International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-9 Issue-2, December 2019

Internet of Things: Quality of Services of RABBITMQ & KAFKA

Abhishek D. Pathak, C. Kalaiarasan

Abstract*: Diverse use of various heterogeneous devices in Internet of Things is providing challenges and several issues in Quality of services related to Internet of Things. This paper provides the overview of Internet of Things (IoT) and standards. Factors affecting the quality of services in communication. Standards used for communication in Internet of Things (IoT) are discussed and reviews related to it is presented. RabbitMQ implementation of AMQP as well popular standard like Kafka is discussed. Quantitative and Qualitative differences between RabbitMQ and Kafka is presented. We have also discussed similarities and dissimilarities of Message Queue Telemetry Transport (MQTT) and Advanced Message Queuing Protocol (AMQP) protocols. The main objective of this review is to provide critical review of a functional view of IoT architecture, protocols and standards.

***Keywords*: RabbitMQ, Kafka, AMQP, IOT.**

I. INTRODUCTION

Internet of Things (IoT) is one of the most demanding fields in recent scenario, where need of IOT is increasing with the increase in demands with network and applications, more enhanced methods and techniques are evolving for automatizing and revolutionizing the world scenario. IOT is a paradigm, where several physical devices are interconnected in network for communication. These devices can be heterogeneous with different standards used in various application areas. This heterogeneity gives rise to several challenges in enhancement of quality of services, security, privacy, energy efficiency, scalability, interoperability and standardization of protocols. Key challenges and issues in security and privacy are upcoming research topics and several methods are addressed as solution to these issues.

In IOT the increased growth of heterogeneous devices is increasing the demand of quality of services. These services can be categorized as services related to Identity, data gathering. This also gives rise to issues related to availability, mobility, performance, scalability, interoperability and reliability thus affecting quality of services.

Revised Manuscript Received on December 30, 2019.
***** Correspondence Author
Abhishek Pathak*, Research Scholar, Computer Science & Engineering, Presidency University, Bengaluru, India. Email : abhishekpathak81@rediffmail.com
Dr. C. Kalaiarasan, Associate Dean-SOE, Presidency University, Bengaluru, India. kalaiarasan@presidencyuniversity.in

To enhance quality of services various methods are proposed with protocols and standards.(Andrey, 2018). The quality of services may degrade with low interoperability. In several applications of Internet of Things interoperability is required at knowledge as well as data level.

Keeping in view the vision of IoT, the security and privacy issue, the research projects are developed to provide key challenges. IoT researchers are addressing various ways and methods to overcome these challenges of interoperability, some of them highlighted on the methods adding semantics (Shi, et al., 2018).

This paper is organized as follows; section II we discussed about the most demanded standards used for server to server or gateway communication i.e. Advanced Message Queuing Protocol (AMQP), RabbitMQ and KAFKA. In section III and RabbitMQ and KAFKA protocols are compared, the functional differences between RabbitMQ and Kafka protocols are highlighted. The similarities and dissimilarities are discussed as presented in (Al-Fuqaha, et al., 2015), (Sethi & Sarangi, 2017). Briefly discussed about the issues and challenges in section V.

The main objective of this survey paper is to provide:

- Brief introduction about RabbitMQ & KAFKA protocols.
- To highlight on key aspects of RabbitMQ and Kafka.
- To provide brief overview of key challenges

II. STANDARDS

As Internet of Things standards is an environment of various distinct environment and behavior, this distinct environment creates issues and challenges in communication. To facilitate a uniform set of rule to all kinds of data, various standards are designed. These standards can be categorized as per the use i.e. based on application, influential and infrastructural level. In IOT data transfer is done between device to server and for server to server. These standards send and receive messages in binary and string form. Internet of Things covers a large scope of different and heterogeneous areas and fields. To facilitate and to provide support to these distinct environments, many groups provide standards and protocols. The World Wide Web Consortium (W3C), Internet Engineering Task Force is fully involved in standardization of the protocols (Madhumitha, Johnsema & Manivannan, 2014). In this we highlighted the standards used at application level.

A. Application Protocols

The Application layer is responsible for data formatting and presentation. For many years HTTP has been considered as reference protocol for communication. The internet applications are especially supported with HTTP protocol, but in a constrained environment HTTP is not suitable to use. Hence, many alternate protocols are developed such as CoAP, MQTT, MQTT-SN, AMQP, XMPP and DDS, etc. The protocols works on the principle of publish/subscribe module as well as stateless client server architecture and some protocols uses the acknowledgment mechanism for confirmation. AMQP, MQTT is most widely used protocols for light weight communication. In this paper we also discussed about the most popular client implementation of AMQP i.e. RabbitMQ and other pub/sub module like KAFKA. In this section the popular protocols like AMQP, RabbitMQ and KAFKA are discussed (Al-Fuqaha, et al., 2015), (Luzuriaga, et al., 2015).

B. Advanced Message Queuing Protocol (AMQP)

AMQP is an ISO/IEC 19464: 2014 defined open internet protocol for business messaging, a reliable exchange defined on binary-wire level. It is an advanced messaging protocol in which advanced queuing concept is used for reliable data transfer of data. AMQP is an open standard application layer protocol for message-oriented environment is presented in (Al-Fuqaha, et al., 2015). Its goal is to enable a wide range of different applications and systems to be able to work together to enhance interoperability. AMQP has a layered architecture as explained in table 2.

Table 2 – Functional architectural view

Layer 1	Defines system and encoding process
Layer 2	Defines transport layer and efficient, binary peer to peer protocol functioning
Layer 3	Defines grouping for atomic transactions.
Layer 4	Defines security aspects.

AMQP serves to deliver messages and to have simple queue system for asynchronous messages thus targeting high durability and scalability. AMQP and MQTT both works on pub/sub model. In AMQP the implementation of Broker and subscriber strategy makes AMQP more reliable and efficient, the message exchange mechanism decides the routing mechanism of data from producer to consumer through queues such as separate queues for the respective subscriber. The same is shown in Fig. 3

It uses a routine and instances to examine the message and route it to proper queue by using key, which is actually a virtual address. This exchange model accepts messages from the publisher and route them to queues according to the predefined criteria. The exchange gets attached with queues by different types of binding keys and patterns.

The various AMQP exchange types are Direct, Fan-out, Topic, and Header.

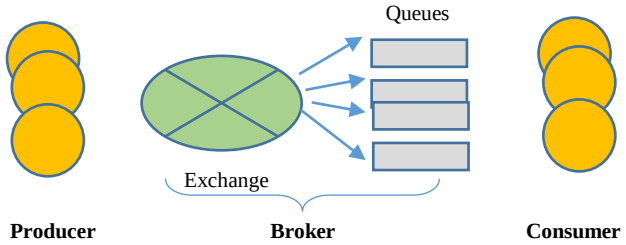

Fig. 3 – Mechanism of AMQP Protocol

The most favorite and globally used open source AMQP implementations.

- RabbitMQ
- OpenMQ
- StormMQ
- ApacheQpid
- RedHat Enterprise MRG.

C. RabbitMQ

RabbitMQ is one of the most favorite clients of AMQP, a publish/subscribe model. Supported by Erlang, a robust messaging system, which is the main reason for scalability capabilities in RabbitMQ fit for cloud scale, business messaging and is supported by adapters like HTTP, STOMP, SMTP and lightweight web messaging. It is easy to deploy. Especially in distributed environment it helps in achieving high availability. RabbitMQ supports more efficient acknowledgement system for the publisher better transactional behavior, better asynchronous batch transfer, high degree of coupling between producer and consumer. It supports better messaging services, it provides delivery acknowledgment mechanism, flexible routing queues and multiple exchange types. In comparison with AMQP, RabbitMQ has efficient acknowledgement mechanism and improved behavior related to transactions. It is developed on Erlang which implements actor model concept for lightweight processes. In RabbitMQ because of increased topics nad queues the clustering capability is increased.

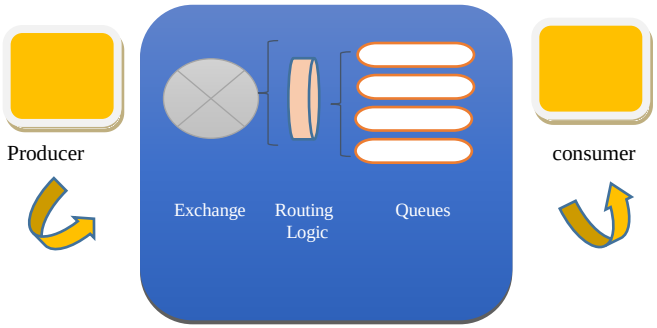

Fig.4 – Mechanism of RabbitMQ protocol

As shown in fig,4 RabbitMQ supports multiple queuing. The producer sends the messages to the exchange. The exchange forwards rhe message based on the routing logic to the respective queue.

D. KAFKA

Apache Kafka is a scalable publish subscribe messaging system (Philippe Dobbelaere et. al 2017).

Kafka was originated at LinkedIn pipelining platform. The main objective of designing Kafka is to handle high throughput by handling multiple consumers i.e. message set by the producer can be consumed by one or more consumers. Kafka works on distributed commit log technique in which records cannot be updated or deleted once it is they are send to Kafka. The communication in Apache Kafka is controlled by Zookeeper module. Cluster management and partitioning in Kafka is managed through zookeeper Kafka achieves high scalability by using partitioning method with topics. To improve scalability and performance with its distributed environment it uses zookeeper for clustering, topics, election and election control.

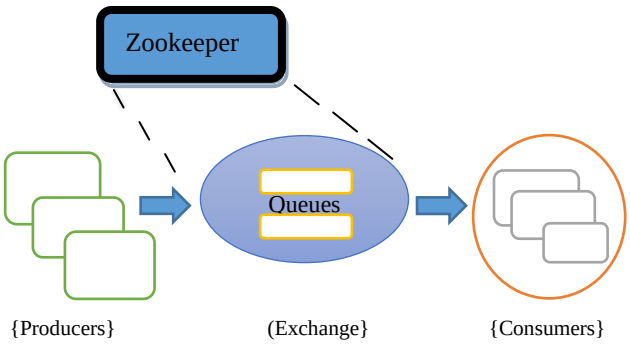

Fig. 5 – Mechanism of Kafka

As represented in the fig.5. Kafka is distributed architecture environments. The consumers are assigned groups. These groups pulls the data from respective topics. This featured mechanism makes Kafka more fault-tolerant and scalable.

III. COMPARATIVE ANALYSIS OF RABBITMQ AND KAFKA

As stated in (Philippe Dobbelaere et. al 2017) the comparative analysis based on Quantitative and Qualitative factors as stated below. Quantitatively RabbitMQ and Kafka can be compared on parameters like time decoupling, routing logic, delivery guarantees, ordering guarantees, and availability and multicast transactions.

In RabbitMQ, the exchanges provides the mechanism to support routing logic. i.e. producer sends the messages to exchanges and then to queues based on routing algorithms. The failure in order of packet might disturb efficiency and throughput factor of the system. In RabbitMQ acknowledgement mechanism is introduced by using at most once and at least once delivery methods, In Kafka the order is not maintained as it uses multiple partitioning method, but the result vary significantly with these methods. This is not the case in Kafka the results improves with this delivery methods. In RabbitMQ when delivery method is in at least once mode then the queue may get drained if messages are not consumed by the consumer and not acknowledged. Thus, resulting the RabbitMQ execution to maximum load, which will deteriorate the latency.

Table 3– Qualitative comparison of RabbitMQ and KAFKA.

Criteria	RABBITMQ	KAFKA
Time Decoupling	RabbitMQ consumes heavy amount of memory if messages are not consumed at the consumer side	While Kafka enables various consumption rates to handle wide scale of time decoupling
Routing logic	Routing logic can be anything. It can be topic based or content based exchange type	Kafka supports only topic based routing mechanism
Delivery Guarantees	In RabbitMQ, confirmation mechanism is introduced for consumer acknowlegement	Kafka cannot preserve order while sending to multiple partitions
Ordering Guarantees	RabbitMQ conserve order for flows using single AMQP channel	Kafka conserve order only inside the partition
Availability	RabbitMQ provides availability by replication	Kafka also provides availability by replication
Multicast	In RabbitMQ dedicated queue is provided to individual consumer	In Kafka copy of messages and index is maintained at the broker side

As shown in above table 3. Both RabbitMQ and Kafka can handle large message string with low consumption rate as stated in (Philippe Dobbelaere et. al 2017).

IV. SUMMARY

Both RabbitMQ and Kafka protocols are very popular pub/sub standards. The comparison of the standards is done on parameters of performance and efficiency.
With the above comparison it is clear that Kafka can be used in such conditions data transactional behavior is not fixed, Kafka only uses topic based routing mechanism thus increasing the security and privacy. On other hand RabbitMQ maintains order of messages thus increasing the complexity in at the consumer side. RabbitMQ is more flexible as compared to Kafka as it uses both topic or content based routing mechanism.

V. CONCLUSION

In this paper, we attempt to present the review of RabbitMQ and KAFKA standards, AMQP protocol is explained and the client implementation of RabbitMQ and one of the recent publish system Kafka the application layer protocols used in IoT applications. Both the protocols very popular in application area, but provides varying behavior and properties with the change in scalability, and delivery methods. Moreover, we have highlighted the comparison, dissimilarities of these protocols. However, this critical review definitely helps the researchers to understand these protocols as well to identify and utilize the appropriate protocol for different real-time IoT applications.

REFERENCES

1. Andriy Mazayey, James A. Martins and Noelia Correia, 23 April,2018, " Interoperability in IOT through Semantic Profiling of Objects,IEEE, 2018.
2. Shi, F., Li, Q., Zhu, T., Ning, H., et al. (2018). A Survey of Data Semantization in Internet of Things. Sensors, 18, 313.
3. Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., Ayyash, M., et al. (2015). Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. IEEE Communication Surveys and Tutorials, 17(4), 2347-2376.
4. Sethi, P. & Sarangi, S. R. (2017). Internet of Things: Architectures, Protocols, and Applications. Journal of Electrical and Computer Engineering, 2017, 25 pages.
5. 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
6. 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.
7. Philippe Dobbelaere and Kyumars sheikh Esmaili, "Industry Paper: KafkaversusRabbitMQ A comparative study of two industry reference publish/subscribe implementations",DEBS,2017, June 19-23.
8. 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.
9. Cohn, R. (2012). Comparison of AMQP and MQTT. White Paper. https://lists.oasis-open.org/archives/amqp/201202/msg00086/ StormMQ_WhitePaper_-Comparison_of_ AMQP_ and_ MQTT. pdf
10. Abhishek Pathak, Dr. C. Kalaiarasan, et al. Internet_of_Things_ (2019). A_Survey_on_AMQP, MQTT & Challenges.pdf.
11. 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).
12. 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.
13. Lin, J., Yuy, W., Zhangz, N., Yang, Z., Zhangx, H., Zhao, W., et al. (2017). A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications. IEEE Internet of Things Journal, 4(5), 1125-1142.
14. Yang, Y., Wu, L., Yin, G., Li, L., Zhao. H., et al. (2017). A Survey on Security and Privacy Issues in Internet-of-Things. IEEE Internet of Things Journal, 4(5), 1250-1258.
15. Bormann, C., Castellani, A. P. & Shelby, Z. (2012). CoAP: An Application protocol for billions of tiny Internet nodes. IEEE Internet Computing, 16(2), 62-67.
16. 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.
17. 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.
18. Clark, A. S. & Troung, 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.
19. Grgić, K., Špeh, I & Heđi, I. (2016), A web-based IoT solution for monitoring data using MQTT protocol. International Conference on Smart Systems and Technologies (SST), Osijek, Croatia.
20. 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. Internet and Distributed Computing Systems. IDCS 2014. Lecture Notes in Computer Science, vol 8729. Springer, Cham.

AUTHORS PROFILE

Abhishek Pathak, Research Scholar, abhishekpathak81@rediffmail.com Departerment of Computer Science & Engineering, Presidency University, Bengaluru– 560064, India

Dr. C. Kalaiarasan, Associate Dean-SOE kalaiarasan@presidencyuniversity.in Presidency University, Bengaluru– 560064, India