


Article

Mathematical Modeling and Validation of Retransmission-Based Mutant MQTT for Improving Quality of Service in Developing Smart Cities

Jawad Ali ¹, Mohammad Haseeb Zafar ^{2,*}, Chaminda Hewage ², Raheel Hassan ³ and Rameez Asif ³¹ Department of Electrical Engineering, University of Engineering and Technology, Peshawar 25000, Pakistan² Cybersecurity and Information Networks Centre (CINC), Cardiff School of Technologies, Cardiff Metropolitan University, Cardiff CF5 2YB, UK³ School of Computing Sciences, University of East Anglia, Norwich NR4 7TJ, UK

* Correspondence: mhzafar@cardiffmet.ac.uk

Abstract: Unreliable networks often use excess bandwidth for data integration in smart cities. For this purpose, Messaging Queuing Telemetry Transport (MQTT) with a certain quality of service (QoS) is employed. Data integrity and data security are frequently compromised for reducing bandwidth usage while designing integrated applications. Thus, for a reliable and secure integrated Internet of Everything (IoE) service, a range of network parameters are conditioned to achieve the required quality of a deliverable service. In this work, a QoS-0-based MQTT is developed in such a manner that the transparent MQTT protocol uses Transmission Control Protocol (TCP)-based connectivity with various rules for the retransmission of contents if the requests are not entertained for a fixed duration. The work explores the ways to improve the overall content delivery probability. The parameters are examined over a transparent gateway-based TCP network after developing a mathematical model for the proposed retransmission-based mutant QoS-0. The probability model is then verified by an actual physical network where the repeated content delivery is explored at VM-based MQTT, local network-based broker and a remote server. The results show that the repeated transmission of contents from the sender improves the content delivery probability over the unreliable MQTT-based Internet of Things (IoT) for developing smart cities' applications.

Keywords: QoS; MQTT; IoT; end-to-end service assurance; smart cities



Citation: Ali, J.; Zafar, M.H.; Hewage, C.; Hassan, R.; Asif, R. Mathematical Modeling and Validation of Retransmission-Based Mutant MQTT for Improving Quality of Service in Developing Smart Cities. *Sensors* **2022**, *22*, 9751. <https://doi.org/10.3390/s22249751>

Academic Editor: Antonio Guerrieri

Received: 25 October 2022

Accepted: 5 December 2022

Published: 12 December 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Physically separated and logically connected applications on distributed nodes are mostly used while developing smart cities, employing the IoE [1]. Such an application may include but is not limited to inventory and stock management, health and security, the entertainment industry, information segregation, information exchange, information integrity, autonomous functionality, transportation and traffic system, energy and smart grid, industrial manufacturing and process engineering, supply chain management, etc. The mode of communication can be Device to Device (D2D) and Device to Server (D2S). D2D is based on the mechanism where devices are directly exchanging information for a certain process or service. On the other hand, in D2S or Server to Device (S2D), the distributed nodes are communicated with a server and the information exchange is on-demand. Integrated services often use a backbone server for an information exchange for various reasons.

For acquiring the optimum functionality of smart cities, a server must possess the required information or instructions that are timely needed by the connected devices [2]. Thus, for the stated purpose, abridged devices use various protocols, varying from application to application. The majority of these protocols serve as the foundation of the Internet of Things (IoT) and Internet of Everything (IoE). The Constrained Application

Protocol (CoAP) is used for Machine to Machine (M2M) communication in smart energy and building automation. The Messaging Queuing and Telemetry Protocol (MQTT) [1] is used in manufacturing industries due to its open-source moldable nature, petroleum and natural gas, health, weather, home automation, smart farming, smart metering, remote, etc. Web Socket Protocol has a variety of uses such as in social media feeds, multiplayer games, collaborative editing/coding, clickstream data, financial tickets, media chats, location-based apps, online education, etc. [3]. The Data Distribution Service (DDS) is used on the top of edge devices/sensors for supervision, integration, and process control. Similarly, the Extensible Messaging and Presenting Protocol (XMPP) is widely employed in content syndication, collaboration tools, geolocation, file sharing, gaming, remote systems control, cloud computing, etc. These application protocols are the extensively used available protocols in the development of smart cities [3].

While talking about MQTT-based connectivity, it is worth noting that MQTT is used on the top of IEEE 802.15.4 and TCP/IP [4]. MQTT requires less power compared to other employed protocols for a range of applications where the power consumption accounts for the network performance. MQTT uses a traditional subscription and publishing-based mechanism, hence the bandwidth efficiency is managed up to a certain level.

In this work, the mathematical models for estimating the end-to-end delay and probability of the content delivery for retransmission-based mutant MQTT with QoS-0 are developed and then examined for their validation by analyzing the reliability of end-to-end MQTT. The attempts of sending information on a fixed subscription are varied per simulation and the results are then checked on a physical network. The emulated network nodes are comprised of fundamental sensing and connectivity features. The end nodes consist of Windows clients on PCs and as a Java application on handheld devices. The requests that are generated in a simulation environment on Linux are thus checked by a physical device for the performance consistency and validity of the mathematical model. The MQTT server is handled on a Virtual Machine, running at the eclipse.org cloud. The end nodes are connected to the Internet via a wired network, whereas the switches use optical fiber connectivity with the Internet. The DNS server is used in this case as well.

1.1. Architecture

MQTT is a subscription and publishing service-based communication with a centralized server [1]. The MQTT-based network is comprised of three basic logical formations, as shown in Figure 1. The transparent gateway has one node connection that is bridged with a remote MQTT server. The aggregating gateway connects local network nodes using a singular gateway to a remote MQTT server. Hybrid gateways route packets from multiple nodes and use mesh topology for connecting with the remote MQTT server [2]. The above-stated gateway formations are opted for various IoT scenarios. For example, a low-density network with large bandwidth requirements may employ the transparent gateway. This way, a single actuator–single sensor-based application may be run with a speedy response [1].

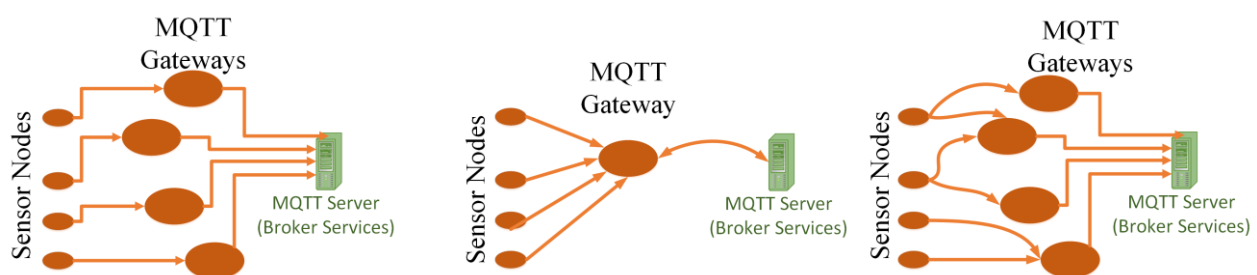


Figure 1. Logical Formation of MQTT gateways in IoT for various applications (a) Transparent Gateway MQTT, (b) Aggregated Gateway MQTT and (c) Hybrid Gateway MQTT.

Each stated formation has its own facilities, uses and drawbacks. Aggregated gateways are not employed for the applications of smart cities when the nodes are geographically

dispersed. It is because several nodes might be acting as actuators and controlling the nodes from physically separated locations, and this means that different MQTT gateways are used that makes the IoT network's formation transparent (see. Figure 1a). Aggregated gateways have the limitations of poor congestion control and packet loss as well as a single point failure issue [1], directly degrading the performance of the applications of smart cities. On the other hand, a hybrid gateway, as in Figure 1c, uses multiple paths for communication with the server that will lack the concept of modularity for communication with a remote server on the Internet in our case. A Virtual Machine, while generating data traffic for the experiment, can only provide transparent gateway-based communication. Thus, the work uses the IoT network formation based on the transparent gateway-based client to server connectivity. The rest of the rules and assumptions are listed below.

- The MQTT subscribe/publish-based communication requests to the server are collectively treated as request-acknowledge messages.
- The end node uses DNS for address resolution in case of connectivity with the remote server that poses a constant delay.
- The SUBSCRIBER nodes as well as the PUBLISHER nodes follow an identical mechanism for communication with the server, thus the request flows through the same pathway.
- The TCP-based communication "fire and forget after n attempts" strategy is considered for all the network nodes that follows retransmission-based QoS-0.
- The remote server holds two sets of request queues. The published contents' queue holds the data for the T_{cnt} interval and the subscribers' requests are held for an interval denoted by T_{req} . These holding intervals T_{hold} are thus defined by the storage capacity of the MQTT server as well as the connected nodes [2].
- QoS-1, that is based on "at least one time delivery", and QoS-2, based on "exactly one time delivery", are covered in [3].

1.2. Problem Statement and System Model

The services that are provided in smart cities are categorized by latency in the information exchange, bandwidth and packet loss. For example, the data communication networks that are used in a hospital environment must be in accordance with the severity of the patient's condition [4,5]. Moreover, it may have several levels and priorities of information exchange w.r.t parameters that are being exchanged. For example, a single patient may require multiple services, each having different sets of the data transmission rate and data update rate. Similarly, the end-to-end delay for services at a normal condition may be compromised. For example, a remote weather station may exchange weather-related information with minimum assurance for its delivery in normal weather [1]. The reliability parameters need an upgraded communication scheme when the situation is of an elevated priority. For example, adverse weather condition may need an immediate data sharing service for smart decision making [6].

The research aims to assure an adequate reliability in terms of the content delivery and guaranteed end-to-end delay in the development of smart cities' applications. Such applications use data from multiple sources/sensors on a single topic or on a range of topics for proper functioning that varies from application to application. The PUBLISH requests from sensing nodes are considered unique while receiving data, irrespective of their physical or logical situation [6]. Thus, the events are considered to have a mutually exclusive w.r.t delivery time or bandwidth requirements. Likewise, the logging of data on the centralized server always put a constrain on the residing time of the data as well as the queue size. We may term this as the packet expiry time or packet replacement time on the server. A single topic data may be required by different subscribers for a range of uses. For example, data from traffic junctions may be used for the re-routing of vehicles or for controlling traffic lights.

QoS-0, QoS-1 and QoS-2 are shown in Figure 2. In Figure 2, PUBLISH means post data to the MQTT Broker, and PUBREC means Publish has been Received, confirming the PUBLISH operation to a server in QoS-2 [3]. After receiving PUBREC, the sender

sends a “Publish Release” request as PUBREL to tell the server that it has discarded the sent data from a local stack. The server also stored the packet identifier at this stage to avoid the processing of duplicate packets. The server responds with a confirmation packet of a Publish Complete (PUBCOMP) message after PUBREL. This way, the sender can repeat the process with new data. According to these QoS levels, either the packet is lost, or the packet is confirmed by an acknowledgement from the MQTT Broker server. The work proposes a one step ahead solution to improve the QoS of the MQTT-based smart cities’ connectivity applications. The MQTT protocol-based nodes are configured for a transparent gateway-based application. The nodes are running the simple process of publishing and subscribing services that are identical in nature. The work considers the number of connected nodes, the PUBLISH and SUBSCRIBE request rate, the time of content holding by the server and gateway for the retransmission-based QoS-0 MQTT IoT, for a system performance improvement in terms of the reliability and end-to-end delay. Here, a mathematical formulation is provided for achieving a guaranteed QoS by a varying number of retransmission attempts.

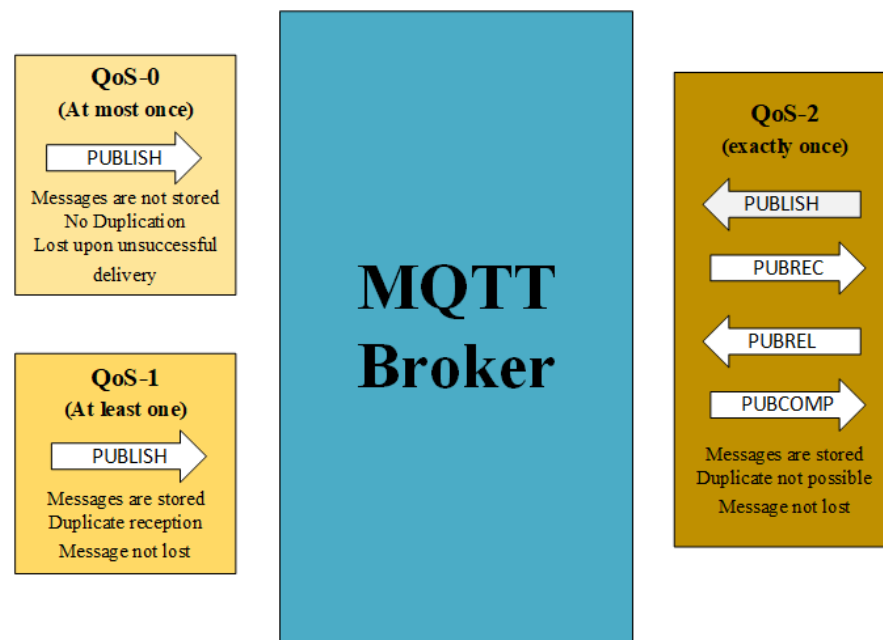


Figure 2. QoS-0-, QoS-1- and QoS-2-based MQTT broker with message delivery, duplication and storage status on the publishing node.

The rest of the work is organized as follows: Section 2 explains the basic literature and characteristics of MQTT. Likewise, the proposed retransmission-based MQTT is explained in the section. Section 3 is the mathematical modeling of the end-to-end delay and expressions for evaluating the probability of the content delivery. The experimental setup and system model is given in Section 4. Section 5 provides a complete insight into the obtained simulation results and its explanation w.r.t scalability of the network. Section 6 concludes the work and gives future perspectives on this research.

2. MQTT Characteristics

MQTT is light weight and content-oriented protocol. Locally, MQTT can be used for the IoE and IoT, i.e., amongst IP-less as well as IP-based network nodes. The explored case uses the wired network of IP-based nodes that are connected to a local gateway on a single node–single gateway scheme basis. Mosquito clients are deployed on local nodes in an emulated environment for real world connectivity by providing the NAT to LAN bridge with the Internet. This provided real world environment, sufficient to receive effective and authentic readings of the parameters that are under observation.

2.1. Retransmission-Based QoS-0 MQTT Mutant Protocol

By the mechanism of retransmission in the QoS-0, the instantaneous delivery probability, that is explored for the static network characteristics in [5], is affected. The requestee node retransmits the TCP/IP-based request if the packet is either lost between the node and gateway or on the Internet. Additionally, the gateway data buffer as well as the request queue at the server are kept of a minimum size. It is for the fact that the MQTT gateways usually have a limited cache for storing data. The queue is only handled at the server, so the gateway limitations are traded off by retransmit protocol. This process makes the midway delays static for usual processes and define (make constant) it instantaneously.

2.2. Proposed Asynchronous MQTT

We have assumed that one node is posting for a single topic and multiple topic-based posting is assumed to be separate nodes and separate events. The rest of the communication scheme between the publisher and subscriber node is explained in the following steps, as in Figure 3.

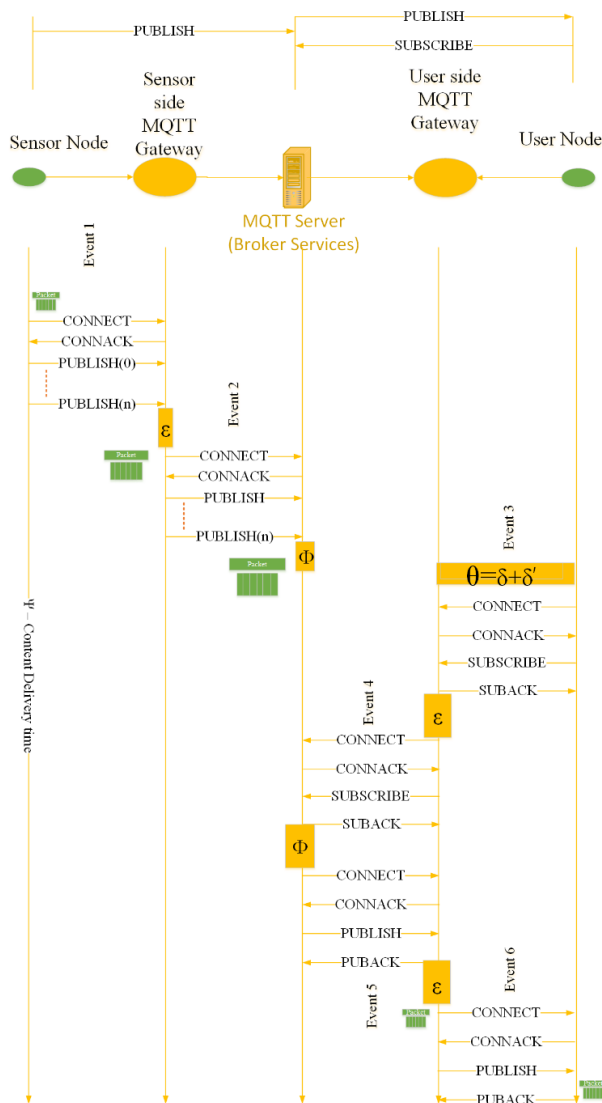


Figure 3. End-to-end delay and packet travel system model for QoS-0 retransmit Mutant MQTT.

Step 1: The end node connects with the gateway using the CONNECT/CONNACK mechanism and then we publish the contents using the publish-retransmit mechanism.

Step 2: The received data from the end node is transmitted by the gateway to the remote MQTT server as CONNECT/CONNACK and then a publish–retransmit basis over a reliable TCP/IP wired network.

Step 3: The end user connects to the gateway using the CONNECT/CONNACK mechanism as in Step 1 and then the same end user registers with the gateway using SUBSCRIBE and SUBACK for an already available topic on the server.

Step 4: The subscriber side gateway registers the end user with the server using the connect request followed by the subscribe request that is acknowledged by the server as the SUBACK.

Step 5: The server sends the available date of the subscribed topic to the end user side gateway via the Internet using PUBLISH and PUBACK instructions.

Step 6: The gateway on the end user’s side fetches the data to the end user using PUBLISH and PUBACK.

The above-stated steps pose a certain probability of failure for a send and forget mechanism. This send–retransmit and forget mechanism is used for the asynchronous IoT system in Figure 3. The availability of the data on the server is arguable.

3. Delay Estimation

The probabilistic model uses an asynchronous mode of data transfer for the collection and distribution of data. Contents are created at sensors using a constant request rate that are sent to the intermediate gateway with a send–retransmission and forget manner. The same method is followed for fetching the sensors’ data to a remote MQTT server from the intermediate gateway. There are two cases for which an end-to-end delay may be estimated while considering the availability of the contents on the server for the requested topic from the end user.

The probability of one successful delivery of a data packet from the sender to receiver is comprised of the exclusive probabilities of two event probability, i.e., the sender to server probability and server to subscriber probability. Now that the subscriber does a handshake in TCP/IP-based data fetching, the number of total attempts are denoted by ‘ i ’ for relating towards the packet probability p_f , and the number of acknowledgements from the communicating node is ‘ j ’ for relating the reverse packet probability p_r , whereas the number of maximum iterations from the sender to server are considered to be ‘ n ’ for the sensor to the gateway and ‘ m ’ for the gateway to the server.

We have considered that attempts m and n are equal. The success probability is given by (1) as:

$$P(i,j) = p_f^j \cdot (1 - p_f)^i \cdot (1 - p_r)^i \quad (1)$$

3.1. Contents Available on Topic

The iteration-based packet sending probability for n events of retransmission is given by (2):

$$P(n) = \sum P_f(i \leq n) \quad (2)$$

A single transmission/retransmission is completed after the delay time td . This ‘ td ’ must be greater than the product of the request rate ‘ σ_k ’, where k is a unique node and the time taken to completely send one request T_r . This shows that the total requests after the retransmission will take time, as in (3).

$$Tr(n) = n\sigma_k t_r \quad (3)$$

The point-to-point latency also includes the synchronization time T_s , that is a single packet traversing time between the two nodes and the server on the end user’s side using TCP. The latency of a successful packet delivery is given by (4)–(6).

$$Lt(i, j) = Lt(n) = \text{unsuccessfulattempts} + \text{successfulattempts} \quad (4)$$

$$L_t(n) = (n)\sigma_k \cdot t_r \quad (5)$$

$$L_t(i, j) = RTT + \left(\sum_{k=1}^{i-1} 2^k T_s \right) + \left(\sum_{l=1}^{j-1} 2^l T_s \right) \quad (6)$$

The overall probability of latency $P[L_t(n, i, j) \leq t]$ on a low data rate channel for the duration t is:

$$P[L_t(n, i, j) \leq t] = \sum_{L_t(i, j)} P(i, j) + \sum_{L_t(n)} P(n) \quad (7)$$

For the fact that two additional TCP connections are established, we assume a ε queueing delay at the gateways. The gateway may hold sensing data to be forwarded to the MQTT server or it may enqueue the data while sending it to the end node. The delay posed by the MQTT server is denoted by ϕ . If the data are already available on the server, two cases may arise. A sensing node will either send a publish request or an end node will make a request for the contents. The connection request delay for a subscription service is denoted by δ .

The content delivery time Ψ for a successful route traversal from the sensing node n via the sensing side gateway g , through the server s and then from the server to the end user u through another gateway g , is given by (8)

$$\Psi = T_{connect-ng} + T_{pub-ng} + \varepsilon + T_{connect-gs} + T_{pub-gs} + \phi + \delta + T_{connect-ug} + T_{sub-ug} + \varepsilon + T_{connect-gs} + T_{sub-gs} + \phi + T_{connect-sg} + T_{pub-sg} + \varepsilon + T_{connect-gu} + T_{pub-gu} \quad (8)$$

By examining (7) w.r.t Figure 3, we may replace the $T_{connect-p2p}$ by $RTT_{connect}$, T_{pub-ng} and T_{pub-gs} with $2(n)\sigma_k t_r$, T_{pub-sg} and T_{pub-gu} with RTT_{pub} and $T_{sub-p2p}$ with RTT_{sub} . Thus, Equation (8) is express as:

$$\Psi = 6RTT_{connect} + 2(n)\sigma_k t_r + 2RTT_{pub} + 2RTT_{sub} + 3\varepsilon + 2 + \delta \quad (9)$$

Since PUBLISH requests are always of the same length. Moreover, the connection requests are identical. So, by keeping a fixed packet size for all the TCP requests, we may combine $RTT_{connect}$, RTT_{pub} and RTT_{sub} and thus receive (10) for a collectively called RTT , the TCP Round Trip Time for each event:

$$\Psi = 10RTT + 2(n)\sigma_k t_r + 3\varepsilon + 2 + \delta \quad (10)$$

3.2. Contents Not Available on Topic

The constant δ in (10) represents the queueing delay experience by the subscription request faced at the server before the content is sent to the end user. If the contents on a requested topic are not available, the subscription request may be answered but the user will have to wait for a δ' duration before the request is answered by the server. We may collectively call this queueing delay θ , and it is calculated by adding δ' with δ , as in (11).

$$\Psi = 10RTT + 2(n)\sigma_k t_r + 3\varepsilon + 2 + \theta \quad (11)$$

3.3. Expressions for θ , ε , RTT

Θ may be given by a uniform distribution between the two events, i.e., the arrival instant of content \check{T}_{cnt} and the arrival time of request \check{T}_{req} , at the server. This interval is given by (12)

$$\theta = [\check{T}_{cnt}, \check{T}_{req}] \quad (12)$$

We will use the relation from [6–8], meaning the random delay from the gateway to the gateway to the server is given by (13):

$$\mu_\theta = (t_{cnt} + 2T_{cnt} - \delta)/2] \quad (13)$$

Also, for the contents already available on the server:

$$\theta = \delta = 3 RTT + \varepsilon \quad (14)$$

Also, the Smooth Round Trip Time is given by (15), thus the RTT is calculated as (16). In (15) and (16), $0 < \beta < 1$ and $0 < \Delta < 1$ and uses $\beta = 0.125$ for a smooth RTT , as suggested by [6–8].

$$SRTT[i] = (1 - \beta)x SRTT[i - 1] + \beta x RTT \quad (15)$$

Additionally,

$$RTT = RTT + \Delta x Diff, \dots, Diff = SRTT - RTT \quad (16)$$

3.4. Service Rate, Capacity of the Network, and Probability of Content Delivery

The queueing delays can be calculated from the virtual service rate S_{rate} . For connected users N_c , the service rate's threshold is given by relation in (17):

$$\sum_{i=1}^{N_c} \lambda_i \leq S_{rate} \quad (17)$$

(17) gives us the average processing time for the contents in terms of the arrival rate λ_i . For the packet length L , the service duration s_i is calculated in (18).

$$s_i = L/\mu_i \quad (18)$$

here,

$$\mu_i = \frac{\lambda_i}{\sum_{i=1}^{N_c} \lambda_i} S_{rate} \quad (19)$$

The request arrival rate is given by (20):

$$\mu_i^{req} = \frac{\sigma_i}{\sum_i^{N_r} \sigma_i} S_{rate} \quad (20)$$

The maximum queueing delay by both the content and request is given by $Q[\lambda_i, \sigma_i]$ in (21), as:

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

Here, $\rho_i^{req} = \sigma_i/\mu_i$ and $\rho_i^{cnt} = \lambda_i/\mu_i$.

The probability of the content delivery for the M/D/1 fixed scheduler-based queue model, having $D = 1/\mu$, is given by (22), as per [9,10].

$$P_{cnt}^D = \frac{\sigma_i}{2} \left(\left[1 - e^{-\zeta_i/T_{cnt}} \right] + \left[1 - e^{-\zeta_i/T_{req}} \right] \right) \quad (22)$$

4. Experimental Setup

The test bed is comprised of an emulated environment for remote nodes that are connected to the MQTT broker, as in [11]. Linux-based nodes are used for creating topic-based data at various data rates. Mosquitto MQTT clients are deployed on Virtual Machines for this purpose. The VMs consist of a range of clients and subscribers to approximate the scenario for the practical results and a realistic environment. The Mosquitto MQTT broker is used on a local server and an Eclipse IoT server. The clients from windows, android and VMs are subscribed to various topics on this server. The setup of the nodes is given in Figure 4.

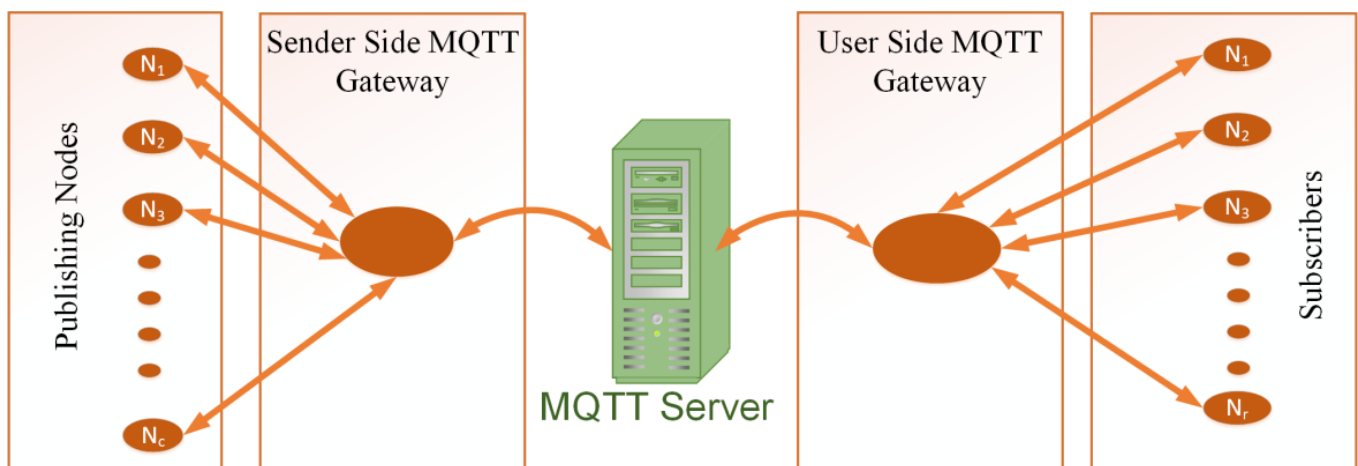


Figure 4. System Model and connectivity.

The paper assumes that Body Area Network (BAN) nodes are connected nodes. The data of various patients are collected and then the subscribers' requests for the collected data, as in [12]. This work is not limited to a single patient or specific parameter because MQTT supports topic-wise data and different patients can be dealt as different topics on the server. For the purpose of the analysis, we have used a temperature sensor, heartbeat sensor, moisture sensor and fall detection sensor data as the primary source.

The arrival rate of contents λ by the sensing nodes and requests rate for the contents σ by the subscribers are treated as the same. Both rates are varied between 100 kbps and 2 Mbps, with no retransmission involved. The client node traffic uses a Constant Bit rate (CBR), which is varied using the Poisson distribution mean [13]. The *RTT* for TCP is obtained by the terminal as the ping6 command, whereas for windows, the power shell is used. These values are saved to a comma separated value (csv) file for a further analysis. The gateway queueing delay is kept at 720 μ s, as in [14]. The payload is set indirectly by CBR [15]. The local server keeps the request for the contents and the contents on the server for 10 ms as a fair queueing policy [14]. The M/D/1 queue is used at the local server. Ref. [11] has used an infinite queue size but due to hardware-based servers, we are limiting the dedicated local server capacity to 5 GB and the Mosquitto server on Eclipse.org uses an 80 GB shared capacity-based service.

5. Results and Discussion

Figure 5 is a graphical depiction of the conclusion drawn by (10). The request rate or content arrival rates are controlled by the connected nodes in the IoT. Starting from 100 kbps, these rates are incremented with a step of 0.1 Mbps up to 2 Mbps. Moreover, the request arrival rate σ and content arrival rates λ are kept the same for the fact that the behavior of both the content provider and subscriber nodes is the same. The local server gives an end-to-end delay of 1.6 ms for an arrival rate of 100 kbps. The behavior remains linear, and the end-to-end delay increases to 2 ms for an arrival rate of 2 Mbps. The results are satisfactory as compared to the work in [11]. The remote server poses a 7.7 ms delay for 100 kbps traffic, whereas the delay slightly increases to 8.5 ms for a 2 Mbps request rate. Nonlinearity is seen between the 0.4 and 0.5 Mbps and the 1.0 and 1.4 Mbps request arrival rate in Figure 5. It is for that fact that VMs share their network and processing power for other processes. Similarly, the remote server is accessed through the Internet over a shared broadband network that causes the unavoidable nonlinearity in the graphs.

The end-to-end delay w.r.t number of topic subscriptions or the clients' requests is also explored for the real time traffic that is graphed in Figure 6. The delay in the delivery is linearly increasing while varying the number of client requests between 1 and 2800. The arrival rate is kept at 1 Mbps. The end-to-end delay increments from 2.2 ms to 3.2 ms, likewise for a locally deployed server. The delay steps up to 7.4 ms when remotely accessing

the server, for which the maximum delay of 8.65 ms is experienced when 2800 requests are handled by the remote server. This also concludes that one request is handled at 500 μ s on average on the server and a total service time (Δt_d) of 1.4 ms is utilized for all the requests. The nonlinearities that are caused by shared processes on the VM and broadband network can be seen between the number of nodes ranging from 1000 to 1200.

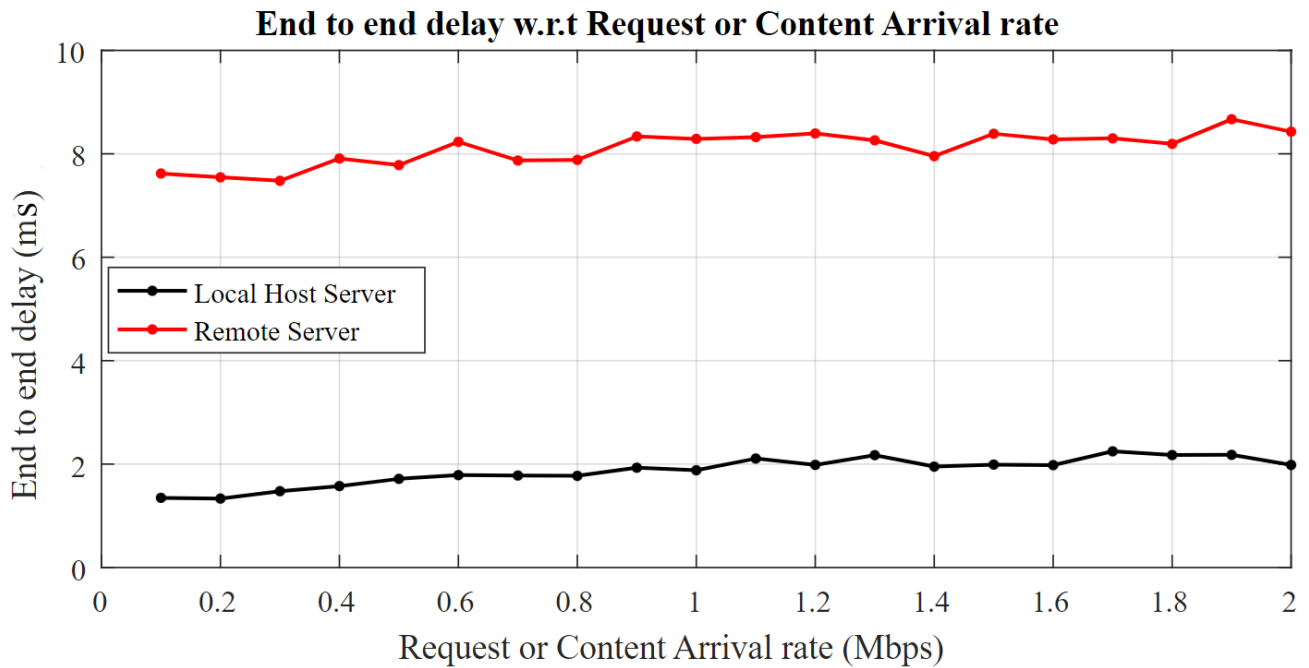


Figure 5. End-to-end delay for request and content arrival.

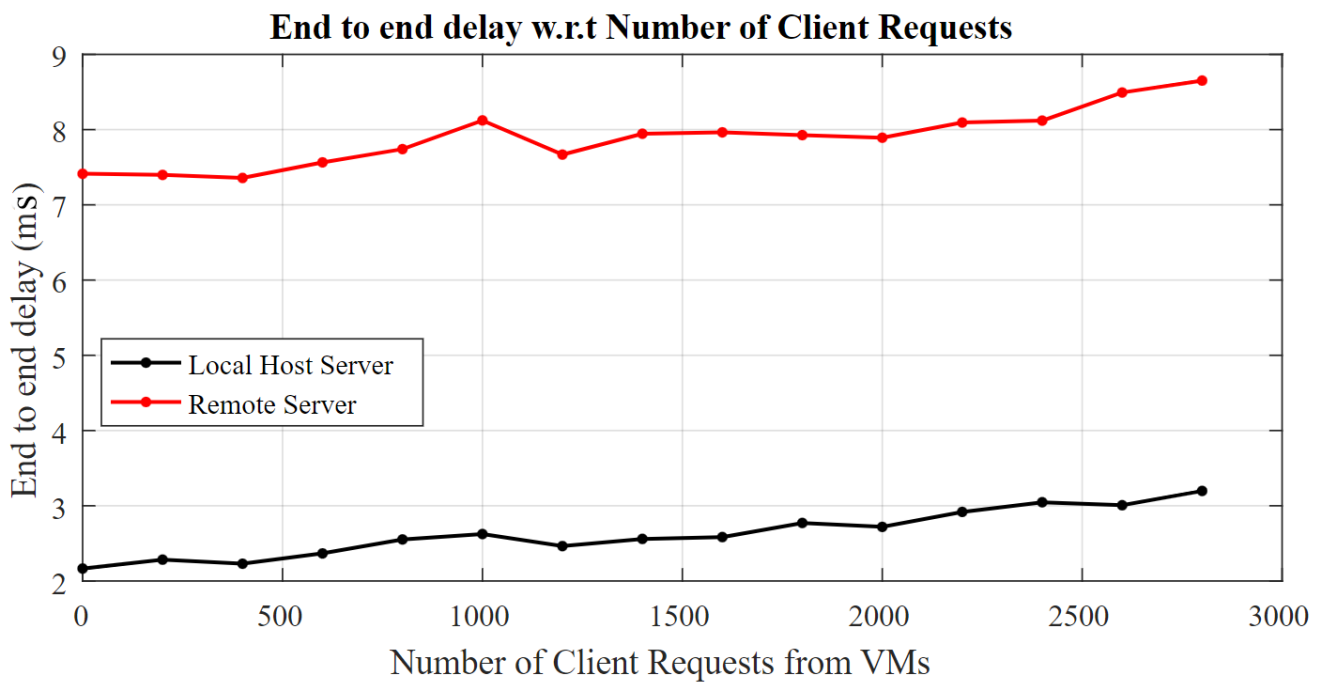


Figure 6. End-to-end delay w.r.t number of client request from connected IoT nodes.

Figure 7 explores the content holding time and gives a notion about the end-to-end delay w.r.t T_{cnt} , and the delay increases drastically with the request hold time on the server. An average of a 300 μ s delay is introduced by increasing the hold time by 50 ms. The slope of the line increases with an increase in the round-trip path through the network.

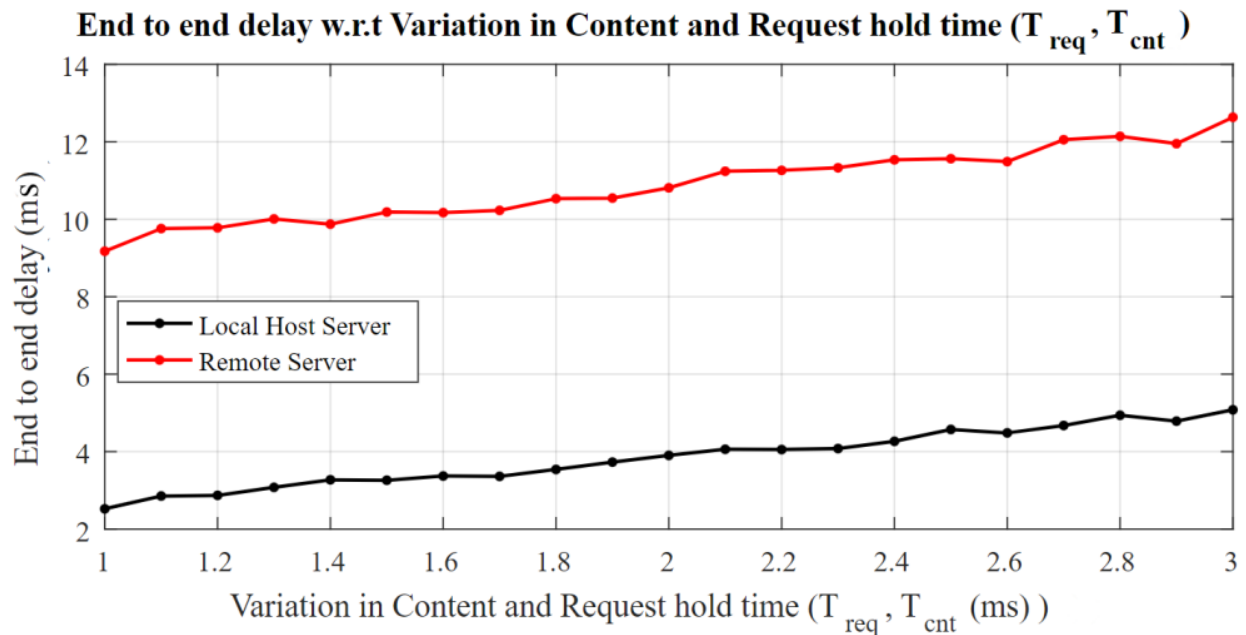


Figure 7. End-to-end delay w.r.t variation in content request time and content holding time.

The analysis probability of the content delivery P_D^{cnt} is analyzed; w.r.t content re-transmissions n and σ is given by Figures 8 and 9. Both figures can be directly used as a benchmark for the scalability of smart cities' services. The request duration is incremented from $1 \mu s$ to $20 \mu s$ with a $4\text{--}5 \mu s$ step, likewise. It is worth noting that P_D^{cnt} linearly improves by increasing the n or T_{req} on the server. Additionally, the maximum achievable P_D^{cnt} can be directly calculated for the various set of retransmission attempts n . While keeping T_{cnt} constant, increasing the arrival rate of the contents also improves the delivery probability, as depicted by Figure 9. The results show an improvement in the achieved maximum probability of the delivery for a 2 Mbps arrival rate in [11] by 4%. The results show that the behavior of the developed scheme exponentially converges the performance of the scalable IoT.

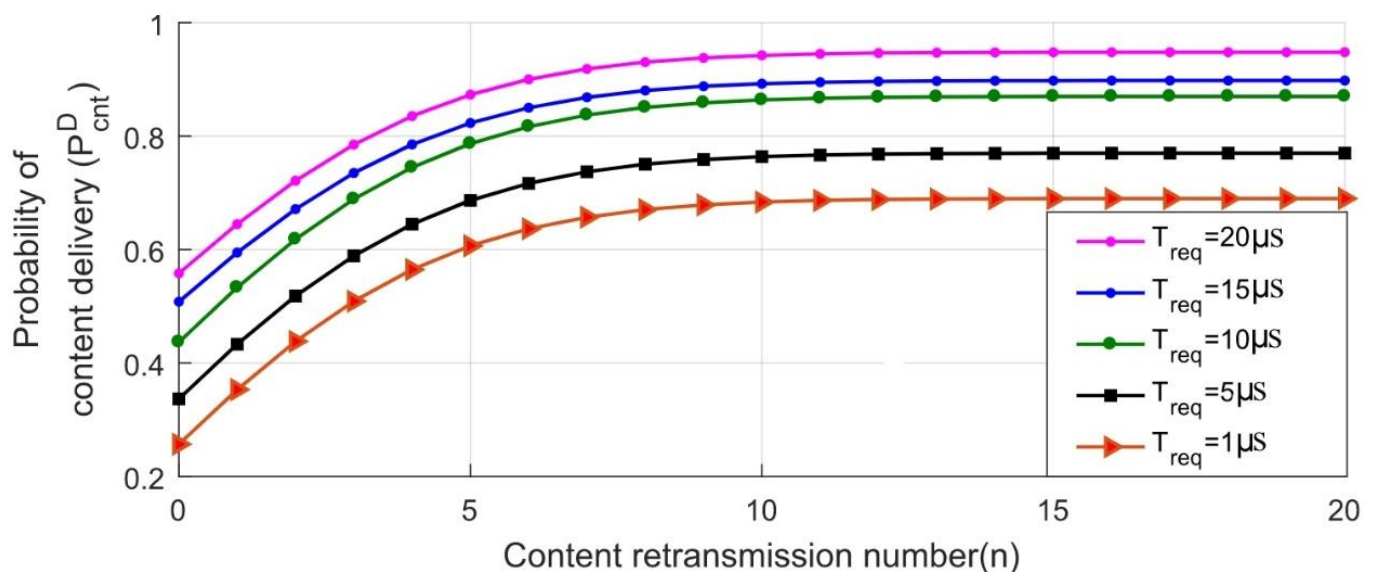


Figure 8. Probability of content delivery on localhost MQTT server over Virtual Machine for content interval (T_{cnt}).

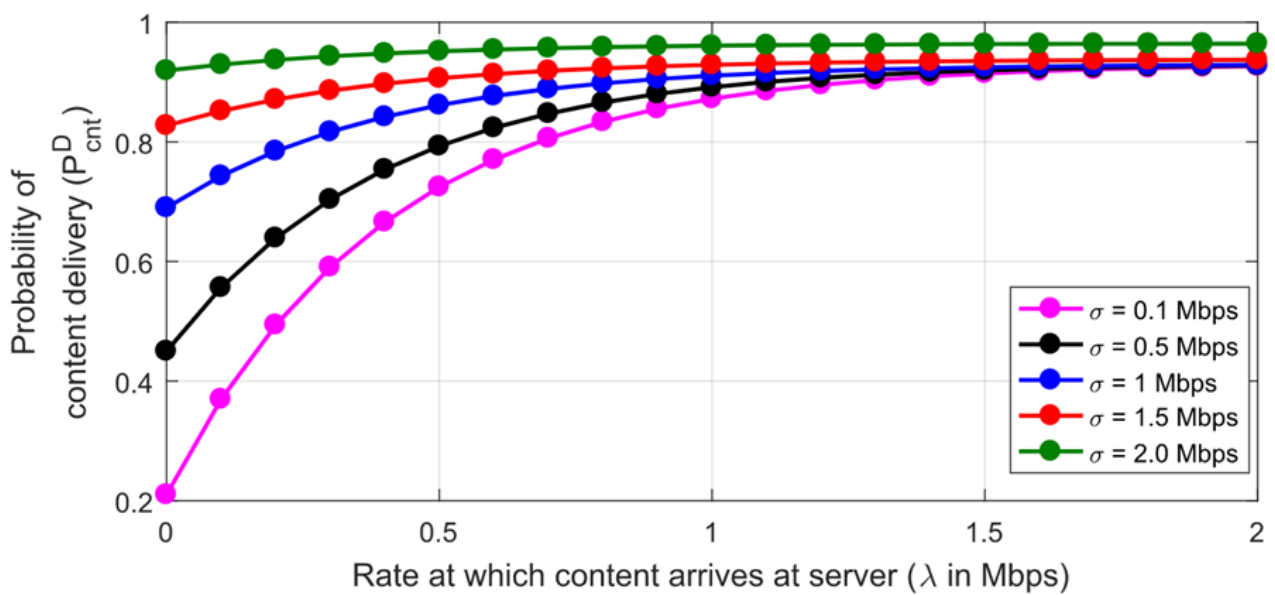


Figure 9. Effect of content arrival rate on probability of content delivery for $n = 1$.

Figure 10 is a broad picture of the developed scheme where the number of connected nodes is varied between 1000 and 11,000. The figure is a depiction of the real time content delivery and service degradation matrix. The remote services are degraded by 13%, whereas a 4% degradation is noted for the locally hosted MQTT server.

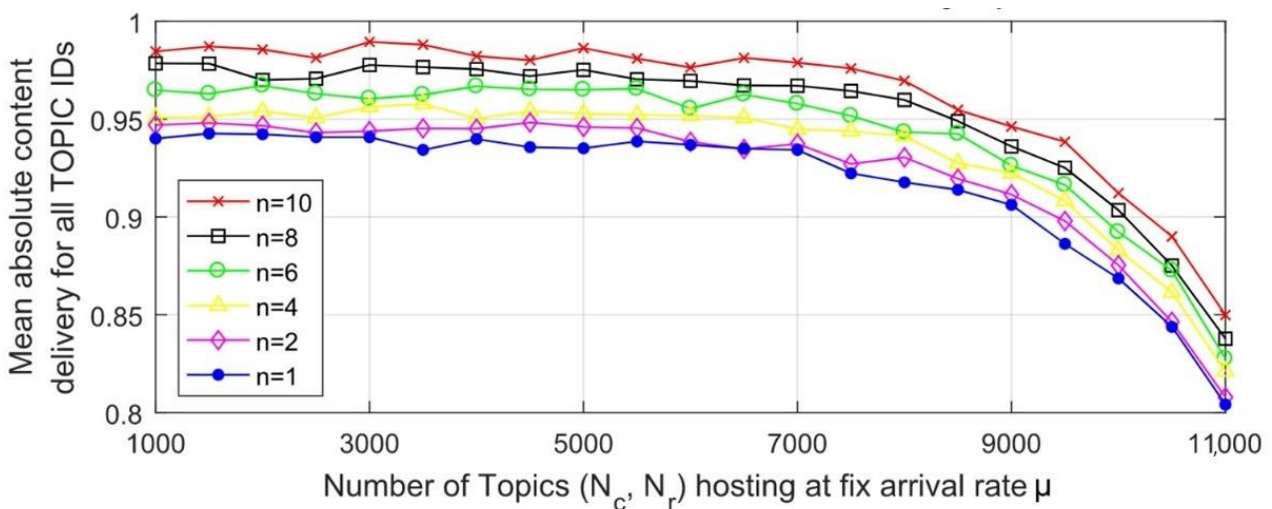


Figure 10. Mean absolute content delivery for collective topic IDs w.r.t connected hosts at fixed arrival rate for $n = 1, 2, 4, 6, 8, 10$.

The portrayed image of the mean absolute content delivery for all the topic IDs is thus conformation that the degradation is practically tolerable for the massively deployed–remotely accessed server with a retransmission-based mutant MQTT scheme.

6. Conclusions and Future Perspectives

The paper has proposed a mutant QoS-0-based model that examines the retransmission of a guaranteed end-to-end service, the probability of the content delivery and the mean absolute content delivery w.r.t number of the connected nodes (N_c and N_r), arrival rates (λ and σ), content hold time (T_{cnt}) and content retransmission n . The parameters are first used in the modeling system performance equations and then examined in a real time environment. Figure 10 bears more importance of all because it directly gives us a clear

picture about the scalability of the IoT. Thus, smart cities can use the concept directly from this graph. A lower Round trip time RTT can be achieved with a lesser means absolute content delivery while keeping the value of n at the minimum. Likewise, a guaranteed packet loss is only possible when certain λ and σ are used in this regard.

Table 1 gives a comparative analysis of the proposed model, i.e., QoS-0 retransmission (for $n = 1$ and $n = 10$) with the currently employed MQTT service models. The CPU usage and transmission latency clearly state that the developed mechanism of mutant QoS-0 retransmission MQTT outperforms both in terms of the scalability as well as the guaranteed QoS with the bare minimum usage of the extra CPU power. The developed QoS-0 retransmission MQTT gives an improved packet delivery with the minimum repetition of the publishing process. Although the JoramMQ is promising for a lossless network for utilizing 3% of CPU power, it is a single transmission and discard-based protocol that degrades the packet delivery drastically over a lossy network. The work is a modified version of the Mosquitto Server scheme and reduces the CPU usage by more than 50% for an “at most once” MQTT mechanism, as in Figure 2.

Table 1. Comparison of the improved MQTT protocol with Benchmark MQTT, JoramMQ [16], Apollo servers [16], and RabbitMQ [17].

Service	Parameter	Apollo	JoramMQ	RabbitMQ	Mosquitto Server	QoS-0 Retransmission MQTT ($n = 1$)	QoS-0 Retransmission MQTT ($n = 10$)
QoS-0	CPU Usage	6%	3%	38%	24%	11%	15%
	Message Transmission Latency	5 ms	1.5 ms	10 ms	10 ms	2 ms	3.4 ms

The future of this work may go in three directions. One can include the re-transmission and acknowledged scheme in QoS-1 services with a fixed rule for the retransmission and for checking the improvement in the availability of the contents at the remote server. The CPU usage, that is not dealt due to the availability of a continuous power source in this work, is also debatable. A lower content delivery using a UDP-based network can be improved by applying same mathematical model of calculating the end-to-end delay of the network; this way MQTT-SN can be merged with retransmission-based mutant MQTT to reduce the network overheads, as in [4].

Author Contributions: Conceptualization, J.A., C.H. and M.H.Z.; methodology, J.A.; software, J.A.; validation, J.A., C.H. and M.H.Z.; formal analysis, J.A.; investigation, J.A.; resources, R.H., R.A.; data curation, J.A.; writing—original draft preparation, J.A., C.H.; writing—review and editing, J.A., C.H., R.A., R.H. and M.H.Z.; visualization, J.A.; supervision, C.H., M.H.Z.; project administration, M.H.Z.; R.H. and R.A.; funding acquisition, R.A. All authors have read and agreed to the published version of the manuscript.

Funding: The authors acknowledge the internal research start-up funding, reference: 1012606FA1, from University of East Anglia (UEA), Norwich, UK.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Khriji, S.; Benbelgacem, Y.; Chéour, R.; Houssaini, D.E.; Kanoun, O. Design and implementation of a cloud-based event-driven architecture for real-time data processing in wireless sensor networks. *J. Super Comput.* **2022**, *78*, 3374–3401. [CrossRef]
2. Andrew, B.; Rahul, G. OASIS MQTT Version. Available online: <http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html/> (accessed on 1 July 2021).

3. Handosa, M.; Gračanin, D.; Elmongui, H.G. Performance evaluation of MQTT-based internet of things systems. In Proceedings of the Winter Simulation Conference (WSC), Las Vegas, NV, USA, 3–6 December 2017; pp. 4544–4545.
4. Kavitha, K.; Suseendran, G. Priority based Adaptive Scheduling Algorithm for IoT Sensor Systems. In Proceedings of the International Conference on Automation, Computational and Technology Management (ICACTM), London, UK, 24–26 April 2019; pp. 361–366.
5. Pham, C.; Bounceur, A.; Clavier, L.; Noreen, U.; Ehsan, M. Radio channel access challenges in LoRa low-power wide-area networks. In *LPWAN Technologies for IoT and M2M Applications*; Academic Press: Cambridge, MA, USA, 2020; Volume 1, pp. 65–102.
6. Govindan, K.; Azad, A.P. End-to-end service assurance in IoT MQTT-SN. In Proceedings of the 12th Annual IEEE Consumer Communications and Networking Conference (CCNC), Las Vegas, NV, USA, 9–12 January 2015; pp. 290–296.
7. Chang, H.-L.; Wang, C.-G.; Wu, M.-T.; Tsai, M.-H.; Lin, C.Y. Gateway-Assisted Retransmission for Lightweight and Reliable IoT Communications. *Sensors* **2016**, *16*, 1560. [[CrossRef](#)] [[PubMed](#)]
8. Strowes, S.D.; Boundary Inc. Passively Measuring TCP Round-trip Times—A close look at RTT measurements with TCP. *ACM Queue* **2013**, *11*, 478–491. [[CrossRef](#)]
9. Günther, A.; Hoene, C. Measuring Round Trip Times to Determine the Distance Between WLAN Nodes. In *NETWORKING 2005: Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications Systems*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2015; Volume 3462, pp. 768–779.
10. Jiang, N.; Deng, Y.; Kang, X.; Nallanathan, A. Random Access Analysis for Massive IoT Networks Under a New Spatio-Temporal Model: A Stochastic Geometry Approach. *IEEE Trans. Commun.* **2018**, *66*, 5788–5803. [[CrossRef](#)]
11. Zanella, A.; Bui, N.; Castellani, A.; Vangelista, L.; Zorzi, M. Internet of things for smart cities. *IEEE Internet Things J.* **2014**, *1*, 22–32. [[CrossRef](#)]
12. Hasan, H.; Aqeel, S. IoT Protocols for Health Care Systems: A Comparative Study. *Int. J. Comput. Sci. Mob. Comput.* **2018**, *7*, 38–45.
13. Zeng, Z.; Che, H.; Miao, W.; Huang, J.; Tang, H.; Zhang, M.; Zhang, S. Towards secure and network state aware bitrate adaptation at IoT edge. *J. Cloud Comp.* **2020**, *9*, 38. [[CrossRef](#)]
14. *Standard Draft IEEE 802.15.4; Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks WPANs*. IEEE: New York, NY, USA, 2006.
15. MQTT Client Library for C. Available online: <https://www.eclipse.org/paho/> (accessed on 1 October 2020).
16. Comparison of Server's Processing Power and Transmission Rate. Available online: http://www.scalagent.com/IMG/pdf/Benchmark_MQTT_servers-v1-1.pdf (accessed on 1 October 2020).
17. Rabbit MQTT Message Broker. Available online: <https://www.rabbitmq.com/mqtt.html> (accessed on 1 July 2021).