**CHAPTER 4**

**Tests and Results of the Proposed System**

This chapter represents about the design and implementation of the proposed

system. It also suggests implementing the comparison between MQTT and HTTP.

**4.1 Internet of Thing**

The Internet of Things (IoT) is being widely discussed. It is a topic of worldwide interest. In IoT, a large number of tiny data blocks from devices, such as various sensors, are transferred across networks. Although Internet Protocol (IP) has been adopted for most types of communication, it will have some problems when it is applied to IoT. Currently, Internet access requires application protocols over TCP/IP or UDP/IP. One of the application protocols is HyperText Transfer Protocol (HTTP), which has been standardized in IETF, and has been applied for general communication over the Internet. However, when HTTP is applied to communication in IoT, in which a huge number of tiny data blocks are transferred, protocol overhead and resulting performance degradation are a serious problem. Moreover, IP addressing depends on physical location, which causes the problem of complexity of network control. To solve these problems, name- based architectures, such as Named Data Networking (NDN), Content Centric Networking (CCN), and Information Centric Networking (ICN) have been discussed. Some of the examples focus on adopting these architectures to IoT.

In these architectures, MQ Telemetry Transport (MQTT) is one of the protocols, as described in [13]. Standard committees such as oneM2M and ETSI have paid considerable attention to MQTT and have also conducted relevant discussions. MQTT reduces protocol overheads and provides high efficiency communication for IoT. It also invokes “Name based routing,” and mitigates IP address based routing for IoT traffic flows. This paper discusses the possibility of considering MQTT

as a candidate for the communication protocols on the IoT platform. It compares the performance of MQTT with that of HTTP. Moreover, it proposes new mechanisms to enhance the current MQTT specifications.

**4.2 HTTP For IoT Communication**

It is assumed that HTTP is applied to communication for IoT. The HTTP must transfer a large number of tiny packets. Protocol overhead of HTTP causes serious problems, such as consumption of network resources and large delays. Since HTTP is operated over TCP/IP, reliable communication is provided. However, connections established by TCP are released on every access, because accessed data is transferred based on IP address and URL and their relationship is changed dynamically. In short, after many times of establishment of release of a connection, communication is completed.Therefore, communication for IoT causes serious overhead and consumption of network resources during this communication.

4.2.1 MESSAGING PROTOCOLS FOR IOT SYSTEMS

This section presents the four widely accepted and emerging messaging protocols for IoT systems: MQTT, and HTTP, which are the 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 clients publish 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 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 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].

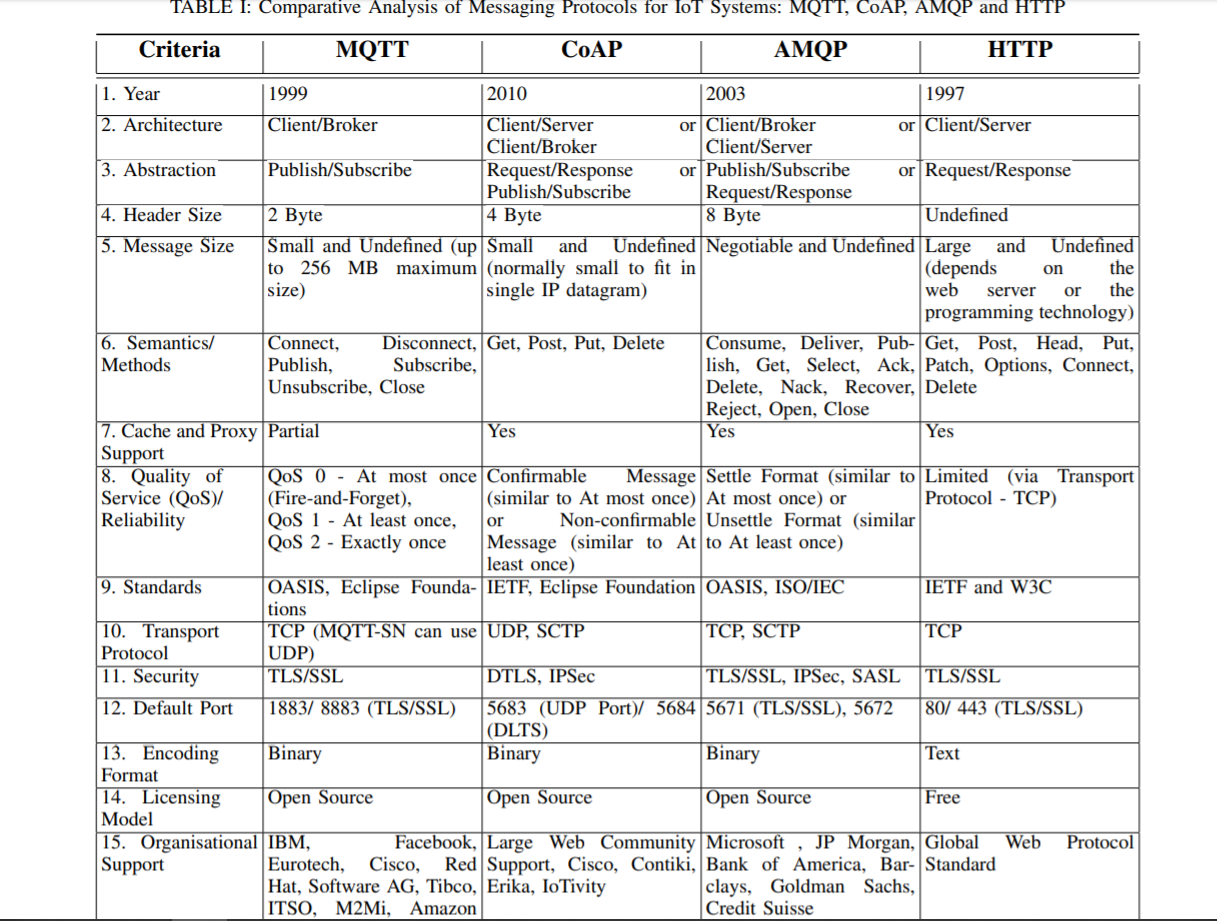
D. HTTP (HyperText Transport Protocol)

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 the client receives data through a particular URI. HTTP is a text-based protocol and it does not define the size of header and message payloads rather it depends 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 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].

**4.3 Comparative analysis of messaging protocols for IoT systems: HTTP, COAP, AMQP AND MQTT**

This section presents a comparative analysis of the widely accepted and emerging messaging protocols for IoT systems MQTT 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



**4.4 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 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 scenarios 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 of 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 have 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 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]. TCP sender gradually opens the congestion window and doubles 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 the same payload under the 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 [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 is 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 relies 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 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 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 India's 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 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].