Received: April 25, 2018

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Development and Testing of Message Scheduling Middleware Algorithm with SOA for Message Traffic Control in IoT Environment

Poonam Gupta[1]* **Kopparti Veera Venkata Satyanarayan[1]** **Dilip Devchand Shah[2]**

[1]*Koneru Lakshmaiah Education Foundation, Vaddeswaram, Guntur, Andhra Pradesh, India*
[2]*Imperial College of Engineering and Research, Pune, Maharashtra, India*
* Corresponding author's Email: poonam77gupta@gmail.com

Abstract: Nowadays, in most of the applications, Internet of Things (IoT) is being used, multiple clients are indirectly connected to sensing devices through messaging broker and their interaction happens through message exchanger. Day by day number of clients and sensing devices are increasing and subsequently message traffic management becoming key research area. In case of overloaded message traffic, IoT broker system faces delayed messaging and so sensor activation and response gets delayed. With this identified problem, present research focused on development of new middleware (broker) architecture for IoT which can handle message queue more efficiently. With changing requirement trends, traditional service-oriented architecture (SOA) model has many architecture design issue. Traditional SOA does not support multiple protocol request communication. Hence, this paper focuses on the design of the message scheduling broker for IoT environment with hybrid protocol routing e.g. Extensible Messaging and Presence Protocol (XMPP), Constrained Application Protocol (CoAP), MQ Telemetry Transport (MQTT) protocols considered like one request and it is filtered by broker based on their calculated priority. Additionally, proposed scheduling algorithm uses priority queue model which is considered as a reference messaging model. Multiple queues model is developed to increase the efficiency of the algorithm. Such middleware provides a solution for message delay issue with multiple protocol request handling. The system is tested for MQTT, XMPP and CoAP message protocols with respect to service time and MQTT protocol performance are found more efficient during IoT sensor test using Apache HIVE testing platform as Apache HIVE is a data warehouse software project for providing data summarization, query and analysis.

Keywords: Internet of things (IoT), Message scheduling, Service oriented architecture (SOA), IoT broker, Priority queue, MQ telemetry transport (MQTT), Extensible messaging and presence protocol (XMPP), Constrained application protocol (CoAP), Quality of service (QoS).

1. Introduction

The purpose of middleware is to deal with sensor data, manage a sensor request, and provide short-term information storage while using the present sensor information [1]. The most crucial one concern is the protocol's "interoperability". The author designed the 'negotiation protocol' that is a group of policies managing the connection amongst brokers which is often initiators as well as contractors dynamically [2]. Determining the most efficient activity models which enhances the broker source utilization is necessary. By middleware message scheduling system can generate the most effective response delay for the requests attained by the brokers. Different author classified the IoT protocols in four major types as: application protocols, assistance breakthrough protocols, infrastructure protocols and influential protocols. However, these kind of protocols need to be designed jointly to deliver SOA supporting IoT application.

Core disadvantages of existing systems are: the need of specific resource, no support for multiple protocol system, Lack of multi-sensor support for SOA, Message scheduling of middleware need to be enhanced for multiple (hybrid) protocol handling.

Hence, over the existing systems, the proposed system provides advantages in terms of the following functionalities:

SOA-IoT interoperability: The proposed system developed the new middleware for SOA which supports IoT systems with multiple sensor support. Nowadays, various companies require multi-functionalities like security systems, machine controls, data center management, energy efficient scheduling. For such applications, companies preferring multitasking control system which can be monitored with central monitoring facility. In proposed system the auto-sensor systems receive requests from sensors and activation sensors responds via SOA with newly developed middleware.

Multiprotocol support: Many existing IoT systems are available but prefer support for MQTT protocol. MQTT protocol is considered an efficient protocol for IoT systems. SOA supports HTTP, XMPP etc. Hence, to support MQTT, CoAPIoT protocols along with HTPP, XMPP protocols proposed system designed hybrid protocol support system.

Elimination of EBS: Enterprise Service Bus (ESB) is mandatory for existing systems to carry request and responses. Due to EBS, message traffic management becomes rigid and the possibility of message delay occurs. To eliminate response delays, a proposed system developed multiple queues model (BrokerMessage algorithm) with priority decentralization concept which is presented as Hybrid Execution of Messaging in section-5.

QoS Performance: The response time is considered as a QoS parameter. Minimum response time is necessary to count any system as an efficient system. Existing systems solely supports either SOA or IoT. But, the proposed system supports minimum response time for sensor activation via new multiple queue scheduling models. This has been tested for live sensor system with HIVE server for MQTT, XMPP, CoAP protocols and discussed in section-5 of this paper.

In summary, this paper presents a systematic study of recent researches and explains conventional Service-Oriented architecture (SOA). When compared with existing literature reviews to design an efficient system that addresses the most significant challenges, this paper makes several distinguished contributions, including message/protocol request sorting and message routing. The paper explores various approaches based on SOA and middleware architecture to highlight possible solutions for IoT messaging challenges. Section 2 presents significant literature review and research gap identification. Section 3 discusses the traditional SOA and limitations of traditional SOA and existing research methods. Section 4 provides the middleware messaging architecture developments for IoT with scheduling model. Section 5 provides computational experiment and BrokerMessage algorithm modeling and performance testing of proposed work. Finally, Section 6 concludes the paper with the future scope.

2. Identification of research gap

This section summarizes existing methods developed for service oriented architecture, message scheduling middleware, protocol study and internet of things (IoT) implementation. The primary literature collected about 185 from Springer, Elsevier, IEEE, Hindawi and ACM databases for period 2014 to 2017. Out of those, 41 papers identified as relevant references for SOA and IoT study. Finally, to identify key research gaps 16 papers are referred. 13 papers are chosen for functionality comparison as depicted in following Table 1.

Table 1. Existing research

Sr. No	**Author/Method Used/Remark**
1	Author: Qiang, Bao-Hua, et al.[1]
	Method Used:Author suggested a SOA message scheduling design for one protocol service with the addition of an overall control queue among service consumers and providers, the high-priority service scheduling reconciled.
	Remark:This technique is made for one protocol assistance and multiple protocol assistance techniques ought to be developed. Additionally, IoT is not examined together with SOA.
2	Author: Da Xu, Li, Wu He, and Shancang Li[2]
	Method Used: Authors analyzed the architectures and also technology for bringing in distributed business programs, highlighted their particular benefits and flaws, and also acknowledged investigation developments along with options within this progressively crucial area.
	Remark: The application of middleware in distributed programs is effectively discussed. Furthermore, SOA-oriented integration environment employing Enterprise Service Bus (ESB) give an ensuring and important platform for inter-enterprise integration.
3	Author: Calvaresi, Davide, et al.[3]

	Method Used: Internal broker schedulers, communication middleware, and negotiation protocols are recognized as co-factors suppressing the real-time concurrence. Agents' communication middleware have been designed.
	Remark: This particular paper offers an evaluation of this kind of Multi-Agent Systems (MAS) elements and also paves the trail for accomplishing the MAS conformity having rigorous timing restrictions, hence cultivating trustworthiness and predictability.
4	Author: Asghar et. al.[4]
	Method Used: Author recommended the method to enhance the power conserving in MQ-service technique. MQTT or MQ Series elements are utilized to tie up linked approach various other software applications in order to operate connected way.
	Remark: This kind of application is generally known as enterprise integration software or middleware. However, this is often useful for protocol specific IoT systems.
5	Author: Al-Fuqaha, Ala, et al.[5]
	Method Used: Author investigated the relationship between IoT along with other promising technologies such as big data analytics and also cloud and fog computing. Additionally, the necessity for superior horizontal integration amongst IoT products and services is examines.
	Remark: Comprehensive service use cases introduced to demonstrate the fact that diverse protocols introduced within the paper can assist collectively to produce preferred IoT services.
6	Author: Happ, Daniel, et al. [6]
	Method Used: Author examined the ideal sustainable throughput and also delay within practical load circumstances applying traces through actual sensors. The examined XMPP methods differ within their blocking functionality, semantic ensures and encoding.
	Remark: This assessment shows that these dissimilarities can offer a significant effect on throughput and delay of cloud-based IoT platforms. Consequently, greater message scheduling improvement is essential.
7	Author: Jiang, Zheng, et al.[7]
	Method Used: Author looked into the spatial degree of freedom of IoT devices dependant on their particular distribution, after which it provides the multiuser shared access (MUSA) that is amongst the standard MUST strategies to spatial area.
	Remark: However, this facilitates simply one protocol and multiple protocol scheduling anticipated to promote the improvement of 5G cellular networks and also needs the productive assistance for numerous simultaneous short message devices.
8	Author: Yaqoob, Ibrar, et al.[8]
	Method Used: Author researched, highlighted, and also reported leading analysis developments produced in IoT architecture recently. Then author classified and grouped IoT architectures and formulatedtaxonomy according to significant variables like purposes, empowering technology, organizational goals, architectural prerequisites, network topologies, and also IoT podium architecture forms.
	Remark: Author determined and discussed the main element prerequisites for upcoming IoT architectures.
9	Author: Machorro-Cano, Isaac, et al.[9]
	Method Used: This book chapter is to provide the effective use of the IoT in the profession, explaining its program areas, platforms and numerous research cases.
	Remark: Author introduced a evaluation analysis of the research cases, along with the developments and issues of the IoT as outlined by each and every domain of application.
10	Author: Gil-TakOh et. al.[10]
	Method Used: Author recommended a DDS (Data Distribution Service) dependent CoAP (Constrained Application Protocol) adaptor so as to resolve the issue with the DDS middleware dependent interoperable system whenever used in combination with additional availability or resource-constrained devices.
	Remark: Utilizing the CoAP adaptor, technique offered a data transfer assistance where in preceding studies were to be known as difficult to access.
11	Author: Kim, Hong Jin, et al.[11]
	Method Used: The contribution of author technique is in discovering solutions that have not necessarily been discovered by preceding approaches.
	Remark: Author developed a system that incrementally contributes dimensions to split up services till all services are determined.
12	Author: Albano, Michele, et al.[12]

	Method Used: Author applied the QoS Set up along with the Monitor services, it is employed to validate and configure QoS within the local cloud, as well as for on-line monitoring of QoS.
	Remark: Paper explains how a QoS Setup and also Monitor services are offered within a Arrowhead-compliant Technique. This technique cannot be employed for SOA with IoT systems. Consequently, Hive testing should be employed for IoT system testing.
13	Author: Hachem et. al.[13]
	Method Used: The author used decoupling the sensing/actuating tasks through the querying for measurements and also requests for actions.
	Remark: As this research limited up to sensing and actuating tasks through queries, future development can be in the direction of multiple protocol handling. Hence, proposed research is focused to develop multiple protocol handling and processing of sensor requests.

Determined by literature analyze regarding existing methods, we recognized the suggested system design need. As there is no research accomplished in the region of hybrid protocol controlling middleware with SOA for the internet of Things (IoT), we acknowledged this as a significant research gap.

3. Existing service oriented architectures

A service-oriented architecture (SOA) is a type of software design whereby services are offered for the other elements by application aspects, through the transmission protocol using a network. The essential key points of service-oriented architecture are independent of clients, products, and services along with technologies.

3.1 Single-protocol service oriented middleware: MobIoT

With intension to develop the solution for research gaps identified in articles [1 - 13] which directs to develop the key trust area of IoT known as "multiple protocol sensor support",proposed work is compared with MobIoT developed by Hachem et. al. [13]. Author designed "MobIoT" that explores probabilistic approaches, according to semantic knowledge to aid interoperability and accomplish users' requests for Thing-based measurements/actions.

The author also focused on the mobile area of the IoT where SOA performs a task of middleware. MobIoT decouples the sensing/actuating tasks through the querying for measurements and also requests for actions. In MobIoT middleware, human interation for raising query is essential. Which in turn must be removed in future development for pure sensory systems. MobIoT was built to transparently provide the functionalities necessary through the suggested Thing-based SOA.

To handle heterogeneity issues like protocol assistance, it is common practice to utilize their particular meta-data, framework details or services. However, not very much effort was focused in the direction of supplying information that assessed by sensors which are at the core of the IoT.

Disadvantages of MobIoT are identified as follows:

- ✓ Human interaction required to input request
- ✓ Wireless network required for activation via mobile which may delay response due to network traffic.
- ✓ Use of ontology is necessary which means, predefined criteria/rules need to be fixed and no random operation executed as per priority.
- ✓ No priority scheduler, hence request collision is possible
- ✓ There is no broker system available with MobIoT, as 'unknown topology' concept used.
- ✓ Hence to eliminate such demerits, proposed multi-protocol service-oriented middleware is developed with BrokerMessage Scheduling to achieve automatic sensor request/response without the need for Enterprise Service Bus.

Asper research gaps identified in section-2, there's a robust requirement for the development of new middleware technique which can make SOA much more ideal for IoT systems or large systems. Consequently, we developed new message queue algorithm "BrokerMessage" with reference to priority queue model and is also mentioned in section 5 of this paper.

4. Computational experiments and modeling of proposed system

For the development of new SOA messaging system for the Internet of Things (IoT), for present research request-response model using MQTT, XMPP and CoAP protocols are considered. The application level scenario is, first system (or client) send the request using any of such protocols which will be forwarded to middleware broker. (For example, System will send security alert signal and the siren will sound. This siren will further forward request protocol to remote server broker to activate camera system or video capturing system.) As

heterogeneous client request can arrive at the broker and every request needs to get the response from the variety of IoT devices (sensors), hence it is necessary to develop SOA architecture with multiple protocol support. So, proposed architecture is developed to handle such hybrid (multiple) protocols (Refer Fig. 1 for the flow of execution).

Further, when the number of clients increases, number of messages (with variable protocol types) also increases and proportionately leads to huge traffic at server/broker end. To make protocol request response model efficient, real-time incoming messages must be scheduled properly to avoid huge traffic at server/broker end (Apache JMeter for performance testing of message scheduling is discussed in the section-5). In proposed system messages are divided into number of classes. Each class has arrival rate λ_n and service rate μ_n where *n=1,2….r*. Every class has traffic intensity denoted by ρ_n.

$$\rho_n = \frac{\lambda_n}{\mu_n} \tag{1}$$

So overall traffic intensity of the system is denoted by σ_n

$$\sigma_n = \sum_{i=1}^{n} \rho_i \sigma_0 = 0, \ \ \sigma_n = \ \rho \ , \rho < 1 \tag{2}$$

Every message of each class is represented by triplet *M (P, S, U)*where P – Requesting Period, S – An Average Service Time, U – Maximum Responding Period

The message for n[th]type of device is represented by $M_n(P_n,\ S_n,\ U_n)$ where P_n is requesting period, S_n is average service time, U_n is maximum responding period. Now arrival rate and service rate for n[th] device message are represented byλ_n, $\mu_n$respectively.

$$\lambda_n = \frac{1}{P_n} \tag{3}$$

$$\mu_n = \frac{1}{S_n} \tag{4}$$

Andtraffic intensity $\rho_n = \frac{\lambda_n}{\mu_n} = \frac{S_n}{P_n}$

Overall traffic intensity ρ will be

$$\rho = \ \sigma_r = \sum_{n=1}^{r} \frac{\lambda_n}{\mu_n} = \sum_{n=1}^{r} \frac{S_n}{P_n} < 1 \tag{5}$$

Initially $P_n = U_n$ for *n = {1, 2, ….., r}*

If overall traffic intensity is less than one then requesting period will be increased by $\frac{U_n}{2^n}$ for all classes *n = {1, 2, … , r}* and once again traffic intensity of each type is calculated and all types ofmessages are rearranged. This procedure is repeated until we get overall traffic intensity less than one. The standard waiting amount of time in the queue for each and every kind of message is considered as an indication of system effectiveness [14, 15].

We have realized if the number of messages is increasing progressively at broker end then using single queue for scheduling is limiting the efficiency of system so the scheduling algorithm is modified for 2-queue and 3-queue algorithms are given below.

We have proposed improvement in earlier scheduling algorithm[16].We have used M/M/n queues over M/M/1 [16]queue where we have tested our system for *n=2 and n= 3*.

4.1 Message scheduling algorithm using single queue reference algorithm:

$M_n(P_n, S_n, U_n)$ Property of messages for n[th]class
i) read (number of classes n);
ii) ρ=0; // ρ is the overall traffic intensity
iii) for*i = 1 to n*
{
readU_i, S_i;
$P_i = U_i$;
$\rho_i = \frac{S_i}{U_i}$;
$\rho = \rho + \rho_i$;
}
iv)while*(ρ > 1)*
{
// rearrange M_n in descending order of ρ_n
For *i = 1 to n*
{
$P_i = P_i + \frac{U_i}{2^i}$;
$\rho_i = \frac{S_i}{P_i}$;
$\rho = \rho + \rho_i$;
}
}

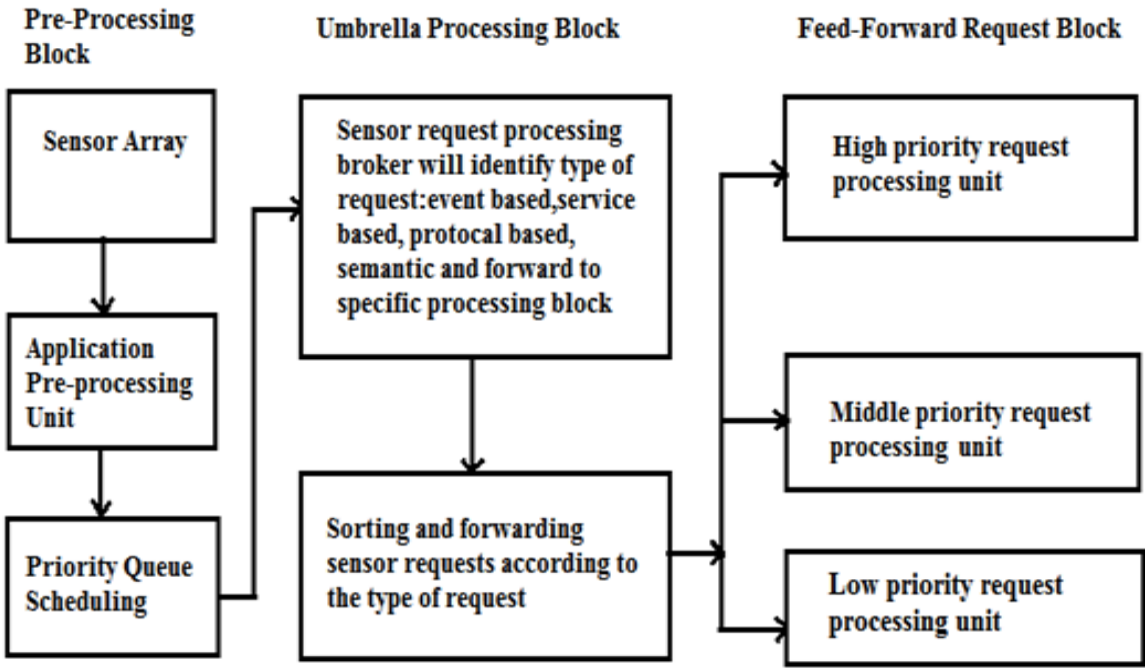

Figure. 1 Hybrid execution of messaging model

4.2 The modified message scheduling algorithm with two queues:

$M_n(P_n, S_n, U_n)$ property of messages for n[th]classof first queue

$M_{n1}(P_{n1}, S_{n1}, U_{n1})$ property of messages for n[th]class of second queue

i) read (number of classes n);

ii) $\rho = \rho_1 = 0$; // ρ, ρ_1 are the overall traffic intensities of first and second queue respectively

iii)for*i = 1 to n*

{

readU_i, S_i, U_{i1}, S_{i1};

$$P_i = U_i;$$
$$P_{i1} = U_{i1}$$
$$\rho_i = \frac{S_i}{U_i};$$
$$\rho_{i1} = \frac{S_{i1}}{U_{i1}};$$
$$\rho = \rho + \rho_i;$$
$$\rho_1 = \rho_1 + \rho_{i1};$$

}

iv)while (ρ*>1 or* ρ_1*> 1*)

{

Rearrange M_n, M_{n1} in descending order of ρ_n, ρ_{n1}

For*i = 1 to n/2*

{

$$P_i = P_i + \frac{U_i}{2^i};$$
$$\rho_i = \frac{S_i}{P_i};$$
$$\rho = \rho + \rho_i;$$
$$P_{i1} = P_{i1} + \frac{U_{i1}}{2^i};$$
$$\rho_{i1} = \frac{S_{i1}}{P_{i1}};$$
$$\rho_1 = \rho_1 + \rho_{i1};$$

}

}

Based on above modeling, system block flow is identified and integrated for hybrid protocol messaging. The IoT Pre-processing block identifies groups of functionalities in a hybrid messaging i. e. Event-based function, service-based, database oriented and semantic-driven, and application dependent.

The present service oriented messaging architecture with BrokerMessage algorithm for IoT brings out new SOA messaging architecture for controlling the complexity of request protocol messages for multitasking of several types of application sensors in even more homogeneous approach.

Umbrella processing block is a heart of architecture as this block works as a primary broker and holds all prioritized and sorted request protocols according to type and priority key-value. Fig. 1 shows hybrid execution of messaging queue. Further, this process block provides input to Feed Forward request block where BrokerMessage Algorithm executes the internal processing. BrokerMessage Algorithm steps are as follows:

BrokerMessage Algorithm (referred as iBroker for live testing with Apache HIVE server)

1) Locate sensor request data and store in arraySensArr[];
2) Store available request types available in system in array SysReq[];
3) Arrange sensor request protocol in key-value pair ProtArr[key,val];
4) Identify request type from SysReq[] and store type of received request in array ReqType[];

5)if count of ReqType[] != SysReq[];
6)repeat step 4
7) Execute queue 'Rq' with FIFO request strategy and forward to feed forward processing unit for further processing
8) ifRq != count ReqType[];
9) repeat step 7
10) else wait for next protocol request and repeat step 3

This algorithm is further tested with HIVE server for performance evaluation in section 5.

5. Result and discussion

The aim of proposed system development is to increase the interopretability of MQTT, XMPP and CoAP protocols which is discussed in previous sections. To show the how efficiency of system is increased by increasing interoperability amongst protocols, proposed systems are compared with article/ reference [13] and comparison is shown in Fig. 10.

Also, proposed system can handle multiple protocol requests at a time which was not possible in existing research [13]. The proposed architecture is very advantageous in terms of IoT application where many sensors used, Cloud of Things (CoT) is being used (which include requests from diversified communication protocols) because of hybrid protocol communication is developed.

Hence, effectiveness of proposed research is wide in terms of successful and efficient communication where multiple protocols communication occurs. In fact, using proposed system many IoT applications can be combined to provide a generalized solution. As an end product, user can provide requirements of application and vendor can add or remove IoT (sensor) facility for client without any need of architecture level or protocol specific modification. In short, proposed system is a single solution for all types of IoT application irrespective of protocol types.

We have compared the results of proposed system with the references [16, 13, 17] on various aspects and comparisons are shown in Figs. 8, 10 and Table 2, respectively.

For illustration, we considered case of fire alert security system along with proposed architecture and results were compared with reference to project specified in reference [17]. For very high profile building like parliament building or laboratories/documentation library etc. For confidential data protection, it is always necessary to provide automatic security contingency plan where unauthorized manual interference must be negligible [18]. Hence, we suggested a fully automated system where automation server managed at the remotely accessed central location.

In our security system, the security alert sends signals to all sensors along with the remotely managed server to activate SOA middleware processing. As soon as fire alarm triggers the request, request protocol sent to the server. Further, Umbrella Processing Unit (UPU) receives signals from requested protocol(s). Here, Umbrella Processing Unit works as a primary broker and verifies the type of incoming data protocol. Further, it logically stores all received data protocols according to the order as it received. In this process, priorities are set as per the severity of application-level issues. Consider the event where alert is activated for a smoke detector. As it is necessary to retrive data about the reason for the fire which further needs another system to identify whether the fire is because of the electrical short circuit or due to the burning of the material. If the fire is due to short circuit, the sensor should not trigger signal for water shower and must trigger signal to chemical fog [19]. If the fire is due to burning of material, then onesensor must keep eye on water level of the tank. Again, if water tank gets empty then sensor must send a request for main water storage plant to drain water to working water tank. Also, sensors that capture the images or records the video must send all image/video data to the server. So, the system becomes complicated with all sensory network and various types of request protocols.

Present research protocols (i.e. MQTT, XMPP, CoAP) tested using HIVE [12] as a third party evaluation tool, as the scope of present work is limited to middleware framework development, we used third-party sensor network for testing. Though Apache HIVE server uses MQTT as a primary protocol [20], proposed research used protocol piping concept to test XMPP and CoAP protocols as a part of hybrid protocol execution. For result

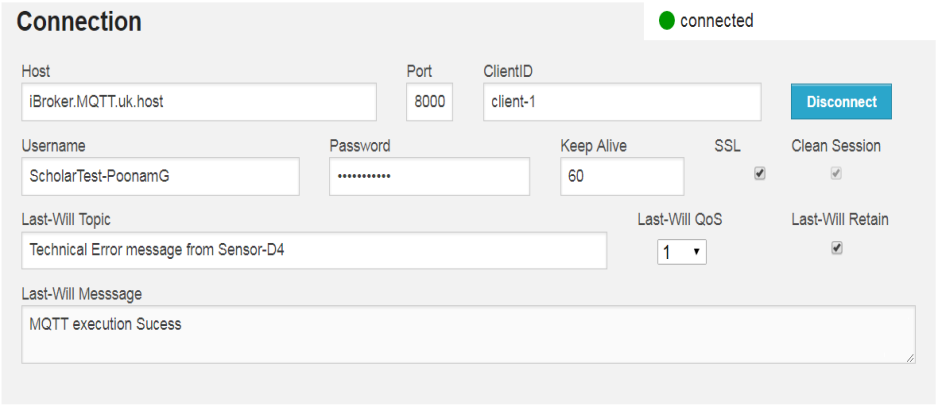

Figure. 2 Testing of MQTT protocol

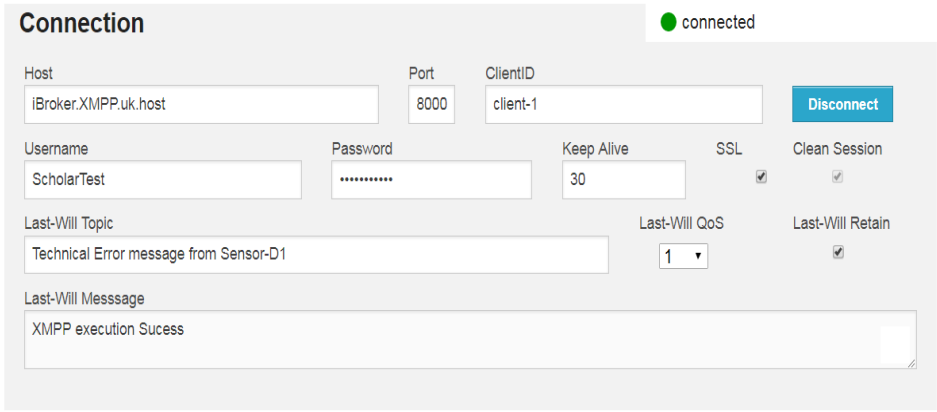

Figure. 3 Testing of XMPP protocol

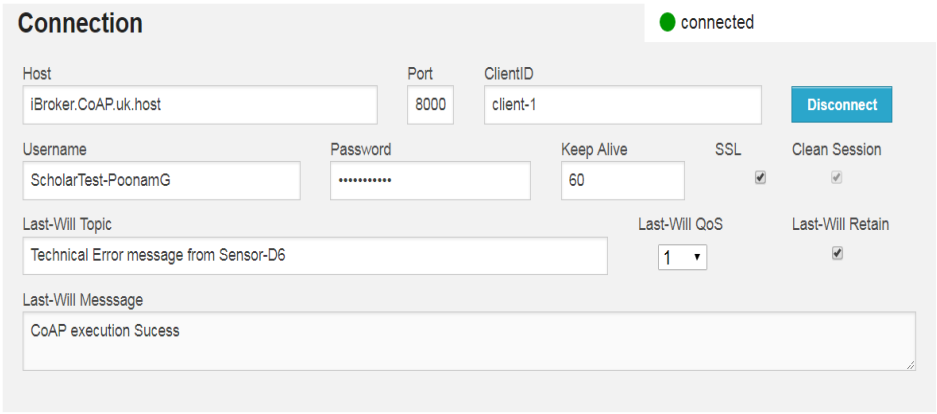

Figure. 4 Testing of CoAP protocol

comparison and testing purpose, we embedded iBroker (Algorithm) and framework (depicted in Fig. 1) for online testing of iBroker request routing and results were compared against the existing work referred from article/reference [17]. Out of 12 sensor networks, we considered sensors compatible for MQTT, XMPP and CoAP protocols for testing purpose. For MQTT-Sensor-D4, XMPP-Sensor-D1 and CoAP-Sensor-D6 are processed.

The connection created using port 8000 with iBroker.MQTT.uk.host and initiated clientID-client-1 as shown in Figs. 2 - 4. The QoS index is assumed 1 and connection session will remain alive for 60 seconds.

We enabled Secure Sockets Layer (SSL) to make protocol transfer more secure with clean session which enable to make buffer free of data. Lastly, retain option keeps records of sensor execution. Further, protocol success message will be displayed after successful connection with the host. Fig. 2 represents MQTT protocol; Fig. 3 shows the connection for XMPP whereas Fig. 4 shows CoAP connection success.

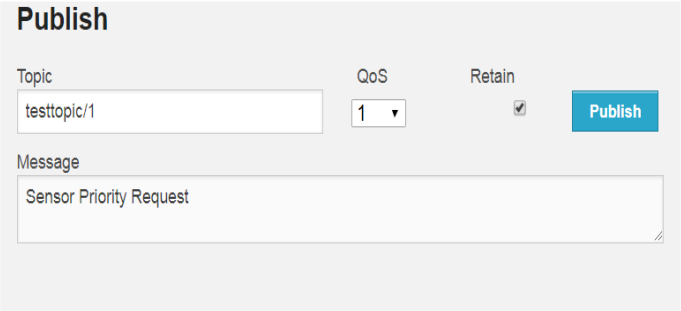

Figure 5:Publishing of subscriber data using iBroker host

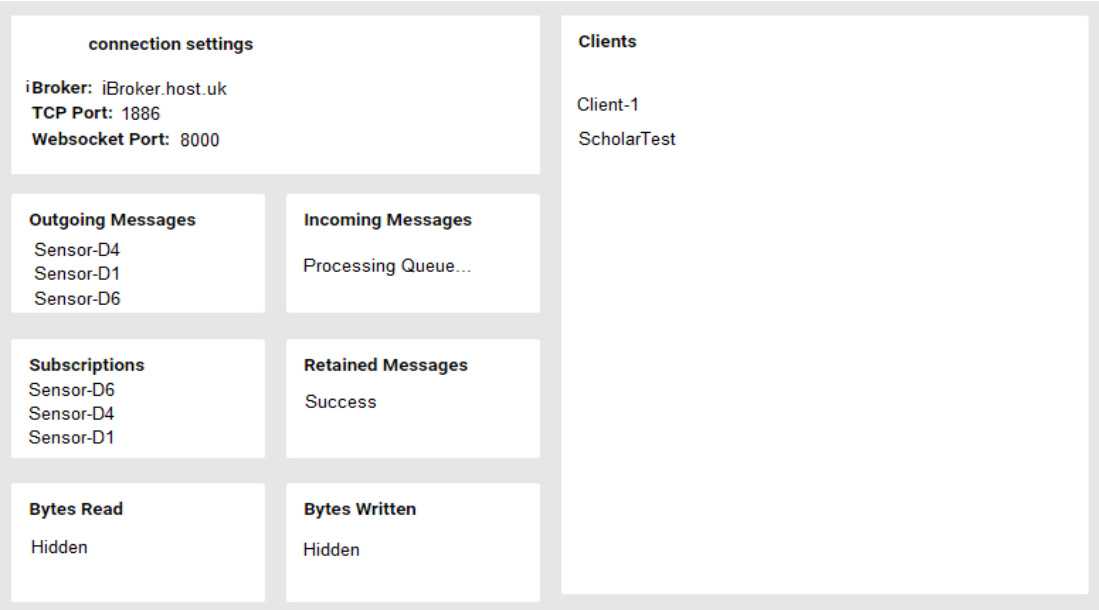

Figure 6:Response to iBroker request protocol priority pool and sensor execution testing

Further with reference to Fig. 5, to test iBroker, 'testtopic/1' will publish message for successful execution with Quality of service (QoS) as a '1'. Here assume 'testtopic/1' is an initial process for umbrella processing block for initial execution of input sensor. This will transfer the protocol credentials which are received from actual sensors.

After successful connection with the host and publish command, the server will handle request via Transmission Control Protocol(TCP) port 1886 with iBroker.host.uk and web socket port 8000. Web socket port works as a pipeline. So, client connection port must be identical as web socket port (to maintain the communication synchronization). As shown in Fig. 6, incoming messages always 'in processing' status as a continual check for client data after successful connection.

The outgoing message box shows the list of 'sensorID' which was processed by 'BrokerMessage' algorithm for each client request. Subscription shows the sequence of sensor request protocol received from 'publish' command by the client. Retained Message shows success or failure message from sensors. If end action sensor fails, Retained Message will show sensorID with a failure notice. Bytes read/written kept hidden to secure data and can be visible by encryption to system administer only. As for present middleware testing, we considered only single client as 'client-1' and username 'ScholarTest', there can be multiple clients to connect to the server. Based on developed messaging algorithm, system tests are carried out. Systems are compared for its efficiency and throughput.

As shown in Table 2, Test number 1 to 4 are tested as a standalone algorithm test for message queues and test number 5 to 7 are tested as application level live server tests (Refer Fig. 2 - 4). Also, live server status is checked for other modified message scheduling algorithm with multi queues to observe messaging traffic impact on efficiency and throughput. The executed test proved that message scheduling algorithm (i.e. BrokerMessage Algorithm) using multiple queues are more efficient with expected throughput as compared to message

Table 2. Experimental and algorithm performance comparisons for 656 message request compared with results from existing system for message queue and protocols [17]

Test No	Message Scheduling Type	Existing system [17]	Efficiency
1	Message scheduling without priority queue algorithm test-msgsch	72.43%	79.4%
2	Message scheduling algorithm using single priority queue reference model algorithm test -msgschb	90.56%	92.1%
3	Message scheduling algorithm using two queues algorithm test - msgschb2	-	94.9%
4	Message scheduling algorithm using three queues algorithm test - msgsch3	-	94.7%
5	Message scheduling algorithm using three queues-MQTT Live Test	92.16%	95.7%
6	Message scheduling algorithm using three queues-XMPP Live Test	79.7%	84.4%
7	Message scheduling algorithm using three queues-CoAP Live Test	78.5%	82.4%

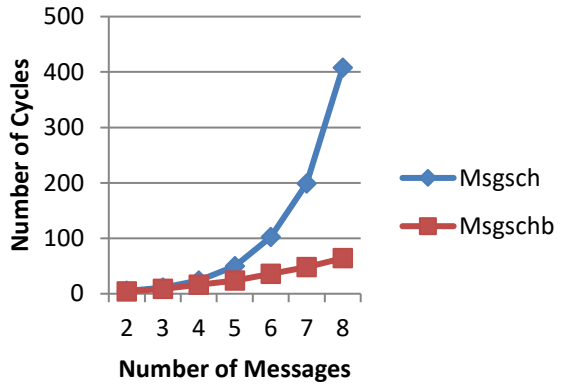

Figure. 7 Comparison of reference algorithm with (msgsch) and without (msgschb) priority queue algorithms

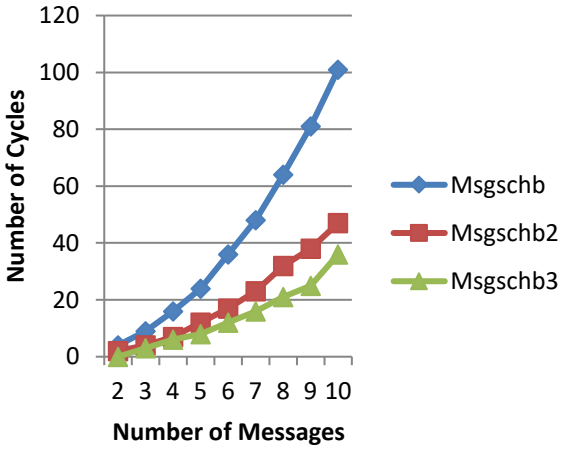

Figure. 8 Comparison of Single [16], Two &Three priority queue algorithms

scheduling without priority queue [16], message scheduling algorithm using single priority queue reference model algorithm and message scheduling algorithm using two queues algorithm (Refer Table 2).

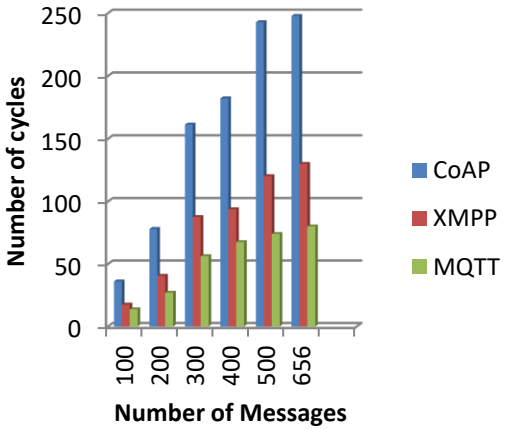

Figure. 9 Comparison of live server test of MQTT, XMPP and CoAP message protocols w. r. t. service time

The test results for variations shown in Table 2 are shown in the graphs given above.

As shown in Fig. 7, reference algorithm i.e. M/M/1 queue [16] with priority queue and without priority queue is compared. Graph shows that with priority queue is powerful as priority message execution gives quick messaging results while queue without priority queue waits for message response and causes more delay.

Similarly, as depicted in Fig. 8, reference algorithm i.e. M/M/1[16], M/M/2 and M/M/3 message queues with priority queue are compared. Graph shows that algorithm with three priority queues gives quick messaging results. But, in case of IoT hybrid protocol messaging, we also need to use BrokerMessage algorithm as discussed in section-4 of this paper.

As per comparison done for message queue, various types of protocols, further application level comparison carried out. Hence, proposed architecture Hybrid Execution of Messaging Model

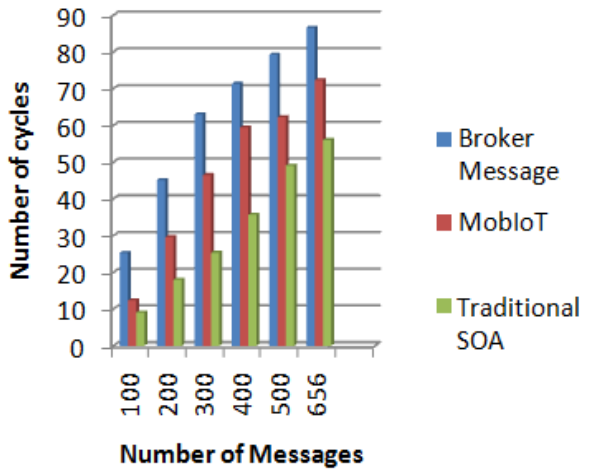

Figure. 10 Comparison of BrokerMessage, MobIoT [13] and Traditional SOA message per cycles

with execution of proposed BrokerMessage algorithm compared with existing MobIoT and Traditional SOA architecture [13] to analyze the application level efficiency of proposed system. The comparison is shown in Fig. 10.

From the comparison chart, BrokerMessage algorithm i.e. Hybrid Execution of Messaging Model performance is better for message delivery than that of MobIoT model and traditional SOA [13]. Based on graphical result discussed, we implemented BrokerMessage algorithm and system tested on live server to get response for MQTT, XMPP and CoAP messaging protocols. And MQTT has proven as a best suited lightweight protocol (Refer Fig. 10) with new middleware broker architecture. Further, as a performance test algorithm is tested for throughput analysis. Throughput is analyzed by loading application with multiple use requests at a time. So, it is necessary to know multi message handling efficiency of middleware for IoT sensor requests.

To evaluate communication efficiency of proposed BrokerMessage algorithm, performance testing is conducted using 'Apache JMeter'. Performance testing is a kind of testing meant to look for the responsiveness, trustworthiness, throughput, interoperability, and scalability of a system and/or application within certain workload. The aim of this test is to analyze the impact of network traffic. For this test, we randomly generated 656 message requests (considered as 656 samples of ApacheJMeter). The request/response is identified in millisecond (ms) units. The system protocol must achieve minimum time to distinguish system as an efficient system. The Apache JMeter test output is shown in Fig. 12.

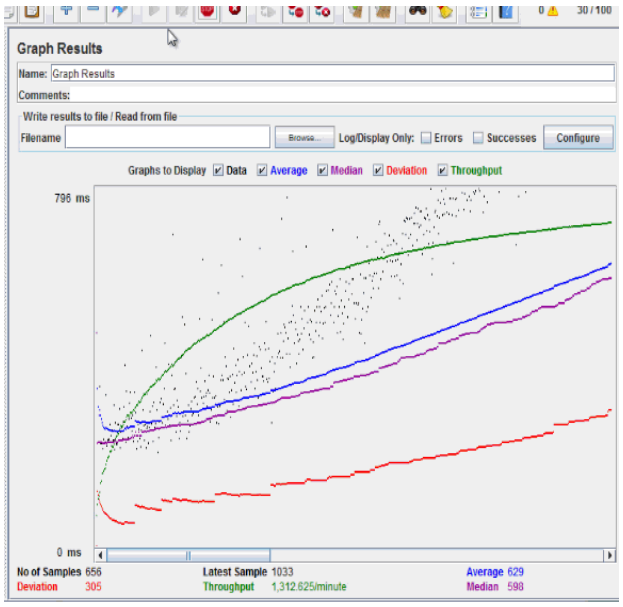

Figure. 11 ApacheJMeter test screen showing throughput for multiple message queue for MQTT

As per Table 2, MQTT gives 95.7% efficiency so, MQTT protocol is tested for throughput by sending 656 messages to IoT device using JMeter. This test provided throughput of 1,312.625/minute with an average of 629 and median is 598. The deviation is comparatively less as 305. Hence, MQTT with BrokerMessage as a middleware algorithm is best for IoT device messaging.

6. Conclusion

The paper presented a solution to 'interoperability of protocols' which is one of the major limitations of SOA for IoT device communication protocols. The message scheduling broker algorithm 'BrokerMessage' is developed for IoT environment where hybrid protocol routing is required. XMPP, CoAP, MQTT protocols considered as a one request and filtered using 'Hybrid Execution of Messaging Model' which uses BrokerMessage algorithm to handle multiple protocol requests. Additionally, this works as a scheduling algorithm and uses priority queue model. Multiple queues model is developed to increase the efficiency of the BrokerMessage algorithm. Such middleware provides a solution for message delay issue with multiple protocols request handling. The system performance is tested using Apache JMeter for MQTT, XMPP and CoAP message protocols with respect to service time and MQTT protocol performance are found more efficient during IoT sensor test. Also, as an application test, system is tested using live server HIVE platform. Due to the proposed BrokerMessage algorithm, protocol communication/messaging overhead is reduced. As

a comparative analysis, MQTT [17] is the best-suited solution if executed with the BrokerMessage algorithm. Performance comparison among BrokerMessage model, MobIoT and traditional SOA [13] revealed that, BrokerMessage algorithm provides better performance in terms of multiple protocol support for IoT as well as message service time. Hence, with proposed research it now possible to use a single IoT architecture for all applications without worry about types of protocol. Further as a future work, implementation of this solution with development of plugin model is intended to build the generalized solution for any IoT middleware irrespective of the type of protocol.

References

[1] B. H. Qiang, Z. J. Jia, P. Wang, S. Wang, and Y. F. Wang, "JMS Scheduling Model Design and Implementation Based on Priority Control in the SOA System", In: *Proc. of International Conference on Information System and Artificial Intelligence*, pp. 214-218, 2016.

[2] L. D. Xu, W. He, and S. Li, "Internet of Things in Industries: A Survey", *IEEE Transactions on Industrial Informatics*, Vol. 10, No. 4, pp. 2233-2243, 2014.

[3] D. Calvaresi, M. Marinoni, A. Sturm, M. Schumacher, and G. Buttazzo "The challenge of real-time multi-agent systems for enabling IoT and CPS", In: *Proc. of the International Conference on Web Intelligence*, pp356-364, 2017.

[4] M. H. Asghar, N. Mohammadzadeh, A. Negi, and T. Kazerouni, "Principal ingredients and framework of Internet of Things (IoT)", In: *Proc. of the Twelfth International Conference on Wireless and Optical Communications Networks*, pp. 1-6, 2015.

[5] A. Al-Fuqaha, A. Khreishah, M. Guizani, A. Rayes, and M. Mohammadi, "Toward better horizontal integration among IoT services", *IEEE Communications Magazine*, Vol. 53, No. 9, pp. 72-79, 2015.

[6] D. Happ, N. Karowski, T. Menzel, V. Handziski, and A.Wolisz, "Meeting IoT platform requirements with open pub/sub solutions", *Annals of Telecommunications*, Vol.72.1-2, pp.41-52, 2017.

[7] Z. Jiang, B. Han, P. Chen, F. Yang, and Q. Bi, "On Novel Access and Scheduling Schemes for IoT Communications", *Mobile Information Systems*, Vol. 2016, Article ID 3973287, 9 pages, 2016.

[8] I. Yaqoob, E. Ahmed, I. Hashem, A. Ahmed, A. Gani, M. Imran, and M. GuIzani, "Internet of Things Architecture: Recent Advances, Taxonomy, Requirements, and Open Challenges", *IEEE Wireless Communications*, Vol. 24, No. 3, pp. 10-16, 2017.

[9] I. Machorro-Cano, G. Alor-Hernández, N.A. Cruz-Ramos, C. Sánchez-Ramírez, and M.G. Segura-Ozuna, "A Brief Review of IoT Platforms and Applications in Industry", *New Perspectives on Applied Industrial Tools and Techniques. Management and Industrial Engineering*, pp.293-324, 2018.

[10] G. T. Oh, M. K. Back, and K. C. Lee, "A Design and Implementation of the CoAP Adaptor for Communication Between DDS-Based Adaptors and External Devices", *Advances in Computer Science and Ubiquitous Computing, Lecture Notes in Electrical Engineering*, Vol 474, pp.901-909, 2017.

[11] H. J.Kim, M.Y.Jung, W.S.Chin, and J.W. Jang, "Identifying Service Contexts for QoS Support in IoT Service Oriented Software Defined Networks", In: *Proc. of International Conference on Mobile, Secure, and Programmable Networking, Lecture Notes in Computer Science*, Vol 10566, pp 99-108, 2017.

[12] M. Albano, P. M. Barbosa, J. Silva, R. Duarte, L. L. Ferreira, and J. Delsing, "Quality of service on the arrowhead framework", In: *Proc. of IEEE 13th International Workshop on Factory Communication Systems*, pp. 1-8, 2017.

[13] S. Hachem, A. Pathak, and V. Issarny, "Service-Oriented Middleware for the Mobile Internetof Things: A Scalable Solution", In: *Proc. of IEEE GLOBECOM: Global Communications Conference*, 2014.

[14] D. Gross, J.F. Shortle, J.M. Thompson, and C.M. Harris, Fundamentals of Queueing Theory, 4th ed. Hoboken, NJ, USA: Wiley, 2008.

[15] S. Abdullah and K. Yang, "A QoS aware message scheduling algorithm in Internet of Things environment", In: *Proc. of IEEE Online Conference on Green Communications*, pp. 175-180, 2013.

[16] J. Leu, C. Chen, and K. Hsu, "Improving Heterogeneous SOA-Based IoT Message Stability by Shortest Processing Time Scheduling", *IEEE Transactions on Services Computing*, Vol. 7, No. 4, pp. 575-585, 2014.

[17] J. Aguirre, A. Ordóñez, and H. Ordóñez, "Low-Cost Fire Alarm System Supported on the Internet of Things", *In: Solano A., Ordoñez H. (eds) Advances in Computing. CCC 2017,*

Communications in Computer and Information Science, Vol 735, 2017.

[18] F. Saeed, A. Paul, A. Rehman, W. H. Hong, and H. Seo, "IoT-Based Intelligent Modeling of Smart Home Environment for Fire Prevention and Safety", *Journal of Sensor and Actuator Networks*, Vol. 7, No. 1, p. 11, 2018.

[19] D. H. Kang, M. S. Park, H. S. Kim, D. Y. Kim, S. H. Kim, H. J. Son, and S. G. Lee, "Room Temperature Control and Fire Alarm/Suppression IoT Service Using MQTT on AWS", In: *Proc. of International Conference on Platform Technology and Service*, pp. 1-5, 2017.

[20] A. Malik and H. Om, "Cloud Computing and Internet of Things Integration: Architecture, Applications, Issues, and Challenges", *Sustainable Cloud and Energy Services*, pp1-24, 2018.