Istanbul Commerce University Journal of Science
İstanbul Ticaret Üniversitesi Fen Bilimleri Dergisi, 17(33), Bahar 2018
http://dergipark.gov.tr/ticaretfbd

ISTANBUL COMMERCE UNIVERSITY

Research Article

EXAMINATION AND COMPARISON OF THE COMMUNICATION PROTOCOLS ON THE APPLICATION LAYER IN IOT*

Cem GÜLTUNCA[1]   Prof. Dr. Abdül Halim ZAİM[2]

[1]İstanbul Ticaret Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı, Küçükyalı, İstanbul, Turkey, info@cemgultunca.com.tr orcid.org/0000-0003-4646-7311

[2] İstanbul Ticaret Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, Küçükyalı, İstanbul, Turkey, azaim@ticaret.edu.tr, orcid.org/0000-0002-0233-064X

Abstract

Today the Internet has become ubiquitous, has touched almost every corner of the globe, and is affecting human life in unimaginable ways. We are now entering an era of even more pervasive connectivity where a very wide variety of appliances will be connected to the web. We are entering an era of the "Internet of Things" (abbreviated as IOT). IOT is defined as a paradigm in which objects equipped with sensors, actuators, and processors communicate with each other to serve a meaningful purpose. Several IOT protocols have been introduced in order to provide an efficient communication for resource-constrained applications. However, their performance is not as yet well understood. I evaluated and compared four communication protocols, namely, AMQP, MQTT, XMPP, and COAP. I implemented a some IOT application using open source software for these protocols and measured their performance. In our tests, we compare AMQP and MQTT protocols. As a result, AMQP protocol transmits data faster than MQTT

Keywords: Internet of things, AMQP, MQTT, XMPP, COAP.

Araştırma Makalesi

NESNELERİN İNTERNETINDE KULLANILAN UYGULAMA KATMANINDA İLETİŞİM PROTOKOLLERİNİN İNCELENMESİ VE KARŞILAŞTIRILMASI

Öz

Bugün internet her yere yayıldı, dünyanın hemen her köşesine dokundu ve insan yaşamını önemli şekillerde etkiliyor. Şimdi, çok çeşitli cihazların ağa bağlanacağı daha da yaygın bir bağlantı çağına giriyoruz. Yani "Nesnelerin İnterneti" (IoT olarak kısaltılır) dönemine giriyoruz. IoT, sensörler, aktüatörler ve işlemcilerle donatılmış nesnelerin anlamlı bir amaca hizmet etmek için birbirleriyle iletişim kurduğu bir paradigma olarak tanımlanır. Kaynak kısıtlı uygulamalar için verimli iletişim sağlamak amacıyla birçok IoT protokolü tanıtılmıştır. Ancak, performansları henüz tam olarak anlaşılmamıştır. AMQP, MQTT, XMPP ve COAP olmak üzere dört iletişim protokolünü değerlendirdim ve karşılaştırdım. Bu protokoller için açık kaynaklı yazılım kullanan bazı IoT uygulamalarını uyguladım ve performanslarını ölçtüm. Bu makalede yaptığımız testlerde, AMQP ve MQTT protokollerini karşılaştırıyoruz. Sonuç olarak, AMQP protokolü verileri MQTT'den daha yüksek bir performans sağlamıştır.

Anahtar Kelimeler: Nesnelerin interneti, AMQP, MQTT, XMPP, COAP.

* Bu çalışma, İstanbul Ticaret Üniversitesi Fen Bilimleri Enstitüsü'nde yapılan "Nesnelerin İnternetinde Uygulama Katmanı Üzerindeki Haberleşme Protokollerinin İncelenmesi ve Deneysel Karşılaştırılması" başlıklı yüksek lisans tezinden hazırlanmıştır.

Received / Geliş tarihi: 07/02/2018   Accepted / Kabul tarihi: 19/05/2018
Corresponding Author/ Sorumlu Yazar :   info@cemgultunca.com.tr

1. INTRODUCTION

In the current era of Internet, “Internet of Things” finds a more substantial place in the life with its incrementally increasing importance. Smart home systems, intelligent cities, wearable technology, and devices with Internet connection became communicative with each other to facilitate our life. Concerning these topics, many projects are being developed and new products are being launched. Inter device connections are ensured through several protocols and communication methods. This article will address to some of these protocols such as MQTT, AMQP, COAP, and XMPP. Their differences and performance will be compared.

2. IOT PROTOCOLS

a. Message Queue Telemetry Transport (MQTT)

Message queue telemetry transport (MQTT) is a publish/subscribe protocol. It’s similar to the client-server model. However, its simplicity, and open source code make this protocol suited only for constrained environments, such as low power, limited computation capability and memory, and limited bandwidth. It’s suitable for IOT applications and machine to machine communications. MQTT protocol can run over TCP/IP. (Banks and Gupta,2004).

MQTT was designed by IBM, and by 2013 it was standardized by OASIS (Salman, bt.), it aims to reduce bandwidth requirement. In addition to guarantee reliability of packet delivery, MQTT provides a set of features that includes: the support of multi-cast communication (one to many message), and the capability to establish communications between remote devices. But the most important feature of this protocol is the minimization of network traffic by reducing transport overhead and protocol exchanges. In addition, it provides a notification mechanism when an abnormal situation occurs (Banks and Gupta,2004; Oh, vd 2010).

MQTT protocol has three options to achieve messaging Quality of Service (QOS) (Oh, vd. 2010).

One Delivery (At Most)

Messages are delivered according to the best effort of the network; an acknowledgment is not required. (Least level of QOS)

One Delivery (At Least)

Message sends at least once, some duplicate message may exist, and an acknowledgment message is required.

On Delivering (Exactly)

Require an additional protocol to ensure that the message is delivered once and only once. (Highest level of QOS)

b. Advanced Message Queuing Protocol (AMQP)

Advanced Message Queuing Protocol (AMQP) is a publish/subscribe model which depends on reliable and efficient messaging queue. It’s standardized by OASIS. Nowadays, AMQP is widely used in business and commercial platforms. The use of a publish/subscribe approach makes this protocol of high scalability (Bloebaum and Johnsen, 2015).

AMQP supports heterogeneity and interoperability characteristic communications among different devices that support different languages. Applications that belong to AMQP protocol are able to exchange messages one to another. AMQP protocol focuses on knowing a set of the specifications of messages to achieve reliability, security and performance (Fernandes vd. 2013)

The AMQP protocol does not tolerate too much loss of messages, so it focuses on losing message data. The AMQP protocol works over TCP as it provides a reliable point-to-point connection. In addition, endpoints must provide a confirmation of receipt of the message for each message. The protocol also defines an optional job order mode with multi-phase sequence processing capability. Looking at their roots, AMQP is a protocol that focuses on tracking all messages, trying to be sure that each message is independent of the error, arriving as desired.

AMQP protocol has four option to send messaging (Rabbitmq, bt)

-Point-To-Point (Direct Exchange)

A direct exchange delivers messages to queues based on the message routing key. A direct exchange is ideal for the unicast routing of messages

-Publish-Subscribe (Fanout Exchange)

A fanout exchange routes messages to all of the queues that are bound to it and the routing key is ignored. If N queues are bound to a fanout exchange, when a new message is published to that exchange a copy of the message is delivered to all N queues. Fanout exchanges are ideal for the broadcast routing of messages.

-Topic Exchange

Topic exchanges route messages to one or many queues based on matching between a message routing key and the pattern that was used to bind a queue to an exchange. The topic exchange type is often used to implement various publish/subscribe pattern variations. Topic exchanges are commonly used for the multicast routing of messages.

Topic exchanges have a very broad set of use cases. Whenever a problem involves multiple consumers/applications that selectively choose which type of messages they want to receive, the use of topic exchanges should be considered.

-Headers Exchange

A headers exchange is designed for routing on multiple attributes that are more easily expressed as message headers than a routing key. Headers exchanges ignore the routing key attribute. Instead, the attributes used for routing are taken from the headers attribute. A message is considered matching if the value of the header equals the value specified upon binding.

It is possible to bind a queue to a headers exchange using more than one header for matching. In this case, the broker needs one more piece of information from the application developer, namely, should it consider messages with any of the headers matching, or all of them? This is what the "x-match" binding argument is for. When the "x-match" argument is set to "any", just one matching header value is sufficient. Alternatively, setting "x-match" to "all" mandates that all the values must match.

Headers exchanges can be looked upon as "direct exchanges on steroids". Because they route based on header values, they can be used as direct exchanges where the routing key does not have to be a string; it could be an integer or a hash (dictionary) for example.

c. Extensible Messaging and Presence Protocol (XMPP)

Extensible Messaging and Presence Protocol (XMPP) nowadays is one of the most common communication and messaging protocol in IOT, it was standardized by the IETF. This protocol is a well-known protocol that was used broadly in all networks. The need of IOT can be addressed by XMPP protocol since it supports small messages and low latency; these characteristics make the XMPP protocol a good choice for IOT communications and messaging.

XMPP protocol supports both request/response and publish/subscribe models; request/response which allows bi- directional communications and publish/subscribe model which allows multi-directional communication (push and pull the data). High scalability in XMPP is provided by decentralized architecture. There are many extensions to XMPP protocol, this allows it to work on the infrastructure- less environment (Bendel vd 2013).

d. Constrained Application Protocol (COAP)

Constrained application protocol (COAP) is request/response protocol; it is similar to client-server model. Nevertheless, this protocol is only sufficient in constrained environment such as: constrained node with low capability in RAM or CPU, and constrained network, such as lower power using wireless personal area network (WPAN). This constrained environment led to bad packet delivery and high overhead. COAP was designed by Internet Engineering Task Force (IETF) which is mainly interested in machine to machine (m2m) applications and the automation of systems to reduce overhead, enhance packet delivery, and to increase the simplicity of work, by using simple interface with HTTP (Shelby vd 2014).

COAP supports publish/subscribe architecture, this architecture provides multicast communications, and the publisher sends the message so on the other hand multi-subscribers can catch the message and takes the actions. This scenario is done in an Asynchronous way. Publish /subscribe architecture is used to support a large number of users and provide better performance than the traditional way (Oh, vd. 2010)..

The most important features in COAP are simplicity and reliability; since it supports unicast and multicast request by taking advantage of UDP, and provide the ability to Asynchronous message exchanges. COAP is a single protocol with two layers, the first layer is the messaging layer and the second one is the request/response layer; messaging layer aims to achieve reliability based on UDP, while request/response layer aims to act the interactions and communication.

COAP uses different types of massages: Conformable Message, Non-conformable Message, Acknowledgement Message, Reset Message, Piggybacked Response, Separate Response, and Empty Message (Salman, bt.; Shelby vd 2014). The following points provide a brief description for each:

Conformable Message

This type of messages guarantees reliable communication by using the acknowledgment method; if the message arrives at the destination, it should propagate a return message of type acknowledgment or reset message.

Non-Conformable Message

In this type there is no need for an acknowledgment message.

Acknowledgment Message

This message means that conformable message arrives.

Reset Message

When a message (conformable, Non-conformable) arrives, but it misses critical and important part required for message interpretation. Propagate resets messages into an empty acknowledgment message.

Piggybacked Response

The receiver responses directly when receiving the message of the acknowledgment message.

Separate Response

The Receiver responses in a different message separate from the acknowledgment message.

3. PROTOCOL COMPRESSION

Table 1. IOT Protocols Comparison

	MQTT	**AMQP**	**XMPP**	**COAP**
Abstraction	Pub/Sub	P2P or Pub/Sub	P2P or Pub/Sub (based on draft spec/ XEP-0060	Request/Rep ly
Implementatio n Architecture	Brokered (most common)	Brokered (most common)	XMPP Server (broker)	Client-Server

User configurable QOS	3	3	None	Confirmable or non-confirmable messages
Interoperability	Partial	Yes	YES	YES
Hard Real-time	No	No	NO	NO
Transports	TCP	TCP	TCP	UDP
Subscription Control	Topics with hierarchical matching	Exchanges, Queues and Bindings in v0.9.1 standard, Queues and message filtering in v1.0 standard	Nodes which are analogous to a Topic defined In draft spec XEP- 0060	Provides support for Multicast addressing
Data Serialization	Undefined	AMQP type system or user defined	XML	Configurable
Standards	Proposed OASIS MQTT standard M	OASIS AMQP	XMPP Standards Foundation	Proposed IETF COAP standard
Encoding	Binary	Binary	Plain Text	Binary
Licensing Model	Open Source & Commercially Licensed	Open Source & Commercially Licensed	Open Source & Commercially Licensed	Open Source & Commercially Licensed

Dynamic Discovery	No	No	YES	YES
Mobile devices (Android, iOS)	Yes	YES	YES	Via HTTP proxy
6LoWPAN devices	Yes	Implementation specific	NO	YES
Multi-phase Transactions	Yes	YES	NO	NO
Security	Simple Username/Password Authentication, SSL for data encryption	SASL authentication, TLS for data encryption	TLS and SASL	DTLS

4. EXPERIMENT

To demonstrate the feasibility of Local Area Network, we have conducted two protocols experiments. We compare MQTT and AMQP

a. Experimental Setting

We have two raspberry pi 3 and macbook pro. Raspbian jessie operating system is running on both raspberry pi 3. Macbook Pro has sierra operating system. We setup network with TP-link switch (8 port – 1 Gbit bandwitdh).

To evaluate the performance of MQTT on a LAN environment, MQTT version 3.1 compliant implementations were used as follows. Mosquitto is an open source broker implementation written in the C language. We used the version 3.1. We wrote a sample code for test with Python programing language. We send 1000 message for test and measure sending time. We use 5 different message size for test.

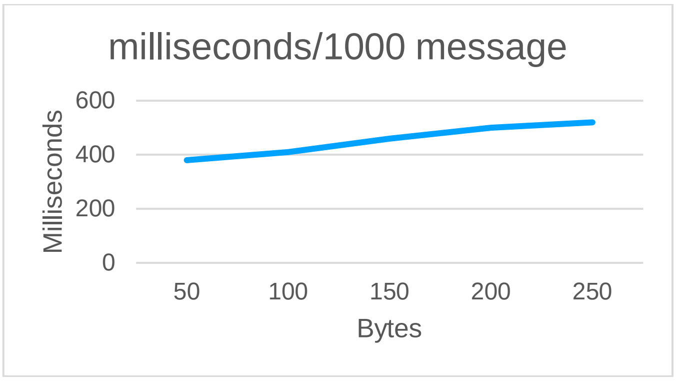

Fig. 1 MQTT transfer time while varying the message size.

Our second test to evaluate AMQP performance. We use RabbitMQ message broker and C# client for sending message. We send same messages for test.

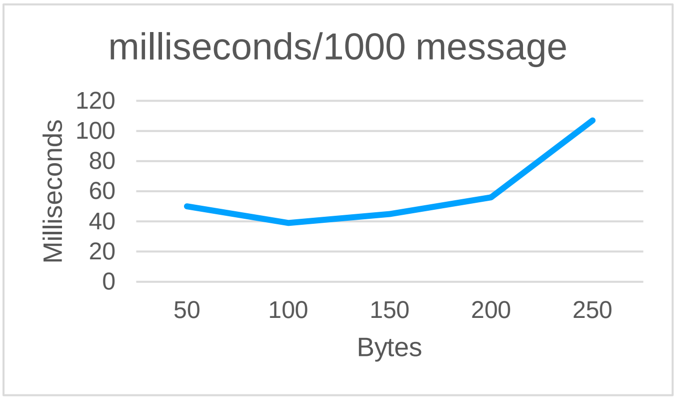

Fig. 2 AMQP transfer time while varying the message size.

5. CONCLUSION AND FUTURE WORK

This paper briefly discusses the most common application layer protocol in IOT environment, and focuses on the evaluation of each protocol in term of the architecture, communication model, security, and achieving the quality of services. This paper provides comprehensive comparison between the existing protocols.

As a result, IOT application layer protocols have advantages and disadvantages relative to each other in performance. In our tests, the AMQP protocol transmits data faster than MQTT. But if you start lot of message same time, CPU usage increased and system spend more power. We plan to develop to demonstrate the feasibility, we have conducted a preliminary performance evaluation of a commodity hardware environment, including Bluetooth Low Energy (BLE) network.

REFERENCES

Banks, A., and Gupta, R., (2014), MQTT Version 3.1. 1. OASIS Standard.

Bendel, S., Springer, T., Schuster, D., Schill, A., Ackermann, R., and Ameling, M., (2013, March), A service infrastructure for the internet of things based on XMPP. In Pervasive Computing and Communications Workshops (PERCOM Workshops), 2013 IEEE International Conference on (pp. 385-388). IEEE.

Bloebaum, T. H., and Johnsen, F. T., (2015), Evaluating publish/subscribe approaches for use in tactical broadband networks. In Military Communications Conference, MILCOM 2015-2015 IEEE (pp. 605-610). IEEE.

Fernandes, J. L., Lopes, I. C., Rodrigues, J. J., and Ullah, S., (2013, July), Performance evaluation of RESTful web services and AMQP protocol. In Ubiquitous and Future Networks (ICUFN), 2013 Fifth International Conference on (pp. 810-815). IEEE.

Oh, S., Kim, J. H., and Fox, G., (2010), Real-time performance analysis for publish/subscribe systems. Future Generation Computer Systems, 26 (3), 318-323.

Salman, T., Internet of Things Protocols and Standards.

Shelby, Z., Hartke, K., and Bormann, C., (2014), The constrained application protocol.

https://www.rabbitmq.com/tutorials/amqp-concepts.html

PrismTech Corporation, Messaging Technologies for the Industrial Internet and the Internet of Things Whitepaper.