Future Generation Computer Systems 89 (2018) 300–316

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Application-aware end-to-end delay and message loss estimation in Internet of Things (IoT) — MQTT-SN protocols

Deepsubhra Guha Roy [a,*,1], Bipasha Mahato [a], Debashis De [a,b], Rajkumar Buyya [c,d]

[a] Centre of Mobile Cloud Computing, Department of Computer Science and Engineering, West Bengal University of Technology, India
[b] Department of Physics, University of Western Australia, 35 StirlingHwy, Crawley, WA 6009, Australia
[c] Cloud Computing and Distributed Systems (CLOUDS) Laboratory, School of Computing and Information Systems, The University of Melbourne, Australia
[d] Manjrasoft Pty Ltd, Melbourne, Australia

HIGHLIGHTS

- A smart gateway selection algorithm is designed for IoT using MQTT-SN protocol.
- Pub-to-sub delay, message loss are analyzed using Paho client for different QoS level.
- It reduces packet delivery time by mapping solution gateway with the sensor nodes.
- It increases packet delivery rate with increasing packet number and payload.
- Dependency between message loss and delay are calculated during transmission.

ARTICLE INFO

Article history:
Received 29 December 2017
Received in revised form 16 April 2018
Accepted 24 June 2018
Available online 3 July 2018

Keywords:
MQTT-SN
Publish/subscribe system
Pub-to-sub delay
Eclipse-Paho
Mosquitto
Android MQTT application

ABSTRACT

Wireless Sensor Network (WSN) is suffering from ailments like a small lifetime of participating nodes, packet loss during delivery, an end-to-end delay etc. Static allocation of sensor nodes and gateways are suitable for the predefined, pre-planned WSN. The more we comprise sensor nodes, the more we need gateways to accomplish the whole IoT (Internet of Things) based scenario. These communication imprecisions have led the trendy IoT network to be self-motivated in nature. This article proposed a gateway to gateway load balancing solution for smart talk static defined WS cluster network to overcome these issues. We have explored the unusual behavior of WSN gateways with suitable theoretical expressions and applied dynamic gateway selection formula to reduce end-to-end delay as well as to increase packet delivery rate. The experimental scenario has been developed using sensor devices, Arduino Uno, Raspberry Pi 3, Amazon web services, MQTT broker and private cloud platform in our Mobile Cloud computing laboratory. Theoretical analysis shows that the occurrence of delay due to increasing number of packets and message payload has reduced compared to the existing approaches. Experimental results confirm the correctness of the proposed model.

1. Introduction

We consider a general model of wireless sensor network (WSN) [1–3] which can be used for various type of applications domain such as crop field monitoring, flood controlling, industrial plant controlling and monitoring, smart home monitoring [1], cultural heritage site monitoring [4] system etc. Data-centric communication approach provides a wrapper of publish/subscribe (pub/sub) messaging system onto these kinds of WSNs and implements IoT (Internet of Things), MQTT-SN protocol to achieve the message flow and exchange content between the sensor node and Broker/Server or user (Application PI) and Broker/Server. This is an approach, where the information is conveyed to the purchasers based on a function of their contents and interests, instead of their network addresses. Message Queuing Telemetry Transport (MQTT) protocol designed specifically for machine-to-machine is one out of various data-centric protocol of Internet of Things (IoT) is based on publish/subscribe messaging systems [5,6]. MQTT protocol is optimized for communications over networks [7] where the network connection could be broken very frequently or where bandwidth is at a premium. However MQTT provides an ordered lossless connection capability in a network such as TCP/IP which

* Corresponding author.
E-mail addresses: roysubhraguha@gmail.com (D. Guha Roy), mahatobipasha.91@gmail.com (B. Mahato), dr.debashis.de@ieee.org (D. De), rbuyya@unimelb.edu.au (R. Buyya).
[1] Presently, Maulana Abul Kalam Azad University of Technology, WB, BF-142, Sector-I, Salt Lake City, Kolkata 700064, West Bengal, India.

https://doi.org/10.1016/j.future.2018.06.040
0167-739X/

is too complex for wireless SA devices that have a small footprint, battery-operated devices with limited processing and storage resources. Lightweight publish/subscribe MQTT-SN protocol is the best suited and exclusively design for WSN environment, offers the real-time data transmissions with minimum publish to subscribe delay. From the gateway to the cloud platform [2] or from the system analysis to system response both totally depends upon available bandwidth service, security prohibitions of the whole system architecture and subscribed packages of the rent cloud platform like real-time factors. A network address is changed with the breaking of wireless links between the SA nodes. To keep the service alive once the SA node fails is replaced by another new node rather than being repaired. In such condition, apply a prediction approach [8] by using network address as announcement mode between the application and the SA node [9,3] may become tough because of the temporal and dynamic nature of the network. SA devices communicate with the gateway through the application in wireless media. Once a sensor node publishes a sensor data within the network it becomes available to the subscribers with the help of MQTT-S broker. A service subscriber can ask for any relevant data to the broker which is later sent to the subscriber through a gateway. Most of the event occurred without knowing or taking interest in the source address or name of the source node either. Applications are more interested in the data content which has to be published than the information about data producer. For example, an application is developed for an asset tracking purpose only concern with the present location of a current asset than the network address [10], received by the GPS. Therefore we are highly motivated to control over the nature of the connectivity in between the sensor nodes and the gateway. Smart talk gateway management selection has proposed over gateway layer to recommend a transmission route with effective packet delivery rate assist with dynamic packet retransmission time. Rather than local and remote sensor nodes, we considered local and remote gateways which reflect our clear approach in constructing the whole WSN cluster network into a smart gateway selective IoT association. Detail of the proposed procedure is presented later in this article.

2. Motivation and contributions

Internet of Things (IoT) is a rapidly exploring research area in the field of ubiquitous computing. Gateway has a big role as an intermediate in this field [8] between IoT devices and MQTT broker. Thus unable to select the right gateway to publish data content and to store a bulk amount of data into the cloud storage is actually power consumable as well as increases network latency. Our motivation is to successfully deal with this anomaly. The intuition of the proposed work is to choose the right gateway to publish relevant data with consideration of load balancing among the existence gateways [3,1] to prevent extra power consumption. The major contributions of this paper are summarized as follows:

- **i.** A smart gateway selection procedure has proposed for static defined IoT network to publish sensor data using MQTT-SN protocol.
- **ii.** The proposed technique focuses on how to map the solution gateway with the sensor node in order to improve packet delivery delay and rate along with other network parameters in consideration.
- **iii.** The performance of proposed technique in term of pub-to-sub delay, message loss has also been analyzed with the help of Eclipse-Paho client subscription using synthetic workloads in different QoS levels.
- **iv.** The correlation coefficient standards between message loss and end-to-end delay during the transmission on various levels of QoS has analyzed to calculate the dependency in between those two variables.

Comparison between our proposed approach and existing schemes of MQTT-SNs in IoT has done to highlight efficiency of the proposed model.

3. Related work

In this section, discuss the prior works in the context of IoT, WSN, MQTT-SN protocols.

3.1. Internet of things

To connect more than one electronics gadgets with the help of their associated IP address through the internet is the general purpose of the IoT connectivity [3]. Noticeably, connectivity is the backbone of IoT architecture, and for well-organized mobility handling it is important to stable the architecture with some reliable connectivity from publishing node to cloud server [10]. MQTT, CoAP, and LWM2M are the most extensively adopted current protocols in the fields of M2M and IoT. Within the IoT architecture [11,6,12] IoT devices are acted as end users and indirectly connected with the broker through gateways. To recognize how the MQTT and further MQTT-SN works for IoT devices [13] first we have to know the nature of WSN.

3.2. Wireless sensor network

WSN is the connection approach of IoT connectivity [14,1]. IoT devices are basically some embedded systems with linked IP address to connect with the broker and further with the subscriber. Within the wireless sensor network, the IoT publishers are defined as nodes. Depending upon the characteristics of the end nodes WSN has two possible architectures [15–18].

- **i.** ***Heterogeneous network*** (Network is built by nodes of different resource).
- **ii.** ***Homogeneous network*** (All nodes are identical in nature).

WSN nodes are dynamic in nature while destination network address changes with the changes of time instance.

3.3. MQTT-SN

It is a publish/subscribe (pub/sub) telemetry protocol, exclusively used in WSN and designed to connect IoT devices on top of the Zigbee which is open industrial consortium with the aim to provide single communication standard worldwide for WSNs. In addition, MQTT-SN mainly uses simple UDP connection for faster and light weighted message passing over wireless media than TCP. Thus MQTT-SN is best suited for the WSN with the compromise of reliability due to signal fading, higher link failure, and interface disturbances. Readers, who wish to gather enormous knowledge over Zigbee, MQTT, and MQTT-SN are encouraged to refer [19,5], and [20]. Fig. 1 show how the MQTT-SN connectivity is related to the simple MQTT protocol with the existing WSN to publish the IoT data.

• Connectivity procedure

With some additional features, MQTT-SN protocol is trying to keep its nature as far similar to MQTT as much possible. Figs. 3.1, 3.2a and 3.2b follow the basic steps to make the entire MQTT-SN connectivity comprehensive, is theoretically described here.

i. *Advertise by the gateway and discovery by other gateways*:

With the help of ADVERTISE message, one or more active gateways can announce their presence into the set-up of WSN. With the help of GWINFO message the client comes to know about the IP addresses of discovered active gateways and with the restriction to connect with only one gateway at a time, the client independently

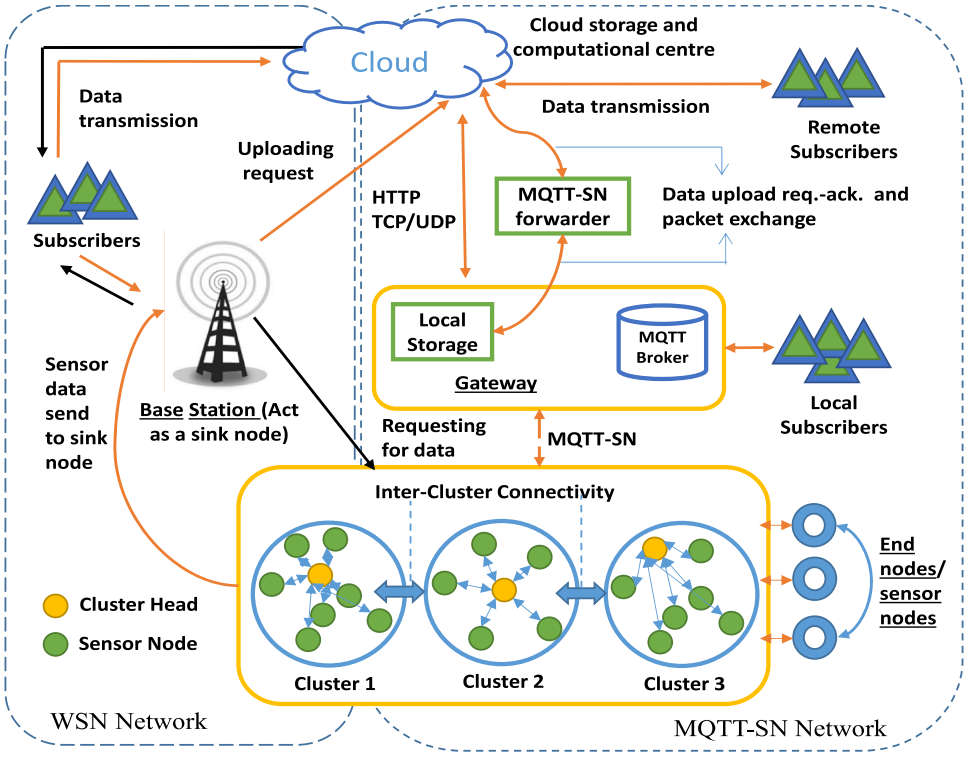


Fig. 1. The existing model of MQTT-SN protocol for WSNs.

can choose the gateway through which it wants to publish its data to the MQTT server [5]. We restricted this random gateway selection method and put a dynamic gateway selection procedure to make the statically defined WSN more efficient in the sense of time and energy requirement with respect to the successfully delivered packet rate of published sensor data.

ii. *Will data updating procedure*:

This is an added facility in MQTT-SN protocol where the client can update its will data by sending WILLTOPICCUPD or by sending WILLMSGUPD. The gateway then updates the previous saved wills topics with the help of new topic id and both of the messages are acknowledged. A model data table has articulated in [21]. In Section 4 it has also discussed.

iii. *Registration procedure of topic id*:

One of the existing limitations in WSN is restricted bandwidth [22]. In MQTT-SN protocol data is published with its topic id in the place of a long topic name used in MQTT. A registration procedure occurs before any PUBLISH message has been sending, where all the subscribers, GWs, and clients come to know about the list of short topic ids parallel to its topic name. Apart from these PING procedure and keep alive, sleeping clients support, procedure of Client's publishing, Client's retransmission, Client's disconnection, publish procedure of Gateway, PUBLISH message with Quality of Service Level 1, topic name registration procedure, clean session, connection setup of Client, Gateway advertisement, and its discovery are some others must focusing points which either recent to or depart from MQTT.

There are very few works have done on the integration of MQTT protocol and the wireless sensor network to analysis the publish-to-subscribe delay including message loss on various level of QoS provided by MQTT. In [23] the authors introduce the MQTT-SN protocol and show how it fits in the wireless sensor networks to revise the behavior of various protocols upon real time systems by implementing an MQTT-SN client–gateway instance. In addition, he explains the working process of the protocol, along with its advantages and challenges. The author in [19] has done the delay and message failure correlation analysis based on real data on different QoS levels of MQTT protocol in both domains wired and wireless network. The analysis of message loss over different payloads of messages and delay has done on wired and wireless network separately. They have also produced the correlation between these two variables. For the first time the pub-to-sub delay and service assurance of MQTT-SN protocol in wireless healthcare IoT system are explored in [24,11]. In MQTT-SN for healthcare application, the authors estimate the pub-to-sub content delivery delay and content delivery probability as functions of MQTT-SN. There are some other papers which compare the most popular application protocol i.e. MQTT and CoAp [25,11]. Sensor movement is an important parameter for accurate analysis of end-to-end QoS analysis. In all the recent-related work, the quality and service assurance of MQTT-SN protocol for wireless sensor network has not been covered. Despite the huge foreseen advantages, there is no more significant work on considering MQTT-SN for wireless sensor network system is done yet. We leverage the functionality of the protocol and applied to the wireless sensor network, also provide the detailed analysis of end-to-end quality assurance.

- **MQTT-SN gateway architecture**

Wireless sensor network is well known for its efficiency, though it has some inbuilt peculiarities in nature. Some of those are dynamic IP changes, short length of a published message, high rate of data link failure, low bandwidth etc. MQTT-SN is the extended version of MQTT protocol with the aim to fulfill the lacks regarding the implementation of the same in vulnerable sensor network, in addition, to optimize the protocol for low-cost, battery-driven IoT devices with limited processing capacity.

End sensor nodes of the network here symbolized as MQTT-SN clients, different types of MQTT-SN gateways, MQTT broker/servers are shown in Fig. 2. To connect with the MQTT-SN gateway, MQTT-SN clients use MQTT-SN protocol. Following this, MQTT-SN gateway connects with the MQTT broker/server via

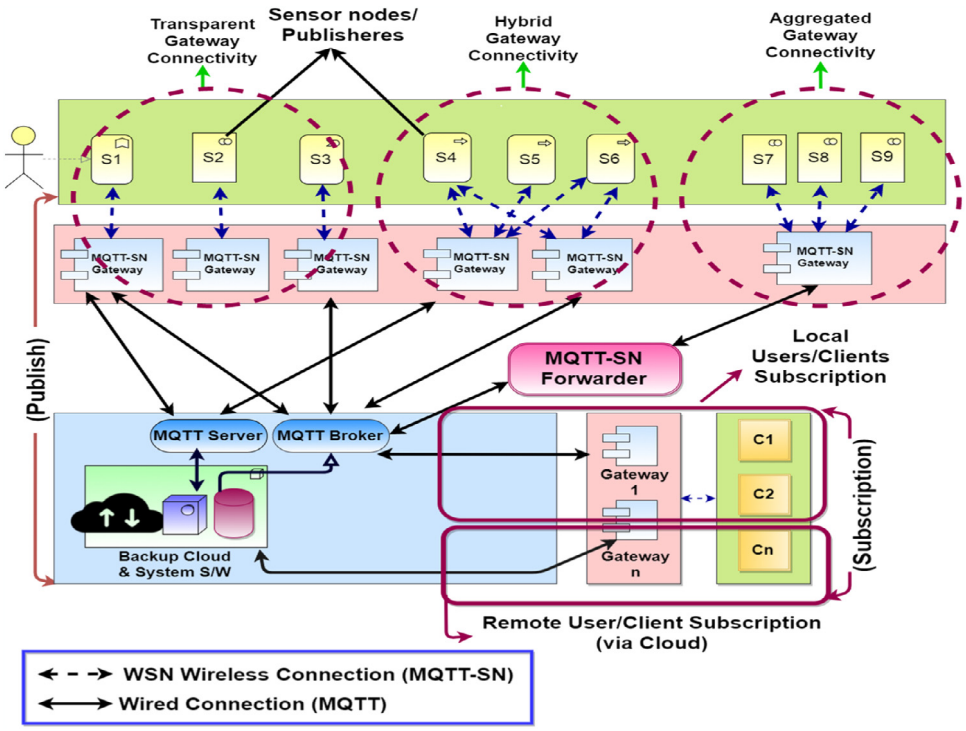


Fig. 2. MQTT-SN gateway architecture.

MQTT protocol. Some of the time MQTT forwarders actively take part between MQTT-SN client and broker. They simply encapsulate the received message from the client side and then without changing the message content it forwards to the broker. Similarly, it encapsulates the return frames coming from the broker/server side and again without changing the message content sent to the client. MQTT-SN gateways simply act as an interface between client and broker/server and mainly perform as a translator. The gateway usually categorized in three different kinds, depending on their performance and how they are connected with the clients and brokers, are shown in Fig. 2. We estimated source node to consumer node delay for the published content from the time it became available in the network and the time it was delivered to the user and also analyze the message/ packet loss under different QoS levels.

Later in Section 5, we calculate the estimated delay of our proposed network model. Emulation setup of our proposed model and extract results are shown in Section 5. We conclude our overall achievement in Table 7 with comparative analysis and also we have talked about the future directions related to our model in Section 6.

4. Proposed gateway selection strategy

Proposed MQTT-SN model for the wireless sensor network is shown in Fig. 2. The user can subscribe for the content either locally (through broker/server) or remotely(through application reside on the internet). Local users make a subscription for the content directly to the Broker/Server, whereas the remote users make subscription through the applications(controlling and monitoring application) which reside on the traditional network(Internet). Using via UDP connection sensor node is connected with the MQTT-SN gateway with the help of MQTT-SN protocol. Whether it is a sensor device, which wants to publish its content data or it is an MQTT gateway, which advertises its presence by periodic broadcasting, for connecting mechanism both of them follows some rule within a particular range. Considering the specific situation: discovering the presence of each other, at the time to send first CONNECT message by the client to the gateway two types of ranges are must define to make out the situation as well as the conditions clear, which are as follows:

4.1. Sensor node to gateway connectivity

Definition 1 (*Transmission Range*)**.** Transmission range is the specific range, within the gateway coverage area or area-of-interest of a gateway, which is satisfactorily covered by particular gateway GW_i

Definition 2 (*Sensing Range*)**.** Sensing range is the area coverage at a particular time t by a gateway GW_i to sense CONNECT message, send by the farthest connected sensor device within the total possible coverage area of a gateway GW_i.

If r_{tn} denotes the radius of the transmission area and r_{sn} denotes the radius of the sensing area of a gateway GW_i at a particular time t, then the ratio of r_{sn} with respect to r_{tn} in between zero and one represent the following states:

$$\frac{r_{sn}}{r_{tn}} \begin{cases} = 1 & \text{[When the farthest sensor node placed on the circumference of the transmission area]} \\ = 0 & \text{[Either no connection has established yet or all sensor nodes are in sleep mode.]} \\ \leq 1 & \text{[At least one node is active]} \end{cases}$$

where $r_{tn} \geq r_{sn}$ and all N number of possibly connected nodes must be within the transmission range. $r_{s1}, r_{s2}, \ldots, r_{sn} \leq r_{tn}$. $\{S_1, S_2, \ldots, S_n\} \subseteq S$ are the publisher nodes, where $S = \sum_N S_i$, is the total set of possible connected sensor node at a time via the same gateway; Coverage and connection possibility is closely related. If $r_{tn} \geq 2 \times r_{sn}$ then it implies connectivity within the coverage.

According to Fig. 2 MQTT broker is subscribed by some local subscribers and others remote users are symbolized by $_{U_l}$ and $_{U_r}$ respectively. Therefore, $U = \sum_{i=1}^{N} U_l(i) + \sum_{i=1}^{N} U_r(i) = \sum_{i=1}^{N} U_l(i) + U_r(i)$;

For easy understanding, we have first considered all delays with respect to the local user and then some extra delays have added in the case of subscription by any remote subscriber. Both of the delays has calculated later in this paper and have shown in Eqs. (5) and (6) respectively. We will state adequate conditions for local optimal delay calculation when these four alternative situations may occur:

i. *Unconstraint or no limit event/state*
ii. *Occurrence of only equality constraint event/state*
iii. *Only inequality constraint event/state*
iv. *Fusion or equality–inequality constraint event/state*

Sensor device, gateway, server, subscriber all are transmission nodes, particularly sensor device and subscriber are end nodes, one in publishing side and another in subscribing side. Easy to understand for now we are considering the scenario when sensor node n_i waiting to connect with the MQTT server via an MQTT-SN gateway GW_i.

i. Definition 3 (Unconstrained State): f is the local delay function changes depending upon local differentiable random distance r^* in between client and gateway, within the transmission range (radius: r_{tn}) then delay occurrence measured by $f\ (r)$ is the delay at client side before receiving the message CONNACK followed by CONNECT request to publish some content to the MQTT server via the MQTT-SN gateway. f has zero gradients at r^* when no connection has established yet. Then the unconstrained state is defined as: $\nabla_r f(r^*) = 0$;

If and only if the connection has established beforehand then only the delay will include estimating total delay. Otherwise, the publish method will end with negative acknowledgment from the server site or timeout will occur. It may happen due to congestion in the network or frequent changes of dynamic IP. Unconstrained or no constrained minimization of the WSN through the delay function is represented by:

$v^t(\nabla^2 f(r^*))v \geq 0$ and for maximization unconstrained delay representation will be: $v^t(\nabla^2 f(r^*))v \leq 0$. Where $f(r^*)$ defines the distance-dependent delay to connect with a gateway at a particular time t and for variable data content v such that $\forall v \in \Re^n$.

ii. Definition 4 (Equality Constrained State): This state occurs when a gateway is already connected with exactly N number of sensor nodes, that is mean $N - \sum_{p=1}^{n} N_p = 0$. No connection is available at the time instant $(t-1)$. Just at time t, the gateway became free and wants to acknowledge new sensor node N_i. The occurrence of delay to get CONNACK from the gateway at ith node side is represents by $f\ (r)$. Feasible region: $\min_{r\in\Re 2} f(r)$ subject to $h(r) = 0$; here $f\ (r) = r_1 + r_2$, $h\ (r) = (r_1^2 + r_2^2 - 2)$ and, r_1, r_2 are the distance of two nodes n_j and n_i, from the same gateway GW_i, mentioned before.

In Fig. 3.1a feasible region $h(r) = 0$, iso-contours of $f(r)$ and $h(r)$ functions is shown. If sensor node n_j has disconnected with GW_i and a new connection has established with n_i we can say that feasible point R_F migrate to R_{F1} for the distance δr. It affects the constrained functions such that: $h(R_F + \alpha\delta r) = 0$ and $f(R_F + \alpha\delta r) \prec f(R_F)$. To establish a new connection it must satisfy $f(r + \delta r) \prec f(r)$ with $\delta r(-\nabla_r f(r)) \succ 0$. Here, $\nabla_r h(r)$ is the normal to the constrained surface. Note that, the direction of the position change is arbitrary and orthogonal within the transmission range imposed as either $h(r) = 0$ or $-h(r) = 0$.

iii. Definition 5 (Inequality Constrained State): A sensor node within a WSN has moved from one place to another and wants to connect with the same gateway which is already known to it the delay occurrence state will be defined as inequality constrained

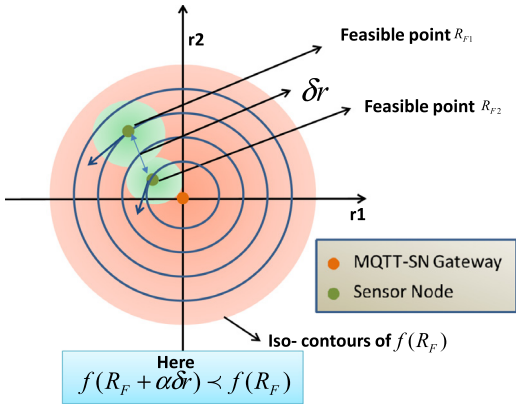


Fig. 3.1a. Equality constrained state within transaction range.

Gateway / Requests	1	2	3	-	n
1	-	-	1	-	-
2	1	-	1	-	-
3	2	-	1	-	-
-	-	-	-	-	-
-	-	-	-	-	-
n-1	1	3	4	-	-
n	1	4	4	-	-

Fig. 3.1b. Gateway allotment with respect to request processing table.

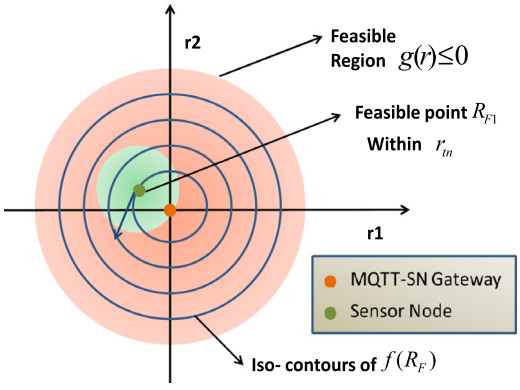


Fig. 3.2a. Inequality constrained state within transaction range, before new sensor node is found.

state. This particular inequality constrained problem is stated as: $\min_{r\in\Re} f(r)$ subject to $g(r) \leq 0$. Where $f\ (r) = r_1^2 + r_2^2$, $g\ (r) = r_1^2 + r_2^2 - 1$ and R_f denotes a feasible point within the range of r_{tn}. If $g(r^*) < 0$ implies deactivation of the ith node at R_F therefore it will be identified by the same state within the network as in the unconstrained state. In Fig. 3.2(a) minimum of $f(r)$, feasible region $g(r) \leq 0$ and sensor device (feasible point) R_F is shown. At $g(r^*) < 0$, the constraint is inactive. Therefore the previous condition will apply. Fig. 3.2b it has shows how the position of R_F has changed to R_{F1} remaining the access point (gateway) at a constant position. Now, due to the activation of equality constraint $g(r) = 0$ new connection has established. Note that, optimality constraint constant find or feasibility search is not our motive.

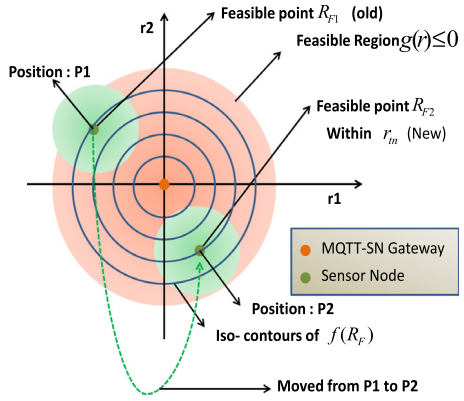


Fig. 3.2b. Inequality constrained state within transaction range, after new sensor node found.

Delay function measurement with the help of changing situations in a feasible condition is mainly pressurized in our work.

iv. Fusion or equality–inequality constraint: Fusion or equality is a state when chances or probability of both occurrences (equality and inequality) constraints are same. Similar scenarios can be seen if the gateway and the sensor nodes will be replaced by MQTT broker/server and gateways respectively. Here we are considering the particular situation when a message has been published into the WSN and wishes to subscribe by a user node with Eqs. (2) to Eq. (7) are using to calculate this particular time-delay and the additional delays which are differ according the situations already discussed and also explained with Figs. 3.1a, 3.2a and 3.2b. These varied times are denoted as network delay or system delay, represented by the α and β, further has use to calculate end-to-end delay calculation. Probability of α and β occurrences are in between (0, 1). Where '0' represents the probability of network failure and tells about some reasons, like data has either not published or not has subscribed by the end nodes. And 1 denotes the possibility of 100% data transmission occurrence.

In the architecture of MQTT-SN clients, MQTT-SN gateways, and MQTT-SN forwarders are three kinds of MQTT-SN components. MQTT-SN forwarders forward the publish request from the clients to the gateways, we ignore the requirement of forwarder in our architecture. In addition, MQTT-SN gateway is optionally incorporated with the MQTT Server/Broker. We integrate the MQTT-SN gateways to the MQTT broker and these combined acts as a gateway to the wireless sensor network. Each publishing request reached to the broker where gateway manager assign a gateway to publish the sensor data. Three kind of publishing request has described with Figs. 3.1a, 3.2a and 3.2b. Fig. 3.1b shows a model allotment table within the manager database which is updated after assigned a gateway against each publishing request. Each allotment is done following Algorithm 1 and after each update the information table is replicated [21] to the manager database. Further when next allotment is required then the table is retrieve to make next decision. This looping process is shown into Fig. 7.

4.2. Proposed load-balancing dynamic gateway selection method for static defined WSN

Network Model: Sensor nodes are deployed at fixed place as per their utilization and requirements, within L*L area having following characteristics:

i. All isomorphic sensor nodes are placed as per user model. Nodes are unable to move after they have deployed. Manual maintenance is not required where all the nodes are not energy renewable in nature.
ii. All the nodes having data fusion capability within WSN, used for particular purpose, able to sense particular attribute.
iii. Communication power is easy to regulate as per the distance between source node and destination.
iv. There are fixed numbers of sink nodes or gateways within the total L∗L WSN scenario. Each gateway has been deployed following fixed cluster head selection method (Reference) within all πr_i^2 cluster area where sensor nodes are almost equally far from that.
v. The model has established upon the power consumption model of threshold gain which can be defined

$$\text{as: } e_0 = \sqrt{\frac{E_{fr}}{E_{mf}}}; \tag{1}$$

Here, E_{fr} denotes the energy consumption while sending 1 bit data without any constrain over the links and E_{mf} is the energy consumption while sending 1 bit data through the service root when all the possible tasks are executing as per the work load.

Publish-to-subscribe delay analysis is defined in this section which is the delay in between publishing the content to the server through the broker by the MQTT-SN client and consumed that exact data by the authorized MQTT user. With the restriction, we have estimate end-to-end delay for the local user only, which directly make a subscription for the particular topic of content to Broker/Server. We estimate pub-to-sub delay at all different levels of QoS. From connection establishment to message deliver from sensor node to the consumer end in case of QoS-1 total six events, we have considered here, shown in Fig. 4.

Event 1: Source node chose a gateway among the discovered gateways and establishes the bond by sending CONNECTS command to the Gateway. The Gateway response with CONNACK command and establish the connection. Once the connection is established the sensor nodes publish data content with the help of PUBLISH method with specific TOPIC-ID to the MQTT-SN Gateway. In response, Gateway sends PUBACK to the sensor node. These message exchanges take place using wireless media over MQTT-SN and UDP.

Event 2: This event initiate at the gateway end. Gateway initiates the connections by sending CONNECT message command to the broker. The broker response with CONNACK command and establish the connection. After the connection is established, then with the help of PUBLISH method the Gateway forwards the frame which it received from the source node to the MQTT server/ broker with TOPIC-ID and PUBACK. MQTT with TCP connection over wired media, this event takes place.

Event 3: This event starts according to the user's will to subscribe the published content from the broker by sending CONNECT command; in response, the broker established the connection with CONNACK. After the establishment of the connection with the help of PUBLISH method, the gateway forwards the frame which it received from the source node to the MQTT server/ broker with TOPIC-ID and PUBACK. MQTT with TCP connection over wired media, this event takes place.

Event 4: Broker established the connection to the user by sending CONNECT message command; in response, the User established the connection with CONNACK. After the connection established the broker take the help of TCP over wired MQTT media to release content on TOPIC-ID requested with the help of PUBLISH and PUBACK methods.

Let assume TD^{SU} as the total publish-to-subscribe delay with total 2 UDP round trips and 6 TCP back to back communications, shown in Fig. 5. When the sensor sends the content to the Gateway, then it performs the translation between MQTT-SN and MQTT, thus

ALGORITHM I: Dynamic Gateway Selection Algorithm with Load Balancing Strategy

INITIALIZATION:

i. WSN with some static allocated IoT devices: $WSN = \{S_i, G_i, CBR\}$

$S = \sum_N S_i$;

$GW = \sum_N GW_i$;

ii. CSC = Cloud Storage Centre
iii. ACK = Success acknowledgement
iv. NACK = Negative response

ESTIMATED OUTPUT:

i. T = Total transmission time/ pub- to- sub delay ;
ii. $Load\_Head$: Load of the gateway;
iii. T_{sg} = Packet delivery time from sensor node to gateway;
iv. T_{gb} = Packet delivery time from gateway to broker ;
v. T_{bs} = Packet delivery time from broker to subscriber ;
vi. T_D = Delay occurrence due to network congestion ;
vii. Connectivity between $n \in N$ numbers of gateways ;

START

1. Initialize n numbers of connectivity table against each gateway into CS ;
2. Connectivity establishment between gateway manager and each alive gateways;
3. Add a new sensor node S_i into WSN ;
4. Check for available connectivity route;
5. Request to publish data content;
6. Send topic code which is already known by the gateway manager;

/* **Calculate** $Load\_Head$ */

7. **For** r number of requests to the gateway manager

$$e(GW_i) = e_0(GW_i) + \sqrt{\frac{E_\Delta^{fr}}{E_\Delta^{mf}}} \text{ (Equation 1) ;}$$

$$\textbf{Where } e_0(GW_i) = \sqrt{\frac{E_0^{fr}}{E_0^{mf}}} ;$$

8. **Estimate** T **Where** $T = T_{sg} + T_{gb} + T_{bs} + T_D$ **;**
9. **If** $Load\_Head < Threshold\_Load$

   Data publish through GW_i;
10. **Else**

    Go to step 7 ;
    Find a new efficient gateway GW_j ;
    Load transfer to GW_j from GW_i;
    Direct connection establishment between sensors node S_i and GW_j ;
    Data publish through GW_j ;
11. **For** n number of requests to the gateway manager Calculate new_Load ;
12. Update Load_Table ;
13. Publish load table to each connected (n-1) Gateways ;
14. **End For**
15. **End If**
16. Backup of Load_Table after 9, 10, 12 ;
17. **END**

content from the sensor not directly forward to the Broker by the Gateway. Similarly, when the user makes the subscription for the content to the Broker, there is some delay before publishing the content to the user. Let the delay at the gateway and delay at the Broker are represented by α and β respectively. After the content from the sensor published on the Broker, assume that the user makes the subscription request after some time, thus we also take the user side delay represented by μ. Thus we have completed publish-to-subscribe delay of the content from the source end to

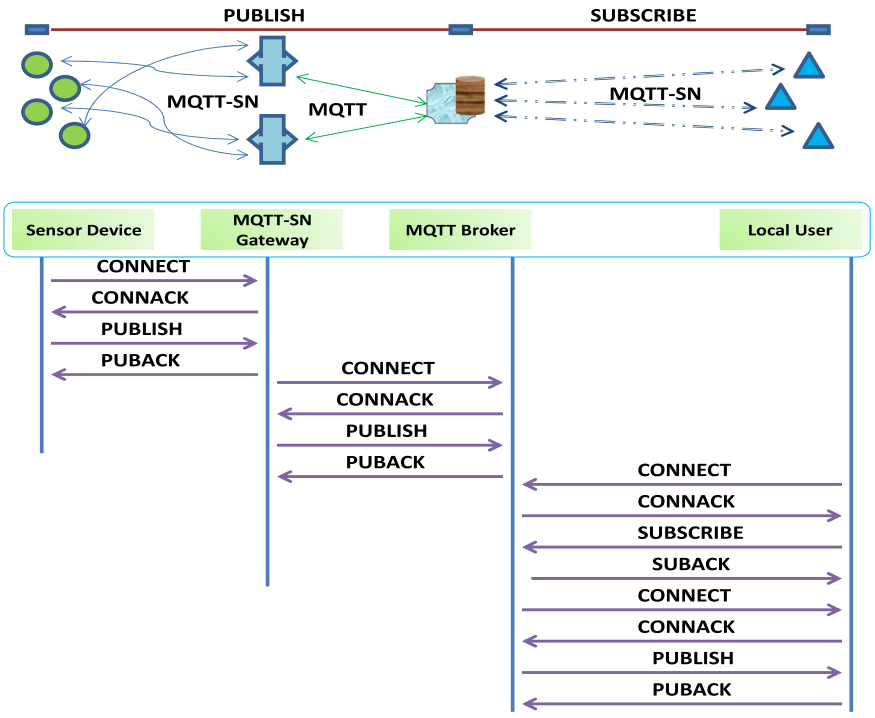


Fig. 4. Estimation of end-to-end delay in QoS-1 level.

the local user as follows:

$$TD^{SU} = T^{SG}_{UDP-CONNECT} + T^{SG}_{UDP-PUB} + \alpha + T^{GB}_{TCP-CONNECT} + T^{GB}_{TCP-PUB} + \mu + T^{UB}_{TCP-CONNECT} + T^{UB}_{TCP-SUB} + \beta + T^{BU}_{TCP-CONNECT} + T^{BU}_{TCP-PUB} \quad (2)$$

In Eq. (1), $T^{SG}_{UDP-PUB}$ denotes the round trip delay for UDP-PUBLISH with PUBACK from the source node to the gateway; assuming the delay is symmetric in between the source node to the gateway, MQTT-SN gateway to the broker and the broker to the user. Eq. (2) derives the UDP-PUBLISH delay and Eq. (3) derives the delay in TCP connection in order to derive total pub-to-sub delay to deliver the published content.

$$T^{SG}_{UDP-CONNECT} = T^{SG}_{UDP-PUB} = T^{SG}_{UDP} \quad (3)$$

$$T^{GB}_{TCP-CONNECT} = T^{GB}_{TCP-PUB} = T^{UB}_{TCP-CONNECT} = T^{UB}_{TCP-SUB} = T^{BU}_{TCP-CONNECT} + T^{BU}_{TCP-PUB} = T^{UB}_{TCP} \quad (4)$$

Then we can write from Eqs. (2) and (3) as follows:

$$TD^{SU} = 2T^{SG}_{UDP} + 6T^{UB}_{TCP} + \alpha + \beta + \mu \quad (5)$$

TD^{SU} is overall time delay of the content which originates from the sensor and delivered to the user at QoS-1 level. Publish-to-subscribe delay estimation of quality in service level-0 and level-2, to be applied in the similar way in which we did QoS-1, the model for message exchange is shown in Fig. 4. Hence in the case of quality of service level o, there is no reply as an acknowledgment from the gateway side to the client side at the time of message publish, which is shown in Fig. 5(a) the pub-to-sub delay estimation expression can be written as:

$$TD^{SU} = T^{SG}_{UDP} + 4T^{UB}_{TCP} + \alpha + \beta + \mu \quad (6)$$

Similarly, in the Quality of Service level-2 in the place of PUBACK, three new messages are executed. PUBREC, PUBREL, and PUBCOMP are those three new messages; ways of execution of the messages are shown in Fig. 5(b). Hence pub-to-sub delay estimation expression for Quality of Service level-2 can be written as:

$$TD^{SU} = 6T^{SG}_{UDP} + 10T^{UB}_{TCP} + \alpha + \beta + \mu \quad (7)$$

All the symbols used in the equation have been described in detail in Table 1.

5. Performance evaluation

We have set up an emulation framework illustrated in Fig. 6, by using sensor nodes, Eclipse-Paho client library, open source Mosquitto Broker/Server, private cloud, Amazon web service and Android MQTT application in MAKAUT, WB cloud hub. Our design framework is divided into four major segments: Sensor client; Broker with gateway; Gateway manager; Subscriber or User. Wireless sensor client is implemented by using the Eclipse-Paho MQTT-SN client library. Convenience to make the design simple, we limited into total 11 Paho clients and some real publishing nodes from the sensor node network. Data are combined and based on time stamp, sending requests to publish the content data to the broker by using proposed Algorithm I.

Note that all the evaluation has done over the data which are going to be published to the broker. The Eclipse-Paho [26,27] is a one of the first open source project, which provides an MQTT and the MQTT-SN client implementations library available and is actively maintained by a huge community [28], it has synchronous API, which is highly callback based and allows to attach event-based logic on the MQTT Paho client, e.g. when a message is received or when the connection to the broker is lost [29,16]. In addition, it does also provide support for a various version of MQTT with possible secure communication via Transport Layer Security (TLS). To implement the MQTT broker part we use Mosquitto, an open source message broker which is able to implement the MQTT protocol a different version for carrying out the messages using a Publish/Subscribe model [15]. It allows the client to publish the content on a certain topic at all three levels of QoS, and provide a secure communication for the MQTT client. For the user side we use Paho Android services, it is an interface to the Paho MQTT client library for the Android application. By using this service the user easily performs messages exchange to the broker after establishment of the connection. Amazon web service allowed us

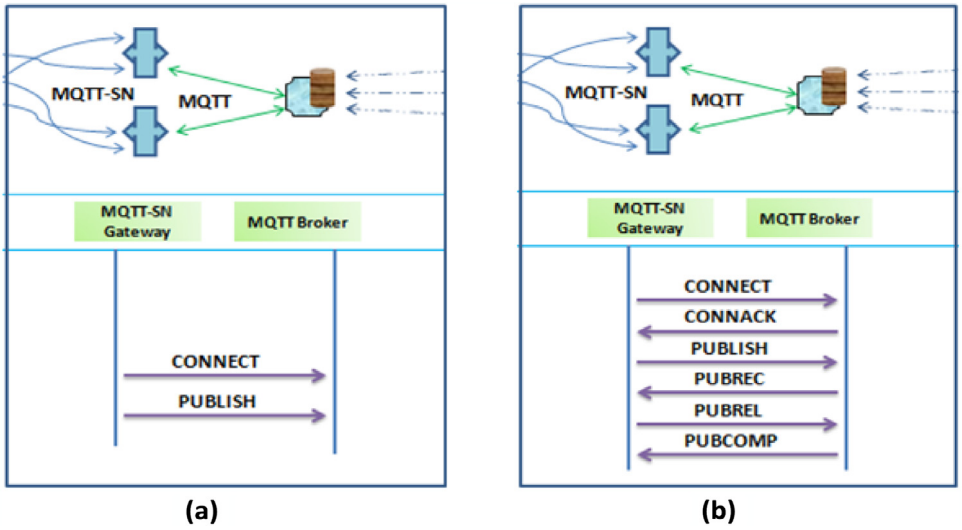


Fig. 5. Message exchange model of QoS-0 (a) and QoS-2 (b).

Table 1
Description of the symbols used in the equation to calculate total end-to-end time delay.

Symbol	Description
TD^{SU}	The total time delay of content from the sensor to the user.
T^{SG}_{UDP}	Time taken by the content from sensor node to gateway over wireless UDP connection.
T^{UB}_{TCP}	Time between the broker and user over wireless TCP connection.
$T^{SG}_{UDP\text{-}CONNECT}$	Time taken by the sensor node to established connection with gateway over UDP.
$T^{SG}_{UDP\text{-}PUB}$	Time taken by the sensor node to published the sensed data to the gateway over UDP.
$T^{GB}_{TCP\text{-}CONNECT}$	Time taken by the gateway to established connection with broker over TCP.
$T^{GB}_{TCP\text{-}PUB}$	Time taken by the gateway to published the sensed data to the broker over TCP.
$T^{UB}_{TCP\text{-}CONNECT}$	Time taken by the user to established connection with the broker over TCP.
$T^{UB}_{TCP\text{-}SUB}$	Time taken by the user to subscribe the sensor data to the broker over TCP link.
$T^{BU}_{TCP\text{-}CONNECT}$	Time taken by broker to established connection with the user in order to published sensor data over TCP.
$T^{BU}_{TCP\text{-}PUB}$	Time taken by the broker to published the content to the user over TCP link.
α	Delay at the gateway while forwarding the content from sensor node to the broker due to aggregation of multiple MQTT-SN request to single MQTT.
β	Delay occurred at the broker after receiving the subscribe request from the user to publish the content.
μ	Delay from the user side.

to make a broker service platform as well as to implement gateway manager using proposed algorithm in smart cloud environment. Private cloud instances have used for three major purposes: Gateway manager; Gateway manager database and Gateway manager database backup. Gateway is implemented by using Raspberry pi 3. The whole workflow of emulation setup is shown in Fig. 7 and equipment used to implement the whole setup, mention in Table 2.

For MQTT-SN the "topic-id" is two bytes long, short-term representation of topic name in PUBLISH message, used in MQTT protocol. Predefined topic-id of the content is known to the source end in advance, to skip the requirement of registration procedure in which, sensor node allow registering their topic name with the broker or gateway and obtained the corresponding topic-id. We have added the registration procedure to the broker side, by assuming that old sensor node is replaced by new one with different topic name [10].

Thus register their topic name in persistence mapping table is needed to maintain dynamically by the broker side. The users subscribe the content by the topic name through Android MQTT application. Broker maintain all the mapping of a topic name with its corresponding topic-id, after receiving the subscribe request from the user. Then the broker publishes the related content to the user.

5.1. Pub-to-sub delay analysis

We analysis publish-to-subscribe for three different MQTT-SN quality of service levels is shown in Figs. 8a–8c, based on the number of published content and the subscription request arrived at the broker.

At the place of publish-to-subscribe alternatively, the end-to-end or pub-to-sub delay are also in consideration. We take a various number of publishing requests and subscription requests; we assume that both requests are equal in number. For each number

Table 2
Configurations of equipment's used in experiment.

Equipment	Equipment quantity	Provider/vender	RAM	HDD	Processor	Operating system
Sensor node	Temperature sensor [TMP36] (5 pcs.)	kjdElectronics	–	–	–	–
	Accelerometer sensor [ADXL335] (4 pcs.)	Omega	–	–	–	–
	Sound Sensor [LM393] (2 pcs.)	HALJIA	–	–	–	–
MQTT broker	MQTT broker: Computation instance	AWS EC2 instance Free Tier	1 GB	8 GB	Intel® Xeon® @2.5 GHz VCUP-4	Ubuntu
	MQTT broker: Database instance	AWS EC2 instance Free Tier	1 GB	8 GB	Intel® Xeon® @2.5 GHz VCUP-4	Ubuntu
	MQTT broker: Backup/restore instance	AWS EC2 instance [Paid]	4 GB	1 TB	Intel® Xeon® @2.5 GHz VCUP-4	Ubuntu
MQTT gateway Manager	MQTT Gateway master manager	Private cloud instance using Openstack	16 GB	2 TB	Intel® Xeon® CPU ES-2667 0 @2.90 GHz (Hexa Core)	Cent OS
	MQTT gateway manager: Database	Private cloud instance using Openstack	16 GB	2 TB	Intel® Xeon® CPU ES-2667 0 @2.90 GHz (Hexa Core)	Cent OS
	MQTT gateway manager: Backup/Restore	Private cloud instance using Openstack	16 GB	2 TB	Intel® Xeon® CPU ES-2667 0 @2.90 GHz (Hexa Core)	Cent OS
Gateway	Gateway	Raspberry Pi 3	1 GB	32 GB	4× ARM Cortex-A53 @1.2 GHz	Noobs
	Gateway	Raspberry Pi 3	1 GB	32 GB	4× ARM Cortex-A53 @1.2 GHz	Noobs
	Gateway	Raspberry Pi 3	1 GB	32 GB	4× ARM Cortex-A53 @1.2 GHz	Noobs
Receiver node	Receiver node 1	Lenovo Ideapad 500	8 GB	1 TB	Intel Core I7 6th Gen @ 2.5 GHz-Dual Core	Windows 8.1
	Receiver node 2	Moto E4	2 GB	32 GB	Quad core@1.3 GHz, Android 7.1.1	Nougat
	Receiver node 3	Redmi 3S Prime	3 GB	64 GB	Octa-core@1.4 GHz, Android 6.0.1	Marshmallow

Table 3
Pub-to-sub delay result obtained at various levels of QoS on particular number of publish request and subscribe request to the broker.

Number of publish requests made by Ecplise-paho client to the broker	Number of subscribe requests made by Android MQTT application to the broker	Pub-to-sub delays obtained at QoS level-0	Pub-to-sub delays obtained at QoS level-1	Pub-to-sub delays obtained at QoS level-2
100	100	160	210	280
200	200	220	430	537
300	300	343	470	590
400	400	410	527	639

of request, we calculate the pub-to-sub delay of the message from the source to destination. We choose the arbitrary number of the requests as shown in Table 3. There are a number of the users making subscription request to the broker. So we analyze the pub-to-sub delay of the message on four different numbers of published content and the subscription request i.e. 100, 200, 300, and 400 at all three levels of QoS, using Eq. (7).

The result obtained at each level of QoS, shown in Table 3. From this result we plot the graph and how pub-to-sub delay of the messages are dependent on a number of publishing requests made by the paho client to the broker and the number of subscribe requests made by the Android MQTT application to the broker are shown in Fig. 8a. The Eclipse-paho client makes a publish request by publishing the messages on particular topic-id to the Mosquitto broker, concurrently many paho client publish the message on different topic-id to the broker. At the subscriber end the users are using Android MQTT application to make a subscription request for particular topic-id of the message.

We select various size of the message payload in bytes as shown in Table 4, and on that size analyze the end-to-end delay on various levels of QoS. From this result we plot the graph and show how pub-to-sub delays of the messages are dependent on payload of the message as shown in Fig. 8b.

In Fig. 8b shows pub-to-sub delay for various QoS levels by varying the payload of the messages. We assume and kept the rate at which the sensor is published its content to broker is equal to the rate of subscription request made by the user to the broker, for sake of simplicity. The payload of the messages is controlled by the sensor devices. In our case, we use Eclipse-paho MQTT-SN client to publish the messages to the broker, so we can control the payload of the message by using paho Application Programming Interfaces (API). Increasing payload of the message also effects the end-to-end delay time.

We can see in Fig. 8b, QoS level-0 has a less pub-to-sub delay than QoS level-1 and QoS level-2, and in the same order, QoS level-1 has less end-to-end or pub-to-sub delay than QoS level-2. Thus we can see in this figure that as the message payload is increased, in same way the pub-to-sub delay also increased.

5.2. Source-to-destination message loss analysis

We also analyze the message loss during Source-to-Destination delivery of the message or content. End-to-end delay calculation has considered with respect to 1000, 2000, 3000 and 4000 bytes of message payload which are published to the broker set by Paho API.

Fig. 6. Tools used for emulating the design Framework.

Table 4
Pub-to-sub delays obtained at various levels of QoS on different payload of the message in bytes which has published to the broker.

Message payload going to be published to the broker set by Paho API (in byte)	Pub-to-sub delays obtained at QoS level-0	Pub-to-sub delays obtained at QoS level-1	Pub-to-sub delays obtained at QoS level-2
1000	183	227	310
2000	293	421	526
3000	385	494	580
4000	498	580	697

Table 5
The percent of message loss occurred during the transmission on various levels of QoS, over different payloads of the message in byte which are going to be published to the broker.

Message payload in byte which are going to published to the Broker set by Paho API (in bytes)	Source-to-destination loss of messages calculation at QoS level-0 (In %)	Source-to-destination loss of messages calculation at QoS level-1 (In %)	Source-to-destination loss of messages calculation at QoS level-2 (In %)
1000	1.00	0.24	0.18
2000	1.40	0.41	0.20
3000	1.60	0.60	0.22
4000	1.80	0.78	0.24

We select a various size of the message payload in bytes as shown in Table 5, and on that size analyze the percent of message loss occurred during the transmission on various levels of QoS. From this result, we plot the graph and show how message losses are dependent on the payload of the message as shown in Fig. 8c.

The sensor simply fires and forgot the message, whereas, in QoS level-1, the message must deliver to the receiver end at least once with the guarantee. Similarly, in QoS level-2, it is guaranteed that sent message must be received by the destination node only once with the support of higher handshaking trade.

We can see in Fig. 8c that QoS-0 has more message loss percentage than QoS level-1 and QoS level-2, and in the same order, QoS level-1 has a less message failure percentage than QoS level-2. This is mainly because, in QoS level-0, the receiver never acknowledge after receiving the message from the broker, or even store or redelivered.

5.3. Comparative analysis of proposed method with existing method

To check whether our proposed method is efficient for IoT based WSN architecture or not, a comparative study has done with following approaches.

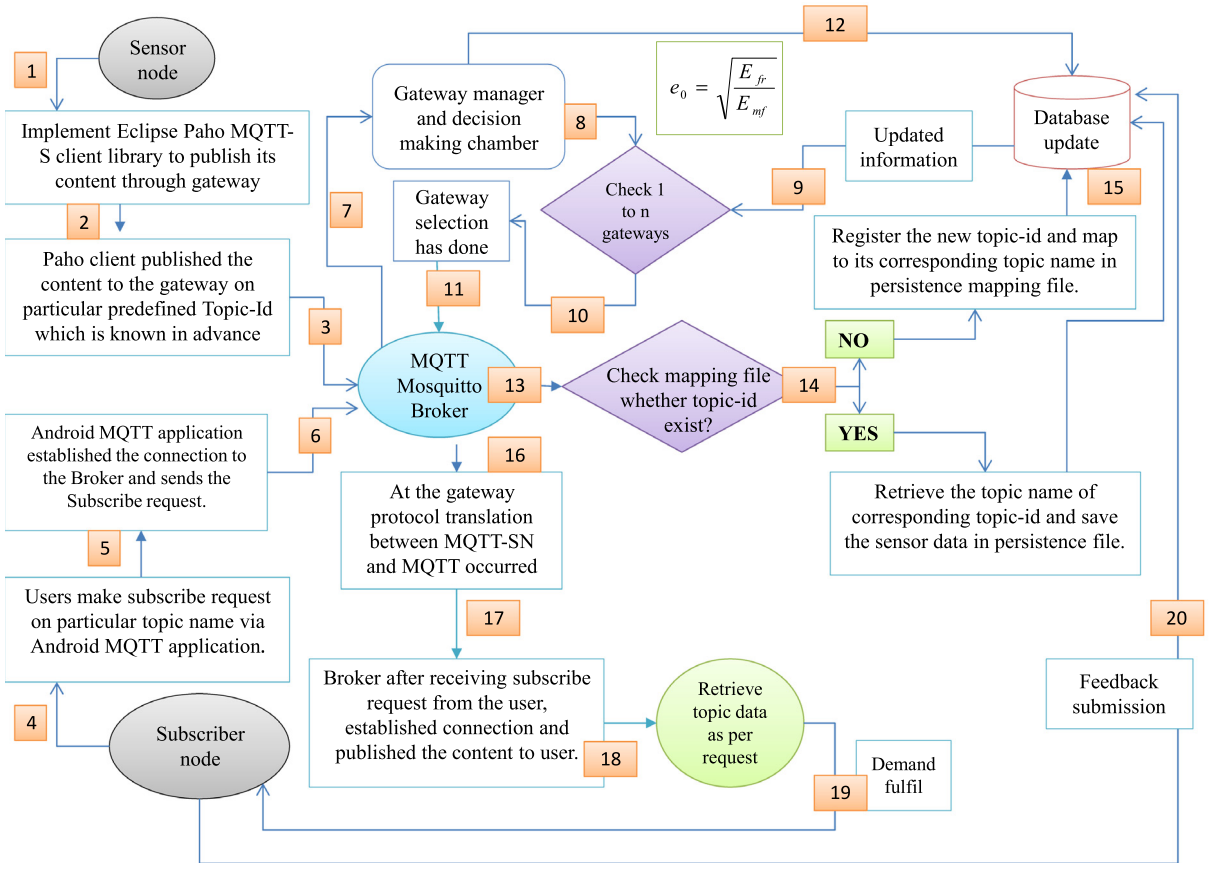


Fig. 7. Workflow of proposed emulation mechanism.

Approach[1]	R. Piyare, S.R. Lee, Towards internet of things (iots): Integration of wireless sensor network to cloud services for data collection and sharing, International Journal of Computer Networks & Communications, 5.5 (2013) 59–72 [19].
Approach[2]	E.G. Davis, A. Calveras, I. Demirkol, Improving Packet Delivery Performance of Publish/Subscribe Protocols in Wireless Sensor Networks, Sensors, 13.1 (2013) 648–680 [5].
Approach[3]	Xu, Zhen, Chuanhe Huang, and Yong Cheng. "Interference-aware QoS routing in wireless mesh networks." Mobile Ad-hoc and Sensor Networks, 2008. MSN 2008. The 4th International Conference on. IEEE, 2008 [30].
Approach[4]	Brannstrom, Robert, ChristerAhlund, and ArkadyZaslavsky. "Maintaining gateway connectivity in multi-hop ad hoc networks." Local Computer Networks, 2005. 30th Anniversary. The IEEE Conference on. IEEE, 2005 [25].
Approach[5]	Our Proposed Load-balancing Dynamic Gateway Selection Method for Static Defined WSN

Fig. 9(a), (b) and (c) has obtained by varying message payload and Fig. 9(d), (e) and (f) has obtained due to different number of packet sending using different approaches at QoS level 0, 1 and 2.

Comparative analysis shows that our proposed method is succeed to eliminate 10%–29% delay for end-to-end data obtained at QoS-0, 14%–28% at QoS level 1 and 7%–23% at QoS level 2, by varying payload of publish request and subscribe request to the broker. Relative study also shows that the delay with particular number in bytes also cut down to 11%–27%, 8%–19% and 6%–23% at QoS level 0, 1 and 2 respectively. Table 7 shows the relative achievement of our proposed method in details.

5.4. Correlation analysis between message-loss and pub-to-sub delay in MQTT-SN QoS level:

The mathematical representation of correlation between message failure and pub-to-sub delay for each quality level has been analyzed, which is shown in Table 6. Correlation coefficient σ value lies in between -1 to 1.

Mean (μ) and Standard Deviation (SD) is calculated by Eqs. (8) and (9) respectively.

$$\mu = \frac{1}{n}\sum_{i=1}^{n}(TR_i - TS_i) \tag{8}$$

$$SD = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(TR_i - TS_i - \mu)^2} \tag{9}$$

TR_i, TS_i are the time when the content was published and the time when the content was received by the consumer end. In Eq. (10), X and Y represent the message loss during n numbers of payload at the time of message publishing and delay to consume by the end user respectively at each QoS level.

$$r = \frac{\frac{1}{n}\sum_{r=1}^{n}(X_r - \overline{X})(Y_r - \overline{Y})}{SD_X SD_Y} \tag{10}$$

With the purpose of analyzing the relationship in between message loss and pub-to-sub delay, we use the mathematical correlation procedure in Eq. (10). $\overline{X}$, $\overline{Y}$ are the average value of X and Y. Standard Deviation value of X and Y are represented by SD_X and SD_Y. Standard Deviation and Correlation Coefficients are shown in Table 6.

Another comparative study has done where it can be shown that error for three level of Quality service are 2.45%, 1.52% and 0.82% comparing pub-to-pub delay for calculative values and real time outcomes by applying the proposed gateway selection method. In Fig. 10, we have validated the experimental

All the data have collected for three different statically defined positions of the sensor nodes. After that the mean values of time

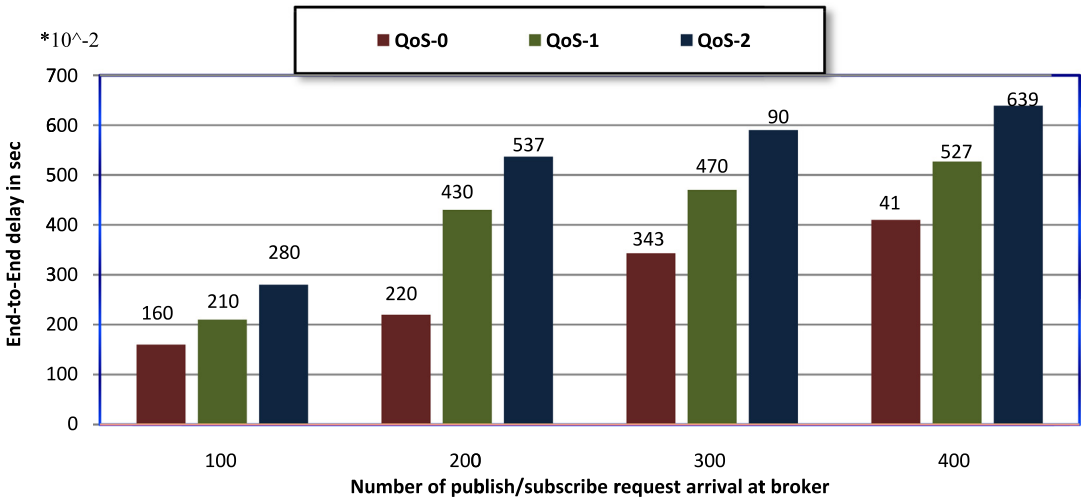


Fig. 8a. Emulation result of pub-to-sub delay by varying number of publish requests at the broker.

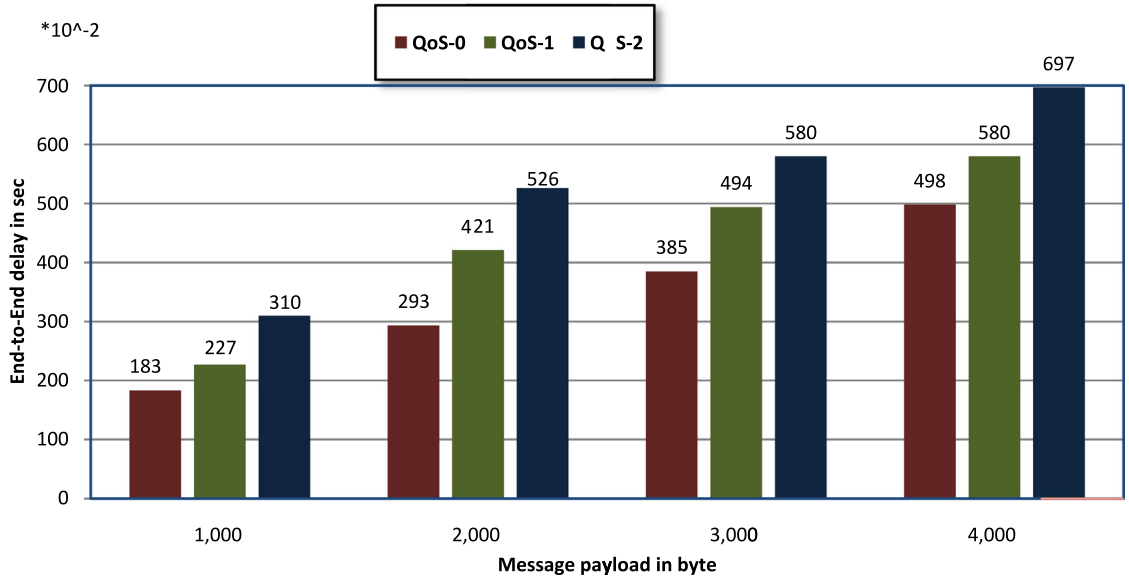


Fig. 8b. Emulation result of pub-to-sub delay by varying payload of the message.

Table 6
The correlation coefficient values in between message loss and end-to-end delay during the transmission, on various levels of QoS, over different payload of the message in bytes which are going to be published to the Broker.

QoS level	Standard deviation		Correlation Coefficient value for Wireless connection in WSN (MQTT-SN)
	SD_X	SD_Y	
QoS-0	7.178	2.959	0.0636
QoS-1	8.992	1.092	0.1039
QoS-2	10.927	0.422	0.1215

delay has considered to be plotted into the graphs. We have then compared our proposed approach respect to other existing methods.

The first case study has occurred through single hop predefined gateway for IoT data publishing [19]. The sensors use MQTT-SN protocol within WSN to publish some real-time sensed data to the MQTT-SN gateway wirelessly by using UDP connection. The gateway acts as a sink node for WSNs directly, it refers to MQTT-SN Gateway, either it is integrated within broker or connected to the broker over wired or wireless TCP connection. Users from the remote location use android applications to get the published data by connecting with the MQTT broker using MQTT protocol over TCP connection to subscribe the sensor data. The end-to-end or pub-to-sub delay is a time taken procedure, to deliver the data by the network, sensed by the sensor nodes to the user-end through the gateway and the MQTT broker/server respectively while keeping important service assurance parameters. The second approach is multi-hop gateway selection [5]. Here a sink node is already assigned and gives security to publish content into the network through broker. The third one is static round trip time (RTT) [30] applied data publish method whereas fourth approach is customized for dynamically choose round trip [25]. All these

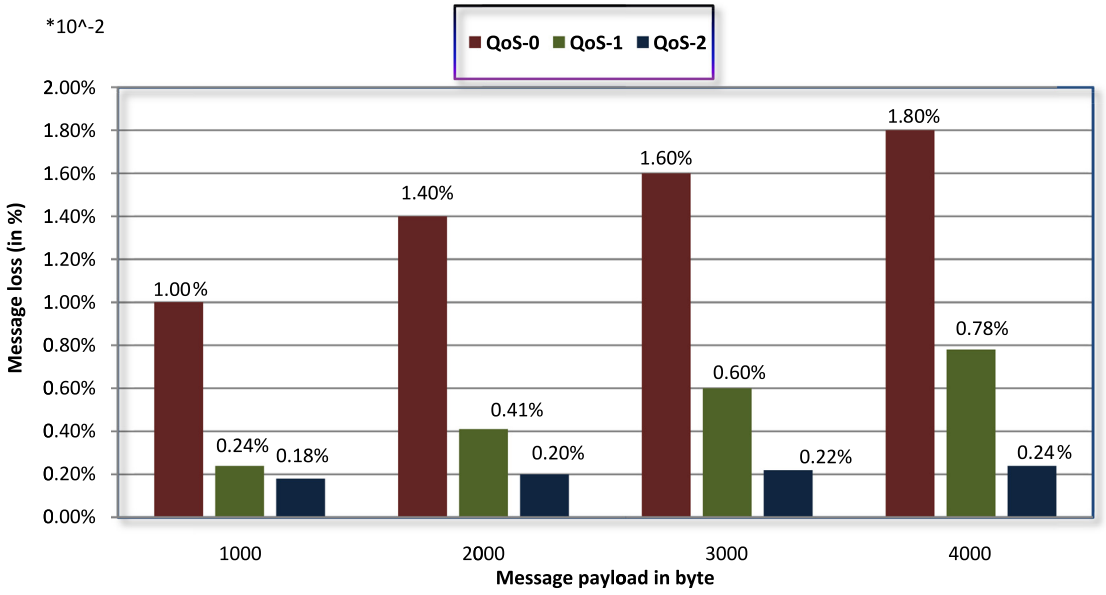


Fig. 8c. Message loss rate in various QoS levels.

Table 7
Comparison of proposed model with existing models of MQTT and MQTT-SN in IoT.

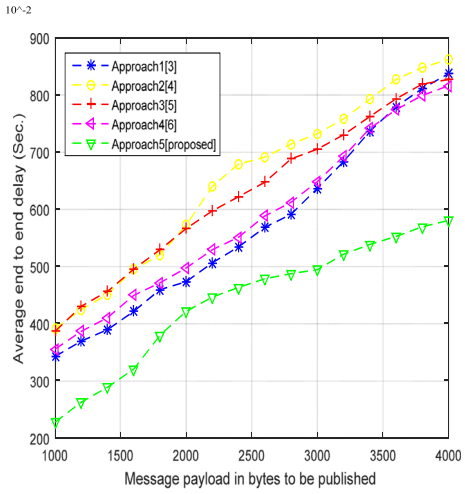
Contributions	IoT MQTT-SN system based on wireless sensor network scenario.	End-to-end delay calculation using open source Mosquitto Broker.	Eclipse-Paho client for MQTT-SN at sensor node side for publishing the content to the Broker.	MQTT-SN Gateway at the sink node forwards the content to broker by performing protocol translation between MQTT and MQTT-SN.	End-to-end delay calculation and message loss are done in all three levels of QoS.	Android MQTT application used to subscribe the content of the sensor node.
Single hop gateway selection [19]	Simple Object Access Protocol (SOAP) and Representational State Transfer (REST)	✔	×	✔	Packet delivery time from source to destination has calculated	✔
Multi-hop gateway selection [5]	[Simulation work done in OMNet platform]	✔	×	✔	[QoS level 0 and level 1 has considered]	×
Static RTT connectivity [30]	✔	×	×	Ns-2 simulation based performance analysis	Success rate of message delivery has calculated	×
Dynamic RTT approach [25]	✔	×	✔	×	✔	×
Dynamic gateway selective mode [Proposed]	✔	✔	✔	✔	✔	✔

	QoS-0	**QoS-1**	**QoS-2**
Reduction of delivery time with increasing of message payload size applying proposed scheme, compared to other existing approaches.	10%–16% [19] 17%–28% [5] 23%–29% [30] 10%–14% [25]	16%–25% [19] 27%–28% [5] 18%–21% [30] 14%–17% [25]	18%–19% [19] Not applicable [5] 20%–23% [30] 07%–13% [25]
Reduction of delivery time with increasing of packet number applying proposed scheme, compared to other existing approaches.	16%–25% [19] 17%–21% [5] 25%–27% [30] 11%–16% [25]	10%–12% [19] 12%–15% [5] 16%–19% [30] 08%–13% [25]	13%–17% [19] Not applicable [5] 14%–23% [30] 06%–09% [25]

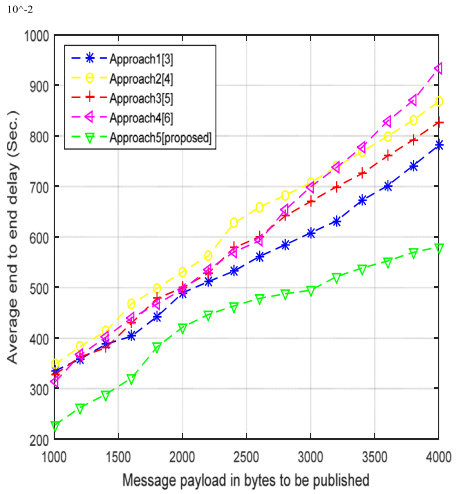
schemes are compared with our proposed solution called as a dynamical selection of the fittest gateway and publishing the data via the same. Detail of these comparison tests has emphasized later in Table 7.

6. Conclusions and future work

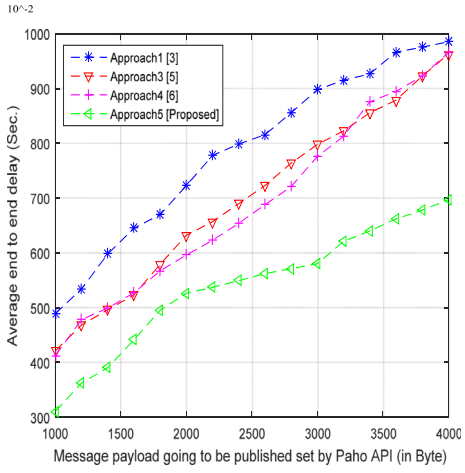
In this paper, we explore well-known MQTT-SN protocol of IoT with a new approach to publish sensor data, called smart gateway selection method. We have proposed a mathematical load balancing model to discover the network using the most recent open source technology. With this proposed model, we have estimated end-to-end content delay and message loss, during the transmission of content in all various levels of QoS. Here we have also shown how the parameters, such a number of publish/subscribe clients and the message payload are influenced by the end-to-end delay and message loss estimation. Simulating sensor node with Eclipse-Paho, MQTT server with Mosquitto broker and Android MQTT application for subscribe the content enable our model to work on real time. As the number of the sensor node and user increased, from the result total time delay far less than a second, so the whole system likely to behave as a real time. Message loss



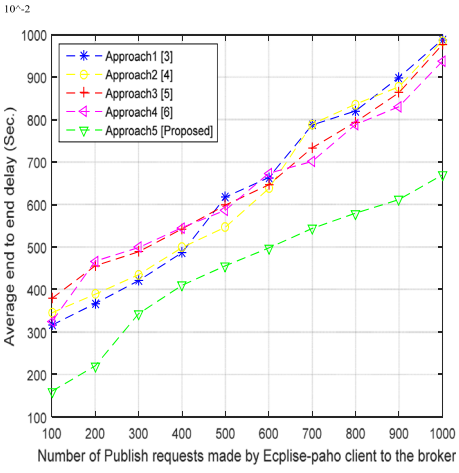
(a) Comparative pub-to-sub delay for QoS level 0 with changing bytes value of publish request to the broker



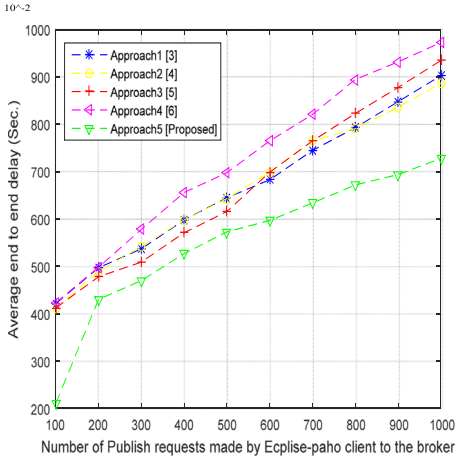
(b) Comparative pub-to-sub delay for QoS level 1 with changing bytes value of publish request to the broker



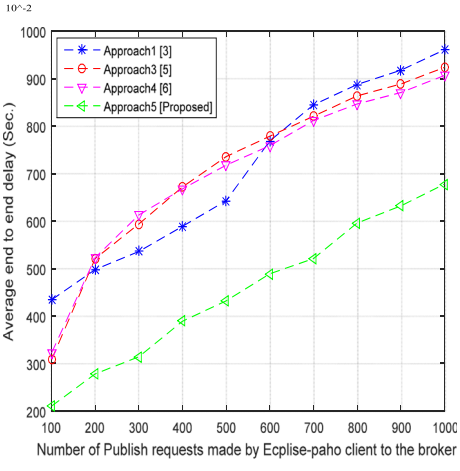
(c) Comparative pub-to-sub delay for QoS level 2 with changing bytes value of publish request to the broker



(d) Comparative pub-to-sub delay for QoS level 0 by varying number of publish requests at the broker



(e) Comparative pub-to-sub delay for QoS level 1 by varying number of publish requests at the broker

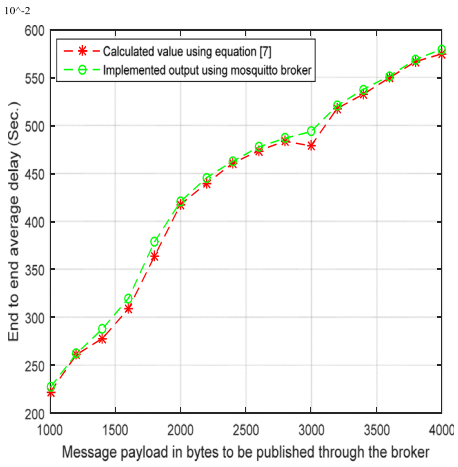


(f) Comparative pub-to-sub delay for QoS level 2 by varying number of publish requests at the broker

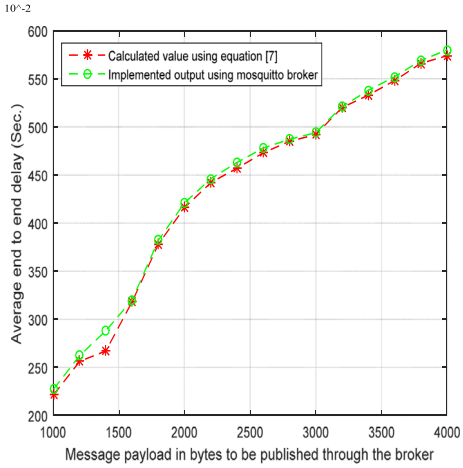
Fig. 9. Comparative pub-to-sub delay for existing methods and proposed dynamic gateway selection method.

during transmission while increasing the message payload is not more than two percent names our model more reliable. In this emulation, we do not consider a delay occurred inside the wireless sensor network during forwarding or routing the data between the nodes. In future, we shall consider time delay estimation inside the wireless sensor network more appropriate.

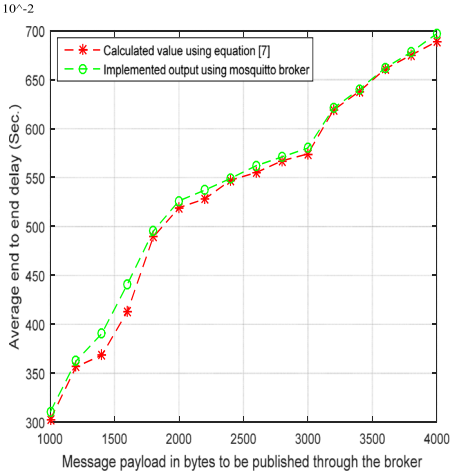
Compared to other existing prototypes MQTT-SN is a preferable prototype in the field of WSN, which is specially made for IoT application in WSN where client's IP addresses are dynamically



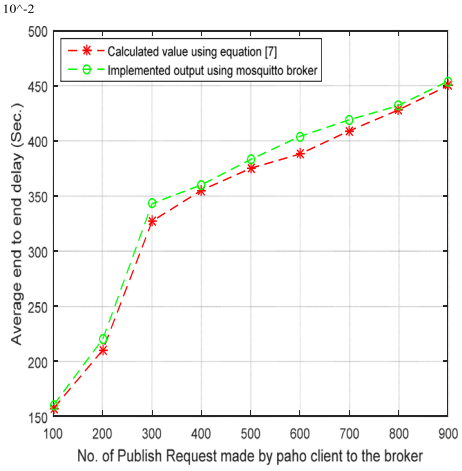
(a) Comparative pub-to-sub delay due to different message payloads for calculative values and real time outcomes by applying the proposed method for QoS level 0



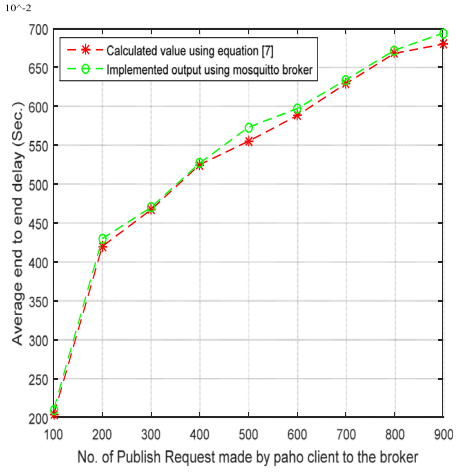
(b) Comparative pub-to-sub delay due to different message payloads for calculative values and real time outcomes by applying the proposed method for QoS level 1



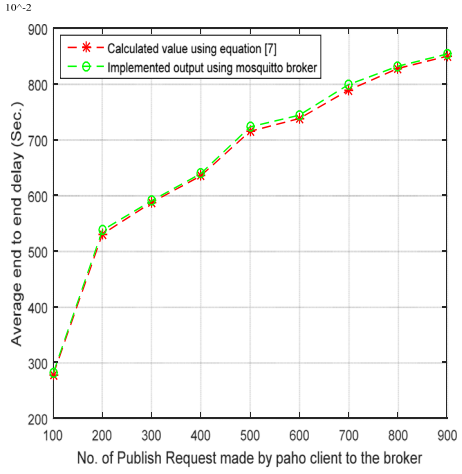
(c) Comparative pub-to-sub delay due to different message payloads for calculative values and real time outcomes by applying the proposed method in QoS level 2



(d) Comparative pub-to-sub delay due to different numbers of packet sending for calculative values and real time outcomes by applying the proposed method in QoS level 0



(e) Comparative pub-to-sub delay due to different numbers of packet sending for calculative values and real time outcomes by applying the proposed method in QoS level 1



(f) Comparative pub-to-sub delay due to different numbers of packet sending for calculative values and real time outcomes by applying the proposed method in QoS level 2

Fig. 10. Comparative pub-to-sub delay for calculative values and real time outcomes by applying the proposed method.

connected with gateways using UDP. In future, we plan to explore topology management along with:

i. Develop machine intelligence-based approach for placement of user nodes within the WSN with respect to the dynamic connectivity to sink.
ii. Design a self-sufficient and energy efficient WSN model for IoT devices in our everyday life.

Acknowledgments

We thank Department of Science and Technology (DST) for sanctioning a research Project entitled "Dynamic Optimization of Green Mobile Networks: Algorithm, Architecture and Applications" under Fast Track Young Scientist scheme reference no.: SERB/F/5044/2012–2013, DST FIST SR/FST/ETI-296/2011 and TEQIP-III under which this research is carried out.

References

[1] Bello Oladayo, Sherali Zeadally, Toward efficient smartification of the Internet of Things (IoT) services, Future Gener. Comput. Syst. (2017).
[2] Wang Tian, et al., Data collection from WSNs to the cloud based on mobile Fog elements, Future Gener. Comput. Syst. (2017).
[3] Masip-Bruin Xavi, et al., Managing resources continuity from the edge to the cloud: Architecture and performance, Future Gener. Comput. Syst. 79 (2018) 777–785.
[4] Perles Angel, et al., An energy-efficient internet of things (IoT) architecture for preventive conservation of cultural heritage, Future Gener. Comput. Syst. 81 (2018) 566–581.
[5] E.G. Davis, A. Calveras, I. Demirkol, Improving packet delivery performance of publish/subscribe protocols in wireless sensor networks, Sensors 13 (1) (2013) 648–680.
[6] Aazam Mohammad, Eui-Nam Huh, Fog computing and smart gateway based communication for cloud of things, in: 2014 International Conference on Future Internet of Things and Cloud (FiCloud), IEEE, 2014.
[7] Varghese Blesson, Rajkumar Buyya, Next generation cloud computing: New trends and research directions, Future Gener. Comput. Syst. 79 (2018) 849–861.
[8] Yao Yi, et al., Admission control in YARN clusters based on dynamic resource reservation, in: Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on, IEEE, 2015.
[9] Rho Seungmin, Yu Chen, Social Internet of Things: Applications, architectures and protocols, 2018, pp. 667–668.
[10] Raza Shahid, et al., SecureSense: End-to-end secure communication architecture for the cloud-connected Internet of Things, Future Gener. Comput. Syst. 77 (2017) 40–51.
[11] S. Wagle, Semantic data extraction over MQTT for IoTcentric wireless sensor networks in International Conference on Internet of Things and Applications (IOTA), 2016.
[12] Zhu Qian, et al., Iot gateway: Bridging wireless sensor networks into internet of things, in: Embedded and Ubiquitous Computing (EUC), 2010 IEEE/IFIP 8th International Conference on, IEEE, 2010.
[13] A. Stanford-Clark, H.L. Truong, Mqtt for sensor networks (mqtt-sn) protocol specification in International business machines (IBM) Corporation version, 2013.
[14] Lin Chuan, et al., Dynamic power management in new architecture of wireless sensor networks, Int. J. Commun. Syst. 22 (6) (2009) 671–693.
[15] Z. Chen, Y. Xiao, X. Li, R. Li, A clustering protocol for wireless sensor networks based on energy potential field, Sci. World J. (2013) 1–7.
[16] Sharma Gaurav, Varsha Sahni, Performance analysis of energy efficient clustering protocolusingTabuin wsn, Int. J. Sci. Manage. Technol. (IJSMT) 15 (2018).
[17] Sankarasubramaniam Yore, Ian F. Akyildiz, S.W. McLaughlin, Energy efficiency based packet size optimization in wireless sensor networks, in: Sensor Network Protocols and Applications, 2003. Proceedings of the First IEEE. 2003 IEEE International Workshop on, IEEE, 2003.
[18] Doudou Messaoud, Tifenn Rault, Abdelmadjid Bouabdallah, Efficient QoS-aware heterogeneous architecture for energy-delay constrained connected objects, in: Wireless and Mobile Networking Conference (WMNC), 2016 9th IFIP, IEEE, 2016.
[19] R. Piyare, S.R. Lee, Towards internet of things (iots): Integration of wireless sensor network to cloud services for data collection and sharing, Int. J. Comput. Netw. Commun. 5 (5) (2013) 59–72.
[20] Tapiwa M. Chiwewe, Gerhard P. Hancke, A distributed topology control technique for low interference and energy efficiency in wireless sensor networks, IEEE Trans. Ind. Inf. 8 (1) (2012) 11–19.
[21] Yang Zhengyu, et al., AutoReplica: automatic data replica manager in distributed caching and data processing systems, in: Performance Computing and Communications Conference (IPCCC), 2016 IEEE 35th International, IEEE, 2016.
[22] A.A. Chandra, Y. Lee, B.M. Kin, S.Y. Maeng, S.H. Park, S.R. Lee, Review on Sensor Cloud and Its Integration with Arduino Based Sensor Network in International Conference on IT Convergence and Security (ICITCS), 2013.
[23] P. Zhang, H. Sun, Z. Yan, A Novel Architecture Based on Cloud Computing for Wireless Sensor Network in 2nd International Conference on Computer Science and Electronics Engineering (ICCSEE), 2013.
[24] A. Botta, W.D. Donato, V. Persico, A. Pescapé, Integration of cloud computing and Internet of Things: A survey, Future Gener. Comput. Syst. 56 (2016) 684–700.
[25] Brannstrom Robert, Christer Ahlund, Arkady Zaslavsky, Maintaining gateway connectivity in multi-hop ad hoc networks, in: The IEEE Conference on Local Computer Networks, 2005. 30th Anniversary, IEEE, 2005.
[26] http://www.eclipse.org/paho/clients/c/embedded-sn/.
[27] https://eclipse.org/paho/clients/android/.
[28] http://mosquitto.org/documentation/.
[29] W. Xinhua, W. Sheng, Performance Comparison of LEACH and LEACH-C Protocols by NS2 in Ninth International Symposium on Distributed Computing and Applications to Business, Engineering and Science, 2010.
[30] Xu Zhen, Chuanhe Huang, Yong Cheng, Interference-aware QoS routing in wireless mesh networks, in: The 4th International Conference on Mobile Ad-Hoc and Sensor Networks, 2008. MSN 2008, IEEE, 2008.

Deepsubhra Guha Roy is currently pursuing his Ph.D in the field of Mobile Cloud Computing. He is the founder developer of Centre of Mobile Cloud Computing in West Bengal University of Technology. His research area is QoS aware offloading strategies in mobile cloud computing.

Bipasha Mahato is currently pursuing her Ph.D in the field of Mobile Cloud Computing. Her research area is QoS analysis in mobile and cloud computing.

Debashis De (M'13-SM'15) is a Professor and presently Head of the Department of Computer Science and Engineering of Maulana Abul Kalam Azad University of Technology, India and Adjunct Research Fellow of University of Western Australia, Australia. His research area includes energy and latency optimization in mobile cloud computing. He has received Young Scientist award both in 2005 at New Delhi and in 2011 at Istanbul from International Union of Radio Science, H. Q., Belgium.

Rajkumar Buyya is a Redmond Barry Distinguished Professor and Director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory at the University of Melbourne, Australia. He is also serving as the founding CEO of Manjrasoft, a spin-off company of the University, commercializing its innovations in Cloud Computing. He has authored over 625 publications and seven text books including "Mastering Cloud Computing" published by McGraw Hill, China Machine Press, and Morgan Kaufmann for Indian, Chinese and international markets respectively. Dr. Buyya is recognized as a "Web of Science Highly Cited Researcher" in 2016 and 2017 by Thomson Reuters, a Fellow of IEEE, and Scopus Researcher of the Year 2017 with Excellence in Innovative Research Award by Elsevier for his outstanding contributions to Cloud computing. For further information on Dr.Buyya, please visit his cyberhome: www.buyya.com.