Development of IoT Middleware Broker Algorithm for Handling Multiple Event Based Protocol Requests

Sayalee Deshmukh, S.B.Vanjale

Abstract*: The IoT Middleware is an integration layer for many Sensors, Actuators, devices, and brokers. The core aim of broker is to receive the request from the gateway and to process the payload and forward to the controlling actuators. Existing middleware uses thediscovery and protocols request management. Proposed work is focused on middleware soft-broker development which can support incoming protocols of IoT gateways. This paper presents the newly developed algorithm for handling CoAPrequests. This can be very useful for industrial IoT systems where sensor systems are powerful.

***Index Terms*: IoT, middleware,communication,CoAP**

I. INTRODUCTION

Device monitoring IoT broker are liable for caching the modern view of all entity gadgets and also serving context updates, queries, and subscriptions. In terms of execution collection, IoT agents are disbursed on the distinct nodes. Subsequently, to help efficient protocol verbal exchange, broker plays crucial role. Also, the middleware broker has to able to supporting many varieties of protocol communication.

Middleware for IoT is acknowledged as the unit which usually present this kind of vital facilities of services and soit became essential for the IoT [1]. Many models are cloud-based platforms. The Pub-Sub pattern presents an option for gadgets to connect and synchronize with each other. Every request is registered with specific event, with a identification code which permits the broker to promptly categorize requests for further delivery of message. Units may possibly inform the broker about requests they are about to process [2]. Genuine request recognition and particular request processing for modern applications, it is necessary to facilitate with programs which efficiently process the focused information from linked IoT gadgets [3].

II. PROPOSED METHODOLOGY

Because of the extremely significant amount of different solutions getting utilized in IoT platforms, a number of solutions are proposed to impose the incorporation and the reliability of units and information in the connected network. Consequently, suggested study aims at the structure of the middleware broker named “UniversalCom”. The system is tested for CoAP protocol with respect to IoT event based requests.

A. Block Diagram

IoT middleware communication is the important aspect of overall IoT system communication. The role of middleware is to receive the requests processed by gateway and to pass those requests to respective in action sensor or device. In short the middleware is a kind of the port which forwards the gateway, device, sensor request and responses. For proposed research we considered event based CoAP [4] protocol communication. Proposed event based middleware broker request handling block diagram is shown in figure 2 below. The sensor request flows from sensor to universal gateway where this gateway identifies the protocol request and further forwards it to the middleware. The middleware broker gets incoming request and passes to the actuators of respective application. In above figure 2, camera and fire alarm systems are shown as part of IoT actuator panel. In some applications actuators send response back to sensor control panel as an acknowledgement. Such acknowledgement is again received by middleware broker and send to universal gateway.

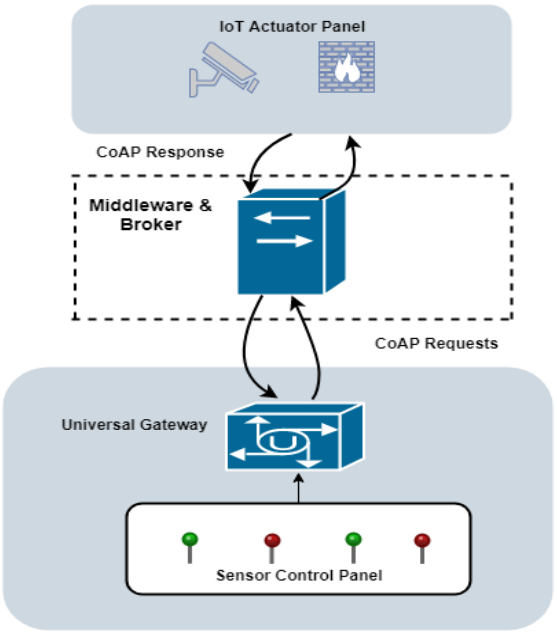

Figure 2–Proposed event based middleware broker request handling

Revised Manuscript Received on August 08, 2019.

Prof. S.A.Deshmukh, Research Scholar at Department of Computer Engineering,Bharati Vidyapeeth (Deemed to be University) College of Engineering,Pune, India. Email: sayalee87.deshmukh@gmail.com

Prof. Dr. S.B.Vanjale, Department of Computer Engineering,Bharati Vidyapeeth (Deemed to be University) College of Engineering,Pune, India. Email: sbvanjale@bvucoep.edu.in

B. UniversalComAlgorithm

1. Input: Incoming CoAP protocol request
2. StringreqID, eventId,actuatorID,messageID
3.arrayeventArr[] // store sequential eventIDs
4. arrayactuatorArr[ ] // store all actuatorID list
5. arraythingsReg[]
6. String Type // Con or NCON
7. arraymsgIDArr[]
8. String payload
9.arrayList [msgID, payload]
10. ifeventArr[] &&thingsReg[] matches reqID
11. responseCode = =Valid
12. else
13. responseCode = = BadRequest
14. Store eventId, actuatorID, messageID, payload
15. ifeventArr[] !=null &&eventArr [] ==1
16. processReq()
17. forwardReq () // to Actuator control panel
18. else
19. apply FIFO message scheduling
20. repeat step 10 to 12
21.if acknowledgement = = true
22. forwardReqGate () // acknowledgement to gateway
23. forwardEventReg() //forward request to event registry
24.forwardSensorCP() // acknowledgement to control sensor panel
25. else
23. Set responseCode = =Valid
24. End

As shown in above algorithmic details, in proposed research we focused on new middleware broker algorithm development for event based CoAP protocol communication. The CoAP message format is shown in figure 3 below which signifies, version, type (CON, NCON etc), length of token, code (as client or server error response), message id and payload.

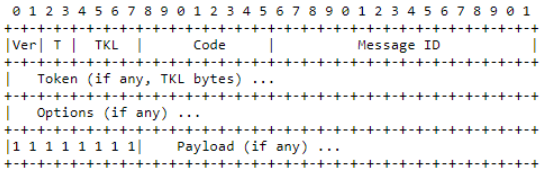

Figure 3 – Standard CoAP message format [5]

Proposed research considers type, message id and payload for testing of UniversalCom algorithm. The detail flow of algorithmic processes is shown in figure 4.

C. Flowchart

The flowchart comprised of two sections. One is sensor control panel section and other is UniversalCom algorithmic process flow. The incoming protocol request is assumed as a CoAP request. The details about sensors are assumed to be in things registry and sensor details of sensor intended for specific events (e.g. Sensor-1 as a fire sensing sensor) are already stored in events registry (which often fixed at the time of control panel commissioning).

The universal gateway is a gateway cluster which supports CoAP, MQTT and XMPP etc IoT protocols is connected as a entry port for all incoming requests. For proposed work we considered only CoAP protocol requests hence, Universal Gateway are for CoAP protocol. Once incoming request is generated it is first validated that whether it's entry is in things registry and sensor is allocated with event registry. The event registry is a pair of sensor ID and type of event so, if malicious request comes, the request will be marked/dumped as a "Bad Request" with client error code 4.00 (as per standard CoAP protocol specifications).

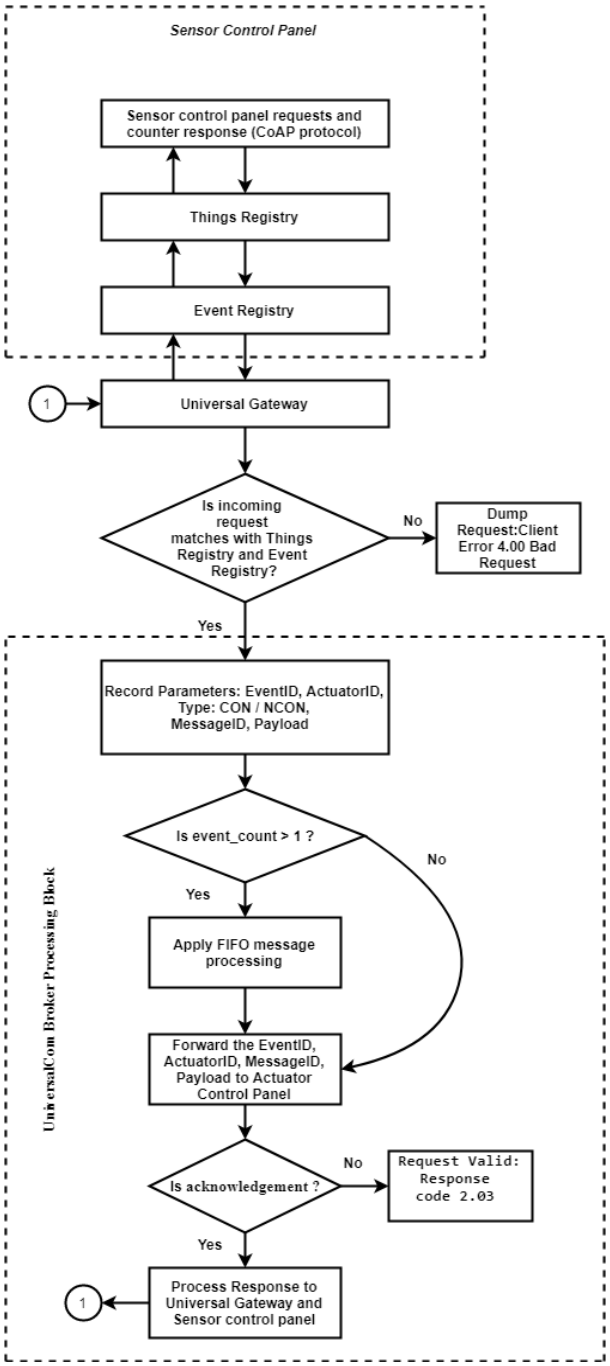

Figure 4 - Proposed UniversalCom Algorithm Flow

III. RESULT ANALYSIS

The testing of protocol is done with "coap.me" open source server and message specific parameters are recorded. The parameter "Type" has two kinds as "CON" i.e. confirmable message and "NON" i.e. non-confirmable message, (refer figure 3) if type of message is CON then it provides Acknowledgement (ACK) message which travel back to sensor control panel via Universal gateway. For valid request the response code 2.03 is generated and for contents response code 2.05 is generated. For invalid request response code will be 4.05 i.e. method not allowed (refer figure 6 and 7 below)

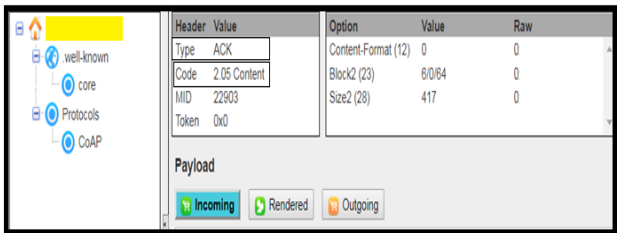

Figure 6 – UniversalCom Output for response code 2.05

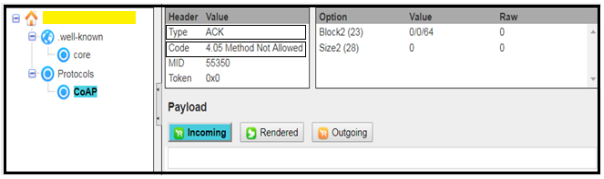

Figure 7 – UniversalCom Output for response code 4.05

TheUniversalCom is tested for pre-defined Things registry and event registry. The payload is generated manually for pilot testing and processed using UniversalCom algorithm. With valid message we noted the response code 2.05 as shown in figure 6 above and for messed request we received response code 4.05 as shown in figure 7 above.

Figure 8 – UniversalCom payload response output

For valid contents the request and response with payload is shown in above figure 8. The universal gateway is applied as a plug-in for this test. Hence, the UniversalCom is successful for CoAP protocol communication.

IV. CONCLUSION

In this paper we presented the UniversalCom algorithm as a soft-broker. The algorithm extracts the CoAP message parameters at run-time and provides the routing of the incoming sensor requests. We also presented the detail flow of requests and response through Universal gateway. For testing purpose we used coap.me server and evaluated response code for payload. As a future development, proposed algorithm can be used to test the multiple types of protocols with Universal gateway.

REFERENCES

1. Tiburski, Ramão Tiago, et al. "The importance of a standardsecurit y architecture for SOA-based iot middleware." *IEEE Communications Magazine* 53.12 (2015): 20-26.
2. Fuentes Carranza, Juan Carlos, and Philip WL Fong. "Brokering Policies and Execution Monitors for IoT Middleware." *Proceedings of the 24th ACM Symposium on Access Control Models and Technologies*.ACM, 2019.
3. Benson, Kyle E., et al. "Firedex: a prioritized iot data exchange middleware for emergency response." *Proceedings of the 19th International Middleware Conference*.ACM, 2018.
4. Iglesias-Urkia, Markel, et al. "Analysis of CoAP implementations for industrial Internet of Things: a survey." *Journal of Ambient Intelligence and Humanized Computing*10.7 (2019): 2505-2518.
5. https://tools.ietf.org/html/rfc7252

AUTHORS PROFILE

Prof. S.A.Deshmukh Research Scholar at Department of Computer Engineering,Bharati Vidyapeeth (Deemed to be University) College of Engineering,Pune. Published over 6 papers in National and International Journals and Conferences

Dr. S. B. Vanjale Professor at Department of Computer Engineering,Bharati Vidyapeeth (Deemed to be University) College of Engineering,Pune.Completed Ph.D. (Computer Engineering) from Bharati Vidyapeeth Deemed UniversityCollege of Engineering, Pune in October 2016. Sir is working as Chairman of Board of Studies Computer & IT department. Sir has 56 Citations and h-index is 5.Published over more than 38 papers in various National and International Journal and over 26 papers in National and International Conference.