Choice of Effective Messaging Protocols for IoT Systems: MQTT, CoAP, AMQP and HTTP

Nitin Naik
Defence School of Communications and Information Systems
Ministry of Defence, United Kingdom
Email: nitin.naik100@mod.gov.uk

***Abstract*—The standard and real-time communication technology is an unalloyed inevitability for the development of Internet of Things (IoT) applications. However, the selection of a standard and effective messaging protocol is a challenging and daunting task for any organisation because it depends on the nature of the IoT system and its messaging requirements. Copious messaging protocols have been developed and employed by various organisations based on their requirements in the last two decades. Though, none of them is able to support all messaging requirements of all types of IoT systems. Messaging protocol is an ongoing dilemma for the IoT industry; consequently, it is important to understand the pros and cons of the widely accepted and emerging messaging protocols for IoT systems to determine their best-fit scenarios. Therefore, this paper presents an evaluation of the four established messaging protocols MQTT, CoAP, AMQP and HTTP for IoT systems. Firstly, it presents the broad comparison among these messaging protocols to introduce their characteristics comparatively. Afterwards, it performs a further in-depth and relative analysis based on some interrelated criteria to gain insight into their strengths and limitations. Thus, based on this detailed evaluation, the user can decide their appropriate usage in various IoT systems according to their requirements and suitability.**

Keywords*—*IoT Systems; M2M Communication; Messaging Protocol; MQTT; CoAP; AMQP; HTTP; Quality of Services; Interoperability

I. Introduction

In the Internet of Things (IoT), everyday things and machines are in the lead role and communicate with each other. These IoT networks employ various radio technologies such as Radio-Frequency Identification (RFID), WLAN (IEEE 802.11), WPAN (IEEE 802.15) and WMAN (IEEE 802.16) for communications at the lower level [1]. Irrespective of the specific radio technology used to deploy the Machine-to-Machine (M2M) network, all end-devices should make their data available to the industrial Internet [2]. Industrial Internet can be considered as the connection of industrial machine sensors and actuators to the Internet that can independently generate value [3]. One of the major factors that determine the performance of this M2M communication is the messaging protocol specially designed for M2M communications within the IoT applications. The selection of a standard and effective messaging protocol is a challenging and daunting task for any organisation [4]. While selecting an appropriate messaging protocol for IoT systems, the pre-requisite is the better understanding of a target IoT system and its message/data sharing requirements.

Unlike the Web, which uses a single standard messaging protocol HTTP, IoT cannot rely on a single protocol for all its need [5]. Consequently, hundreds of messaging protocols are available to choose for various types of requirements of the IoT system. Some of them have been designed to address applications requiring fast and reliable business transactions such as AMQP and JMS [3], [6]. A numerous have been designed to address applications requiring data collection (e.g. sensor updates) in constrained network such as MQTT and CoAP [7], [8], [9]. Many of them have been designed to address applications requiring instant messaging (IM) and online presence detection such as XMPP and SIP [3]. A few of them have been designed to address web applications requiring communicating over the Internet such as RESTful client/server protocols HTTP and CoAP [7], [10]. This clearly shows that the future of the IoT lies on several messaging protocols and any one protocol cannot deal with all possible IoT use cases. Consequently, it is necessary to investigate the pros and cons of the widely accepted and emerging messaging protocols for IoT systems to determine their best-fit scenarios. Therefore, this paper presents an evaluation of the four messaging protocols MQTT, CoAP, AMQP and HTTP. Firstly, it presents the general comparison among these protocols to introduce their characteristics comparatively. Subsequently, it performs a further in-depth and relative analysis based on some interrelated criteria to gain insight into their strengths and limitations. For making this relative analysis easy, it is illustrated using simple graphs to render a nimble and broader view of each protocol with respect to other protocols for an ordinary user. Accordingly, the user can decide their relevant usage in IoT systems based on their requirements and suitability.

The remainder of this paper is organised as follows: Section II elucidates the theoretical background of the four widely accepted and emerging messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP; Section III demonstrates a comparative analysis of these four messaging protocols for providing their general information; Section IV presents an in-depth and relative analysis of these messaging protocols for revealing their relative strengths and limitations; Section V concludes the paper and suggests some future work.

II. Messaging Protocols for IoT Systems

This section presents the four widely accepted and emerging messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP, which are shown at the top of the protocol stack for IoT systems in Fig. 1.

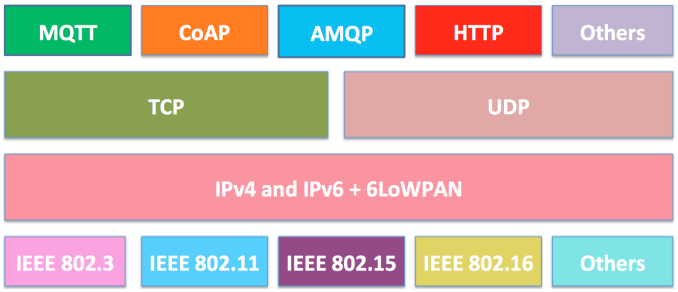

Fig. 1: Protocol Stack for IoT Systems

A. MQTT (Message Queuing Telemetry Transport Protocol)

MQTT is one of the oldest M2M communication protocols, which was introduced in 1999. It was developed by Andy Stanford-Clark of IBM and Arlen Nipper of Arcom Control Systems Ltd (Eurotech). It is a publish/subscribe messaging protocol designed for lightweight M2M communications in constrained networks [7]. MQTT client publishes messages to an MQTT broker, which are subscribed by other clients or may be retained for the future subscription. Every message is published to an address, known as a topic [11]. Clients can subscribe to multiple topics and receives every message published to the each topic. MQTT is a binary protocol and normally requires fixed header of 2-bytes with small message payloads up to maximum size of 256 MB [9]. It uses TCP as a transport protocol and TLS/SSL for security. Thus, communication between client and broker is a connection-oriented. Another great feature of MQTT is its three levels of Quality of Service (QoS) for reliable delivery of messages [7]. MQTT is most suitable for large networks of small devices that need to be monitored or controlled from a back-end server on the Internet. It is neither designed for device-to-device transfer nor for multicast data to many receivers [11]. It is a very basic messaging protocol offering only a few control options.

B. CoAP (Constrained Application Protocol)

CoAP is a lightweight M2M protocol from the IETF CoRE (Constrained RESTful Environments) Working Group. CoAP supports both request/response and resource/observe (a variant of publish/subscribe) architecture [7]. CoAP is mainly developed to interoperate with HTTP and the RESTful Web through simple proxies. Unlike MQTT, CoAP uses Universal Resource Identifier (URI) instead of topics [9]. Publisher publishes data to the URI and subscriber subscribes to a particular resource indicated by the URI. When a publisher publishes new data to the URI, then all the subscribers are notified about the new value as indicated by the URI. CoAP is a binary protocol and normally requires fixed header of 4-bytes with small message payloads up to maximum size dependent on the web server or the programming technology [9]. CoAP uses UDP as a transport protocol and DTLS for security [12]. Thus, clients and servers communicate through connectionless datagrams with less reliability. However, it uses "confirmable" or "non-confirmable" messages to provide two different levels of QoS. Where, confirmable messages must be acknowledged by the receiver with an ACK packet and non-confirmable messages are not. CoAP offers more functionality than MQTT such as it supports content negotiation to express a preferred representation of a resource; this allows client and server to evolve independently, adding new representations without affecting each other.

C. AMQP (Advanced Message Queuing Protocol)

AMQP is a lightweight M2M protocol, which was developed by John O'Hara at JPMorgan Chase in London, UK in 2003. It is a corporate messaging protocol designed for reliability, security, provisioning and interoperability [3]. AMQP supports both request/response and publish/subscribe architecture [13]. It offers a wide range of features related to messaging such as a reliable queuing, topic-based publish-and-subscribe messaging, flexible routing and transactions [3]. AMQP communication system requires that either the publisher or consumer creates an "exchange" with a given name and then broadcasts that name. Publishers and consumers use the name of this exchange to discover each other. Subsequently, a consumer creates a "queue" and attaches it to the exchange at the same time. Messages received by the exchange have to be matched to the queue via a process called "binding". AMQP exchanges messages in various ways: directly, in fanout form, by topic, or based on headers. AMQP is a binary protocol and normally requires fixed header of 8-bytes with small message payloads up to maximum size dependent on the broker/server or the programming technology [14], [15]. AMQP uses TCP as a default transport protocol and TLS/SSL and SASL for security [13]. Thus, the communication between client and broker is a connection-oriented. Reliability is one of the core features of AMQP, and it offers two preliminary levels of Quality of Service (QoS) for delivery of messages: Unsettle Format (not reliable) and Settle Format (reliable) [3].

D. HTTP (Hyper Text Transport Protocol)

HTTP is predominantly a web messaging protocol, which was originally developed by Tim Berners-Lee. Later, it was developed by IETF and W3C jointly and first published as a standard protocol in 1997 [13]. HTTP supports request/response RESTful Web architecture. Analogous to CoAP, HTTP uses Universal Resource Identifier (URI) instead of topics. Server sends data through the URI and client receives data through particular URI. HTTP is a text-based protocol and it does not define the size of header and message payloads rather it depend on the web server or the programming technology. HTTP uses TCP as a default transport protocol and TLS/SSL for security [10]. Thus, communication between client and server is a connection-oriented. It does not explicitly define QoS and requires additional support for it. HTTP is a globally accepted web messaging standard offers several features such as persistent connections, request pipelining, and chunked transfer encoding [4], [5], [10].

III. Comparative Analysis of Messaging Protocols for IoT Systems: HTTP, CoAP, AMQP and MQTT

This section presents a comparative analysis of the four widely accepted and emerging messaging protocols for IoT systems MQTT, CoAP, AMQP and HTTP based on several criteria to introduce their characteristics comparatively. This complete comparative study is shown in Table I.

TABLE I: Comparative Analysis of Messaging Protocols for IoT Systems: MQTT, CoAP, AMQP and HTTP

Criteria	MQTT	CoAP	AMQP	HTTP
1. Year	1999	2010	2003	1997
2. Architecture	Client/Broker	Client/Server or Client/Broker	Client/Broker or Client/Server	Client/Server
3. Abstraction	Publish/Subscribe	Request/Response or Publish/Subscribe	Publish/Subscribe or Request/Response	Request/Response
4. Header Size	2 Byte	4 Byte	8 Byte	Undefined
5. Message Size	Small and Undefined (up to 256 MB maximum size)	Small and Undefined (normally small to fit in single IP datagram)	Negotiable and Undefined	Large and Undefined (depends on the web server or the programming technology)
6. Semantics/ Methods	Connect, Disconnect, Publish, Subscribe, Unsubscribe, Close	Get, Post, Put, Delete	Consume, Deliver, Publish, Get, Select, Ack, Delete, Nack, Recover, Reject, Open, Close	Get, Post, Head, Put, Patch, Options, Connect, Delete
7. Cache and Proxy Support	Partial	Yes	Yes	Yes
8. Quality of Service (QoS)/ Reliability	QoS 0 - At most once (Fire-and-Forget), QoS 1 - At least once, QoS 2 - Exactly once	Confirmable Message (similar to At most once) or Non-confirmable Message (similar to At least once)	Settle Format (similar to At most once) or Unsettle Format (similar to At least once)	Limited (via Transport Protocol - TCP)
9. Standards	OASIS, Eclipse Foundations	IETF, Eclipse Foundation	OASIS, ISO/IEC	IETF and W3C
10. Transport Protocol	TCP (MQTT-SN can use UDP)	UDP, SCTP	TCP, SCTP	TCP
11. Security	TLS/SSL	DTLS, IPSec	TLS/SSL, IPSec, SASL	TLS/SSL
12. Default Port	1883/ 8883 (TLS/SSL)	5683 (UDP Port)/ 5684 (DLTS)	5671 (TLS/SSL), 5672	80/ 443 (TLS/SSL)
13. Encoding Format	Binary	Binary	Binary	Text
14. Licensing Model	Open Source	Open Source	Open Source	Free
15. Organisational Support	IBM, Facebook, Eurotech, Cisco, Red Hat, Software AG, Tibco, ITSO, M2Mi, Amazon Web Services (AWS), InduSoft, Fiorano	Large Web Community Support, Cisco, Contiki, Erika, IoTivity	Microsoft , JP Morgan, Bank of America, Barclays, Goldman Sachs, Credit Suisse	Global Web Protocol Standard

IV. Relative Analysis of Messaging Protocols for IoT Systems: MQTT, CoAP, AMQP and HTTP

This section presents a further in-depth and relative analysis of these four messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP. It critically analyses the two closely associated criteria to provide corresponding strengths and limitations of each messaging protocol. These messaging protocols are very extensive and different from each other because they have been evolved through different processes and needs. Also, their precise and relative comparisons depend on the types of IoT systems, devices, resources, applications, and specific conditions and requirements of the system. However, this relative comparison is based on a linguistic range "Lower" and "Higher" to render a nimble and broader view of each protocol with respect to other protocols. There is one caveat here that this relative comparison may vary in some circumstances due to the above IoT components and may reflect different comparative results than shown here. Additionally, this evaluation is based on static components and some empirical evidence from the literature. Nonetheless, it does not consider the dynamic network conditions and overheads incur in the retransmission of packets, which may also change comparison results.

A. Message Size vs. Message Overhead

Fig. 2 shows the relative comparison of these messaging protocols based on their common message size and message overhead. The graph illustrates that HTTP incurs the highest message size and overhead, and then it decreases for the other

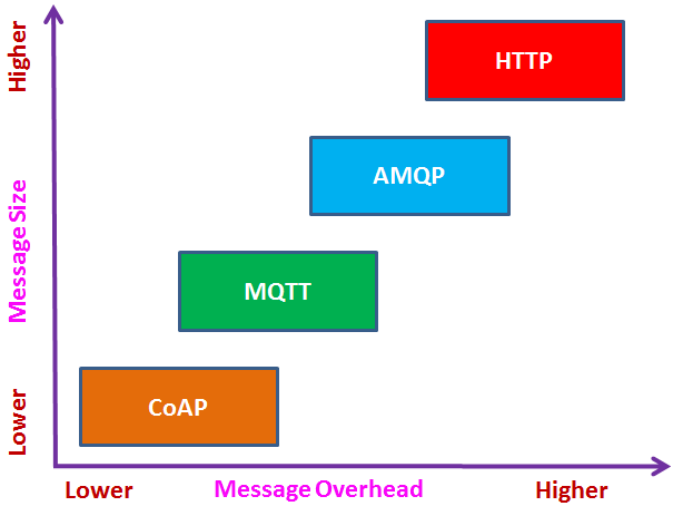

Fig. 2: Message Size vs. Message Overhead

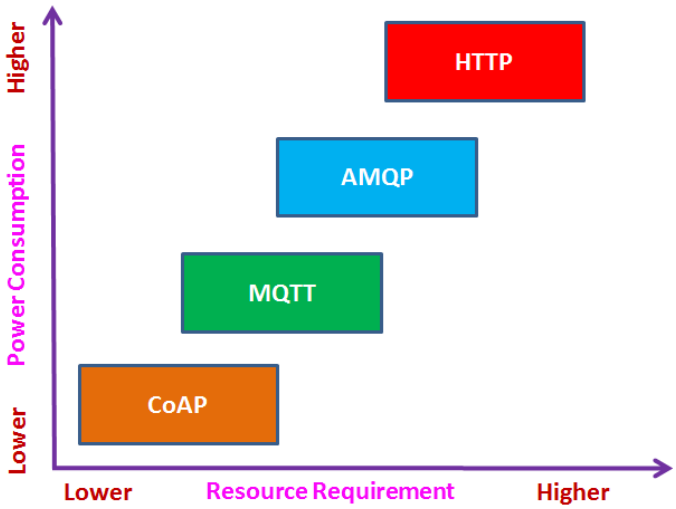

Fig. 3: Power Consumption vs. Resource Requirement

protocols with CoAP incurring the lowest message size and overhead [7], [8], [9], [16], [17]. MQTT, AMQP and HTTP run on TCP; therefore, they incur all TCP connection overheads for connection establishment and closing. However, MQTT is lightweight and has the least header size of 2-byte per message but its requirement of TCP connection increases the overall overhead, and thus the whole message size. CoAP runs on UDP; consequently, it does not incur connection overheads as UDP works in fire and forget basis [7], [9], [17], [18]. This reduces the overall overhead considerably, and thus the whole message size. AMQP is also a lightweight binary protocol; however, its support for security, reliability, provisioning and interoperability increases the overhead and message size [14], [19]. Finally, HTTP among all four is the most verbose and heavyweight protocol [17]. It was originally designed for the Web and not for the IoT; therefore, it requires maximum overhead and message size among all. As previously mentioned, this comparison does not consider retransmission scenario that can completely change overall overheads and amount of transmitted data and, thus, comparison results.

B. Power Consumption vs. Resource Requirement

Fig. 3 exhibits the relative comparison of these messaging protocols based on their normal power consumption and resource requirement. The graph highlights the similar patterns as the first one, where HTTP requires highest power and resource than any other protocols, and then it decreases for the other protocols with CoAP requires lowest power and resource [7], [8], [9], [16], [17], [20], [21], [22]. Both CoAP and MQTT are designed for low bandwidth and resource-constrained devices and can be used on an 8-bit controller and 100s bytes of memory. Various experimental studies found that CoAP consumes slightly less power and resources in similar circumstances: unreliable scenario (MQTT QoS 0 vs. CoAP NON), and reliable scenario (MQTT QoS 1 or 2 vs. CoAP CON), while assuming that no packet losses are happened [7], [8], [9], [17], [18]. AMQP requires slightly higher power and resources due to performing other necessary operations for provisioning and reliability [14], [19]. Finally, HTTP is a bigger than all and needs greater processing power and resources for the same operation [17], [20]. Again, this comparison does not consider dynamic network conditions and overheads incur in the retransmission of packets.

C. Bandwidth vs. Latency

Fig. 4 elicits the relative comparison of these messaging protocols based on their average bandwidth and latency. The graph reveals the very similar patterns as the first two, where HTTP involves largest bandwidth and latency than any other protocols, and then it decreases for the other protocols with CoAP involves lowest bandwidth and latency [7], [8], [9], [16], [17], [21], [22]. The use of TCP in MQTT, AMQP and HTTP is a major factor in determining the latency and bandwidth requirement. Unfortunately, TCP does not help in improving latency. It does not fully utilize the available network bandwidth for the first few roundtrips of a connection because of its slow start approach to avoid network congestion [23]. Where, TCP sender gradually opens the congestion window and doubling the number of packets in each round-trip time (RTT). In CoAP, a UDP transaction requires only two UDP datagrams, one in each direction; this reduces the network load response times. Various experimental studies found that MQTT consumes higher bandwidth than CoAP for transferring same payload under same network condition (MQTT QoS 1 or 2 vs. CoAP CON) [7], [8], [9], [17], [18]. Moreover, when comparing MQTT QoS 2 with CoAP CON, the bandwidth usage of MQTT was approximately double than CoAP. This is because of the four-way handshake mechanism of QoS 2. AMQP's extra services demand moderately higher bandwidth and latency [14], [19]. HTTP takes significantly larger bandwidth and latency time [4], [5], [17], [20].

D. Reliability/QoS vs. Interoperability

Fig. 5 displays the relative comparison of these messaging protocols based on their Quality of Services (QoSs) and interoperability. The graph divulges that MQTT offers the highest level of quality of services with least interoperability among four, whereas HTTP was designed for greatest interoperability on the Web and did not include reliability as a core feature

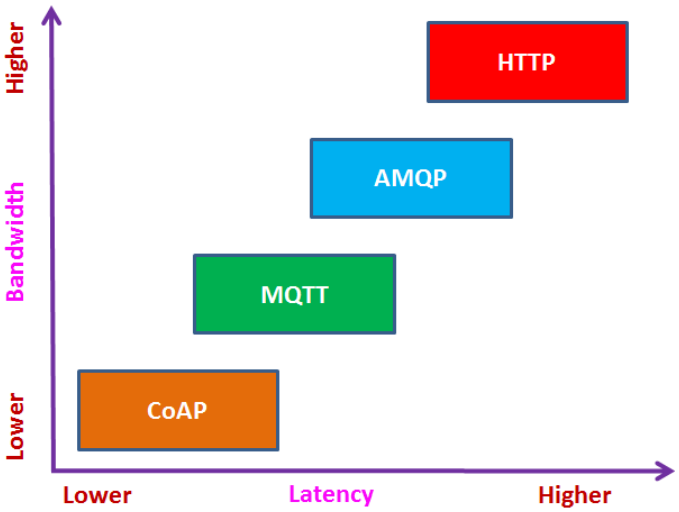

Fig. 4: Bandwidth vs. Latency

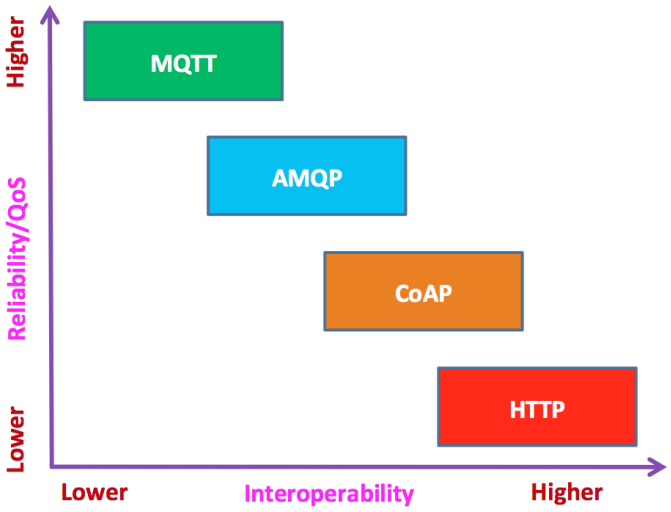

Fig. 5: Reliability/QoS vs. Interoperability

[3], [13], [14], [15], [19]. One of the biggest benefits of using TCP as a transport protocol by MQTT, AMQP and HTTP is the guaranteed delivery of a packet. MQTT, AMQP and CoAP protocols have different levels of QoS support. MQTT defines three QoS levels: 0- at most once (only TCP guarantee), 1- at least once (MQTT guarantee with confirmation), 2- exactly once (MQTT guarantee with handshake) [7]. Additionally, it also provides "last will and testament" message facilities (guarantee after disconnect). AMQP defines two QoS levels: Settle Format (similar to MQTT QoS 0) and Unsettle Format (similar to MQTT QoS 1). CoAP, which deprived of the reliability of TCP, compensates for the unreliability of UDP protocol by defining a retransmission mechanism and providing resource discovery mechanism with resource description [18]. Though CoAP does not provide explicit QoS, it facilitates the use of non-confirmable messages (NON) and confirmable messages (CON), which is very similar to MQTT QoS 0 and QoS 1 [21]. The QoS is not a default service of HTTP; therefore, its default reliability is the TCP guarantee [13]. Interoperability is the biggest issue among all IoT protocols. MQTT only supports the publish/subscribe pattern of communication, which barely covers all use cases within the IoT. In AMQP, it is common to use serialization formats such as Protocol Buffers, MessagePack, Thrift, and JSON to serialize structured data in order to publish it as the message payload [3]. CoAP is a part of the Web architecture and best suited for devices that support UDP or a UDP analogue, however, making it limited to a few special kinds of IoT devices [12]. HTTP-based RESTful clients and servers are the most interoperable because all that is needed to support message exchanges, is an HTTP stack (either on the client or the server) [4], [5], [10].

E. Security vs. Provisioning

Fig. 6 demonstrates the relative comparison of these messaging protocols based on the security and provisioning support provided by them. The graph discloses that AMQP has the highest level of support for security and additional services, while MQTT is barely a messaging protocol and supports the lowest level of security and additional services [3], [6], [11], [13], [14], [15], [19]. Except TLS/SSL, MQTT has minimal authentication features and only rely on simple username and password [6], [11]. The CoAP uses two methods DTLS and IPsec for authentication, integrity and encryption. HTTP facilitates two authentication approaches: HTTP Basic and HTTP Digest [10]. HTTP basic authentication uses unencrypted Base64-encoding username and password to authenticate a service client over TLS/SSL. HTTP digest authentication uses an encrypted username and password to authenticate over on non-TLS/SSL connection. AMQP provides the strongest security with different approaches to TLS negotiation: Single-port TLS Model, Pure TLS and WebSockets Tunnel TLS Model. It has explicitly facilitated the integration of TLS (e.g. TLS virtual server extensions, known as SNI) and SASL [3], [6]. MQTT does not offer any extra services even message labelling; consequently, messages can be used for any purpose; therefore, all clients must know the message formats up-front to allow communication [11]. In CoAP, there are several extensions for enhanced services depending on the requirements of the IoT system such as support for observers, multicast group communications, resource discovery and block-wise transfers [9]. HTTP is a full web standard and offers several services such as multiplexing and concurrency, stream dependencies/prioritization, header compression and server push [4], [5], [10]. AMQP is the preferred choice for businesses because of its wide range of services related to messaging such as reliable queuing, topic-based publish-and-subscribe messaging, flexible routing and transactions [15]. It provides various ways to exchange route messages: directly, in fanout form, by topic, and based on headers [3]. For enhancing the security of IoT systems across multiple clouds, these messaging protocols can be combined with identity and access management protocols [24], [25], [26], [27], [28], [29]. Similarly, for the better provisioning of IoT systems, the Docker-based design may be an alternative option for users [30], [31], [32].

F. M2M/IoT Usage vs. Standardisation

Fig. 7 expresses the relative comparison of these messaging protocols based on their usage in M2M/IoT and accreditation from standard organisations. The graph indicates that MQTT

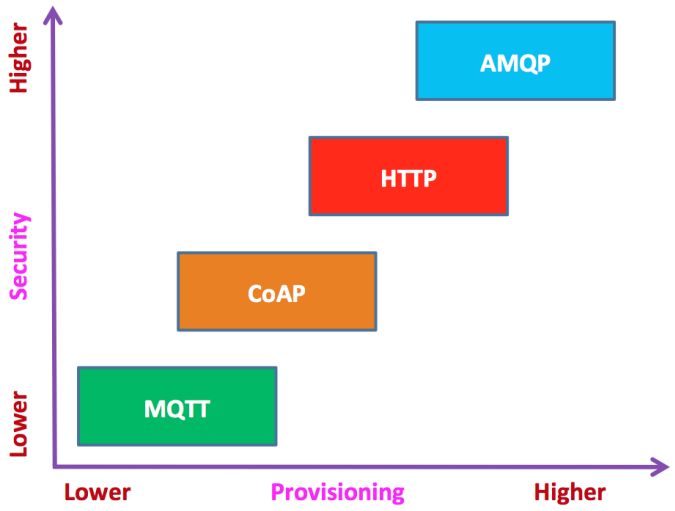

Fig. 6: Security vs. Provisioning

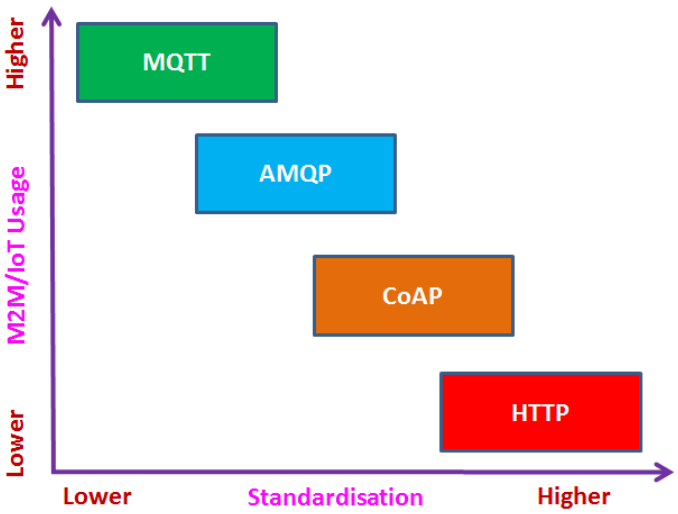

Fig. 7: M2M/IoT Usage vs. Standardisation

has been employed by the large number of organisations but it is still not a global standard, while, HTTP is a global web standard but mostly not suitable and used in the IoT industry [3], [6], [7], [8], [9], [10], [11]. MQTT is an established M2M protocol and has been used and supported by the large number of organisations such as IBM, Facebook, Eurotech, Cisco, Red Hat, M2Mi, Amazon Web Services (AWS), InduSoft and Fiorano [7], [8], [9]. Besides, AMQP is the most successful IoT protocol that has been employed in the worlds biggest projects such as Oceanography's monitoring of the Mid-Atlantic Ridge, NASA's Nebula Cloud Computing and Indias Aadhar Project [3], [6], [11]. CoAP has been swiftly gaining momentum and supported by many large companies such as Cisco (Field Area Network), Contiki, Erika and IoTivity [7], [8], [9]. Finally, the usage of HTTP in the IoT is limited due to its heavyweight size and slow performance. MQTT is an emerging as a de facto protocol for the IoT and hosted by OASIS open standards consortium and Eclipse Foundation [11], [33]. AMQP is an OASIS adopted international standard ISO/IEC 19464:2014 [3]. CoAP is an IETF standard specially designed to integrate the IoT and Web and supported by Eclipse Foundation [11]. Finally, HTTP is an IETF and W3C standard and already established as a global standard for the Web [4], [5], [10].

V. Conclusion

This paper has presented an evaluation of the four widely accepted and emerging messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP. Firstly, it has presented the overall comparison among these protocols to introduce their characteristics comparatively. Subsequently, it has performed a further in-depth and relative analysis based on some interrelated criteria to gain insight into their strengths and limitations. For making this relative analysis easy, it was illustrated using simple graphs to render a nimble and broader view of each protocol with respect to other protocols for an ordinary user. Accordingly, the user can decide their relevant usage in IoT systems based on their requirements and suitability. This critical evaluation has demonstrated a bigger and comparative picture of messaging protocols; which was based on the static components and some empirical evidence from the literature. Nonetheless, it did not consider dynamic network conditions and overheads incur in the retransmission of packets, which may produce the different results from the comparison shown here. Additionally, this is the rapidly growing and changing area that might change the presented scenario in the future. In the future, it may be interesting to practically evaluate these protocols in the same IoT system.

References

[1] V. Gazis, M. Gortz, M. Huber, A. Leonardi, K. Mathioudakis, A. Wiesmaier, F. Zeiger, and E. Vasilomanolakis, "A survey of technologies for the internet of things," in *2015 IEEE International Wireless Communications and Mobile Computing Conference*, 2015, pp. 1090–1095.

[2] V. Karagiannis, P. Chatzimisios, F. Vazquez-Gallego, and J. Alonso-Zarate, "A survey on application layer protocols for the internet of things," *Transaction on IoT and Cloud Computing*, vol. 3, no. 1, pp. 11–17, 2015.

[3] A. Foster, "Messaging technologies for the industrial internet and the internet of things whitepaper," *PrismTech*, 2015.

[4] N. Naik, P. Jenkins, P. Davies, and D. Newell, "Native web communication protocols and their effects on the performance of web services and systems," in *16th IEEE International Conference on Computer and Information Technology (CIT)*. IEEE, 2016, pp. 219–225.

[5] N. Naik and P. Jenkins, "Web protocols and challenges of web latency in the web of things," in *2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*. IEEE, 2016, pp. 845–850.

[6] R. S. Cohn, "A comparison of AMQP and MQTT," 2011.

[7] S. Bandyopadhyay and A. Bhattacharyya, "Lightweight internet protocols for web enablement of sensors using constrained gateway devices," in *Computing, Networking and Communications (ICNC), 2013 International Conference on*. IEEE, 2013, pp. 334–340.

[8] N. De Caro, W. Colitti, K. Steenhaut, G. Mangino, and G. Reali, "Comparison of two lightweight protocols for smartphone-based sensing," in *Communications and Vehicular Technology in the Benelux (SCVT), 2013 IEEE 20th Symposium on*. IEEE, 2013, pp. 1–6.

[9] D. Thangavel, X. Ma, A. Valera, H.-X. Tan, and C. K.-Y. Tan, "Performance evaluation of MQTT and CoAP via a common middleware," in *Intelligent Sensors, Sensor Networks and Information Processing, 2014 IEEE Ninth International Conference on*. IEEE, 2014, pp. 1–6.

[10] I. Grigorik, "Making the web faster with HTTP 2.0," *Communications of the ACM*, vol. 56, no. 12, pp. 42–49, 2013.

[11] T. Jaffey. (2014, February) MQTT and CoAP, IoT protocols. [Online]. Available: https://eclipse.org/community/eclipse_newsletter/2014/february/article2.php

[12] A. Ludovici, P. Moreno, and A. Calveras, "TinyCoAP: a novel constrained application protocol (CoAP) implementation for embedding RESTful web services in wireless sensor networks based on tinyos," *Journal of Sensor and Actuator Networks*, vol. 2, no. 2, pp. 288–315, 2013.

[13] N. S. Han, "Semantic service provisioning for 6LoWPAN: powering internet of things applications on web," Ph.D. dissertation, Institut National des Télécommunications, 2015.

[14] J. E. Luzuriaga, M. Perez, P. Boronat, J. C. Cano, C. Calafate, and P. Manzoni, "A comparative evaluation of AMQP and MQTT protocols over unstable and mobile networks," in *12th Annual IEEE Consumer Communications and Networking Conference*, 2015, pp. 931–936.

[15] G. Marsh, A. P. Sampat, S. Potluri, and D. K. Panda, "Scaling advanced message queuing protocol (AMQP) architecture with broker federation and infiniband," *Ohio State University, Tech. Rep. OSU-CISRC-5/09-TR17*, 2008.

[16] K. Kuladinithi, O. Bergmann, T. Pötsch, M. Becker, and C. Görg, "Implementation of coap and its application in transport logistics," *Proc. IP+ SN, Chicago, IL, USA*, 2011.

[17] S. S. Ngo Manh Khoi, K. Mitra, and C. Ahlund, "Irehmo: An efficient IoT-based remote health monitoring system for smart regions," 2015.

[18] J. Stansberry. (2015, October 7) MQTT and CoAP: Underlying protocols for the IoT. [Online]. Available: http://electronicdesign.com/iot/mqtt-and-coap-underlying-protocols-iot

[19] J. E. Luzuriaga, M. Perez, P. Boronat, J. C. Cano, C. Calafate, and P. Manzoni, "Testing AMQP protocol on unstable and mobile networks," in *Internet and Distributed Computing Systems*. Springer, 2014, pp. 250–260.

[20] S. Nicholas. (2012, May 31) Power profiling: HTTPS long polling vs. MQTT with SSL, on android. [Online]. Available: http://stephendnicholas.com/posts/power-profiling-mqtt-vs-https

[21] W. Colitti, "Communication stacks: Constrained Application Protocol," *ISN Interoperable Sensor Networks Deliverable*, 2011.

[22] W. Colitti, K. Steenhaut, and N. De Caro, "Integrating wireless sensor networks with the web," *Extending the Internet to Low power and Lossy Networks (IP+ SN 2011)*, 2011.

[23] M. Mellia, M. Meo, and C. Casetti, "TCP smart framing: a segmentation algorithm to reduce TCP latency," *Networking, IEEE/ACM Transactions on*, vol. 13, no. 2, pp. 316–329, 2005.

[24] N. Naik and P. Jenkins, "Securing digital identities in the cloud by selecting an apposite federated identity management from saml, oauth and openid connect," in *2017 11th International Conference on Research Challenges in Information Science (RCIS)*. IEEE, 2017, pp. 163–174.

[25] N. Naik, P. Jenkins, and D. Newell, "Choice of suitable identity and access management standards for mobile computing and communication," in *2017 24th International Conference on Telecommunications (ICT)*. IEEE, 2017, pp. 1–6.

[26] N. Naik, "Connecting Google cloud system with organizational systems for effortless data analysis by anyone, anytime, anywhere," in *IEEE International Symposium on Systems Engineering (ISSE)*. IEEE, 2016.

[27] N. Naik and P. Jenkins, "An analysis of open standard identity protocols in cloud computing security paradigm," in *14th IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC 2016)*. IEEE, 2016.

[28] ——, "A secure mobile cloud identity: Criteria for effective identity and access management standards," in *2016 4th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*. IEEE, 2016, pp. 89–90.

[29] N. Naik, P. Jenkins, N. Savage, and V. Katos, "Big data security analysis approach using computational intelligence techniques in R for desktop users," in *IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2016.

[30] N. Naik, "Migrating from Virtualization to Dockerization in the cloud: Simulation and evaluation of distributed systems," in *IEEE 10th International Symposium on the Maintenance and Evolution of Service-Oriented and Cloud-Based Environments, MESOCA 2016*. IEEE, 2016.

[31] ——, "Building a virtual system of systems using Docker Swarm in multiple clouds," in *IEEE International Symposium on Systems Engineering (ISSE)*. IEEE, 2016.

[32] ——, "Applying computational intelligence for enhancing the dependability of multi-cloud systems using Docker Swarm," in *IEEE Symposium Series on Computational Intelligence (SSCI)*, 2016.

[33] OASIS.org. (2015, December 10) MQTT 3.1. 1. edited by Andrew Banks and Rahul Gupta. 29 october 2014. OASIS Standard. [Online]. Available: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html