A Survey on Application Layer Protocols for the Internet of Things

Vasileios Karagiannis[1], Periklis Chatzimisios[1], Francisco Vazquez-Gallego[2], Jesus Alonso-Zarate[2]
[1] CSSN Research Lab, Department of Informatics, Alexander TEI of Thessaloniki, Greece
basilkaragiannis@gmail.com, peris@it.teithe.gr
[2] Centre Tecnologic de Telecomunicacions de Catalunya (CTTC), Spain
[francisco.vazquez, jesus.alonso]@cttc.es

ABSTRACT

It has been more than fifteen years since the term Internet of Things (IoT) was introduced. However, despite the efforts of research groups and innovative corporations, still today it is not possible to say that the IoT is upon us. This is mainly due to the fact that a unified IoT architecture has not yet been clearly defined and there is no common agreement in defining communication protocols and standards for all the IoT parts. The framework that current IoT platforms use consists mostly in technologies that partially fulfill the IoT requirements. While developers employ existing technologies to build the IoT, research groups are working on adapting protocols to the IoT in order to optimize communications. In this paper, we present and compare existing IoT application layer protocols as well as protocols that are utilized to connect the “things” but also end-user applications to the Internet. We highlight IETF’s CoAP, IBM’s MQTT, HTML 5’s Websocket among others, and we argue their suitability for the IoT by considering reliability, security, and energy consumption aspects. Finally, we provide our conclusions for the IoT application layer communications based on the study that we have conducted.

Keywords: *Internet of Things (IoT), Application Layer Protocols, Request/Response, Publish/Subscribe.*

1. INTRODUCTION

The IoT envisions hundreds or thousands of end-devices with sensing, actuating, processing, and communication capabilities able to be connected to the Internet [1]. These devices can be either directly connected using cellular technologies, such as 2G/3G/Long Term Evolution and beyond (5G), or through a gateway, forming a local area network, to establish connection to the Internet. The latter is the case where the end-devices usually form Machine to Machine (M2M) area or capillary networks using various radio technologies, such as Zigbee (based on the IEEE 802.15.4 Standard), Wi-Fi (based on the IEEE 802.11 Standard), 6LowPAN over Zigbee (IPv6 over Low Power Personal Area Networks), or Bluetooth (based on the IEEE 802.15.1).

Regardless the specific wireless technology used to deploy the M2M network, all the end-devices should make their data available to the Internet [2]. This can be achieved either by sending the information to a proprietary web server accessible from the Internet or by employing the cloud. Online platforms such as ThingSpeak.com or Open.Sen.se, among many alternatives, are M2M clouds able to receive, store, and process data. Besides acting as remote data bases, M2M clouds also provide the following key services:

1. They offer Application Programming Interfaces (API) with built-in functions for end-users, thus providing the option to monitor and control end-devices remotely from a client device.
2. They act as asynchronous intermediate nodes between the end-devices and final applications running on devices such as smart phones, tablets or desktops.

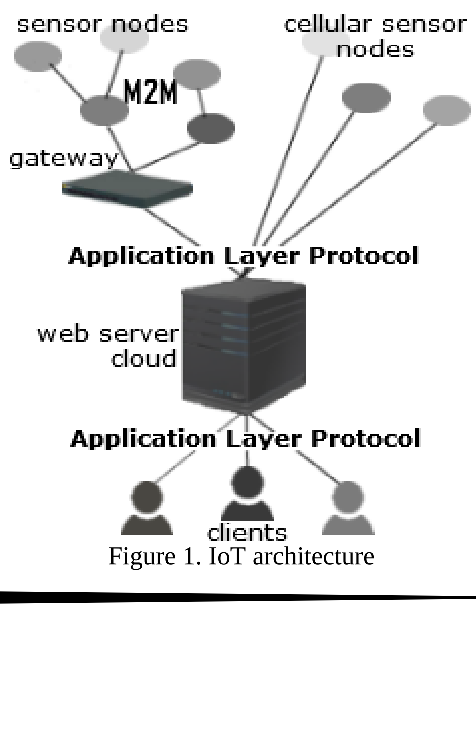

Figure 1. IoT architecture

Our paper focuses on the protocols that handle the communication between the gateways, the public Internet, and the final applications (Figure 1). They are application layer protocols that are used to update online servers with the latest end-device values but also to carry commands from applications to the end-device actuators.

The rest of the paper is organized as follows. Section 2 describes our research motivation, whereas each of the other sections is dedicated to a specific application layer protocol. At the first part of each section, we introduce an application layer protocol, we present its usage, we discuss its reliability and security features, and we then compare its suitability for the IoT with other application layer protocols. Finally, in Section 9, we present overall conclusions based on the previous sections and we provide further research areas.

2. RESEARCH MOTIVATION

The IoT is a term used for a huge wave of innovation originated in industries, but currently heading to urban centers, in-home environments, and individuals.

Our main motivation was to create an IoT testbed in which we could test communications protocols and also innovative applications that could be applied to a gamut of scenarios. While searching for the appropriate application layer protocols to use, we found out that while comparisons can be found between two protocols, there is no paper overviewing all the possible alternatives with pros and cons.

The main motivation of this paper is to fill this gap and to provide a brief yet accurate description of the key protocols that are being used today to implement the IoT. More specifically, we will discuss the following list of protocols being used alternatively or jointly to solve different needs of the communication between machines:

1) **CoAP**: Constrained Application Protocol.
2) **MQTT**: Message Queue Telemetry Transport.
3) **XMPP**: Extensible Messaging and Presence Protocol.
4) **RESTFUL Services**: Representational State Transfer.
5) **AMQP**: Advanced Message Queuing Protocol
6) **Websockets**.

3. COAP

The Constrained Application Protocol (CoAP) [5] is a synchronous request/response application-layer protocol that was designed by the Internet Engineering Task Force (IETF) to target constrained-recourse devices. It was designed by using a subset of the HTTP methods making it interoperable with HTTP [3].

CoAP runs over UDP to keep the overall implementation lightweight. It uses the HTTP commands GET, POST, PUT, and DELETE to provide resource-oriented interactions in a client-server architecture. CoAP is a request/response protocol that utilizes both synchronous and asynchronous responses. The reason for designing a UDP-based application layer protocol to manage the resources is to remove the TCP overhead and reduce bandwidth requirements [4]. Additionally, CoAP supports unicast as well as multicast, as opposed to TCP, which is by its nature not multicast-oriented.

Running on the unreliable UDP, CoAP integrated its own mechanisms for achieving reliability. Two bits in the header of each packet state the type of message and the required Quality of Service (QoS) level. There are 4 message types:

1. *Confirmable:* A request message that requires an acknowledgement (ACK). The response can be sent either synchronously (within the ACK) or if it needs more computational time, it can be sent asynchronously with a separate message.
2. *Non-Confirmable:* A message that does not need to be acknowledged.
3. *Acknowledgment:* It confirms the reception of a confirmable message.
4. *Reset:* It confirms the reception of a message that could not be processed.

There is also a simple Stop-and-Wait retransmission mechanism for confirmable messages and a 16-bit header field in each CoAP packet called *Message ID* which is unique and used for detecting duplicates.

CoAP–HTTP Mapping enables CoAP clients to access resources on HTTP servers through a reverse proxy that translates the HTTP Status codes to the Response codes of CoAP [5].

Even though CoAP was created for the IoT and for M2M communications, it does not include any built-in security features. The protocol that is proposed to secure CoAP transactions is the Datagram Transport Layer Security (DTLS). DTLS runs on top of UDP and is the analogous of TLS for the TCP. It provides authentication, data integrity, confidentiality, automatic key management, and cryptographic algorithms [6]. Even though DTLS secures UDP transfers, it was not designed for the IoT, thus its suitability can be argued. To begin with, DTLS does not support multicast [6], which is a prime advantage of CoAP compared to other application layer protocols. DTLS handshakes [7] require additional packets that increase the network traffic, occupy additional computational resources, and shorten the lifespan of mobile devices that run on batteries, an essential part of the IoT. Being designed for the IoT, CoAP is HTTP-compatible, but CoAP over DTLS might create additional confusion to the HTTP servers due to its diverse packet structure. Other protocols (IPsec, Lithe) for securing CoAP can be found in the literature including approaches that are still being under research [6]-[7].

4. MQTT

Message Queue Telemetry Transport (MQTT) [8] was released by Andy Stanford-Clark of IBM, and Arlen Nipper of Arcom and targets lightweight M2M communications. It is an asynchronous publish/subscribe protocol that runs on top of the TCP stack. Publish/subscribe protocols meet better the IoT requirements than request/response since clients do not have to request updates thus, the network bandwidth is decreasing and the need for using computational resources is dropping.

In MQTT there is a broker (server) [8] that contains topics. Each client can be a publisher that sends information to the broker at a specific topic or/and a subscriber that receives automatic messages every time there is a new update in a topic he is subscribed. The MQTT protocol is designed to use bandwidth and battery usage sparingly, which is why, for example, it is currently used by Facebook Messenger [9].

MQTT ensures reliability by providing the option of three QoS levels:

1. *Fire and forget:* A message is sent once and no acknowledgement is required.
2. *Delivered at least once:* A message is sent at least once and an acknowledgement is required.
3. *Delivered exactly once:* A four-way handshake mechanism is used to ensure the message is delivered exactly one time.

Even though MQTT runs on TCP, it is designed to have low overhead compared to other TCP-based application layer protocols [10]. Moreover, the publish/subscribe architecture that it used, is more suitable for the IoT than request/response of CoAP, for example, because messages do need to be responded. This means lower network bandwidth and less message processing that actually extends the lifetime of battery-run devices.

To ensure security, MQTT brokers may require username/password authentication which is handled by TLS/SSL (Secure Sockets Layer), i.e., the same security protocols that ensure privacy for HTTP transactions all over the Internet.

By comparing MQTT with the aforementioned CoAP, it is possible to see that the UDP-based CoAP has lower overhead than the TCP-based MQTT. However, due to the lack of TCP's retransmission mechanisms, packet loss is more likely to happen when using CoAP. According to a recent research study [10], MQTT experiences lower delays that CoAP for low packet losses, but CoAP generates less extra traffic for ensuring reliability. However, results can vary depending on the network conditions. Additionally packet loss and delays depend on the QoS of the messages. In both protocols, packet loss degrades and delays increase when the QoS level is higher.

5. XMPP

The Extensible Messaging and Presence Protocol (XMPP) was designed for chatting and message exchanging. It was standardized by the IETF over a decade ago, thus being a well-proven protocol that has been used widely all over the Internet. However, being an old protocol, it falls short to provide the required services for some of the new arising data applications. For this reason, last year, Google stopped supporting the XMPP standard due to the lack of worldwide support [11]. However, lately XMPP has re-gained a lot of attention as a communication protocol suitable for the IoT.

XMPP runs over TCP and provides publish/subscribe (asynchronous) and also request/response (synchronous) messaging systems. It is designed for near real-time communications and thus, it supports small message footprint and low latency message exchange [12]. As the name explicitly states, XMPP is extensible and allows the specification of XMPP Extension Protocols (XEP) that increase its functionality.

XMPP has TLS/SSL security built in the core of the specification. However, it does not provide QoS options that make it impractical for M2M communications. Only the inherited mechanisms of TCP ensure reliability.

XMPP supports the publish/subscribe architecture that is more suitable for the IoT in contrast to CoAP's request/response approach. Furthermore, it is an already established protocol that is supported all over the Internet as a plus with regard to the relatively new MQTT [13]. However, XMPP uses XML messages (eXtensible Markup Language) that create additional overhead due to unnecessary tags and require XML parsing that needs additional computational ability which increases power consumption.

6. RESTFUL SERVICES

The Representational State Transfer (REST) is not really a protocol but an architectural style. It was first introduced by Roy Fielding in 2000 [14], and it is being widely used ever since.

REST uses the HTTP methods GET, POST, PUT, and DELETE to provide a resource-oriented messaging system where all actions can be performed simply by using the synchronous request/response HTTP commands. It uses the built-in accept header of HTTP to indicate the format of the data that it contains. The content type can be XML or JSON (JavaScript Object Notation) and depends on the HTTP server and its configuration. REST is already an important part of the IoT because it is supported by all the commercial M2M cloud platforms. Moreover it can be implemented in smartphone and tablet applications easily because it only requires an HTTP library which is available for all the Operating Systems (OS) distributions. The features of HTTP can be completely utilized in the REST architecture including cashing, authentication, and content type negotiation [15].

RESTful services use the secure and reliable HTTP which is the proven worldwide Internet language. It can make use of TLS/SSL for security. However, today most commercial M2M platforms do not support HTTP requests. Instead, they provide unique authentication keys that need to be in the header of each request to achieve some level of security.

Even though REST is already used widely in commercial M2M platforms, it is unlikely that it will become a dominant protocol due to not being easily implementable. It uses HTTP which means no compatibility with constrained-communication devices. This leaves its use for final applications.

Given the current tendency for applications running on smartphones, tablets and pads, the additional overhead associated to request/response protocols affect battery usage, as it also does the continuous polling or long polling for values especially when there are no new updates and the overhead becomes useless. Issues that can be avoided if a publish/subscribe protocol is used such as MQTT or XMPP. CoAP on the other hand, which is the lightweight version of REST, bears the same disadvantages of the request/response architecture. However it is designed to run over UDP making it capable of being used by constrained resource devices, counter to REST.

7. AMQP

The Advanced Message Queuing Protocol (AMQP) is a protocol that arose from the financial industry. It can utilize different transport protocols but it assumes an underlying reliable transport protocol such as TCP [16].

AMQP provides asynchronous publish/subscribe communication with messaging. Its main advantage is its store-and-forward feature that ensures reliability even after network disruptions [17]. It ensures reliability with the following message-delivery guarantees [16]:

1. *At most once:* means that a message is sent once either if it is delivered or not.
2. *At least once:* means that a message will be definitely delivered one time, possibly more.
3. *Exactly once:* means that a message will be delivered only one time.

Security is handled with the use of the TLS/SSL protocols over TCP.

Recent research has shown that AMQP has low success rate at low bandwidths, but it increases as bandwidth increases [17]. Another study shows that comparing AMQP with the aforementioned REST, AMQP can send a larger amount of messages per second [18]. Additionally, it has been reported that an AMQP environment with 2,000 users spread across five continents can process 300 million messages per day [18]. Furthermore, JPMorgan which is an American banking and financial services company uses AMQP to send 1 billion messages per day [19].

AMQP is already in use and its performance has been outstanding. Its main difference comparing it to MQTT and CoAP is that AMQP targets transactions and aims at being an efficient messaging system, while CoAP and MQTT target hardware devices and M2M networks. Nonetheless, implementing the IoT requires both messaging systems and lightweight protocols for the machines.

8. WEBSOCKET

The Websocket protocol [20] was developed as part of the HTML 5 initiative to facilitate communications channels over TCP. Websocket is neither a request/response nor a publish/subscribe protocol. In Websocket, a client initializes a handshake with a server to establish a Websocket session. The handshake process is intended to be compatible with HTTP-based server-side software so that a single port can be used by both HTTP and Websocket clients [20]. However, what comes after the handshake does not comply with the HTTP rules. In fact, during a session, the HTTP headers are removed and clients and servers can exchange messages in an asynchronous full-duplex connection. The session can be terminated when it is no longer needed from either the server or the client side. Websocket was created to reduce the Internet communication overhead while providing real-time full-duplex communications. There is also a Websocket sub-protocol called Websocket Application Messaging Protocol (WAMP) that provides publish/subscribe messaging systems.

Websocket runs over the reliable TCP and implements no reliability mechanisms on its own. If needed, the sessions can be secured using the Websocket over TLS/SSL.

During the session, Websocket messages have only 2 bytes of overhead. As reported by relevant studies [21], the HTTP polling (in REST) repeats header information when the data transmission rate increases, thus increasing latency. Websocket is estimated to provide a three-to-one reduction in latency against the half-duplex HTTP polling. Websocket is not designed for resource constrained devices as the previous protocols and its client/server based architecture does not suit IoT applications. However it is designed for real-time communication, it is secure, it minimizes overhead and with the use of WAMP it can provide efficient messaging systems. Thus, it can compete any other protocol running over TCP.

9. CONCLUSIONS & FUTURE WORK

In this paper, we have presented a common IoT architecture by describing the parts where application layer protocols are needed to handle communication. We have presented the most representative application layer protocols that have gained attention for IoT while providing a comparison among each other and argue their suitability for the future of the IoT. Among them, we have identified IETF's CoAP as the only one that runs over UDP, thus making it the most lightweight, followed by HTML 5's Websocket that significantly reduces the communication's overhead. The computational and communication ability of the devices involved should also be taken into consideration when choosing the most appropriate protocol. If constrained communication and battery consumption is not an issue, RESTful services can be easily implemented and interact with the Internet using the worldwide HTTP. This can be proved very useful in testbeds as it can work as proof of concept for final applications. On the contrary, MQTT, which is used by Facebook Messenger, is not as widely used as HTTP but has proved to be more energy efficient for battery-operated devices. Additionally if the target applications require massive updates of the same value, publish/subscribe protocols (e.g. MQTT, XMPP, AMQP) are more suitable.

Protocol	**Transport**	**QoS options**	**Architecture**	**Security**
CoAP	UDP	YES	Request / Response	DTLS
MQTT	TCP	YES	Publish / Subscribe	TLS/ SSL
XMPP	TCP	NO	Request / Response Publish / Subscribe	TLS/ SSL
REST	HTTP	NO	Request / Response	HTTPS
AMQP	TCP	YES	Publish / Subscribe	TLS/ SSL
Web socket	TCP	NO	Client / Server Publish / Subscribe	TLS/ SSL

Table 1. Major differences among protocols

To sum up, there are several factors that influence the selection of an application layer protocol. The most important factors are the computational and communication ability of the end-devices, energy consumption and final application in mind. For this reason, opinions differ. An overview of major differences among the aforementioned protocols can be found above (Table 1).

Having seen this paper purely qualitatively, future work will be aimed at implementing all these protocols in order to obtain an experimental and quantifiable comparison among them. Moreover, we plan to explore the possibility of creating a server that supports multiple application layer protocols and dynamically chooses the most appropriate according to the network's conditions.

Such an innovative approach not designed so far, would optimize the overall performance of the IoT in various application scenarios.

REFERENCES

[1] Tasos Kaukalias and Periklis Chatzimisios, “Internet of Things (IoT) – Enabling technologies, applications and open issues”, in Encyclopedia of Information Science and Technology (3rd Ed.), IGI Global Press, 2014.

[2] Periklis Chatzimisios, Industry Forum & Exhibition Panel on "Internet of Humans and Machines", IEEE Global Communications Conference (Globecom 2013), Atlanta, USA, December 2013.

[3] Angelo P. Castellani, Mattia Gheda, Nicola Bui, Michele Rossi, Michele Zorzi, “Web Services for the Internet of Things through CoAP and EXI”, *IEEE International Conference on Communications Workshops (ICC),* 5-9 June 2011, pp. 1 – 6.

[4] Sye Loong Keoh, Sandeep S. Kumar, Hannes Tschofenig, “Securing the Internet of Things: A Standardization Perspective”, *Internet of Things Journal IEEE (Volume: 1, Issue: 3),* June 2014, pp. 265 – 275.

[5] Maria Rita Palattella, Nicola Accettura, Xavier Vilajosana, Thomas Watteyne, Luigi Alfredo Grieco, Gennaro Boggia, Mischa Dohler, “Standardized Protocol Stack for the Internet of (Important) Things”, *Communications Surveys & Tutorials IEEE (Volume:15 , Issue: 3),* 2013, pp. 1389 – 1406.

[6] Thamer A. Alghamdi, Aboubaker Lasebae, Mahdi Aiash, “Security Analysis of the Constrained Application Protocol in the Internet of Things”, *Second International Conference on Future Generation Communication Technology (FGCT),* 12-14 Nov. 2013, pp. 163 – 168.

[7] Shahid Raza, Hossein Shafagh, Kasun Hewage, René Hummen, Thiemo Voigt, “Lithe: Lightweight Secure CoAP for the Internet of Things”, Sensors Journal, IEEE (Volume: 13, Issue: 10), Oct. 2013, pp. 3711 – 3720.

[8] Shinho Lee, Hyeonwoo Kim, Dong-kweon Hong, Hongtaek Ju, “Correlation Analysis of MQTT Loss and Delay According to QoS Level”, *International Conference on Information Networking (ICOIN),* 28-30 Jan. 2013, pp. 714 – 717.

[9] http://mqtt.org/2011/08/mqtt-used-by-facebook-messenger, cited 28 Jul 2014.

[10] Dinesh Thangavel, Xiaoping Ma, Alvin Valera, Hwee-Xian Tan, Colin Keng-Yan Tan, “Performance Evaluation of MQTT and CoAP via a Common Middleware”, *IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP),* 21-24 April 2014, pp. 1 – 6.

[11] http://www.zdnet.com/google-moves-away-from-the-xmpp-open-messaging-standard-7000015918/, cited 28 Jul 2014.

[12] Sven Bendel, Thomas Springer, Daniel Schuster, Alexander Schill, Ralf Ackermann, Michael Ameling, “A Service Infrastructure for the Internet of Things based on XMPP”, *IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops),* 18-22 March 2013, pp. 385 – 388.

[13] Michael Kirsche, Ronny Klauck, “Unify to Bridge Gaps: Bringing XMPP into the Internet of Things”, *IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops),* 19-23 March 2012, pp. 455 - 458.

[14] Roy Thomas Fielding, "Architectural Styles and the Design of Network-based Software Architectures", PhD thesis, University of California, Irvine, USA, 2000.

[15] Bipin Upadhyaya, Ying Zou, Hua Xiao, Joanna Ng, Alex Lau, ”Migration of SOAP-based Services to RESTful Services”, *13th IEEE International Symposium on Web Systems Evolution (WSE),* 30 Sept. 2011, pp. 105 – 114.

[16] http://en.wikipedia.org/wiki/Advanced_Message_Queuing_Protocol, cited 28 Jul 2014.

[17] Frank T. Johnsen, Trude H. Bloebaum, Morten Avlesen, Skage Spjelkavik, Bjørn Vik, “Evaluation of Transport Protocols for Web Services”, *Military Communications and Information Systems Conference (MCC),* 7-9 Oct. 2013, pp. 1 – 6.

[18] Joel L. Fernandes, Ivo C. Lopes, Joel J. P. C.Rodrigues, Sana Ullah, “Performance Evaluation of RESTful Web Services and AMQP Protocol”, *Fifth International Conference on Ubiquitous and Future Networks (ICUFN),* 2-5 July 2013, pp. 810 – 815.

[19] Notable AMQP users, http://www.amqp.org/about/examples, cited 28 Jul 2014.

[20] I.Fette, A.Melnikov, “The WebSocket Protocol”, RFC 6455, Dec 2011.

[21] Victoria Pimentel, Bradford G. Nickerson, “Communicating and Displaying Real-Time Data with WebSocket”, *Internet Computing IEEE (Volume: 16, Issue: 4)*, July-Aug. 2012, pp. 45 – 53.