Internet of Things: Reliability & Availability of Advanced Queuing Mechanism in RabbitMQ, Issues & challenges

Abhishek D. Pathak¹ and C. Kalaiarasan²

¹Research Scholar, Presidency University, Bengaluru- 560064, India,

²Associate Dean, University, Bengaluru- 560064, India,

ABSTRACT

Internet of Things (IOT) is a evolutionary domain which is impacting the day to day life all over the world. In IOT the non-intelligent peripherals/devices are used for smart applications, as well as for becoming the sources for getting the needed information. The devices and platforms used in IOT are of different types and uses different platforms which makes the IOT more scalable and parallel more challenging. The standards/protocols used in IOT are basic infrastructure used to combine and synchronize these heterogeneous behavior of devices thus making the communication more reliable and accurate. In this paper an introduction about the basic levels of IOT are highlighted. Publish and subscribe model is discussed with the factors affecting the quality of services. The popular standard AMQP and its client implementation RabbitMQ is discussed with implementation and exception. The method for handling the exception i.e DLX is discussed. The various issues are highlighted. With this paper the critical review on RabbitMQ standard is provided.

KEY WORDS: INTERNET OF THINGS, AMQP, RABBITMQ, DEAD LETTER EXCHANGE

INTRODUCTION

In growing era, Internet of Things (IOT) is emerging technological domain, the technological enhancements in IOT is providing more relevance with societal aspects. In future the wide growth will generate the horizontal needs to infrastructure. Today the infrastructure depends on specific needs but as the use of applications will grow then the heterogeneity with the devices may become difficulty to synchronize all the devices with better performance. The devices work independently, and in future with increase in growth the plug and play infrastructure will be more needed so that wide variety of application may run simultaneously. The need in requirements

and infrastructure also create the issues related to interoperability and semantic modelling of devices and standards. Infrastructure should also support and provide the facilities to automatic update of the IOT data whenever new things are available and needed. As mentioned (by Dr. Ovidiu Vermesan et al.) The infrastructure should in IOT should support functionalities, plug and play integrations, semantic modelling of things, physical locations and positions and security and privacy.

The devices and platforms used in Internet of Things mostly will be ready to plug and play with the environment and should get easily synchronized with infrastructure and standards used in it. So interoperability becomes important area of consideration. IOT interoperability is a key challenge and will be the center of consideration in future because of the rapid increase in devices of heterogeneous configurations and continuous updating device nature. This will lead to the continuous updates in standardization process. In networking system of IOT the various levels use different types of standards/protocols. This networking level system can be categorized based on Data linking, Networking & session. The most popular

ARTICLE INFORMATION

*Corresponding Author: abhishekipathak81@rediffmail.com

Received 19th Oct 2020 Accepted after revision 30th Dec 2020

Print ISSN: 0974-6455 Online ISSN: 2321-4007 CODEN: BBRCBA

Thomson Reuters ISI Web of Science Clarivate Analytics USA and Crossref Indexed Journal

NAAS Journal Score 2020 (4.31)

A Society of Science and Nature Publication, Bhopal India 2020. All rights reserved.

Online Contents Available at: <http://www.bbrc.in/>

Doi: <http://dx.doi.org/10.21786/bbrc/13.14/5>

and most commonly used standards are shown in fig.1 As depicted in above diagram the level wise protocols for IOT. The communication system needed for infrastructure can be categorized in three levels of system.

Figure 1: Level wise protocols for IOT

1.1 Datalink Level: This level provides the needed support for connecting two or multiple devices to aggregates the information from the devices for this specialized communicating protocols like 802.11ah, wireless HART, Z-Wave, ZigBee etc. are used to synchronize the operating behavior of heterogeneous devices to access the system.

1.2 Network Level: This layer encapsulates the information to secure for communication as well as decides the routing method. For this various protocols such as RPL, 6LowPan etc. provides the services.

1.3 Session Level: The session level provides the session services for transmission i.e mostly responsible for managing the communication method between the devices. The protocols used at this level are AMQP, MQTT, DDS & COAP.

In this paper we emphasized mostly on session layer protocol AMQP and its client RabbitMQ. These protocols used for management of information at session level can be of simple subscribe and Access i.e. request and response types or of publish subscribe model. Now a days publish and subscribe models are attracting the researchers as it provides more interoperable functionalities to the devices connected infrastructure. Publish subscribe model provides routing logic and more reliable functionalities by making system more scalable and loosely coupled. The major deciding factors i.e. Quality of Service in standards used in IOT communication are Availability, Reliability & Accuracy.

Publish/Subscribe models: Publish/subscribe systems are like messaging systems used in application where messaging is implemented. This approach is most suitable in constrained type of environment as well as provides less congestion of data traffic in communication [2]. There are various protocols like MQTT, XMPP, DDS uses the publish/subscribe model but Advanced Messaging Queuing Protocol (AMQP) is widely accepted and concluded suitable for wide number applications. RabbitMQ and Kafka are the popular implementations of AMQP protocol. The basic architecture of publish and

subscribe model consists of broker concept. As shown in fig2.

Figure 2: Publish/Subscribe model

2.1 Functionalities: The main functionality of publishers and subscriber is decoupling. Which makes it most suitable for loosely coupled and scalable systems. As Eugster et al. discussed that it can be with the following schemes with dimensions such as.

- **Entity decoupling:** In this scheme publish and subscriber do not need to be aware each other's functionalities. The infrastructure is independent of terminating the process at any time thus reducing the overheads in communication which makes it more suitable for loosely coupled environment
- **Time decoupling:** This schemes allows the infrastructure to passively participate in the process making it stronger in interaction.
- **Synchronization decoupling:** This allows more independent access to both publisher and subscriber, without any synchronous control permissions.

Messaging system in publish and subscriber model uses routing logics to manage the messages, The routing logic is used to bind the queues with the exchanges. The binding process can be of Topic based, Direct, Header base and Fan-out. The most popular implementation of AMQP protocol is RabbitMQ.

Figure 3: RabbitMQ Broker

2.2. RabbitMQ: RabbitMQ is an efficient and scalable implementation of AMQP protocol. In simple way in RabbitMQ multiple queue implementation is possible. Queues are used to temporarily manage the data. Queue can also be implemented with multiple exchanges.

The multiple queues can be implemented and are bind with exchange. The exchange is responsible for arranging the messages in the queue by using routing logic. Mostly round robin logic is used to arrange the messages in the queues. The basic mechanism on which RabbitMQ

works is Queuing mechanism. The queues gets bind with exchanges, exchange routes the messages produced by the producer and consumed by the consumer from queue. These producers and consumers may be of different and varying configurations and applications, which gives rise to the issues related with Accuracy and interoperability. i.e mismatch between number of message packets generated per second by the producer (publisher) and message packets consumer by the consumer because of different configuration platforms. Depending upon applications RabbitMQ can be implemented in following cases.

1. Implementation

In our work we have implemented RabbitMQ 3.7.1. In our work the cases considered are :

- Case1: One Queue with single consumer.
- Case2: One Queue with multiple consumers.
- Case3: Multiple queues with multiple consumers.

In case1: Where single queue with single consumer is bind the messages are moved accessed sequentially as only a single consumer is invoked.

In case2: A single queue gets bind with multiple queues and multiple consumer access the queue.

In case3: Where multiple queues gets bound with multiple consumers

In all three cases mentioned above when multiple consumers are involved the rate of pulling the messages from queues may differ from rate of pushing the messages in queues, which may create the unordered message packets issue i.e. the order in which messages are pulled are different from the order of messages pushed in queues.

2. Exceptions: Consider X New(X-N) & Y New(Y-N) as transactions with their invoice receipts as X Paid(X-P) and Y Paid(Y-P). Arranged in order in queue as shown in figure.

Figure 4: Messages in queue in ordered manner

Figure 5: Timeline of no. packets consumed by consumers

Two consumers are bound with queue for consuming the packets. Fig.4 messages in queue in ordered manner So as per Round Robin scheduling. defined in RabbitMQ the 1st and 2nd messages will get distributed to consumer1 and 2nd and 4th messages to consumer2.

In above example consumer (C1) consumes 1st and 2nd message in total 2-T-states and status of Y process is updated as paid and when consumer (C2) consumes another message will complete consuming next message till 3rd T-states, but till that time status of Y is updated as paid by consumer1, So this is because of scheduling the message packets when multiple consumers are involved. The producer will publish messages at faster rate and consumer will consume messages at slower rate, Message can live forever in a queue, if no consumer that consumes message in queue gets crashed, due to this the no messages will piled up in the queue. Due to this queues starts rejecting the messages which may get lost without acknowledging the producer thus gives rise to unreliable communication. As well as the rejection of messages from queue deteriorates the queue performance.

Handling Exceptions on Consumer: Messages can live forever in a queue if no consumer consumes it. If no consumer consumes it then message can be discarded. In RabbitMQ this issue handled in a better way by using a mechanism known as Dead Letter Exchange (DLX). Instead of discarding the messages are arranged to in a queue of DLX. The DLX mechanism republishes the messages for the consumers. The reasons that messages may get republished are:

- The messages is rejected by queue if queue is full.
- The TTL (Time to live) for message expires.
- The queue length limit exceeds the maximum.

This mechanism helps in handling the invalid messages without discarding it. Ex. If image size too large to be handled by the broker then exceptions might happened, So Dead Letter Exchange method (DLX) helps avoids these exceptions. Whenever any message is message rejected by the exchange then it is updated in DLX for once again getting pulled by the consumer this helps in getting true responses about the status of broker.

Drawbacks: DLX mechanism implemented in RabbitMQ is discussed may impact the reliability, availability and accuracy of RabbitMQ standard. In RabbitMQ mechanism if subscriber fails to consume the message then message remains in the queue till it is moved to DLX mechanism or it exceeds the Time to live parameter thus due to the faster publisher/producer, the queue receives the messages with faster rate and thus queue locations gets consumes in very short time. So after this if any message is send by exchange the message gets rejected without acknowledging the publisher and thus gives rise to message unavailability. It also increases the latency level of messages because of which the throughput deteriorates. The issue which effects the Quality of Service (QoS) is accuracy. In RabbitMQ due to the faster producers and slower consumer problem the order of messages gets

affected as shown and explained with fig.5. This simply affects the order of receiving and updating the messages, which deteriorates the accuracy level in communication.

CONCLUSION

In this paper, RabbitMQ, an implementation of AMQP standard is highlighted. The Publisher and subscriber model is discussed with mechanisms used in RabbitMQ with different case scenarios and the chances of getting issues are discussed. Thus this contribution will help the researchers to view a scope on counter measures of avoiding the issues.

REFERENCES

- A. Corsaro et al. Quality of Service in Publish/Subscribe Middleware. *Global Data Management*, 19(20):1–22, 2006.
- AlaAl-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, and Moussa Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Communication Surveys and Tutorials*, Vol. 17, No. 4, 2015, pp. 2347-2376.
- Amaran, M. H., Noh, N. A. M., Rohmad, M. S., Hashim, H., et al. (2015). A Comparison of Lightweight Communication Protocols in Robotic Applications. *IEEE International Symposium on Robotics and Intelligent Sensors*, *Procedia Computer Science*, 76, 400-405.
- Cohn, R. (2012). Comparison of AMQP and MQTT. White Paper. https://lists.oasis-open.org/archives/amqp/201202/msg00086/StormMQ_White_Paper_Comparison_of_AMQP_and_MQTT.Pdf
- Clark, A. S. & Truong, H. (2013). MQTT for sensor networks (MQTT-SN) protocol specification. http://mqtt.org/new/wp-content/uploads/2009/06/MQTT-SN/_spec_v1.2.pdf.
- Dibyajyoti Guha ; S.S. Pathak, "Load balancing using past information of queue", 2011 IEEE Recent Advances in Intelligent Computational Systems.
- Evans, D. (2011). The Internet of Things: How the Next Evolution of the Internet is Changing Everything. Cisco Internet Business Solutions Group: San Jose, CA, USA.
- Grgic, K., Špeh, I. & Hedi, I. (2016), A web-based IoT solution for monitoring data using MQTT protocol. *International Conference on Smart Systems and Technologies (SST)*, Osijek, Croatia.
- Knaian, A. N., Paradiso, J. & Smith, A. C. (2000). A Wireless Sensor Network for Smart Roadbeds and Intelligent Transportation Systems. Mass. Internet Technol. <https://dspace.mit.edu/handle/1721.1/9072> (accessed on 17 January 2018).
- L. Novelli, L. Jorge, P. Melo and A. Koscianski, "Application Protocols and Wireless Communication for IoT: A Simulation Case Study Proposal," 2018 11th International Symposium on Communication Systems, Networks & Digital Signal Processing (CSNDSP), Budapest, 2018, pp. 1-6, doi: 10.1109/CSNDSP.2018.8471765.
- Luzuriaga, J. E., Perez, M., Boronat, P., Cano, J. C., Calafate, C., Manzoni, P., et al. (2014). Testing AMQP Protocol on Unstable and Mobile Networks. *Inter- net and Distributed Computing Systems. IDCS 2014. Lecture Notes in Computer Science*, vol 8729. Springer, Cham.
- Luzuriaga, J. E., Perez, M., Boronat, P., Cano, J. C., Calafate, C., Manzoni, P., et al. (2015). A comparative evaluation of AMQP and MQTT protocols over unstable and mobile networks. *IEEE 12th Consumer Communications and Networking Conference*, Las Vegas, NV, USA.
- Madhumitha, P., Johnsema, B. & Manivannan, D. (2014). Domination of Constrained Application Protocol: A Requirement Approach for Optimization of Internet of Things in Wireless Sensor Networks. *Indian Journal of Science and Technology*, 7(3), 296-300.
- Oasis. (2014). MQTT Version 3.1.1 Plus Errata 01. <http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.pdf>.
- P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec. (2003). Many Faces of Publish/Subscribe. *ACM Computing Surveys (CSUR)*, 35(2):114–131, 2003.
- Philippe Dobbelaere and Kyumars sheikh Esmaili, "Industry Paper: Kafka versus RabbitMQ A comparative study of two industry reference publish/subscribe implementations", DEBS, 2017, June 19-23.
- RabbitMQ: Protocol Extensions. URL <https://www.rabbitmq.com/extensions.html>.
- Shi, F., Li, Q., Zhu, T., Ning, H., et al. (2018). A Survey of Data Semantization in Internet of Things. *Sensors*, 18, 313.
- Wazid, M., Das, A. K., Odelu, V., Neeraj Kumar, Conti, M., Jo, M., et al. (2018). Design of Secure User Authenticated Key Management Protocol for Generic IoT Networks. *IEEE Internet of Things Journal*, 5(1), 269-282.
- XinWei, Pingchuan Ding, Fang Zhou, JingleiLou, Yun Gao, "A Load Balancing Strategy based on Request Queue for P2P-VoD System," *IEEE digital explorer*, 22 July 2019, DOI: 10.1109/IWCMC.2019.8766536.