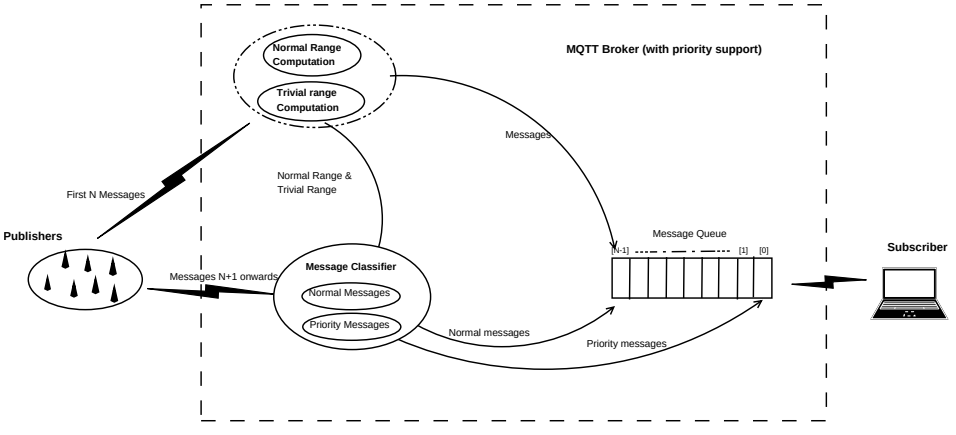
4 Proposed system framework

This section presents an innovative and efficient approach for identifying and rapidly processing high-priority messages of elementary data types in IoT applications. This algorithm not only improves message processing speed but also reduces network traffic. The proposed method addresses the limitations of most existing works in this field. The MQTT protocol, which is widely utilised in the IoT environment, is chosen for implementation.

Initially, input client nodes, acting as publishers and subscribers, establish a connection with the broker server. The publisher clients gather data from the surrounding environment and publish the collected information to the broker at regular intervals. It is assumed that a publisher node can send two types of messages: *normal messages*, which provide regular updates on the sensing environment, and *alert messages* (priority messages), which indicate a need for immediate attention. The publisher generates and publishes messages to the broker using the standard MQTT message PUBLISH command. Upon receiving a message from the publisher, the broker determines whether the incoming message is a normal or a priority message based on its value. If the message is identified as a priority message, the PR-MQTT broker places it at the front of the message queue and promptly forwards it to the subscriber, prioritising it ahead of other normal messages. On the other hand, if the incoming message is classified as normal, the broker assigns it to the next available position in the message queue.

The proposed PR-MQTT approach plays a vital role in mitigating congestion in IoT networks by reducing message transfers. If the values of two consecutive messages from a publisher are nearly identical, the PR-MQTT broker identifies them as duplicates and refrains from forwarding the latter message. Disregarding the processing of such duplicate or identical data may not significantly impact the overall application outcome but significantly improves the system's performance. This reduction in message transfer between the broker and subscriber clients substantially improves network efficiency. The framework of the proposed system is illustrated in Figure 3.

Figure 3 Proposed system framework



The suggested framework is more suitable for applications where multiple publishers publish data simultaneously and a single subscriber receives the data. This environment is ideal for many real-life IoT applications, such as remote patient monitoring (Bashir and Mir, 2021) and industrial environment, where thousands of sensors collect data about their sensing environment regularly and forward them to a central server for analysis and monitoring (Mahmood et al., 2021). This work is divided into three modules:

1 *Normal range computation:* Determine each publisher's normal message range (min-max range).

2 *Trivial range calculation:* Identify each publisher's trivial range to find identical messages and reduce traffic and congestion in the network.

3 *Message classification:* Identify priority messages and forward them ahead of normal messages. Prevent the forwarding of identical messages.

4.1 Normal range computation

This module identifies each publisher's min-max range (normal range) to distinguish high-priority messages from normal ones. Any message value outside this range is considered a priority message. The proposed algorithm monitors each publisher's first n messages and identifies the minimum and maximum values. This period can be considered a training period. The size n (training data) may vary from a few hundred messages to data of one or more days, depending on the application. When each publisher p_j publishes a message to the broker during this period, the broker updates the current minimum and maximum values of p_j using equations (1) and (2).

$$\min(p_j) = \min[p_j m^i, Curr - \min(p_j)] \tag{1}$$

$$\max(p_j) = \max[p_j m^i, Curr - \max(p_j)] \tag{2}$$

where

$\min(p_j)$	new minimum value of publisher p_j
$\max(p_j)$	new maximum value of publisher p_j
$Curr - \min(p_j)$	present minimum value of publisher p_j
$Curr - \max(p_j)$	present maximum value of publisher p_j
$p_j m^i$	i^{th} message of publisher p_j.

The minimum and maximum values identified for each publisher from its first n initial message are stored in corresponding positions in the arrays min[] and max[].

4.2 Trivial range calculation

Most messages transmitted by publishing clients contain information related to environmental factors like temperature or humidity, which typically change gradually.

Disregarding non-critical data from such sensors may not have severe consequences but significantly enhances network performance. To selectively ignore non-critical data and reduce network traffic, as well as minimise message queuing delays, our algorithm employs a trivial interval concept within the min-max region. This trivial interval, denoted as t, is calculated using equation (3).

$$t = \sum_{i=1}^{n} \frac{(m_i - m_{i-1})}{n-1} \tag{3}$$

If the value of a normal message falls within the trivial interval t with its preceding message, both messages are considered identical, and the latter message does not need to be forwarded to the broker. However, this approach may lead to skipping large number of messages from certain sensors, such as temperature or atmospheric pressure. The values of these sensors normally change very slowly. This can result in an indefinite delay for a new message from such publishers to reach the broker. Furthermore, if no message is received from a publisher for an extended period, it may cause confusion for the MQTT broker regarding the client's existence. A *skip limit* is set to prevent such situations, determining the maximum number of messages that can be skipped in a sequence. Only the specified number of messages will be skipped, as defined by the *skip limit*. The subsequent message will be forwarded to the subscriber, even if it falls within the trivial range of the previous message. The computation procedure for determining the normal range of each publisher and its associated trivial interval is provided in Algorithm 1.

Algorithm 1 *min-max* and trivial range computation

Input : publishers p_1, p_2, p_3, $\cdots$, p_k
n, number of initial messages
message queue mq[]
Output: trivial interval t and arrays min[] and max[]

for *publishers* p_1, p_2, p_3, $\cdots$, p_k **do**
  for *messages* $m^{i=1}$ *to* $m^{i=n}$ **do**
    Compute p_j(min) = minimum($p_j m^i$,p_j(min));
    Compute p_j(max) = maximum($p_j m^i$,p_j(max));
    Insert $p_j m^i$ to message queue mq[]; ▷ *note:p$_j m^i$ – i^{th} message of publisher p_j;
  end for
  Store p_j(min) in array min[];
  Store p_j(max) in array max[];
  Compute trivial interval $t = \sum_{i=1}^{n} \frac{(m_i - m_{i-1})}{n-1}$;
end for

4.3 Message classification

Message values within the min-max range are considered normal and treated using the standard MQTT message forwarding procedure. A message value outside this range is abnormal or critical, and should be treated urgently. Such a message is considered a high-priority (hpr) message.

$$hpr(p_j) = p_j m^i < \min(p_j) \text{ or } p_j m^i > \max(p_j) \tag{4}$$

A high-priority message is placed at the first position in the message queue, bypassing all normal messages; $mq[0] = hpr(p_j)$.

Algorithm 2 Identifying and processing priority messages

```
Input : publishers p1, p2, p3, ···, pk.
        trivial interval, t.
        arrays min[] and max[].
        skip limit, skip.
Output: message queue mq[] with priority messages at the front
for p1, p2, p3, ···, pk do
    for messages m^{i=1} to m^{i=n} do
        if m^i ≤ pj(min) or m^i ≥ pj(max) then
            Set m^i a high priority message;
            place m^i at first position of mq[]                ▷  ∗at mq[0]
            else if pj(min) ≤ m^i ≤ pj(max) then
                if (m^i − m^{i−1}) ≤ t then
                    if not skip limit then
                        Skip m^i
                        else
                            Set m^i as a normal priority message;
                            place m^i at the last position of mq[]  ▷  ∗at mq[n+1]
                        end if
                    end if
                    else
                        Set m^i as a normal priority message;
                        Place m^i at the last position of mq[]      ▷  ∗at mq[n+1]
                    end if
                end if
            end if
            else
                m^i is undecided;
                skip m^i;
            end if
        end if
    end for
end for
```

If the value of the new message is within the *(*min-max) range of the publisher, it is treated as a normal message (npr).

$$npr(p_j) = \min(p_j) < p_j m^i < \max(p_j) \tag{5}$$

Such normal messages are placed at the back end of the queue in the next available position; $mq[N+1] = npr(p_j)$, where N denote the number of elements present in the message queue currently. Algorithm 2 discusses the procedure to identify and process

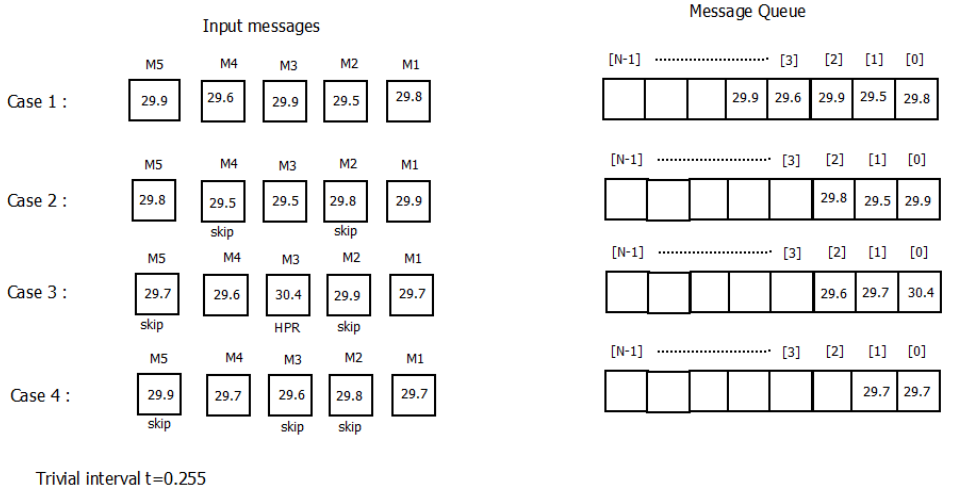
high-priority and normal messages of a publisher. This algorithm also explains how the proposed method reduces network congestion by ignoring unwanted messages.

An illustrative example is provided to explain the proposed algorithm for identifying and processing priority messages and reducing network traffic within the MQTT communication protocol. In this example, a skip limit is set as 3. This means two consecutive messages occurring within a trivial interval will be skipped, and only the third message will be forwarded to the broker. To illustrate this concept, let us consider a scenario where a temperature sensor initially publishes ten messages, which serve as its initial set of messages for determining the minimum-maximum temperature range.

Temperature (°C) (first n values)	29.8 29.5 29.6 29.8 29.6 29.9 29.5 29.8 29.6 29.9
Minimum temperature	$29.5°\text{C}$
Maximum temperature	$29.9°\text{C}$
min-max range	$29.5°C - 29.9°\text{C}$
Trivial interval (t)	$(0.3 + 0.1 + 0.2 + 0.2 + 0.3 + 0.4 + 0.3 + 0.2 + 0.3)/9]$ $= 0.255$

Based on these min, max and trivial values, the behaviour of our new message priority algorithm with various message sequences is illustrated in Figure 4.

Figure 4 Message passing performance of PR-MQTT algorithm



In Figure 4, case 1 illustrates a scenario in which all messages are categorised as normal messages, and there are no instances of two consecutive messages falling within the trivial range. In case 2, certain messages are omitted when they occur within the trivial interval of the preceding message. Case 3 demonstrates how the proposed algorithm handles high-priority messages and messages within the trivial interval. Similarly, in case 4, it showcases how the algorithm effectively reduces network traffic by omitting redundant messages.

5 Results and discussion

In this section, the simulation details of the proposed system are presented. The primary purpose of the experiment is to prove that the modified broker with priority support, PR-MQTT, treats urgent messages ahead of normal messages. The experiments also show how the PR-MQTT algorithm improves IoT network performance compared with standard MQTT broker. A discussion of the proposed PR-MQTT protocol and the experiment results follows.

5.1 Experimental setup

The HBMQTT broker, the only MQTT broker written in Python, is selected to implement the work. The proposed algorithm is incorporated into the HBMQTT broker code. The modified broker runs on a Raspberry Pi 3 processor with 1.4 GHz clock speed and 1 GB RAM. We use ESP32 3.3V MCU as publisher clients and BME280 sensor nodes to read pressure, temperature and humidity data from the atmosphere. The BME 280 sensor nodes are connected to the ESP 32 MCUs. A laptop equipped with a core i7 CPU runs the subscriber clients. We assume multiple publishers publish data simultaneously and are received by a single subscriber.

Table 2 Experimental setup

Experimental setup	
Processor (broker)	Raspberry Pi 3 1.4 GHz 1 GB RAM
Broker	HBMQTT Boker
Publisher nodes	ESP 32 3.3 V MCU
Sensors used	BME 280
Data measured	Temperature, pressure and humidity
Subscriber node	Laptop with i5 processor, 8 GB RAM

Table 3 Testing scenarios

Parameters	*Values*	
	Scenario 1	*Scenario 2*
No. of publishers	10 publishers	40 publishers
No. of subscribers	1 subscriber	1 subscriber
Publishing rate	50 messages/sec. (approx.)	50 messages/sec. (approx.)
QoS of published messages	QoS 0	QoS 0
Normal messages (for each publisher)	250 messages	250 messages
Priority messages (for each publisher)	25 messages (approx.)	25 messages (approx.)
Trivial range (for each publisher)	10 points	10 points
Skip limit (for each publisher)	5 messages	5 messages

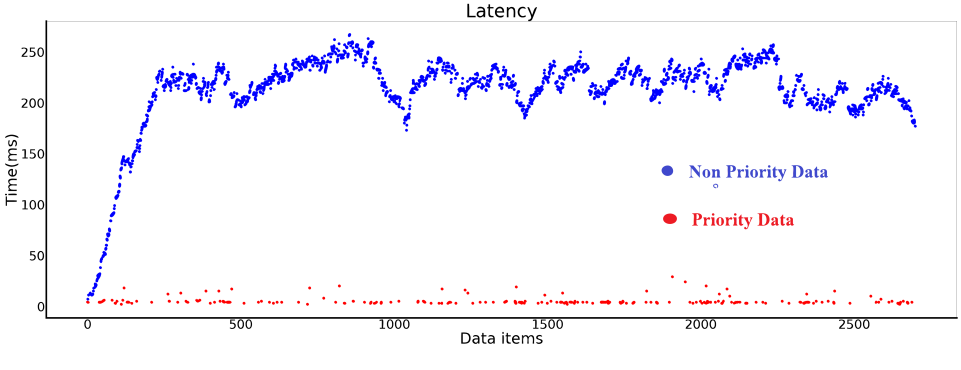
A testing scenario with two publisher client sizes was used for evaluation purposes. Most of the IoT devices that frequently sense and publish data use the reliability level QoS 0 and such data form a major part of IoT network traffic. Hence the QoS of the published messages are selected as QoS 0 for the experimental evaluation. Ten

clients generate and publish data in the first scenario, and in the second scenario, 40 clients publish data. From the testing point of view, we assume all publishing clients work simultaneously. Each client generates and publishes approximately 250 normal and 25 priority messages during the evaluation period at an average of 50 messages per second. For each publisher, the trivial interval is set as 10 points and the skip limit is five messages. The performance of the normal HBMQTT broker and the proposed PR-MQTT broker is compared with different evaluation metrics. The testing environment can be restructured to increase the number of clients and to change the messages per second if required. The experimental setup and testing scenarios are summarised in Tables 2 and 3, respectively.

5.2 Performance analysis

The evaluation of the proposed work involves several performance metrics, including message latency, CPU and RAM resource utilisation, network traffic, and transmission delays. These metrics are essential for assessing the effectiveness of the PR-MQTT method in comparison to the standard MQTT protocol, which does not incorporate priority data handling. Initially, the latency experienced in delivering both normal and priority messages in the PR-MQTT method is evaluated. This analysis provides insights into the efficiency of message prioritisation and its impact on message delivery times. Subsequently, the resource utilisation, specifically CPU and RAM, as well as transmission delays, are compared between the standard MQTT and prioritised MQTT approaches. This comparison allows for an assessment of the resource efficiency and performance gains achieved by incorporating message prioritisation. Furthermore, the evaluation takes into account the network traffic under different datasets. This analysis provides an understanding of the impact of prioritisation on overall network utilisation and congestion. The evaluation results of the proposed work, considering these various performance metrics, are presented to demonstrate the effectiveness and benefits of the proposed PR-MQTT approach.

Figure 5 Message latency of PR-MQTT (see online version for colours)



5.2.1 Message latency

In the proposed PR-MQTT protocol, any input publisher node has the capability to generate priority data. The determination of whether the generated data is normal or

priority data is done by the broker based on the value of the sensed data. Figure 5 presents a scatter plot that illustrates the latency of both normal and priority messages in scenario one, as per the proposed algorithm. The horizontal axis of the plot represents the message index, while the vertical axis represents the time in milliseconds. The plot differentiates between normal and priority data using colours, with blue dots representing normal data and red dots representing priority data. The results obtained from the plot clearly demonstrate that priority data is received by the subscriber much faster compared to normal data. As the message size and index position increase, the latency of normal data also increases. However, the latency of priority data remains relatively constant, regardless of the message index.

5.2.2 Resource utilisation

The CPU and memory requirements were selected as metrics to evaluate the resource utilisation of the protocols under consideration. The analysis aimed to compare the resource utilisation of the standard MQTT protocol and the proposed PR-MQTT protocol using the same dataset. Figure 6 illustrates the CPU and RAM utilisation of the standard MQTT protocol, while Figure 7 represents the performance of the PR-MQTT protocol. In both figures, the red colour indicates the CPU utilisation during the execution period, while the blue colour represents the RAM utilisation in megabytes. Initially, the proposed PR-MQTT protocol was expected to consume more resources due to the additional computation required for filtering out priority data and skipping identical data. However, the analysis graph shows that the proposed algorithm utilises fewer CPU resources and almost the same amount of RAM as the standard MQTT protocol. This advantage is attributed to the trivial interval concept utilised in the PR-MQTT protocol, which allows for skipping unwanted data. The extra computation required by the new algorithm is offset by the reduced number of messages forwarded to the subscriber. Consequently, the proposed method is proven suitable for resource-constrained devices, as it achieves resource efficiency without significantly increasing CPU and RAM utilisation.

Figure 6 Standard MQTT resource usage (see online version for colours)

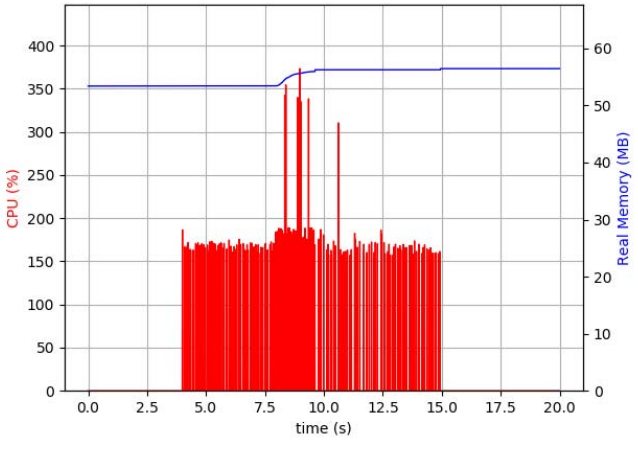
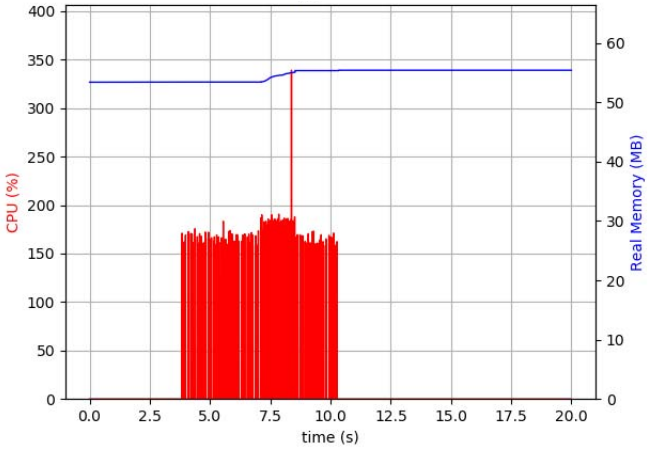


Figure 7 Priority MQTT resource usage (see online version for colours)



5.2.3 Network traffic

The proposed PR-MQTT algorithm offers a significant advantage regarding reduced network traffic. Utilising the trivial interval concept, the algorithm prevents the unnecessary transmission of similar and duplicate data, effectively decreasing the number of messages passed through the network. A significant proportion of data congestion within IoT networks can be attributed to messages employing QoS level 0 reliability. Most IoT devices, which frequently sense and publish data like temperature and pressure sensors, opt for QoS 0 reliability. In environments where sensor data changes infrequently, the loss or omission of a few data elements does not substantially affect the outcome of the application. Therefore, one practical approach to mitigate IoT network congestion is reducing or controlling the transmission of QoS 0 data. Figure 8 shows the number of messages forwarded to subscriber clients for different datasets under two scenarios: using the proposed PR-MQTT broker and the standard MQTT broker. The plot demonstrates that when the proposed algorithm is applied, there is a nearly 25% reduction in network traffic observed across all datasets. This reduction in network traffic is highly beneficial as it allows IoT networks to accommodate many devices and effectively handle the vast amount of data generated and transmitted by these devices. The proposed algorithm contributes to improved network efficiency and scalability in IoT applications by minimising unnecessary message transmission.

5.2.4 Normal message transmission delay

The transmission delay experienced by normal messages in both the standard MQTT (represented by blue dots) and the proposed PR-MQTT method (represented by red dots) for the same dataset is illustrated in Figure 9. The plot demonstrates that when the filtering algorithm is employed, even for normal data, the transmission delay is significantly reduced. This achievement can be attributed to the concept of skipping off unnecessary data, which is a core aspect of the proposed algorithm. By eliminating the transmission of redundant or identical data, the algorithm effectively reduces the overall transmission delay experienced by normal messages.

Figure 8 Network traffic comparison (see online version for colours)

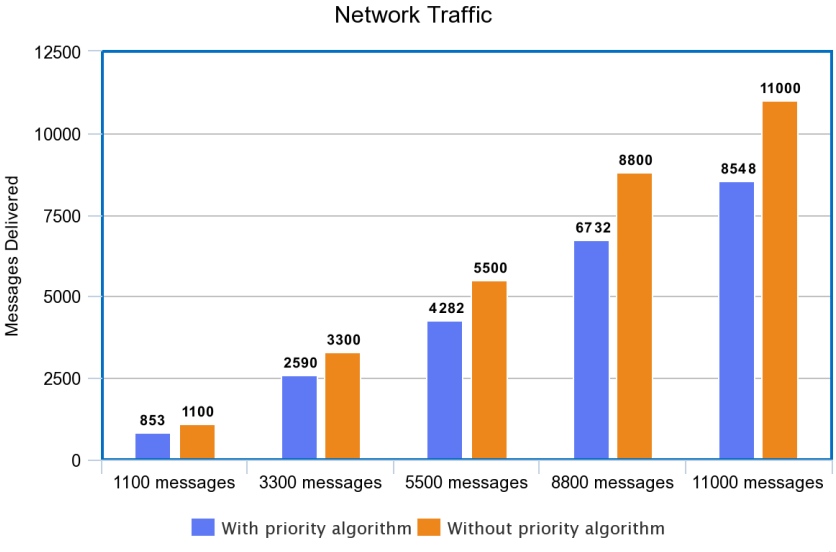
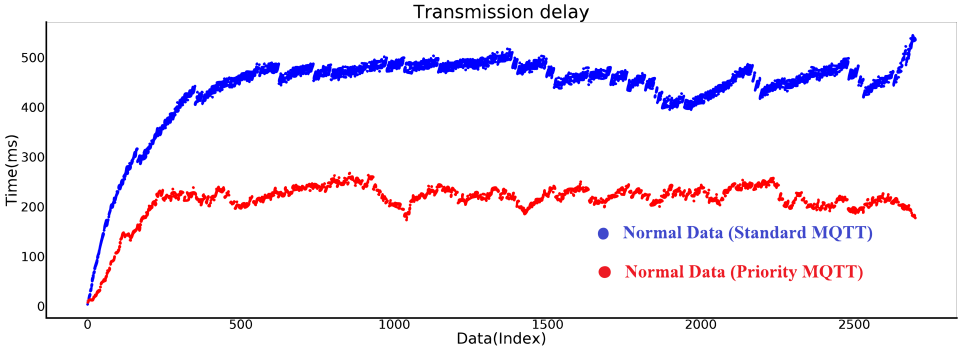


Figure 9 Transmission delay of normal messages (see online version for colours)



The evaluation results clearly demonstrate that the MQTT broker integrated with the proposed algorithm outperforms the standard MQTT broker in all the evaluation metrics that were considered. The proposed algorithm effectively prioritises urgent messages of utmost importance, ensuring their prompt delivery to the receiver. By identifying and skipping the forwarding of redundant or similar data, the PR-MQTT algorithm successfully reduces the overall message transfer through IoT networks.

5.3 Discussion

A study on attempts in the literature to prioritise IoT messages and reduce network traffic reveals that it is a research field that still requires exploration. Many attempts in the literature assign the responsibility of identifying priority messages to the constrained client nodes or require modifications to the standard MQTT packet structure or message

header size. However, such approaches are not always desirable or encouraged in IoT networks.

The priority is managed from the broker side in the proposed PR-MQTT protocol. No modification of the standard MQTT packet structure or fixed header is required. Publisher clients collect data from surroundings and transmit data packets to the broker using the standard MQTT packet forwarding procedure. A broker identifies the high-priority messages using the PR-MQTT algorithm explained. No additional overhead or computation is required at the constrained client nodes. The computational complexity of the broker server is also unaffected, as the new algorithm uses simple mathematical computations.

For evaluation purposes, a testing scenario is created with two publisher client sizes: ten and forty clients, each sending fifty messages per second. The performance of the MQTT protocol is compared with and without the proposed algorithm applied. With the standard MQTT protocol, the latency of the messages increases as the number of clients and messages increases. When the priority algorithm is applied to the same dataset, the latency of the normal messages increases, but at a different pace than in standard MQTT. The priority messages are delivered much faster than normal messages. The latency of the priority messages remains almost constant, regardless of the index position of the message. The priority messages are delivered within 10–15 milliseconds, even for larger data sizes. The latency for normal messages increases as the index position increases. The CPU and RAM resource utilisation of the proposed algorithm is better than that of the standard MQTT algorithm. The new approach employs the concept of a trivial interval to eliminate the transmission of identical or irrelevant data to the client node, which helps reduce resource consumption.

One significant setback experienced by IoT networks is the heavy traffic and congestion problem. As the number of connected clients increases, the load on the broker become heavier and network traffic increases. The proposed algorithm reduces the network traffic by avoiding the transmission of identical unnecessary data, thanks to the help of the trivial interval concept incorporated. Therefore, this approach can support more input clients than normal IoT networks. With the proposed PR-MQTT broker, even the transmission delay of normal messages is reduced considerably. As transmission of identical and insignificant messages is restricted with the trivial interval concept, the overall amount of data transferred between broker and subscriber nodes are also reduced. This improves the transmission time of the normal messages.

The proposed method handles urgent messages efficiently and reduces network traffic. The proposed PR-MQTT protocol improves the performance of the MQTT broker The improved performance of the proposed algorithm is reflected in various aspects, including reduced latency for priority messages, efficient utilisation of CPU and RAM resources, decreased network traffic, and minimised transmission delays for normal messages. These results highlight the effectiveness of the PR-MQTT algorithm in optimising the performance and efficiency of IoT applications. Overall, the proposed algorithm offers significant advantages over the standard MQTT broker, enhancing the delivery of critical messages while minimising unnecessary data transfer. It provides a valuable solution for improving the performance of IoT networks and ensuring the timely and efficient communication of important information.

Various industries, such as automotive, entertainment, gaming, finance and healthcare, are experiencing significant transformations due to the integration of machine

learning techniques. The primary goal of incorporating learning models into these domains is to minimise error rates and enhance real-time results while minimising costs and time investments. When we look at the future scope of learning models, several areas stand out, including robotics, computer vision, quantum computing, the automotive industry, and cybersecurity. These fields are expected to see substantial advancements and applications of machine learning technologies. A significant upcoming trend is optimising machine learning speed by utilising quantum computing. Quantum computing enables the execution of multiple complex operations simultaneously, offering the potential for substantial reductions in execution times, mainly when processing high-dimensional vectors. Another anticipated development is creating a versatile, all-purpose model called a 'big model'. This model is designed to handle various tasks simultaneously, allowing it to be trained in multiple domains according to specific needs. Furthermore, the years ahead are likely to witness advancements in distributed machine learning portability, enabling the running of machine learning tools directly on various platforms and computing engines. This advancement will eliminate the need for transitioning to new toolkits when working with machine learning tasks.

6 Conclusions and future research directions

IoT applications generate and transfer a significant portion of internet data. Many IoT networks experience network traffic and congestion issues due to the large number of connected input devices. Timely processing of critical messages is crucial in various IoT applications, including patient healthcare and industrial data monitoring. Most IoT communication protocols do not prioritise incoming messages. Existing approaches to prioritise IoT data often overload the constrained input clients or require modifications to the standard IoT protocol, which is undesirable. This paper presents a novel approach to reduce network traffic and handle priority input data in IoT networks, regardless of the number of client nodes and the volume of data they produce. The MQTT protocol, a widely used communication protocol in IoT systems, is selected for experimental purposes. The proposed PR-MQTT broker is integrated into the open-source MQTT broker HBMQTT. It prioritises urgent messages and reduces data traffic in IoT networks. A test system is constructed, and experiments are conducted using various metrics to compare PR-MQTT and standard MQTT. The results demonstrate that the proposed approach efficiently processes urgent messages and reduces network congestion. The latency of the priority messages remains nearly constant, irrespective of the message's index position. Priority messages are consistently delivered within 10–15 milliseconds, resulting in a speed improvement of over 90% compared to regular messages. Additionally, the proposed approach reduces CPU resource utilisation and network traffic by 25% and the transmission delay of normal messages by 50%. The client nodes are not overloaded, and this approach does not affect standard MQTT protocol specifications. The responsibility of determining the priority of data is to the powerful MQTT broker. This technique reduces network traffic by avoiding transmitting identical messages and enables IoT networks to accommodate more devices. Hence, the proposed algorithm helps improve network performance and message transmission in IoT applications.

Implementing the proposed algorithm in the Mosquitto broker, the most popular broker in the MQTT circle, is a planned future expansion. Assigning priority concepts in other popular IoT protocols, such as CoAP and AMQP, is also planned as future work.

References

Abdul Ameer, H.R. and Hasan, H.M. (2020) 'Enhanced MQTT protocol by smart gateway', *Iraqi Journal of Computers, Communications, Control and Systems Engineering*, Vol. 20, No. 1, pp.53–67.

Akshatha, P.S., Kumar, S.D. and Venugopal, K.V. (2022) 'MQTT implementations, open issues, and challenges: a detailed comparison and survey', *International Journal of Sensors Wireless Communications and Control*, Vol. 12, No. 8, pp.553–576.

AlEnany, M.O., Harb, H.M. and Attiya, G. (2021) 'A new back-off algorithm with priority scheduling for MQTT protocol and IoT protocols', *International Journal of Advanced Computer Science and Applications*, Vol. 12, No. 11, pp.349–357.

Alhaidari, F.A. and Alqahtani, E.J. (2020) 'Securing communication between fog computing and IoT using constrained application protocol (COAP): a survey', *J. Commun.*, Vol. 15, No. 1, pp.14–30.

Ali, J. and Zafar, M.H. (2023) 'Improved end-to-end service assurance and mathematical modeling of message queuing telemetry transport protocol based massively deployed fully functional devices in smart cities', *Alexandria Engineering Journal*, Vol. 72, No. 1, pp.657–672.

Anitha, P., Vimala, H. and Shreyas, J. (2023) 'Comprehensive review on congestion detection, alleviation, and control for IoT networks', *Journal of Network and Computer Applications*, Vol. 221, No. 1, p.103749.

Ashima, R., Haleem, A., Javaid, M. and Rab, S. (2022) 'Understanding the role and capabilities of internet of things-enabled additive manufacturing through its application areas', *Advanced Industrial and Engineering Polymer Research*, Vol. 5, No. 3, pp.137–142.

Bashir, A. and Mir, A.H. (2021) 'Lightweight secure MQTT for mobility enabled e-health internet of things', *Int. Arab J. Inf. Technol.*, Vol. 18, No. 6, pp.773–781.

Bayılmış, C., Ebleme, M.A., Çavuşoğlu, Ü., Küçük, K. and Sevin, A. (2022) 'A survey on communication protocols and performance evaluations for internet of things', *Digital Communications and Networks*, Vol. 8, No. 6, pp.1094–1104.

Bressoud, T., White, D., Bressoud, T. and White, D. (2020) 'The hypertext transfer protocol', *Introduction to Data Systems: Building from Python*, pp.609–648, Springer, Cham.

Broker API Reference – HBMQTT 0.6 Documentation (2023) [online] https://hbmqtt.readthedocs.io/en/latest/references/broker.html (accessed 5 April 2023).

Chen, D., Li, D. and Guo, R. (2022) 'Design of topic priority based on industrial publish-subscribe system', *Frontiers in Energy Research*, Vol. 10, No. 1, p.979174.

Donta, P.K., Amgoth, T. and Annavarapu, C.S.R. (2020) 'Congestion-aware data acquisition with q-learning for wireless sensor networks', *2020 IEEE International IoT, Electronics and Mechatronics Conference (IEMTRONICS)*, pp.1–6, IEEE.

Donta, P.K., Srirama, S.N., Amgoth, T. and Annavarapu, C.S.R. (2022) '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.

Donta, P.K., Srirama, S.N., Amgoth, T. and Annavarapu, C.S.R. (2023) 'iCoCoa: intelligent congestion control algorithm for coap using deep reinforcement learning', *Journal of Ambient Intelligence and Humanized Computing*, Vol. 14, No. 3, pp.2951–2966.

OASIS Standard Incorporating Approved Errata (2015) *MQTT Version 3.1.1 Plus Errata 01*.

Gruener, S., Koziolek, H. and Rückert, J. (2021) 'Towards resilient IoT messaging: an experience report analyzing MQTT brokers', *2021 IEEE 18th International Conference on Software Architecture (ICSA)*, IEEE, pp.69–79.

Houimli, M., Kahloul, L. and Benaoun, S. (2017) 'Formal specification, verification and evaluation of the MQTT protocol in the internet of things', in *2017 International Conference on Mathematics and Information Technology (ICMIT)*, IEEE, December, pp.214–221.

Hwang, K., Jung, I.H. and Lee, J.M. (2022a) 'U-Mosquitto: extension of Mosquitto broker for delivery of urgent MQTT message', *International Journal of Computational Vision and Robotics*, Vol. 12, No. 1, pp.39–52.

Hwang, K., Lee, J.M. and Jung, I.H. (2022b) 'Performance monitoring of MQTT-based messaging server and system', *Journal of Logistics, Informatics and Service Science*, Vol. 9, No. 1, pp.85–96.

Jain, V.K., Mazumdar, A.P., Faruki, P. and Govil, M.C. (2022) 'Congestion control in internet of things: classification, challenges, and future directions', *Sustainable Computing: Informatics and Systems*, Vol. 35, No. 1, p.100678.

Johansson, L. and Dossot, D. (2020) *RabbitMQ Essentials: Build Distributed and Scalable Applications with Message Queuing Using RabbitMQ*, Packt Publishing Ltd., Mumbai, ISBN: 978-1-78913-166-6.

Jung, C. (2020) 'Prioritized data transmission mechanism for IoT', *KSII Transactions on Internet and Information Systems (TIIS)*, Vol. 14, No. 6, pp.2333–2353.

Kim, G., Park, J. and Chung, K. (2018a) 'Priority-based multi-level MQTT system to provide differentiated IoT services', *Journal of KIISE*, Vol. 45, No. 9, pp.969–974.

Kim, S-j. and Oh, C-h. (2017) 'Method for message processing according to priority in MQTT broker', *Journal of the Korea Institute of Information and Communication Engineering*, Vol. 21, No. 7, pp.1320–1326.

Kim, Y-S., Lee, H-H., Kwon, J-H., Kim, Y.S. and Kim, E-J. (2018b) 'Message queue telemetry transport broker with priority support for emergency events in internet of things', *Sensors and Materials*, Vol. 30, No. 8, pp.1715–1721.

Koziolek, H., Grüner, S. and Rückert, J. (2020) 'A comparison of MQTT brokers for distributed IoT edge computing', *Software Architecture: 14th European Conference, ECSA 2020, Proceedings*, 14–18 September, Springer, L'Aquila, Italy, pp.352–368.

Lawton, G. (2004) 'Machine-to-machine technology gears up for growth', *Computer*, Vol. 37, No. 9, pp.12–15.

Lazidis, A., Tsakos, K. and Petrakis, E.G. (2022) 'Publish-subscribe approaches for the IoT and the cloud: functional and performance evaluation of open-source systems', *Internet of Things*, Vol. 19, No. 1, p.100538.

Liu, Y. and A-Masri, E. (2021) 'Evaluating the reliability of MQTT with comparative analysis', *2021 IEEE 4th International Conference on Knowledge Innovation and Invention (ICKII)*, IEEE, pp.24–29.

Mahmood, A., Beltramelli, L., Abedin, S.F., Zeb, S., Mowla, N.I., Hassan, S.A., Sisinni, E. and Gidlund, M. (2021) 'Industrial IoT in 5G-and-beyond networks: vision, architecture, and design trends', *IEEE Transactions on Industrial Informatics*, Vol. 18, No. 6, pp.4122–4137.

Mosquitto MQTT Broker (2023) [online] http://www.steves-internet-guide.com/mosquitto-broker/ (accessed 5 April 2023).

Naik, N. (2017) 'Choice of effective messaging protocols for IoT systems: MQTT, COAP, AMQP and HTTP', *2017 IEEE International Systems Engineering Symposium (ISSE)*, IEEE, pp.1–7.

Oh, S-C. and Kim, Y-G. (2019) 'A study on MQTT based on priority topic for IIoT', *The journal of the institute of internet, broadcasting and communication*, Vol. 19, No. 5, pp.63–71.

Oyewobi, S.S., Djouani, K. and Kurien, A.M. (2021) 'Using priority queuing for congestion control in IoT-based technologies for IoT applications', *International Journal of Communication Systems*, Vol. 34, No. 4, p.e4709.

Park, K., Kim, I. and Park, J. (2018) 'An efficient multi-class message scheduling scheme for healthcare IoT systems', *International Journal of Grid and Distributed Computing*, Vol. 11, No. 5, pp.67–77.

Park, K., Park, J. and Lee, J. (2017) 'An IoT system for remote monitoring of patients at home', *Applied Sciences*, Vol. 7, No. 3, p.260.

Ray, P.P. (2018) 'A survey on internet of things architectures', *Journal of King Saud University-Computer and Information Sciences*, Vol. 30, No. 3, pp.291–319.

Singh, S., Rathore, S., Alfarraj, O., Tolba, A. and Yoon, B. (2022) 'A framework for privacy-preservation of IoT healthcare data using federated learning and blockchain technology', *Future Generation Computer Systems*, Vol. 129, No. 1, pp.380–388.

Tatyasaheb, D.N. and Kumar, B. (2021) 'Implementation and comparison of MQTT protocol to check the drawbacks for future enhancement', *2021 International Conference on Computing, Communication and Green Engineering (CCGE)*, IEEE, pp.1–6.

Uy, N.Q. and Nam, V.H. (2019) 'A comparison of AMQP and MQTT protocols for internet of things', *2019 6th NAFOSTED Conference on Information and Computer Science (NICS)*, IEEE, pp.292–297.

Wazid, M., Das, A.K., Bhat, V. and Vasilakos, A.V. (2020) 'LAM-CIoT: lightweight authentication mechanism in cloud-based IoT environment', *Journal of Network and Computer Applications*, Vol. 150, No. 1, p.102496.

Yudidharma, A., Nathaniel, N., Gimli, T.N., Achmad, S. and Kurniawan, A. (2023) 'A systematic literature review: messaging protocols and electronic platforms used in the internet of things for the purpose of building smart homes', *Procedia Computer Science*, Vol. 216, No. 1, pp.194–203.

Yugha, R. and Chithra, S. (2020) 'A survey on technologies and security protocols: reference for future generation IoT', *Journal of Network and Computer Applications*, Vol. 169, No. 1, p.102763.