Alexandria Engineering Journal (2023) **72**, 657–672

HOSTED BY

ELSEVIER

Alexandria University

Alexandria Engineering Journal

www.elsevier.com/locate/aej
www.sciencedirect.com

ORIGINAL ARTICLE

Improved End-to-end service assurance and mathematical modeling of message queuing telemetry transport protocol based massively deployed fully functional devices in smart cities

Jawad Ali [a,1], **Mohammad Haseeb Zafar** [b,*]

[a] *Faculty of Electrical and Computer Engineering, University of Engineering & Technology, Peshawar 25120, Pakistan*
[b] *Cardiff School of Technologies, Cardiff Metropolitan University, Cardiff CF5 2YB, UK*

Received 22 August 2022; revised 8 March 2023; accepted 7 April 2023

KEYWORDS

QoS;
MQTT;
End-to-end service assurance;
Smart Cities;
FFDs

Abstract QoS is an arguable feature of Message Queuing Telemetry Transport Protocol (MQTT) that is employed under constrained environment for communication over an unreliable network in smart cities. It gives extra control to the client for matching its needs according to the application but with confiscated network performance. For a reliable end-to-end service assurance, guaranteed QoS and controlled scalability, MQTT based IoT must be examined for various network parameters. In this work, an improved TCP based transparent MQTT network is proposed for massively deployed connected devices in smart cities. The mathematical modeling is carried out by analyzing end-to-end quality assurance. Network latency, content delivery rate, number of subscribe/publish requests and services offered by the MQTT broker for various topic IDs are then verified on physical network. These parameters are examined over a transparent gateway-based network for verifying the agreement of devised probability based mathematical model with the actual content delivery, service rate and request hold time at virtual machine based MQTT broker, local network broker and remote server respectively. The Improved MQTT model surpasses Apollo, RabbitMQ and Mosquitto server by evidencing the message queueing delay of 3.5 ms for QoS-0 and 3.6 ms for QoS-1 service IoT.

© 2023 THE AUTHORS. Published by Elsevier BV on behalf of Faculty of Engineering, Alexandria University This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

1. Introduction

Smart cities are considered as a network of physically separated and logically connected devices. Various services are provided using smart city concept that include automatic functionality, information exchange, stocks and inventory

* Corresponding author.
E-mail address: mhzafar@cardiffmet.ac.uk (M. Haseeb Zafar).
[1] Department of Electrical Engineering, University of Engineering & Technology, Mardan 23200, Pakistan.

https://doi.org/10.1016/j.aej.2023.04.014
1110-0168 © 2023 THE AUTHORS. Published by Elsevier BV on behalf of Faculty of Engineering, Alexandria University
This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

management, health [1] and security, entertainment, Energy, and smart grid [2], transportation, and connected industrial manufacturing. These services may require only its comprising devices to communicate with one another in Device to Device (D2D) based communication scheme or they may interact with a server as Device to Server (D2S) or Server to Device (S2D). Thus, while acquiring fully functionality of smart city, various technological bridges or network bridges are encountered [1]. Various communication protocols are involved in the development of smart cities. Majority of these protocols are covered as a base for Internet of Things (IoT) [3] and Internet of Everything (IoE) [4]. Constrained Application Protocol (CoAP), Messaging Queuing and Telemetry Protocol (MQTT), Web Socket, Data Distribution Service (DDS) and Extensible Messaging and Presenting Protocol (XMPP) are the available protocols that are used extensively for achieving different tasks in the development of smart cities [5]. Of all these protocols, it appears that MQTT is used on the top of IEEE 802.15.4 and TCP/IP for providing bandwidth efficient and its low power consumption-based connectivity. MQTT is thus already widely implemented for Supervisory Control and Data Acquisition (SCADA) [6], Delta-rail – Rail signaling [7], automation [8], healthcare [9,10], Facebook and Instagram applications [11], Real time transport monitoring and automation [12,13].

In this paper, we develop mathematical model for estimating end-to-end delay and validate it by analyzing the end-to-end MQTT based real time traffic over the Internet between an MQTT client and a remote server. MQTT requires a reliable TCP/IP connection which is a complex base for a very simple communicating scenario. If the nodes comprise of fundamental sensing and connectivity features, then MQTT-SN [14] is used. MQTT-SN uses UDP and thus compromises the reliability, as achieved by TCP [14] and [15]. This protocol is adapted along with wireless communication for low data rates at reliable, energy and cost-efficient data transmission [16]. In our case, MQTT server is handled on a virtual machine, running at the eclipse.org cloud. On the other hand, the clients are deployed on Linux, and as windows clients or as a java application. The end nodes are connected to the Internet via Wi-Fi whereas the routers are then using Broadband connection for connectivity to the server as in [16]. Note that both MQTT and MQTT-SN may use DNS Server in our case.

1.1. Architecture

The MQTT application layer protocol works on the top of TCP transport layer protocol. TCP thus guarantees the "at least one time" reaching of data onto the server as well as connected receiving nodes. MQTT involves a TCP based communication of publisher nodes as well as subscriber nodes with the MQTT broker as shown in Fig. 1. The dotted line represents the flow of message from a publisher onto the server. The Server handles the received data in the form of topics. MQTT server topic can be a replaceable logical buffer that holds data for a specific interval of time before discarding it or replacing it with fresh data from the publisher. The subscriber nodes receive the data when it is updated on the server.

The MQTT based network may comprise of three different kinds of logical formation, shown in Fig. 2.

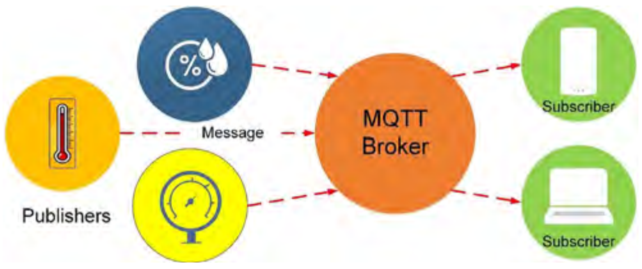

Fig. 1 MQTT broker communication with publishers and subscribers.

A transparent Gateway architecture has one node connected to one gateway that is bridging the node with remote MQTT server. Hybrid gateways can accept requests of multiple nodes and can be served subsequently as access point to a collection of local nodes in mesh topology [17]. An aggregating gateway has support to the whole local node network. It holds the connection of each node in star topology whereas the MQTT server is accessed from a single gateway. Each logical network grade, as discussed, has its own facilities as well as drawbacks. For example, transparent gateway is the only option on low density, single sensor/actuator MQTT based IoT. An aggregated gateway will be handling a high-density network, but a single point failure and frequent requests to/from gateway can affect network performance. A hybrid gateway rectifies and address the limitations of transparent and aggregated Gateways [18]. An aggregating gateway can't be employed for smart city's application as nodes are geographically dispersed and coverage of gateways are limited which may cause channel congestion and more packet loss as a rule of thumb [19]. This work uses transparent gateway-based network. The paper also considered the following messaging rules that are obvious for an MQTT based IoT.

The overall communication, in general, consists of request and acknowledge messages.

The SUBSCRIBE as well as the PUBLISH requests goes through the same pathway except when DNS is involved. In such case, DNS will first find the IP for requested server.

TCP based communication is considered for all nodes on the network.

The contents that are published on the server, are kept for T_{hold}, here referred as $\boldsymbol{T_{cnt}}$ interval of time whereas the subscription request is kept for T_{req} time on the same server. The actual hold time depends upon the scenario implemented for storage capacity on client and server independently [20].

The CONNECT request has QoS-1 when it is secured by TCP whereas it is ensured by CONNACK. QoS-0 and QoS-2 are the extended version which will be addressed as well [21].

1.2. Motivation, system model and problem statements

Smart Cities involve communication between various types of sensors and actuators as its building blocks [1,22,23]. These communicating nodes are interacting in such a manner that the system is monitored for collective services rather than individual functionality. The services, offered in a smart city, are

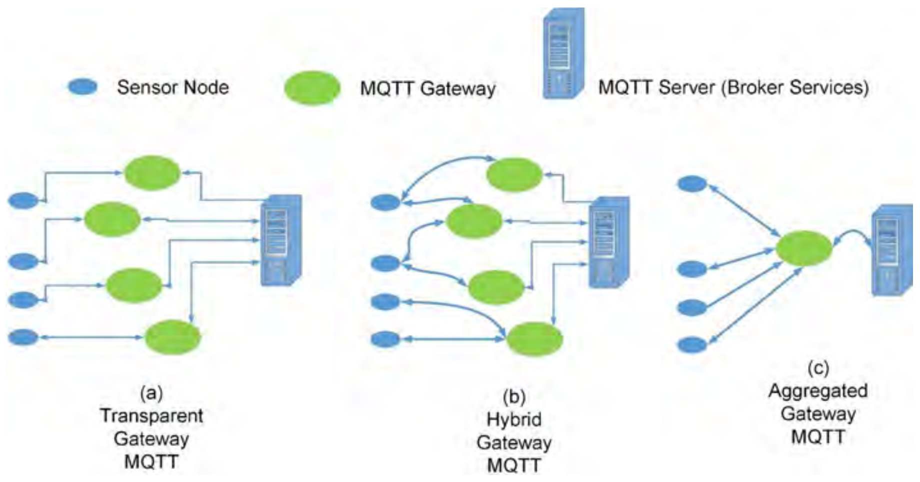

Fig. 2 Logical formation of MQTT network with server.

categorized based on the latency in the information exchange and process execution. For example, an IoT used inside a healthcare unit, may have several priority levels based on the severity of the conditions of a patient [24]. Moreover, a patient may require different services if the condition of patient is changing at every instant. Similarly, a traffic system may use high end-to-end delay for highways in normal condition and operation whereas in adverse weather, the condition needs frequent updates with maximum assurance and least probability of packet loss in end-to-end delivery.

Our aim is to assure sufficient end-to-end reliability in terms of content delivery and delay assurance for smart city application, where multiple sensors can add data to the server log on either same topic or on different topics, based on the application. Thus, PUBLISH requests are all considered as mutually exclusive in terms of its delivery time as well as packet size. Also, the published contents have limited time for residing on the server before it is discarded or replaced. For the sake of brevity, we have used logging of different attributes of a same phenomenon, being published from different nodes. These data nodes may be required by different users to analyze the same problem differently. For example, a traffic data that is collected from different intersections may be used for possibility of low traffic, calculated from a local ambulance to pass through these junctions and reach its destination. On the other hand, the same data can be analyzed for controlling the traffic lights to reduce congestion on a user selected route. Thus, contents are kept on same topic but different subtopics for optimum usage in each selected application.

The estimation of end-to-end latency [25] and probability of content delivery has advantages beyond mentioned scenarios. This paper is first step towards analyzing the end-to-end transmission in Smart Cities, w.r.t MQTT based IoT and its service estimation. First, we model the scenario for transparent gateway based IoT for remote services. This MQTT based network considers number of connected nodes, subscription and publish rates, number of requests and time of content holding by server and gateway for system performance function modeling. This function is then used to estimate the system design parameters to meet a given quality of service, end-to-end delay, and content delivery assurance. We provide the guidelines for achieving a rated service assurance by varying the number of sensors, actuators, gateways, changing the Internet characteristics, publish and subscription rates.

2. MQTT characteristics

2.1. Why MQTT?

MQTT is selected for its light weight, content-oriented messaging characteristics. The local network of MQTT can be used over IP-less data as well as IP based nodes. In our case, we use it for TCP based node to gateway connection. The nodes use wired connectivity with the gateway which is emulated inside environment and by deploying Mosquito clients. The data coordinates to local gateway where an interface with the Internet is provided with real world queuing delay.

2.2. Communication in MQTT

MQTT provides TCP/IP oriented solution to event-based request, publish and connect requests, and the instantaneous probability of end-to-end delivery is affected each time [26]. The subscription-based services depend upon the data provided by other sensors to the server to which the receiver has subscribed. This subscription is topic based and thus here, the received data is dependent on the availability of the data on the server from self or other nodes. The function of midway gateway is to relay received subscribe requests and publish/-connect requests to the MQTT server. This gateway has a limited time and space to save the channeled data. The delay of, say, putting data from input port of gateway to output port of gateway, is of directly additive nature. This queue is only to be maintained at the server and the gateway delay is static or constant. There is no such case where the gateway receives data from different nodes so the addition of data or its queuing is not calculated. The assumption that one node can post to more than one topic is addressed as separate events.

The following cases may occur while communicating between sender node and receiver node, as shown in Fig. 3.

Case 1: End nodes connecting and publishing to gateway with CONNECT/CONNACK and PUBLISH. The process either connects the end node with the gateway or sends/re-

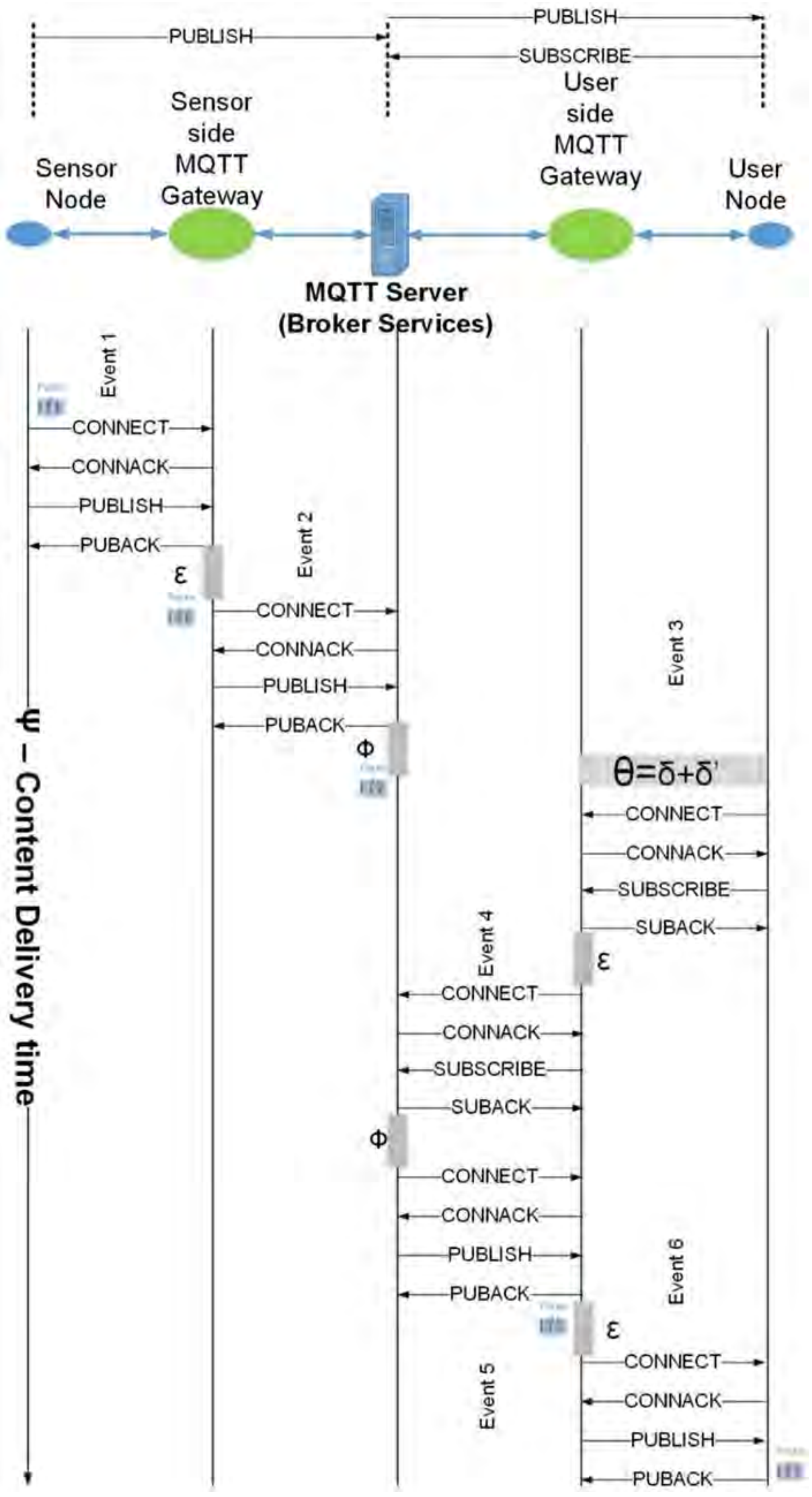

Fig. 3 End-to-end delay and packet travel system model (QoS-1).

ceives contents to an already registered TOPIC-ID, with CONNECT. This communication is TCP for wired (in our case) and UDP for MQTT-SN based connectivity [27].

Case 2: Gateway sends the received data from connected sensor to the server as CONNECT-CONNACK-PUBLISH-PUBACK. This exchange always occurs over wired link and this use the reliable TCP/IP QoS. Sensors are abridged from the main server using gateway.

Case 3: The end user registers with gateway, to a topic that exists on the server using SUBSCRIBE and SUBACK. The process is accompanied by CONNECT and CONNACK that is done before registering to topic. Note that at this point, the end user is not in direct connection with the server, so we need to connect and subscribe the user side gateway with the server, as discussed in next case.

Case 4: Server lets the user side gateway, to register with itself. The gateway sends CONNECT request which is acknowledged by CONNACK from the MQTT server side using TCP/IP. The gateway then sends SUBSCRIBE request to TOPIC-ID which is acknowledged by SUBACK.

Case 5: The server then sends the available data on the TOPIC-ID to its user side gateway using PUBLISH and getting PUBACK.

Case 6: The gateway then puts the same data on its sender side, to be connected on demand with the end user. The gateway repeats the process of Case 5. This process only completes if the information received from the server is not expired and is applicable to be sent to the end user.

The end user has certain probability of its availability thus a queuing delay is expected at this node here. CONNECT is acknowledged by the CONNACK from the end user side whereas TOPIC-ID based data is PUBLISH-ed and acknowledged as PUBACK. The topic ID is initially subscribed in Case 3.

The system, as seen from Fig. 3, is asynchronous by the nature of problem. It is for the fact that after each SUBSCRIBE and PUBLISH content request, the server as well as the end node go to an undefined idle state. Even defining a topic (as TOPIC-ID) is not enough to activate and make the server or end user to stay awake for communication. The SUBSCRIBE and PUBLISH requests thus induces a randomness in estimating the content delivery to/from server.

The aim of this work is to conclude a probabilistic model for calculating end-to-end content travel time. This includes the time of sending of contents and its reach time to the end user after a request. We will discuss two scenarios for estimating the end-to-end delay. First, the Asynchronous communication of the contents when it is available on the server and second, content delivery - when it is not available on the server. We will consider these two scenarios in estimating network latency and QoS.

3. *Delay* estimation

The delay is estimated using probabilistic model. Contents that are created at the sensor nodes are sent to the intermediate gateway which then requests the server (MQTT Broker) to put it on the server. We have considered the synchronous as well as asynchronous mode of data transfer for the delay estimation in coming sections.

3.1. *Content already available for data request*

We have assumed that MQTT server has a TOPIC and is already receiving data from sensing node Ni. In this case, the SUBSCRIBE request takes place after the PUBLISH request. Moreover, each request, whether it is for SUBSCRIBE or CONNECT, follows the packet loss due to TCP behavior and channel characteristics. Therefore we include the packet loss probability for i failures, 1 successfully sent connection request but failed acknowledgement CONNACK/SUBACK, j number of failures from server/-gateway side to the sender, in such way that $(j+1)th$ connection has successful connect/subscribe and CONNACK/SUBACK. We call this the probability of handshake $P_h(i,j)$, provided i is the number of attempts done by the sensing node to connect with the gateway and j is the number of attempts done by the gateway to acknowledge the connect request. Let the probability of failures of the connect request from the sensing node is $P_f(i)$, i.e., the forward packet loss probability, and that of success is **(1-pf)**. Then the probability of successful connection for one side delivery of request is given by (1) as;

$$P_f(i) = p_f^i.(1 - p_f) \tag{1}$$

The same phenomena will occur when the gateway **G** does an acknowledgement for the received request. Here, the probability of the successful request is mathematically shown from the probability of reverse packet loss, $\mathbf{p_r^j}$. The probability of this event is calculated from a single successful attempt and all the unsuccessful attempts from the gateway to the sensing node, given by (2) as;

$$P_r(j) = p_r^j.(1 - p_r) \tag{2}$$

Keeping these two-sided chances in view, the probability of a successful handshake is given by

$$P_h(i,j) = P_f(j).P_r(i) \tag{3}$$

$$P_h(i,j) = p_f^j.(1 - p_f).p_r^i.(1 - p_r) \tag{4}$$

Since TCP usually takes 4–6 times before establishing connection between sender and the receiver, thus the calculated probability $\mathrm{P_h(i,j)}$ causes initial latency for sending a transfer request, called Round Trip Time (RTT) [28]. This **RTT**, accompanied with the probability of failure, gives us the total latency for a successful handshake, given by (5) such that

$$L_h(i,j) = RTT + \left(\sum\nolimits_{k=1}^{i-1} 2^k T_s\right) + \left(\sum\nolimits_{k=1}^{j-1} 2^k T_s\right)$$

$$L_h(i,j) = RTT + (2^i - 1)T_S + (2^j - 1)T_S$$

$$L_h(i,j) = RTT + (2^i + 2^j - 2)T_S \tag{5}$$

Ts is the SYN (Synchronization) time for a single packet to go from sender to receiver or vice versa, in TCP connection. The overall probability of latency $\boldsymbol{P}[\boldsymbol{L_h} \le \boldsymbol{t}]$ for successful acknowledgement and successful request on a low data rate channel is thus calculated for time interval t by;

$$P[L_h \le t] = \sum_{L_h(\mathrm{i,j})} P_h(i,j) \tag{6}$$

The delay estimation can be given by the combination of round trips and delays in each TCP SYN-ACK process. The assumption for this part suggests 6 times of $L_h(i,j)$, i.e., connection establishment between any sensing node to the gateway, gateway connection to the server, user request to subscribe with a topic to the gateway of user, passing the subscription request of user to the server by the gateway and then for the data transfer two additional TCP connections are accomplished. The server connects and publish data to the gateway of the user and then finally, the user gets the data when the gateway requests for connection and it publishes the data on the user side.

Here we assume a queuing delay that is experienced by the TCP transmission in the sensing side gateway, denoted by **ε**. The gateway that receives TCP request from user, adds a queuing delay **ε** to the transmission RTT. The MQTT server and related gateway also embed a queuing delay that is denoted by **Φ**. The user can either request the gateway to connect to the server for registering/subscribing to a topic or it may receive connection request from the gateway if the gateway has some data to be fetched by MQTT user. This delay is denoted by **δ**.

The content delivery time is thus calculated by adding up the delays and RTTs for each segment in the communicating device route. The trip from sensor node **n** to user **u,** via gateway **g** and server **s,** is denoted by **ψ** and is calculated as under.

$$\hat{\hat{I}} = T_{connect-ng} + T_{pub-ng} + \varepsilon + T_{connect-gs} + T_{pub-gs} + \phi + \delta + \delta' + T_{connect-ug} + T_{sub-ug} + \varepsilon + T_{connect-gs} + T_{sub-gs} + \phi + T_{connect-sg} + T_{pub-sg} + \varepsilon + T_{connect-gu} + T_{pub-gu} \quad (7)$$

Considering the assumption that the RTT is symmetric for sensor-gateway and user-gateway, we can combine the connection request times as **6xRTT**, as in [27]. Also, the scenario suggests that there is no packet segmentation in TCP since the sensor data is not big enough to be segmented for TCP Packets. This infers that T_{pub}, $T_{connect}$ and T_{sub} are of same header size and packet length. The sensor to user data travel duration is thus reduced to (8.1) and (8.2)

$$\Psi = 6RTT_{connect} + 4RTT_{publish} + 2RTT_{subscribe} + 3\varepsilon + 2\phi + \delta \quad (8.1)$$

$$\Psi = 12RTT_{pub} + 3\varepsilon + 2\phi + \delta \quad (8.2)$$

All connections are assumed to be using TCP.

3.2. Content not available for data request

Here we consider the assumption that contents are not available at the server. Whereas the overall connection from user to sensor needs to be established for subscribing to a topic. After the user subscribes to a topic with the central server, it requests for sensed data. This leads the server to run an MQTT query (a request) for sending data from the sensor. Thus we can say that the delay **δ'** is a variable that may take any value between $\check{T}_{req}$ and $\check{T}_{cnt}$. This $\check{T}_{cnt}$ is arrival instant (time stamp) of the content. Collectively calling this δ' delay and δ (the request time for content if it is available on the server) as **θ**.

By combining delay due to not availability of information and request for contents, the expression for Ψ is written as (9.1) and (9.2)

$$\Psi = 12RTT_{pub} + 3\varepsilon + 2\phi + \delta + \delta' \quad (9.1)$$

$$\Psi = 12RTT_{pub} + 3\varepsilon + 2\phi + \theta \quad (9.2)$$

3.3. Estimation of θ, RTT, Φ and ε

In next section, we derive θ for specifying the random delay that is experienced by the in the user before sending CONNECT request. The CONNECT request backs up by SUBSCRIBE request. Know that this delay is uniformly distributed between the instant of arrival of request and instant of arrival of contents, i.e. $\theta = [\check{T}_{cnt}, \check{T}_{req}]$. The propagation delay δ is already added to this content delivery time. Thus, the uniform distribution for θ becomes.

U[T_{cnt}, T_{cnt} + T_{cnt} -δ], δ denotes the propagation/handshake process delay for CONNACK/CONNACT. This expression is a plausible response capture time whereas the propagation delay is given by (10).

$$\delta = 3RTT + \varepsilon \quad (10)$$

Also, the mean random delay for user to Gateway and Gateway to Server is given by (11).

$$\mu_\theta = (t_{cnt} + 2T_{cnt} - \delta)/2 \quad (11)$$

In case of prior availability of contents at the server, the above mean delay expression uses T_{req}, such that

$$\mu\theta = \left(t_{req} + 2T_{req} - \delta\right)/2 \quad (12)$$

Here, t is the duration after an event has occurred and T is the interval of presence of a request/response.

Now we calculate the Round-Trip Time for the request/response for the MQTT multilayer service. Our assumption suggests that all the connections are in TCP thus we can use Jacobson algorithm [29], for calculating the delay incurred by RTT. This RTT consists of CONNECT, followed by CONNACK for a single round trip. The Smooth Round Trip Duration (SRTT) [30,31,32] is thus calculated for a steady state data transmission in TCP, using the relation in (13).

$$SRTT[i] = (1-\beta).SRTT[i-1] + \beta.RTT \quad (13)$$

Also,

$$RTT = RTT + \Delta.Diff, Diff = SRTT - RTT \quad (14)$$

Where $0 < \beta < 1$ and $0 < \Delta < 1$. The variation in RTT is checked and compared from packet to packet and is termed as smooth (at a steady rate) after reaching a certain threshold. The work in [27] uses β = 0.125 for smooth RTT.

The queuing delay Φ depends upon scheduling mechanisms. Here we assume two types of schedulers on each server. One scheduler is dedicated for publishing services whereas the second scheduler is responsible for holding subscription requests from various MQTT nodes. Fig. 4 is an illustration of the content-topic matching queuing.

As suggested by [27], we are considering exponential distribution for the generation of random data packets on the sensing nodes, i.e., Poisson arrival. Since MQTT is a profile aware forwarding mechanism [33], the periodic packets are filtered for relevant topic and only relevant packets are released by the sensor for transmission to the MQTT server. Thus, the PUB scheduler may be considered as fixed job schedule scenario. For the same reason, round robin fixed job quantum

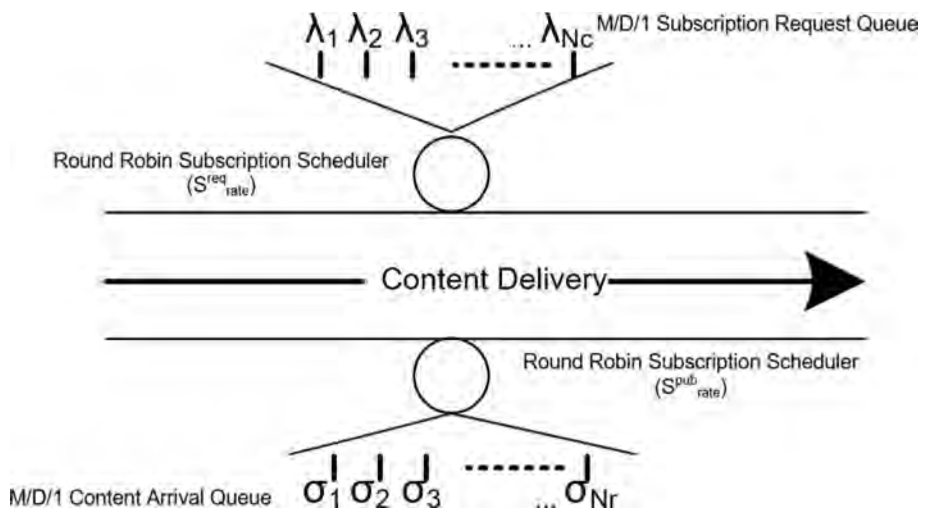
λ1 λ2 λ3 ... λNc
M/D/1 Subscription Request Queue
Round Robin Subscription Scheduler (S^req_rate)
Content Delivery
Round Robin Subscription Scheduler (S^pub_rate)
M/D/1 Content Arrival Queue
σ1 σ2 σ3 ... σNr

Fig. 4 M/D/1 content and request arrival queue in MQTT server.

scheduler is considered [34]. The capacity is assumed to be $\mathbf{S_{rate}}$ bits/sec.

Let the contents generating on different topics are independent and each node Ni in total nodes $\mathbf{N_c}$ generates contents on distinct topic called TOPIC ID. This makes each topic-based packet independent. For the sake of brevity, the length of the packets is considered similar and equal to **L** bits. The bitrate (s) of these L bits are not similar and thus each stream of bits in a packet are denoted with a separate transfer rate **λi**. Subsequently, for a stable round robin fixed job quantum scheduler, the aggregated arrival holds (15). For connected users $\mathbf{N_c}$,

$$\sum_{i=1}^{Nc} \lambda_i \leq S_{rate}$$

This also suggests that the optimized virtual service, for topic **i**, will take an average arrival process time as given by (16)

$$\mu_i = \frac{\lambda_i}{\sum_{i=1}^{Nc} \lambda_i} S_{rate} \tag{16}$$

The service duration for a full packet of length L is thus calculated as $\boldsymbol{s_i = L/\mu_i}$. For such fixed valued service duration, as suggested by fixed length of the packet, the queue model is M/D/1 as in [35], sometimes referred as **M/G/1**.

Now, let $\mathbf{D_i}$ be the number of packets waiting on a station in queue $\mathbf{Q_i}$, the Poisson distribution of **M/D/1** suggests mean of the packet service as in (17);

$$E[D_i] = \frac{2-\rho_i}{2\mu_i(1-\rho_i)} \tag{17}$$

where, $\rho_i = \lambda_i/\mu_i$.

The total service time, as done for CoAP [36], is given by multiplying transfer rate of packet with the mean service time, as given by (18)

$$E[Q_i] = \lambda_i E[D_i] \tag{18}$$

The maximum delay for packet arrival for a symmetric as well as non-symmetric communication scheme is thus given in (19)

$$E[D_i^{req}] = \frac{2-\rho_i^{req}}{2\mu_i^{req}(1-\rho_i^{req})} \tag{19}$$

Also, the $\boldsymbol{E[Q_i^{req}]}$ is calculated from this delay as

$$E[Q_i^{req}] = \frac{(2-\rho_i^{req})\rho_i^{req}}{2(1-\rho_i^{req})} \tag{20}$$

Therefore, the maximum mean queuing delay is calculated by content arrival rate $\boldsymbol{\lambda_i}$ and request arrival rate $\boldsymbol{\sigma_i}$ as

$$Q[\lambda_i, \sigma_i] = \frac{2-\rho_i^{cnt}}{2\mu_i(1-\rho_i^{cnt})} + \frac{(2-\rho_i^{req})\rho_i^{req}}{2(1-\rho_i^{req})} \tag{21}$$

$$\rho_i^{req} = \sigma_i/\mu_i \tag{22}$$

$$\rho_i^{con} = \lambda_i/\mu_i \tag{23}$$

We can also use,

$$\mu = \frac{1}{D} \tag{24}$$

D is the fixed service delay by the server or the gateway in **M/D/1** Queue model. This delay is used for a streamline processing instead of $\boldsymbol{\mu_i}$, whenever required.

This queuing delay $\mathbf{Q}[\lambda_i, \sigma_i]$ is calculated for a virtual service rate $\boldsymbol{\mu_i^{req}}$, for a fixed capacity of transmission rate **S** of the scheduler, This $\boldsymbol{\mu_i^{req}}$ is further given by (25)

$$\mu_i^{req} = \frac{\sigma_i}{\sum_i^{Nr} \sigma_i} S \tag{25}$$

3.4. Estimation of sensor to user trip for QoS-0

Now that we have derived the expression of **Ψ** for service acknowledgement, we can easily convert it to **QoS-0** by removing the PUBACK RTTs. Initially we have **12xRTT** in the (7). In the current consideration, the **RTT** for PUBLISH doesn't get PUBACK, reducing the number of Packets transmitted and thus the network engagement due to **Tpub-ng, Tpub-sg, Tpub-gs** and **Tpub-ug** is reduced. Thus, the new **Ψ** is given in (26).

$$\Psi = 8RTT_{pub} + 3\varepsilon + 2\phi + \theta \tag{26}$$

This is a confirm improvement of 25% in latency with a maximum introduction of 93.75% failure causing events.

3.5. Estimation of sensor to user trip for QoS-2

A confirmation is sent back in case of each successful delivery for achieving QoS-2 [37]. Here 3 extra packets are communicated between MQTT client and MQTT broker. After the client publishes data for a specific topic to a broker, the broker sends a PUBREC packet back to the client. When the client receives this packet, it discards the sent packet of that topic for further retransmission(s). Then again, if the client does not get affirmation as PUBREC, it sends the same PUBLISH request with a DUP flag [38]. This process is repeated until the client receives a PUBREC. The client saves this reply packet and sends an acknowledgement as PUBREL. The broker can now safely discard its current state of reception and send a PUBCOMP packet. This packet enables the client for reusing the packet identifier used for previous data exchange as given by Fig. 5.

The introduction of above scheme is in QoS-0 scenario for confirming its reliability. Thus PUBACK is not necessary as QoS-1. PUBREC, PUBREL and PUBCOMP are included 2 times as these packets are communicated between sensor-gateway, gateway-server and server-user pairs. A complete round trip across network i.e., sensor through gateway, MQTT server and user's gateway to user adds 6 RTTs two times. A total of 8 RTTs as that of QoS-0 is thus increased to 20 RTTs for a full trip.

$$\Psi_{sensor-user} = 20RTT_{pub} + 3\varepsilon + 2\phi + \theta \tag{27}$$

The process is partly shown for a single connection by Fig. 5.

3.6. Content delivery probability

For content delivery, we have assumed that each TOPIC ID subscription is requested before the accessing process of content. Also, the TOPIC ID is dealt independently for unique TOPIC in terms of its TOPIC ID. This may be concluded for the fact that matching station works on the SUB request of the TOPIC based on TOPIC ID in MQTT. Contents on a single TOPIC is delivered if the TOPIC ID is not expired and the content is not expired or replaced. This time is denoted by $\mathbf{T_{cnt}}$. Each request is honored if the content is available, defined by this $\mathbf{T_{cnt}}$.

The discussed fact poses two scenarios [39]. The one in which content is arrived before the subscription request as in Fig. 6. Here, the $\mathbf{T_{req}}$, time for a content request is overlapping the $\mathbf{T_{cnt}}$, i.e., the duration up till which, the content is available. In the second scenario the request for content is arrived before the user is subscribed to the same topic, as in Fig. 7.

We assume that $\check{T}_{req}$ is the time stamp of the arrival of content request and T_{req} is the duration for which this request sustains as in [40] and [41].

$$P_{cnt}(X = \hat{\mathbf{x}}) = P(\check{T}_{req} > \check{T}_{cnt}) + P(\check{T}_{req} \le \check{T}_{cnt}) \tag{28}$$

$\check{T}_{req}$ is the instant at which contents are published on a TOPIC ID and T_{cnt} is the duration till which this content remains on the sharing point/server.

The pdf of this overlapping arrival event (X) is given by the (29)

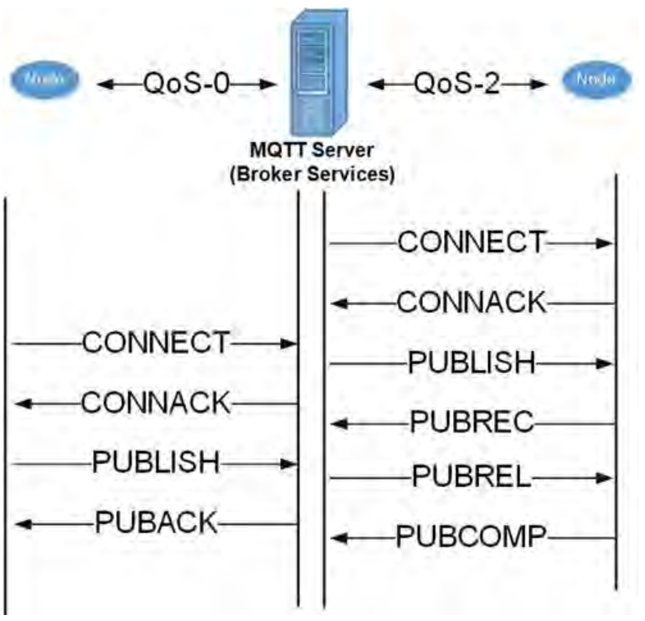

Fig. 5 QoS-0 and Qos-2 Handshake and Publishing mechanism for a TOPIC in MQTT.

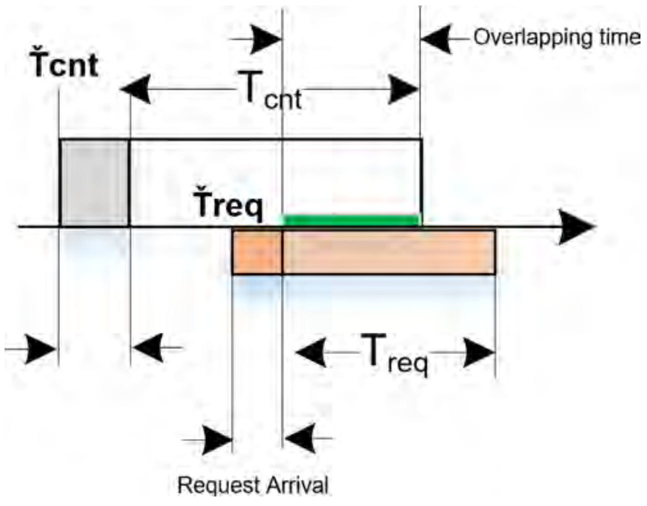

Fig. 6 Timeline of contents availability before user request.

$$P^{D}_{cnt} = P(\check{T}_{req} > \check{T}_{cnt}) \int_{t=0}^{\check{T}_{req}} P(X = X_{req}) f_i^t(X)dt \, P(\check{T}_{req} \le \check{T}_{cnt}t) \int_{t=0}^{\check{T}_{cnt}} P(X = X_{cnt}) f_i^t(X)dt \tag{29}$$

Here, $\boldsymbol{f_i^t(X)} = \zeta_i e^{-\zeta_i t}$ is the probability density function of next arrival i at time t. Also, $\zeta_i = \sigma_i + \lambda_i$ which is the summation of request arrival rate of requests and arrival rate of packets. (29) suggests that the arrived packet may either be a request for content or the content itself. Thus, the probability that an anticipated packet is a request is calculated from the ratio of the rate at which the requests are received, given by

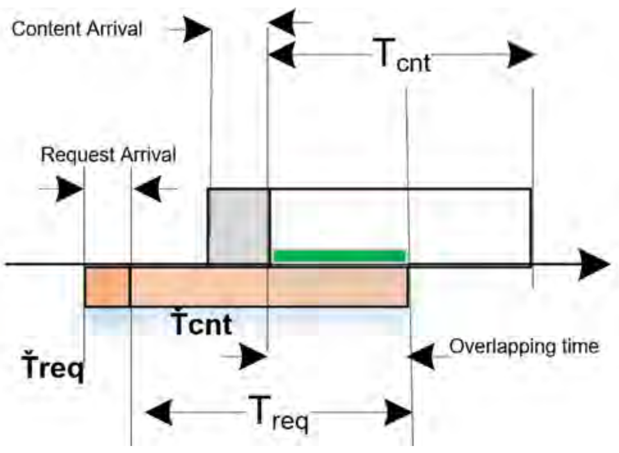

Fig. 7 Timeline of contents availability after user request.

$$P(\hat{\mathbf{x}} = \hat{\mathbf{x}}_{req}) = \frac{\sigma_i}{\zeta_i} \tag{30}$$

Also, the probability that the next packet is a content packet is given as

$$P(\hat{\mathbf{x}} = \hat{\mathbf{x}}_{cnt}) = \frac{\lambda_i}{\zeta_i} \tag{31}$$

By substituting the above probabilities in (29), we get

$$P^D_{cnt} = P(\check{T}_{req} > \check{T}_{cnt}) \int_{t=0}^{\check{T}_{req}} \frac{\sigma_i}{\zeta_i} \zeta_i e^{-\zeta_i t} dt + P(\check{T}_{req} \leq \check{T}_{cnt}) \times \int_{t=0}^{\check{T}_{cnt}} \frac{\lambda_i}{\zeta_i} \zeta_i e^{-\zeta_i t} dt \tag{32}$$

For homogenous and independent probabilities, the probability of $\boldsymbol{P}(\check{T}_{\mathbf{req}} \leq \check{T}_{\mathbf{cnt}})$ must follow Poisson process, i.e., $\boldsymbol{P}(\check{T}_{\mathbf{req}} \leq \check{T}_{\mathbf{cnt}}) = \frac{\sigma_i}{\zeta_i}$.

Similarly, $\boldsymbol{P}(\check{T}_{\mathbf{req}} > \check{T}_{\mathbf{cnt}}) = \frac{\lambda_i}{\zeta_i}$, which upon substituting in (32), reduces it to the following as;

$$P^D_{cnt} = \frac{\lambda_i}{\zeta_i} \int_{t=0}^{\check{T}_{req}} \frac{\sigma_i}{\zeta_i} \zeta_i e^{-\zeta_i t} dt + \frac{\sigma_i}{\zeta_i} \int_{t=0}^{\check{T}_{cnt}} \frac{\lambda_i}{\zeta_i} \zeta_i e^{-\zeta_i t} dt \tag{33}$$

By rearranging (33), we get the probability of content delivery as a function of request arrival rate, content arrival rate, total packet arrival rate and time, in respect to the event occurred of content arrival $\check{T}_{cnt}$ and request arrival $\check{T}_{req}$, as in (34)

$$P^D_{cnt} = \frac{\lambda_i \sigma_i}{\zeta_i} \left(\left[1 - e^{-\zeta_i / T_{cnt}}\right] + \left[1 - e^{-\zeta_i / T_{req}}\right] \right) \tag{34}$$

A special case where the request comes first on Fully Functional Devices [42] in IoT, (34) can be transformed to (35) as

$$P^D_{cnt} = \frac{\sigma_i^2}{\zeta_i} \left(\left[1 - e^{-\zeta_i / T_{cnt}}\right] + \left[1 - e^{-\zeta_i / T_{req}}\right] \right) \tag{35}$$

This evident the fact that probability of content delivery improves while the requests are made beforehand. For requests that are arrived before availability of contents, whereas symmetric data rate is observed via channel, $\boldsymbol{\zeta_i}$ can be approximated as $\mathbf{2\sigma_i}$ or $\mathbf{2\lambda_i}$. Reducing (35) to

$$P^D_{cnt} = \frac{\sigma_i}{2} \left(\left[1 - e^{-\zeta_i / T_{cnt}}\right] + \left[1 - e^{-\zeta_i / T_{req}}\right] \right) \tag{36}$$

Thus, the maximum serving probability of a request, an ideal communication, equals to the request rate $\boldsymbol{\sigma_i}$. (36) is useful while the IoT comprises of actuators that take data from a local or remote server for its functionality.

The overall probability for either Node to gateway communication over TCP or Node to gateway communication over UDP doesn't change the probability of packet delivery though it may pose latency in the packet delivery [43]. This is decided from the ratio of the content and request rates, where the total number of requests averaged over time is normalizing the ratio.

3.7. Discussion on End-to-end service assurance:

Expression in (8.2) is the summation of all types of delays, including round trip time delay RTT, sensor side gateway's queuing delay ε, user side gateway's queuing delay **Φ** and delay caused for fetching the requested contents **δ** and the request time **δ'**, collectively denoted by **θ** in (9.2), representing service duration.

Again, **Φ** is the function of T_{cnt} and T_{req}, that are directly related to the content arrival rate λ and request arrival rate **σ**. The queueing delay **Φ** also depends upon the number of connected end users Nr and number of connected sensors nodes $\mathbf{N_c}$.

The delay due requested content **δ** w.r.t time gives the content arrival rate **λ** and that points towards the same ingredients as that of **Φ**. Same can be stated for request arrival rate σ that is connected to delay posed by entertaining a request with a delay **δ**.

Thus, the variables that should be concerned before modeling a delay tolerant MQTT based IoT include number of connected content providers $\mathbf{N_c}$, number of end users $\mathbf{N_r}$, the request arrival rate σ, the content arrival rate **λ** and the constant service rate **θ**. The end-to-end service assurance policy is determined by considering the system parameters likewise. The scalability of such IoTs is determined by recognizing the effect of each concerned variable regarding content/request rates and services it offers that pose constant and/or variable delays to information streams [44]. Here, we determine the maximum number of connected nodes in an IoT for a defined end-to-end content delivery time.

One may also be interested in determining the end-to-end delay and content delivery assurance based on the available information of service time **θ**, content arrival **λ** and request arrival rates **σ**. (34) is responsible for assuring the desired level of content delivery as the pdf of content delivery probability $\mathbf{P^D_{cnt}}$, in terms of holding the time for which the request is held by the content server before finishing the request, given by T_{req}, the time till when the contents are put on content server, represented by T_{cnt}, the individual packet of content request rate σ_i, and individual content arrival rate , λ_i. Note that ζ_i is the summation of the last two parameters described.

Since, MQTT works on topic wise content arrivals and request arrivals. And our assumption consisted of one TOPIC per connected nodes, individual **Nc** and **Nr** can be related directly to the TOPIC currently in service at any instant i, on a MQTT broker/server. Increasing TOPIC_IDs will have a direct effect on the **Nc** and **Nr** collectively even when hosted from a virtually similar node for different TOPIC IDs [45]. Remember that TOPIC ID is the actual service that requires delivery of contents from Nc and request for contents from **Nr**.

4. Server and client testbed setup

We have considered three different environments for content and request delivery over various nodes. The emulated environment has been organized by deploying MQTT [27] broker on a Linux server/broker. The clients are users and actuators that publish data to a set of 4 user define topics. The nodes are connected over standard 2.5 GHz Wi-Fi and CAT6, to this MQTT broker/server. The server communication with its nodes is analyzed in three different manners [27,46,47]. These setups are categorized based on the placement of MQTT server in the network. The said MQTT server is deployed in three different manners, i.e., deployment on Windows OS [48], deployment on Linux server on VM [48], deployment on remote MQTT server [48].

4.1. Scenarios for server deployment

In first setup, the localhost server is used in which VM based clients are used for communication with the server. This type of configuration will pose minimum realistic delay and packet loss in listing performance-based aspects of the MQTT broker. The second scenario is set up by enabling local network clients to connect with the Linux server on VM and its performance is checked from Terminal where the activity log is exported to a text file. Time stamp-based analysis is done in this setup that consists of range of clients' connection with the server for publish and subscription requests. Diversity is obtained by managing various topics IDs on clients. This gives us the idea of network traversing over local network in realistic environment. We run multiple client's deployment on PC connected on the same network without network overheads of external inbound and outbound traffics. In the third scenario, Mosquitto broker is used on Eclipse IoT server and clients from Windows and VM are subscribed and registered for various topics.

Each VM and Windows PC is working as Gateway. For approximating the scenarios, the paper assumes a hospital environment as in [49]. The work doesn't constrain to a single room and covering a patient with BAN but rather data on same topic is published on server. Here patient is considered as multiple clients because a patient might provide data on different topics to the server.

For the sake of brevity, we have considered heartbeat sensor, temperature sensor, fall detection sensor and moisture sensor as primary data.

4.2. Other connectivity assumptions and calculations

The content arrival rate is taken as same the request arrival rate. Further, both λ and σ varies between 100kbps and 2Mbps. The client node traffic is generated using Constant Bitrate (CBR) that is randomized for the value of arrival rate as its mean and following Poisson distribution [50].

The RTT for TCP is obtained in real time traffic using the data from trace file of ping6 function in terminal on Linux and same function on Windows by employing Power shell utility. Value of gateway queueing delay $\boldsymbol{\varepsilon}$ is set to 720 µs as in [51]. Maximum Payload size is set indirectly by CBR traffic [52]. Availability time of request for content and the content on the server is set to 10 ms as in [51]. Note that two M/D/1 queues are operated at the broker, following the design in [28]. For that, the $\mathbf{E}[\boldsymbol{D}_i]$ and $\mathbf{E}[\boldsymbol{Q}_i^{req}]$, as given in (19) and (20), estimated the QoS. The research in [27] uses an infinite queue size but this paper limits it to the hardware of the server that are also behaving as a non-over-flowing space, i.e., local server of 5 GB dedicated capacity and Mosquitto Eclipse server of 80 GB shared capacity.

5. Result and discussion

5.1. End-to-end service delay analysis in FFDs

Fig. 8 shows the path delay in reaching the services of deployed servers in each scenario. Readings are taken using Wireshark logs and trace file via Linux terminal commands and Windows Power shell. A client, as content provider, behaves as same as a user, requesting for.

contents or subscribed to a topic. That is the reason why $\boldsymbol{\lambda}$ and $\boldsymbol{\sigma}$ are considered as same. Topics are incremented between 1000 and 11,000 with an increment or 100 new client's additions (each as a topic subscription) for averaging the delay time and the request time is kept constant during each run. Results in Fig. 8 shows that the path delay is directly engaging with the QoS. Locally hosted server gives response to connected VM based clients in 2.6 ms with QoS-2 that increases to 3.7 ms with increase in request rate. For QoS-1, same pattern is followed but the end-to-end delay reduces to 2.35 ms instead 2.6 ms. At 2Mbps request rate, this delay rises to 3.5 ms, lower than QoS-2. The performance is exceptional, partly identical to QoS-1 in terms of end-to-end delay. With an initial end-to-end delay of 2.25 ms at 0.9Mbps request rate, QoS-0 outperforms both QoS-1 and QoS-2. The delay is noted to be crossing 3.4 ms mark at $\boldsymbol{\lambda}$ of 2Mbps.

MQTT server hosted on Virtual Machine with local clients is also analyzed in Fig. 8. In this case, the initial end-to-end service delay ranges between 6.1 ms and 6.36 ms for QoS-0, QoS-1 and QoS-2. A maximum delay of 6.8 ms is noted for a request rate of 2Mbps at QoS-2. Due to the PUBACK and SUBACK packets, higher end-to-end delay is experienced in remote server that rises from 8.28 ms to 8.33 ms. The behavior of QoS-1 is slightly tilted towards QoS-0 that can be seen from the end-to-end delay values in each drawn graph in Fig. 8.

Results of the analysis of End-to-end delay while increasing the number of subscribers and publishing nodes, i.e., Nr and Nc are graphed in Fig. 9. These contents are varied from 1 topic up to 2800 topics for which the PUB/SUB request can be entertained by the server. The graph gives estimated number of services that can be run on a single service with given QoS. Note that topic is associated to a single client due to which these numbers can be treated as same quantity. The results in Fig. 9 shows that QoS-0 is ranging from 2 ms to 3.2 ms in terms of end-to-end delay. The delay stretches to 3.65 ms for QoS-2 for 2800 active subscribers while maintaining a steady request rate of 2.0Mbps. Analysis concludes on

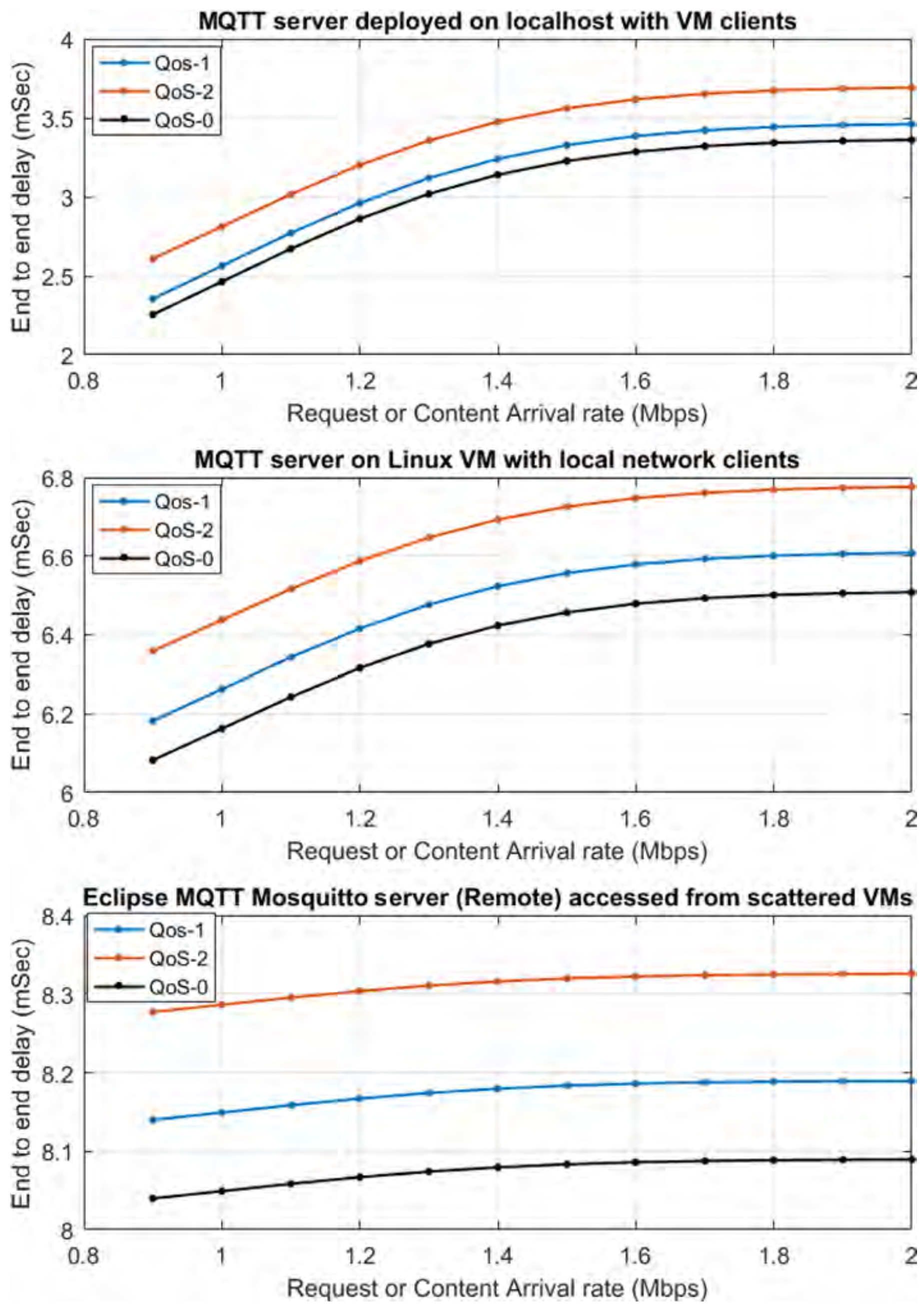

Fig. 8 QoS-0, QoS-1 and QoS-2 real time analysis for calculating end-to-end delay in localhost, Linux VM and Remote MQTT broker.

the fact that a delay Δt_d of 1.35 ms is experienced while increasing the numbers of users from 1 to 2800. An average process-delay of 0.5 µs is experienced by the server regardless of its placement in the network.

End-to-end delay for content and request availability duration is analyzed in Fig. 10. For the sake of brevity, service time is kept same so is $\mathbf{T_{cnt}}$ and $\mathbf{T_{req}}$.

As in [28], the effect can only be seen if one in the $\mathbf{T_{cnt}}$ or $\mathbf{T_{req}}$ is stretched. It was observed that by fixing any of the these, the variation of other term gives drastically different results in terms of network performance. In this paper, we have kept $\mathbf{T_{req}}$ constant and tweaked $\mathbf{T_{cnt}}$. Increasing T_{req} doesn't increase the end-to-end delay but improves the $\mathbf{P_D^{cnt}}$. Likewise, increasing $\mathbf{T_{cnt}}$ increases and also increases end-to-end delay. Graphed results show that for an average increase in the $\mathbf{T_{cnt}}$ of 2.9 ms, a delay of 2 ms is experience in end-to-end information exchange. So, for a fixed number of subscribers, a 600 ms additional holding of content, the end-to-end delivery service will be affected by a rounded figure of 600 µs. The experiments were performed for a fixed Nc of 1500 node count.

5.2. Probability of content delivery (P_D^{cnt})

(33) and (34) gives an insightful notion about the performance of setup smart network w.r.t $\mathbf{T_{cnt}}$, $\boldsymbol{\lambda}$, $\boldsymbol{\sigma}$ and $\boldsymbol{\zeta}$, whereas $\boldsymbol{\zeta}$ is the sum of $\boldsymbol{\lambda}$ and $\boldsymbol{\sigma}$. The exact same picture is given by Fig. 11(a) where $\mathbf{P_D^{cnt}}$ is estimated for $\mathbf{T_{cnt}}$ values in microseconds. These

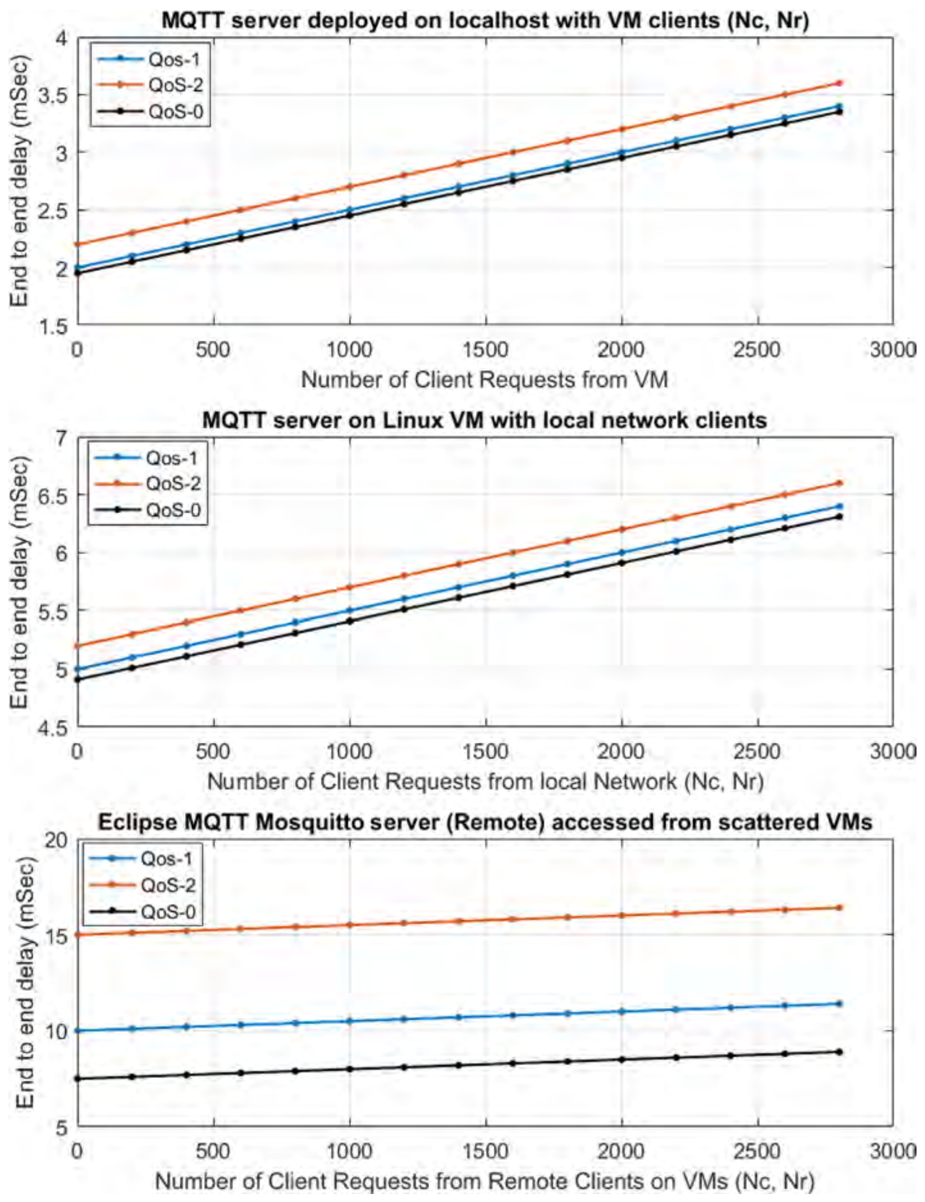

Fig. 9 End-to-end delay analysis w.r.t number of clients' requests in localhost, Linux VM and Remote MQTT broker.

probabilities are analyzed for a $\mathbf{T_{req}}$ duration of 1 µs, 5 µs, 10 µs, 15 µs and 20 µs. The curves depict that a linear increment in $\mathbf{T_{req}}$ causes the exponential graphs.

to get closer to one another with increase in T_{cnt}, from 0 µs to 20 µs. Now the same probability equation (34) is applicable on all 3 scenarios of deployment of the MQTT broker/server. Fig. 11(a) unveils the direct relationship of content availability assurance with probability of content delivery. Each T_{req}, puts an upper bound of the maximum achievable content delivery probability. Increasing T_{req}, to a value of 20 µs makes the MQTT service extremely reliable at given T_{cnt}. The results are only valid for **M/D/1** queue with infinite buffer size for holding requests and contents at the MQTT server.

Variation in the content delivery assurance w.r.t content arrival rate is also portrayed in Fig. 11(b). The content arrival rate as well as request arrival rates are varied from 0.1Mbps to 2.0Mbps, where the buffer size for holding contents as well as request is kept in same range as the buffer size. The figure depicts that $\boldsymbol{\lambda}$ and $\boldsymbol{\sigma}$ are inversely related in terms of $\mathbf{P_D^{cnt}}$. More requests lead to less delivery of content in such case. Also, both $\boldsymbol{\lambda}$ and $\boldsymbol{\sigma}$ has effect on the content delivery when their values are closer to the maximum supported network data rate. The graph also gives us an idea about the future of smart cities in terms of higher data rates as we expect higher data rates in future and with same set of devices, we can achieve much higher performance by increasing $\boldsymbol{\lambda}$ and $\boldsymbol{\sigma}$**.**

5.3. Real time content delivery

At the end of the day, we all are interested in the performance of MQTT servers is real time. Fig. 12 is the elaboration of the fact that increasing the clients drastically degrades the services of MQTT brokers. For this experiment, both $\boldsymbol{\lambda}$ and $\boldsymbol{\sigma}$ are kept at 2.0Mbps. Here, the services are less effected when the client

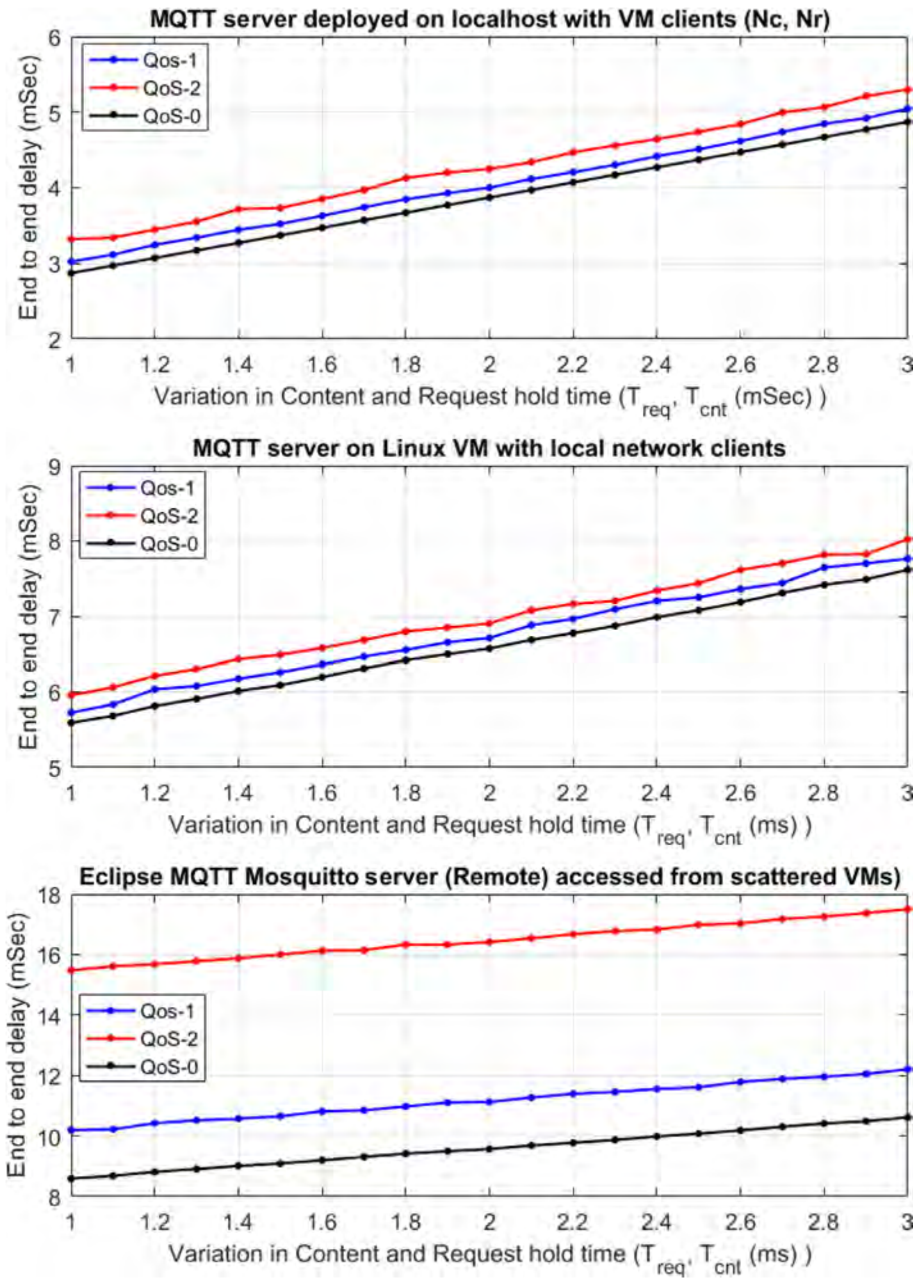

Fig. 10 End-to-end delay analysis w.r.t $\mathbf{T_{cnt}}$ and $\mathbf{T_{req}}$ in localhost, Linux VM and Remote MQTT broker.

has lower RTT and $\mathbf{\Psi}$. The remote MQTT broker, that is represented by red line in Fig. 12 collectively explains the degradation of the server. The Mean Absolute content delivery reduces to 0.59 while the clients $\mathbf{N_c}$ exceeds a mark of 11000.

6. Comparative study and discussion

The paper gives an insight of various parameters that assure the end-to-end service in Fully Functional Devices (FFD) used in smart cities. We have examined three different placements of MQTT servers and three QoS levels are analyzed. The study penetrates deep into the core components of end-to-end delay for content delivery, Probability of content delivery and real time failure rate. It was mathematically proved that the performance evaluators depend on the processing time, queueing time, rate at which the contents and requests are generated, duration till when contents and requests are to be kept at server, Number of connected clients and subscribed topics and QoS. The comparison of CPU Usage and Message Transmission latency $\mathbf{\Delta t_d}$, is given in Table 1 which clearly indicates that the proposed improved MQTT outperforms Apollo, JoramMQ, RabbitMQ, Basic MQTT in both Qos-0 and Qos-1 with a cost of processing power.

The placement of server is exploited for real time content delivery and service assurance in the work. All these distinguished parameters and system design aspects gives a clear pic-

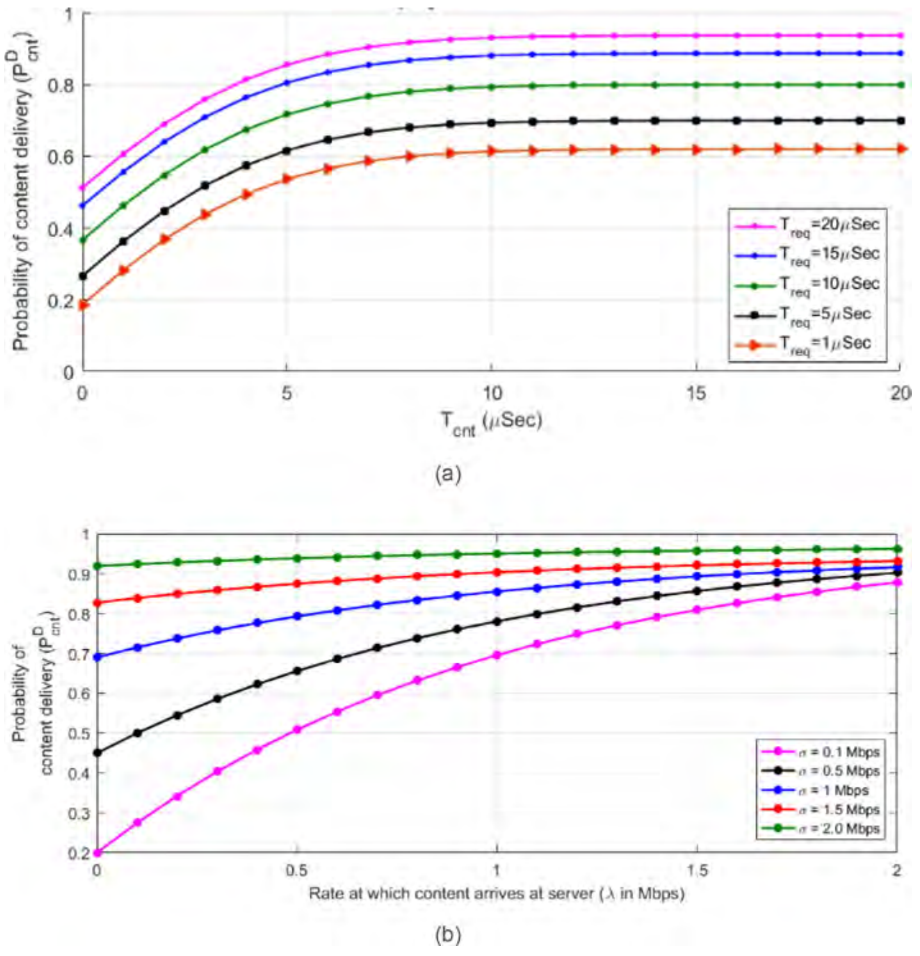

Fig. 11 Probability of content delivery w.r.t (a) T_{cnt} and T_{req} (b)λ and σ (Mbps).

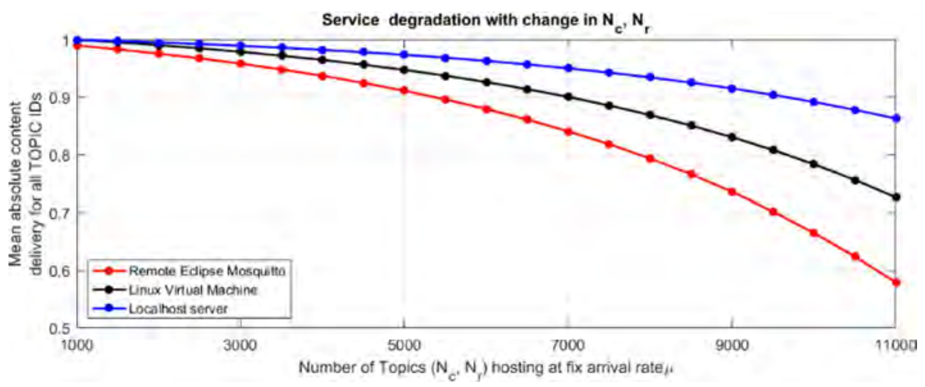

Fig. 12 Mean Absolute real time content delivery w.r.t Nr, Nc (subscribers and publishers) for various server setups.

Table 1 Comparison of the improved MQTT protocol with Benchmark MQTT, RabbitMQ, JoramMQ and Apollo servers.

Service	Parameter	Apollo [45]	JoramMQ [45]	RabbitMQ [45]	Mosquitto Server	Improved MQTT on Linux
Qos-0	CPU Usage	6%	3%	38%	24%	11%
	Message Transmission Latency	5 ms	1.5 ms	10 ms	10 ms	3.5 ms
Qos-1	CPU Usage	6%	6%	34%	24%	12%
	Message Transmission Latency	6 ms	34 ms	> 10sec	6 ms	3.6 ms

ture to the design experts to achieve certain level of performance by setting the physical values as per mathematical formulation given in the paper. The research also gives free hand to the system designer to evaluate the system at any stage and thus determine its employability in a wider set of applications. The physical deployment sheds light on the viability of the system model with real figures that will help the system designers to cop various issues while dealing with MQTT brokers and clients in development of effective smart cities' IoT.

7. Future work

The future work can go in two different directions. First, the performance of the deployed network can be improved by adding another level of QoS that works as QoS-0 with a fixed set of rules for retransmission of contents if requests are not entertained for a fixed duration. Thus, an On-Demand QoS level can be introduced to achieve higher content delivery probability. Secondly, the work uses TCP as the backbone for MQTT services. Though certain nodes may have UDP based connectivity that will not follow the same equations as developed in the work, a much higher content delivery rate can be proposed with lower content delivery probability with a fully UDP dependent network. The idea can be further extended while promoting MQTT-SN that uses UDP for local gateway interaction and TCP for gateway to server communication [24].

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Jawhar, I., Mohamed, N. & Al-Jaroodi, J., Networking architectures and protocols for smart city systems, Applied Journal of Internet Service, vol 9, no. 26, 2018.
[2] X. Huang, D. Zhang, X. Zhang, Energy management of intelligent building based on deep reinforced learning, Alexandria Engineering Journal 60 (2021) 1509–1517.
[3] C.C. Sobin, A Survey on Architecture, Protocols and Challenges in IoT Wireless Pers, Commun. 112 (2020) 1383–1429.
[4] W. Mardini, S. Aljawarneh, A. Al-Abdi, Using Multiple RPL Instances to Enhance the Performance of New and Internet of Everything (6G/IoE)-Based Healthcare Monitoring Systems, Applied Mobile Network 26 (2021) 952–968.
[5] B. Weiss, U. Hunkeler, A. Munai, W. Schott, L. Truong, A Publish/Subscribe Messaging System for Wireless Sensor Communication, 34th IEEE Local computer networks conference, LCN Zurich, Switzerland 34 (2009) 1–2.
[6] B. N. Alhasnawi and B. H. Jasim, SCADA controlled smart home using Raspberry Pi3 , International Conference on Advance of Sustainable Engineering and its Application (ICASEA) , Wasit, Iraq, vol. 3, pp. 1-6, 2018.
[7] P. Fraga-Lamas, T.M. Fernández-Caramés, L. Castedo, Towards the Internet of Smart Trains: A Review on Industrial IoT-Connected Railways, MDPI Journals – Sensors 17 (6) (2017) 1–44.
[8] L Serger, Solution MQTT, Accessed on: April 5, 2019, [Online]. Available: http://www.scalagent.com/en/mqperf-69/produits-70/presentation-370 /.
[9] B. S. Sarierao and A. Prakasarao, Smart Healthcare Monitoring System Using MQTT Protocol, 3rd International Conference for Convergence in Technology (I2CT), pp. 1-5, 2018.
[10] H. Hamoud, Alshammari, The internet of things healthcare monitoring system based on MQTT protocol, Alexandria Engineering Journal 69 (2023) 275–287.
[11] Eclipse Foundation Facebook using MQTT, Accessed on: Jan 1, 2023, [Online]. Available: https://mosquitto.org/blog/2011/08/facebook-using-mqtt/.
[12] M. Sarrab, S. Pulparambil, M. Awadalla, Development of an IoT based real-time traffic monitoring system for city governance, Global Transitions 2 (2020) 230–245.
[13] Yen, Y-H. Implementation of IoT communication technology on automated guided vehicle, National Central University, Taiwan (R.O.C.), 2017.
[14] Light R A, Mosquitto: server and client implementation of the MQTT protocol Open Source Journal Software, vol 2, 2017.
[15] J. Wirges and U. Dettmar, Performance of TCP and UDP over Narrowband Internet of Things (NB-IoT), IEEE International Conference on Internet of Things and Intelligence System (IoTaIS) Bali, Indonesia, pp. 5-11, 2019.
[16] A. Sahadevan, D. Mathew, J. Mookathana and B. A. Jose, An Offline Online Strategy for IoT Using MQTT, 4th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud), NY, USA, pp. 369-373, 2017.
[17] M. U. H. Al Rasyid, F. Astika and F. Fikri, Implementation MQTT-SN Protocol on Smart City Application based Wireless Sensor Network, 5th International Conference on Science in Information Technology (ICSITech), Yogyakarta, Indonesia, pp. 7-12, 2019.
[18] P. Sethi, S.R. Sarangi, Internet of Things: Architectures, Protocols, and Applications, Journal of Electrical and Computer Engineering 2017 (1) (2017) 1–25.
[19] S. Khriji, Y. Benbelgacem, R. Chéour, et al, Design and implementation of a cloud-based event-driven architecture for real-time data processing in wireless sensor networks. Open Access, J Supercomput (2021).
[20] Andrew Banks and Rahul Gupta. OASIS MQTT Version.
[21] 1.1. [Online]. Available: http://docs.oasis-open.org/mqtt/mqtt/v1.1/mqtt-v1.1.html/.
[22] M. Handosa, D. Gračanin and H. G. Elmongui, Performance evaluation of MQTT-based internet of things systems, Winter Simulation Conference (WSC), Las Vegas,USA, pp. 4544-4545, 2017.
[23] B. Kantarci, S.F. Oktug, S. Issue, Wireless Sensor and Actuator Networks for Smart Cities, Journal of Sensor and Actuator Networks 7 (49) (2018) 1–5.
[24] M. Chen, J. Wan, S. Gonzalez, X. Liao, V. Leung, A survey of recent developments in home m2m networks, IEEE Commun. Surveys Tuts. 16 (1) (2014) 98–114.
[25] K. Kavitha and G. Suseendran, "Priority based Adaptive Scheduling Algorithm for IoT Sensor Systems,"International Conference on Automation, Computational and Technology Management (ICACTM), London, UK, pp. 361-366, 2019.
[26] S. Lee, H. Kim, D. Hong and H. Ju, "Correlation analysis of MQTT loss and delay according to QoS level," 27th The International Conference on Information Networking (ICOIN), Bangkok, Thailand, pp. 714-717, 2013.
[27] Co. Pham, A. Bounceur, L. Clavier, U. Noreen, M. Ehsan, Radio channel access challenges in LoRa low-power wide-area networks, LPWAN Technologies for IoT and M2M Applications, Academic Press, Pages 65-102, 2020.
[28] K. Govindan and A. P. Azad, "End-to-end service assurance in IoT MQTT-SN," 12th Annual IEEE Consumer Communications and Networking Conference (CCNC), Las Vegas, NV, USA, pp. 290-296, 2015.

[29] Martin-Escalona, Israel, and Enrica Zola, ‘‘Passive Round-Trip-Time Positioning in Dense IEEE 802.11 Networks’’ MDPI Journal Electronics, vol. 9, no. 8, 2020.

[30] V. Jacobson, Congestion avoidance and control, ACM SIGCOMM conference on applications, technologies, architectures, and protocols for computer communication 188 (4) (1988) 314–329.

[31] H.-L. Chang, C.-G. Wang, M.-T. Wu, M.-H. Tsai, C.Y. Lin, Gateway-Assisted Retransmission for Lightweight and Reliable IoT Communications, Sensors 16 (10) (2016) 1560.

[32] S.D. Strowes, B. Inc, Passively Measuring TCP Round-trip Times-A close look at RTT measurements with TCP, ACM queue 11 (8) (2013) 478–491.

[33] Günther, Andre & Hoene, Christian, Measuring Round Trip Times to Determine the Distance Between WLAN Nodes. Lecture Notes in Computer Science., no. 3462. pp. 768-779, 2005.

[34] Josep Diaz, Alberto Marchetti-Spaccamela, Dieter Mitsche, Paolo Santi, Julinda Stefa. Social-Aware Forwarding Improves Routing Performance in Pocket Switched Networks, European Symposium on Algorithms, Saarbrücken, Germany, 2011,

[35] Hayatunnufus, M. Riasetiawan and A. Ashari, ‘‘Performance Analysis of FIFO and Round Robin Scheduling Process Algorithm in IoT Operating System for Collecting Landslide Data,’’ International Conference on Data Science, Artificial Intelligence, and Business Analytics (DATABIA), Medan, Indonesia, pp. 63-68, 2020.

[36] C. Raskar, S. Nema, IoT Based Real Time Traffic Monitoring System Using M/G/1 Queuing, International Journal of Engineering and Advanced Technology (IJEAT) 8 (6) (2019) pp.

[37] I. Ullah, Muhammad Sohail Khan, DoHyeun Kim, ‘‘IoT Services and Virtual Objects Management in Hyperconnected Things Network’’, Mobile Information Systems (2018).

[38] R. Duan, X. Chen and T. Xing, ‘‘A QoS Architecture for IOT’’, International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing, pp. 717-720, 2011.

[39] B. Mishra, B. Mishra, A. Kertesz, Stress-Testing MQTT Brokers: A Comparative Analysis of Performance Measurements, Energies no. 14 (2021).

[40] D. R. C. Silva, G. M. B. Oliveira, I. Silva, P. Ferrari and E. Sisinni, ‘‘Latency evaluation for MQTT and WebSocket Protocols: An Industry 4.0 perspective,’’ IEEE Symposium on Computers and Communications (ISCC), Natal, Brazil, pp. 1233-1238, 2018.

[41] N. Jiang, Y. Deng, X. Kang, A. Nallanathan, Random Access Analysis for Massive IoT Networks Under a New Spatio-Temporal Model: A Stochastic Geometry Approach, IEEE Transactions on Communications 66 (11) (2018) 5788–5803.

[42] A. Zanella, N. Bui, A. Castellani, L. Vangelista, M. Zorzi, Internet of things for smart cities, IEEE Internet of Things J. 1 (1) (2014) 22–32.

[43] J.K. Parmar, A. Desai, IoT: Networking technologies and research challenges, International Journal of Computer Applications 154 (7) (2016) 1–6.

[44] N.N. Srinidhi, S.M. Dilip Kumar, K.R. Venugopal, Network optimizations in the Internet of Things: A review, Journal of Engieering Science and Tech. 22 (1) (2019) 1–21.

[45] I. Zyrianoff, F. Borelli, G. Biondi, A. Heideker and C. Kamienski, ‘‘Scalability of Real-Time IoT-based Applications for Smart Cities,’’ IEEE Symposium on Computers and Communications (ISCC), Natal, Brazil, pp. 688-693, 2018.

[46] MQTT Server Bechmarking, Scalagent Tech. , Accessed on: Jan 1, 2023, [Online]. Available: http://www.scalagent.com/IMG/pdf/Benchmark_MQTT_servers-v1-1.pdf.

[47] E. Jorge, Luzuriaga, Miguel Perez, Pablo Boronat, Juan Carlos Cano, Carlos Calafate, Pietro Manzoni, ‘‘Improving MQTT Data Delivery in Mobile Scenarios: Results from a Realistic Testbed’’, Mobile Information Systems 2016 (2016) 1–11.

[48] B. Bendele, D. Akopian, A study of IoT MQTT control packet behavior and its effect on communication delays, Electronic Imaging 10 (2017) 120–129.

[49] Beginners guide to the MQTT protocol, 2021, Accessed on: Jan 1, 2023, [Online]. Available: http://www.steves-internet-guide.com/mqtt/.

[50] H. Hasan, S. Aqeel, IoT Protocols for Health Care Systems: A Comparative Study, International Journal of Computer Science and Mobile, Computing 7 (11) (2018) 38–45.

[51] Z. Zeng, H. Che, W. Miao, et al, Towards secure and network state aware bitrate adaptation at IoT edge, Journal of Cloud Comp 9 (38) (2020) pp.

[52] Ieee, Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal aea networks WPANs‘‘, Standard draft IEEE 802.15.4 (2006).