

Message Delivery Guarantee and Status Update of Clients Based on IoT-AMQP

Purvi Bhimani and Gaurang Panchal

Abstract Advanced Message Queue Protocol (AMQP) is an open-standard application layer protocol for IoT focusing on message-oriented middleware. It provides asynchronous publish/subscribe communication with messaging. It is store-and-forward feature that ensures reliability even after network disruptions, which is its main advantage. When compared all other IoT protocols with AMQP protocol, it gives better performance. In this paper, we provide features for some cases or situations like when any client disconnected ungracefully or when any client connected and subscribed for a particular topic which it is interested in. This is because these features are used to notify other client(s) about disconnected client and help newly subscribed clients to get a status update immediately after subscribing and do not have to wait until the publishing clients send the new update. So AMQP protocol provides the guarantee of message delivery and provides reliable communication even after a network failure.

Keywords AMQP · Status update · Publish/subscribe
Message delivery guarantee

1 Introduction

A ubiquitous network enables monitoring and control of the physical environment by collecting, processing and analysing of data, which are generated by sensors, actuators or smart objects referred as an Internet of things (IoT) [1–3]. It should be capable of interconnection through the Internet that connects billions and trillions of objects. There are three main parts of IoT: first is a network terminal part that includes sensors, second is the transmission network that refers to information

P. Bhimani (✉) · G. Panchal

U & P. U. Patel Department of Computer Engineering, Chandubhai S Patel Institute of Technology, Changa, India
e-mail: bhimanipurvi56@gmail.com

G. Panchal

e-mail: gaurangpanchal.ce@charusat.ac.in

© Springer Nature Singapore Pte. Ltd. 2018

Y.-C. Hu et al. (eds.), *Intelligent Communication and Computational Technologies*, Lecture Notes in Networks and Systems 19,
https://doi.org/10.1007/978-981-10-5523-2_2

transmission and third is the data processing centre that controls the processing centre. It grows the world's economy and improves the quality of life [4]. For example, smart home enables their residents to automatically open their garage when reaching home, prepare their coffee or tea, and control climate control systems, TVs and other appliances.

The individual networks are connected together with security, analytics and management that referred as a network of networks for IoT [5, 6]. It is a global concept that provides communication between human-to-human, human-to-machine, machine-to-machine communication [3, 7]. A special set of rules that used by end points when they communicate in telecommunication connection is referred as a Protocol, which specifies the interaction between communication entities. It allows information to be transmitted from one system to another or one device to another device over the Internet [8–16].

2 Brief About AMQP

AMQP [3, 5–7] is an open-standard wire-level application layer protocol for IoT [6]. It is focusing on message-oriented environment. It provides asynchronous publish/subscribe communication with messaging. It is based on reliable transport protocol TCP. It is a store-and-forward feature that ensures reliability even after a network failure. AMQP provides reliable communication via message delivery guarantee that is at least once, at most once, exactly once [17]. As shown in Fig. 1a, at most once guaranteed, delivery referred as the message is sent one time either it is delivered or not. At least once guaranteed delivery referred as the message will be delivered one time and possibly more times (See Fig. 1b). Exactly once guaranteed delivery referred as the message will be delivered only one time (See Fig. 2).

2.1 AMQP Model Structure

To create a desired functionality, there are three main types of components that connected to the processing chain in the server (Fig. 3).

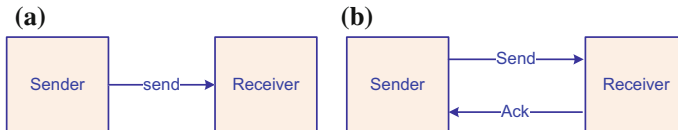


Fig. 1 a At most one delivery; b At least one delivery

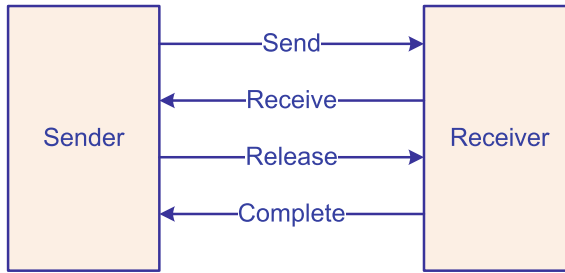


Fig. 2 Exactly once delivery

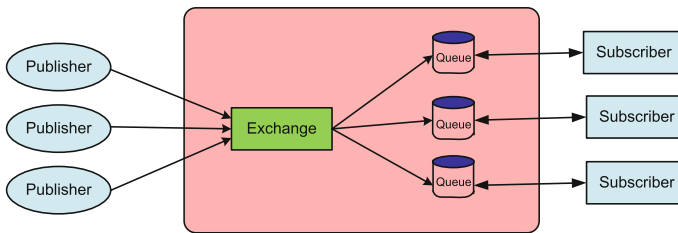
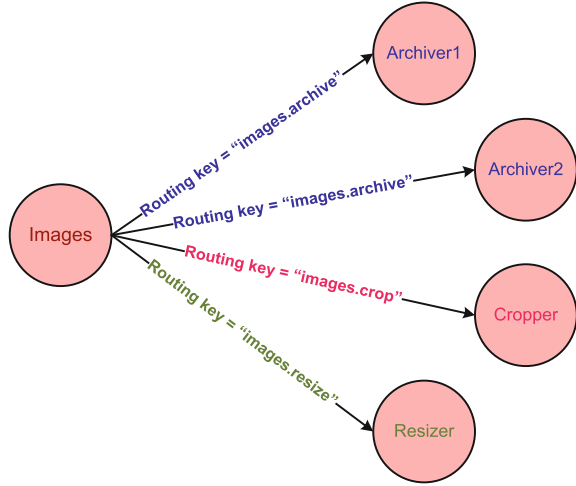
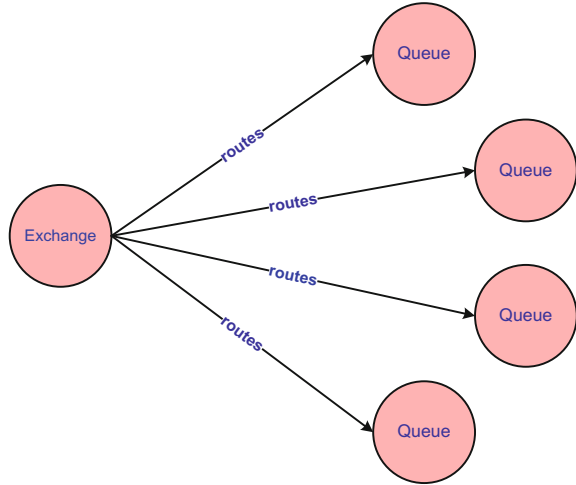


Fig. 3 AMQP model structure

Exchange: Exchange is routing and matching engines. It receives messages from publisher application and forwards it to the message queues according to predefined rules or criteria. It never stores messages. They use routing pattern to route the message to particular queues. There is a number of attributes of exchange namely name and durability. It survives until the broker restarts; auto-delete means it is deleted when all queues have finished to use it, i.e., arguments. There are four types of standard exchange types:

1. **Direct exchange:** Direct exchange routes message to the queue based on the message routing key. It is suitable for unicast as well as multicast routing. In this type of exchange task, messages are distributed between multiple workers in round robin manner. Direct exchange can be represented graphically as shown in Fig. 4.
2. **Fanout exchange:** This type of exchange routes message to the entire bounded queue. It is suitable for broadcast routing. If there are N number of queues bounded to fanout exchange, when a new message is published to that exchange by publisher, then a copy of a message delivered to all N queues. It basically delivers a copy of a message to every queue bounded to it, for example, group chat. Fanout exchange can be represented graphically as shown in Fig. 5.
3. **Topic exchange:** It is suitable for multicast routing of the messages. It matches routing key with routing patterns and based on that it routes the message to the particular queue. The routing pattern follows the same rules as the routing key with addition that single word matches with $*$ and zero or more word matches

Fig. 4 Direct exchange type**Fig. 5** Fanout exchange type

with #. For example, the routing pattern **.stock.#* matches the routing keys *abc.stock* and *abc.stock.mn* but not *stock.empty* [18, 19].

4. Header exchange: Header exchange is designed for routing on multiple message header attributes than a routing key. It ignores the routing key attribute. If the value of a header is equal to the value of binding, then the message is said to be matching. Header exchange type uses “x-match” binding argument. When bind queue to the header exchange using more than one header value, in this case, broker needs one more piece of information that message matches with any of the headers or all of them that referred to the x-match binding argument. If it is set to “any”, just one matching header value is sufficient; or if it is set to “all” then all the header values must match.

Message Queue: Message queues are stored and distributed entities. It stores messages on disk or in memory. Message queues are freely created, shared, used and destroyed within the limits to their authority. Messages have some priority level; in the same queue higher priority message is sent first and then the lower priority message. The server will first discard low-priority messages in order to maintain a specific service quality level.

Binding: To route the message from exchange to the message queue based on some specific criteria or rules that referred to as binding, it specifies the relation between message queue and exchange.

Needs of AMQP: AMQP provides full functional communication between client and message middleware server and queueing for later delivery that provides message delivery guarantee.

3 Message Life Cycle

The publisher applications create the message, place the content, set some message properties, label the message with routing information and then send the message to an exchange on the server. The message is received by exchange and routes it to the message queues. If a message is not routable, exchange may be silently dropped it or return the message to the producer.

AMQP provides functionality that a single message can exist on many message queues by doing copies of the message or by reference counting. When a message queue receives the message, it tries to immediately pass it to the subscriber application. If it is not possible, queue stores the message until subscriber is ready. When a message queue delivers the message to subscriber application, it removes the message from its internal buffer immediately or after subscriber's acknowledgement.

4 AMQP Frame Format

In AMQP, information is organised into a frame. As shown in Fig. 6, a frame is divided into three parts: the first is fixed-sized (8 bytes) frame header, the second is variable-sized extended header or payload and the third is a frame end. Frame header includes information necessary to parse the rest of the frames like the type of the frame, channel and the size of the frame. In a frame header, if "type = 1" then it is "Method frame" which carries remote procedural call (RPC) request and response; if "type = 2" then it is "Header frame" that use to establish the connection; if "type = 3" then it is "Body frame" which includes actual content of the message. If a message is too big, then split it into multiple frames and if "type = 4", then it is "Heartbeat frame" that is used to confirm that client is still alive. If frame channel value is zero,

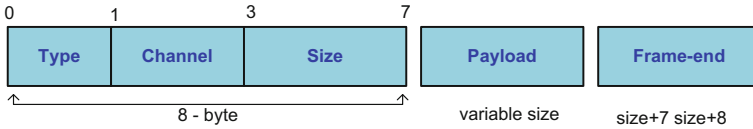


Fig. 6 Frame format

then it is used for global connection; if its value is between 1 and 65535, then it is a frame for specific channels. Payload depends on the type of the frame. Frame end is used to detect the framing error.

5 Proposed Implementation of AMQP

AMQP implements three types of message brokers or servers: RabbitMQ, OpenAMQ and Apache Qpid. Standard AMQP protocol is supported by RabbitMQ. It is an open-source message broker and queueing server. RabbitMQ server is written in Erlang. It is created on the Open Telecom Platform framework for fail over and clustering. When exchanging large amount of data between client and server, the best approach is to use the RabbitMQ server and use a back-end service to consume the messages, process them and send them to the database. In RabbitMQ, data is sent to back-end services and server queue. So, if the back-end service fails, the data is stored in the server. This approach is used to resource saving, prevent data loss and a better organisation of the messages.

OpenAMQ has a faster speed of message routing and high degree of robustness, WireAPI for publish/subscribe application programming. It is in the normal working state and OpenAMQ server opens the direct mode to improve the message forwarding performance. In direct mode, it is five times faster than the normal rate of message forwarding to improve efficiency and reduce delay. Apache Qpid is an open-source messaging system. It provides transaction management, queueing, distribution, security, management, clustering, federation and heterogeneous multi-platform support.

5.1 Status Update of Clients

In future, AMQP provides the feature for some cases or situations like when any client disconnected ungracefully or when any client connects and subscribes for particular topic which it is interested in. These features are used to notify another client about the disconnected client and help newly subscribed clients to get a status update immediately after subscribing and do not have to wait until publishing clients send the new update.

To notify other clients about an ungracefully disconnected client:

Some clients lost the connection; the battery is empty or in any other imaginable case, they will disconnect ungracefully from time to time. In order to take appropriate action, it would be good to know that if a connected client has disconnected gracefully or not. So this feature is used to notify other clients about an ungracefully disconnected client.

When each client is connected to a broker, they specify its message which includes Topic Name, Flag, Quality of Service (QoS) and Payload. Broker detects that the client has disconnected ungracefully till it stores the message. The broker sends the message to all subscribed clients on the topic, when the client has disconnected and stored message will be discarded by sending a DISCONNECT message. It helps to implement strategies when drops the connection of a client or at least informs to other clients about the offline status.

To help newly subscribed clients to get a status update immediately after subscribing to a topic:

A publishing client can only make sure that its message gets delivered safely to the broker. It has no guarantee that a message is actually received by a subscribing client. When a new client is connected and subscribed to the interesting topics, there is no guarantee when the subscriber will get the first message, because it totally depends on a publisher on that topic. It can take a few seconds, minutes or hours until the publisher sends a new message update on that topic. Until then, the new subscribing client is totally in the dark about the current status of that topic.

In the message, if Flag is set to "True", then the broker will store the last message and the corresponding QoS for that topic. So, when a new client is connected and subscribed to interesting topics, it will receive the message immediately after subscribing. For each topic only one message will be stored by the broker. So messages can help newly subscribed clients to get a status update immediately after subscribing to a topic and do not have to wait until a publishing client sends the next update.

6 Conclusion

AMQP provides asynchronous publish/subscribe communication with messaging. It is store-and-forward feature that ensures reliability even after network disruptions, which is its main advantage. It supports reliable communication via message delivery guarantee primitives including at most once, at least once and exactly once delivery. When compared all other IoT protocols with AMQP protocol, it gives better performance compared to others. Therefore, AMQP protocol provides the guarantee of message delivery and it is a reliable communication even after a network failure.

References

1. Al-Fuqaha, Ala, et al. *Internet of things: A survey on enabling technologies, protocols, and applications*, IEEE Communications Surveys & Tutorials 17.4 (2015): 2347–2376.
2. Karagiannis, Vasileios, et al., *A survey on application layer protocols for the internet of things*, Transaction on IoT and Cloud Computing 3.1 (2015): 11–17.
3. Luzuriaga, Jorge E., et al., *A comparative evaluation of AMQP and MQTT protocols over unstable and mobile networks*, 2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC). IEEE, 2015.
4. Maciej, Krzysztof Grochla, and Aleksander Seman, *Evaluation of highly available and fault-tolerant middleware clustered architectures using RabbitMQ*, Computer Science and Information Systems (FedCSIS), 2014 Federated Conference on. IEEE, 2014.
5. Xiong, Xuandong, and Jiandan Fu, *Active Status Certificate Publish and Subscribe Based on AMQP*, Computational and Information Sciences (ICCIS), 2011 International Conference on. IEEE, 2011.
6. Subramoni, Hari, et al., *Design and evaluation of benchmarks for financial applications using Advanced Message Queuing Protocol (AMQP) over InfiniBand*, High Performance Computational Finance, 2008. WHPCF 2008. Workshop on. IEEE, 2008.
7. Fernandes, Joel L., et al., *Performance evaluation of RESTful web services and AMQP protocol*, 2013 Fifth International Conference on Ubiquitous and Future Networks (ICUFN). IEEE, 2013.
8. Vinoski, Steve, *Advanced message queuing protocol*, IEEE Internet Computing 10.6 (2006): 87.
9. G. Panchal , A. Ganatra, Y. Kosta, D. Panchal, “Forecasting Employee Retention Probability using Back Propagation Neural Network Algorithm”, *IEEE 2010 Second International Conference on Machine Learning and Computing (ICMLC)*, Bangalore, India, pp. 248–251, 2010.
10. G. Panchal, A. Ganatra, P. Shah, D. Panchal, “Determination of over-learning and over-fitting problem in back propagation neural network”, *International Journal on Soft Computing*, vol. 2, no. 2, pp. 40–51, 2011.
11. G. Panchal, A. Ganatra, Y. Kosta, D. Panchal, “Behaviour analysis of multilayer perceptrons with multiple hidden neurons and hidden layers,” *International Journal of Computer Theory and Engineering*, vol. 3, no. 2, pp. 332–337, 2011.
12. G. Panchal and D. Panchal, “Solving np hard problems using genetic algorithm,” *International Journal of Computer Science and Information Technologies*, vol. 6, no. 2, pp. 1824–1827, 2015.
13. G. Panchal, D. Panchal, “Efficient attribute evaluation, extraction and selection techniques for data classification,” *International Journal of Computer Science and Information Technologies*, vol. 6, no. 2, pp. 1828–1831, 2015.
14. G. Panchal, D. Panchal, “Forecasting electrical load for home appliances using genetic algorithm based back propagation neural network,” *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, vol. 4, no. 4, pp. 1503–1506, 2015.
15. G. Panchal, D. Panchal, “Hybridization of Genetic Algorithm and Neural Network for Optimization Problem,” *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, vol. 4, no. 4, pp. 1507–1511, 2015.
16. Y. Kosta, D. Panchal, G. Panchal, A. Ganatra, “Searching most efficient neural network architecture using Akaike's information criterion (AIC),” *International Journal of Computer Applications*, vol. 1, no. 5, pp. 41–44, 2010.
17. *AMQP: Advanced Message Queuing, version 0.8, AMQP working group protocol specification*, June 2006 [Online] Available. <https://www.iana.com/opensource>.
18. Programming WireAPI, <http://www.openamq.org/>
19. Pivotal Software, Inc., Messaging that just works, [Online] Available: <https://www.rabbitmq.com>, 2014

Intelligent Communication and Computational
Technologies

Proceedings of Internet of Things for Technological
Development, IoT4TD 2017

Hu, Y.-C.; Tiwari, S.; Mishra, K.K.; Trivedi, M.C. (Eds.)

2018, XXIV, 392 p. 173 illus., Hardcover

ISBN: 978-981-10-5522-5