An extended review of the application layer messaging protocol of the internet of things

Ronok Bhowmik, Md. Hasnat Riaz
Department of Computer Science and Telecommunication Engineering (CSTE), Noakhali Science and Technology University (NSTU), Noakhali, Bangladesh

ABSTRACT
In the internet of things (IoT), there are resource-constrained and immense heterogeneous electronic gadgets worldwide. Till now, no single IoT application layer messaging protocol is the best, nor axiomatic for every requirement. This paper exhaustively summarizes information on the messaging protocols from the available previous research sources online. Our goal is to encapsulate a simple guideline so that users can choose an optimal messaging protocol quickly according to development requirements and specifications. For this purpose, we have reviewed the literature on six enabling and evolving application layer messaging protocols used for IoT systems namely, message queuing telemetry transport (MQTT), advanced message queuing protocol (AMQP), the constrained application protocol (CoAP), extensible messaging and presence protocol (XMPP), data distribution service (DDS), and simple text-oriented messaging protocol (STOMP) in terms of some interrelated metrics. Additionally, we represented a critical analysis of the application layer messaging protocols. This study will be helpful to readers with valuable insights and guide research scholars and developers in choosing optimal application layer messaging protocols based on development specifications and requirements.

1. INTRODUCTION
British entrepreneur and executive director of AUTO-ID center Kevin Ashton first introduced the word internet of things (IoT), as the title of a presentation of Proctor and Gamble in 1999 [1]. Afterward, IoT opened new windows of opportunities for technological and scientific development in every touch point we imagine. IoT systems are associated with intelligent devices, smart objects, and people [2]. Smart devices are self-configurable, self-functional, wireless-based, and can process all the work without manual or human intervention. Dependencies on the internet and internet-based services are increasing rapidly worldwide. Approximately 75.44 billion devices will be appended worldwide through the internet by 2025 [3]. IoT shoves an unbounded number of new applications in a wide range of fields like smart home systems, animal farms, productivity, supply chains, precision agriculture, environmental monitoring (low energy monitoring systems and telemetry), e-health, industrial applications, informatics, automobiles and transportation systems, high-security applications, law enforcement, defense, logistics systems, space research, entertainment systems, and wearable gadgets. IoT-related services have made everything easier than ever. IT industries named Apple, Amazon, and Google use IoT to bring innovative technological changes. In IoT, various
messaging protocols are introduced based on application deployment, communication mode, suitability for applications, intrinsic nature of the devices (heterogeneity of electronic gadgets), kinds of security provided, and the nature of transmission of messages over the internet. The popularity of IoT-related devices, standards, technologies, and platforms is constantly changing and improving, and the present conditions, specifications, and requirements might not be the same in the future. So, picking up every detail (advantages and disadvantages) of the existing messaging protocols is important. Finding an optimal and cost-effective IoT messaging protocol is a mammoth task for developers. If we finalize the system design without selecting an optimal protocol and proper requirement analysis, it will cost a lot of time and money hence a project failure.

During the last fifteen years, several standardization bodies, research groups, technologists, and organizations have searched for a unique messaging protocol; unfortunately, none of them can meet all the requirements of IoT gadgets. In the context of IoT, there are hundreds of protocols of different characteristics and specifications. Some popular IoT application layer protocols are message queuing telemetry transport (MQTT) [4], constrained application protocol (CoAP) [5], extensible messaging and presence protocol (XMPP) [6], advanced message queuing protocol (AMQP) [7], data distribution service (DDS) [8], simple text-oriented messaging protocol (STOMP) [9], representational state transfer (RESTful) hypertext transfer protocol (HTTP), simple media control protocol (SMCP), lightweight local automation protocol (LLAP), simple sensor interface (SSI), lightweight machine-to-machine (LWM2M), M3DA, XMPP-IOT, ONS 2.0, simple object access protocol (SOAP), WebSocket, reactive streams, HTTP/2, and JavaScript IoT. We have noticed this work has the subsequent contribution: i) explored six well-established application layer messaging protocols of IoT systems and ii) represented a critical analysis where we compared the performance, characteristics, and behaviors of the application layer messaging protocols.

This survey paper is exhibited as follows: section 2 includes some related works in literature along with the research gap. Section 3 briefly reviews the six-application layer messaging protocols and exemplifies a critical analysis of the messaging protocols based on different parameters. In the end, in section 4, we summarize our conclusions in section 5 about this study and provide future work directions.

2. RELATED WORKS

Numerous qualitative reviews and experimental illustrations have been conducted in terms of application layer messaging protocols. As we know, IoT system standards, gadgets, specifications, and requirements are constantly changing, it is quite difficult to summarize all aspects of the system. In this paper, we have encapsulated the information from several review papers. The following section will discuss the previous works related to this review study.

Al-Fuqaha et al. [10] started the discussion with an overview of the IoT. The study also focused on IoT-related technologies, protocols, applications, and challenges of recent literature. In addition, the relationship between the IoT, big data analytics, cloud, and fog computing are also illustrated. Lee et al. [11] provided an overview of MQTT. Here, the architecture, message format, scope, and quality of service (QoS) of MQTT are described thoroughly. The authors stated that MQTT is an open standard, publish-subscribe messaging protocol which uses transmission control protocol (TCP) for transport. A comprehensive study on IoT protocols is conducted in article [12], where the authors document the recent developments of IoT protocols and standardization initiatives from the perspective of interoperability. Pathaka and Tembhurne [13] discusses the overview and the standards used in IoT. In addition, the architecture, protocols, and standards are reviewed critically. The authors also find similarities and dissimilarities between MQTT and AMQP protocols. Yughra and Chithra [14] highlighted the issues and challenges related to security and the use of IoT protocols. They also reviewed the research trends and simulation tools used for the analysis purpose of IoT application layer protocols. The goal of the survey conducted by Al-Joboury and Al-Hemari [15] was to facilitate some guidelines for academic researchers in IoT protocols. IoT-related applications, open issues, architecture, explanation of IoT protocols, operations, functionalities, and data cloud integration are also discussed by the authors. Wtyrąbłowicz et al. [16] tried to select an appropriate protocol based on specific communication requirements. Furthermore, the authors represent a comparison of the protocols based on an analysis of the protocol specifications and also provide some recommendations for proper protocol selection. Dizdarévić et al. [17] tried to focus our attention on the implementation of fog and cloud-based IoT systems. The authors have tried to discuss interoperability, system integration, latency, power consumption, throughput, and some common features of widely used IoT protocols. Interestingly, Bayilmuş et al. [18] emphasized the overview of lightweight communication protocols, along with their strengths and limitations. They also draw our attention to the advancement of application layer protocols for the IoT ecosystem. The authors also explained how MQTT, CoAP, and WebSocket protocols are better choices for small IoT devices. Similar to the article [13], research by Bibi et al. [7] represented a survey on

An extended review of the application layer messaging protocol of the internet of things (Ronok Bhownik)
CoAP, MQTT, AMQP, and XMPP. The papers discuss the architecture, advantages, disadvantages, and applications of each protocol. Later, a comparative analysis is also presented.

Although the above-mentioned studies have collected a large amount of information, there is still a lack of advanced research on selecting the right IoT protocol. Surprisingly, there are inadequacies in the review of telemetry transmission, security, data integrity, and interoperability for the application layer protocols of IoT. Moreover, in our investigation, we found limited studies on DDS and STOMP protocol. Because of these shortcomings, an extended guideline has been prepared to find an optimal protocol quickly.

3. REVIEW THE APPLICATION LAYER MESSAGING PROTOCOLS FOR THE INTERNET OF THINGS

In recent years, the most ardent drift of IoT is the application layer messaging protocols and heterogeneity of electronic gadgets. One specific messaging protocol cannot satisfy all the requirements and provide a guarantee to facilitate secure, scalable, traceable, energy-optimized, time and cost-saving, and lossless communications. This section exhibits a review of the six widely accepted and emerging messaging protocols for IoT systems: MQTT, AMQP, CoAP, XMPP, DDS, and STOMP. In this section, we concisely review and compare the performance, characteristics, and behaviors of the six above-mentioned application layer messaging protocols used for IoT systems.

3.1. Message queuing telemetry transport

MQTT is a widely-used, simply designed, lightweight broker-based (server-based) message transport connectivity protocol developed in 1999 and released by Andy Stanford-Clark (IBM) and Arlen Nipper (Arcom) control systems limited [19]. It was integrated with the IBM WebSphere application server, and standardized by the organization for the advancement of structured information standards (OASIS) in 2013. OASIS aims to reduce the bandwidth requirement. It targets M2M communications and a resource-constrained environment. It follows an asynchronous publish-subscribe communication way similar to the client-server model. MQTT assumes a connection-oriented, reliable transport protocol TCP handled by transport layer security/secure sockets layer (TLS/SSL) to ensure security [20]. This protocol is suitable for sensor networks and wireless sensor networks. The architecture of MQTT contains two main components: client and broker. A client may be a publisher or subscriber, and the server is the broker. There may be more than one publisher and subscriber. Publishers/subscribers always try to make a connection to the server. Here, the server is known as the broker. A broker creates a link between physical devices and enterprise systems. Broker contains topics that are a UTF-8 string. For filtering messages to interested clients, the broker uses topics [20]. Frequently used MQTT brokers are Mosquitto, really small message broker (RSMB), MQTT.js, HiveMQ, and paho MQTT [21]. Open-source code, less message processing, lower overhead, lower network bandwidth, the ability to deal with delay or latency in the network, lower battery usage, faster response times, and straightforward implementation are the most noticeable features of MQTT. However, there are two variants of MQTT (MQTT v.3.1/v.3.1.1/v.5.0 and MQTT-SN). MQTT v.5.0. is the latest and current version [18]. MQTT-S is suitable for the non-TCP/IP stacks, and MQTT-SN is for sensory networks [22]. QoS functionalities are available for MQTT, and it has three QoS levels [11]. These QoS are denoted as QoS 0, QoS 1, and QoS 2. In QoS 0, the message is delivered at most once, in QoS 1 the message is delivered exactly once, and in QoS 2 the message is delivered at least once. MQTT isn’t suitable for multicasting (one-to-many messages) [23].

In this day and age, MQTT is one of the leading open-source protocols in the IoT industry. If the circumstances are such that we have to work from remote locations where the network bandwidth and power are limited, constrained resources, and unreliable networks then we can choose MQTT. For example, in remote health monitoring scenarios (automated medical alerts), MQTT is the first choice for system development. In addition, good QoS, high security, a central broker, and a flexible subscription pattern are the notable features of MQTT. So it is clear that MQTT is the best choice for constrained environments. In most M2M communications, the extensively used protocols are the MQTT and CoAP. Slower transmit cycles than CoAP, lack of security encryption, and scalability are the drawbacks of MQTT. Lightweight applications, home automation, healthcare, social networking, enterprise-level applications, and utilities are the sphere where we are using MQTT. In addition, MQTT is used on a higher scale in various sectors in the industries such as automotive, logistics, manufacturing, smart home, consumer products, transportation, and so forth. Amazon web services, Facebook Messenger, and Microsoft Azure IoT are the sectors where MQTT is widely used nowadays [24].
3.2. Advanced message queuing protocol

AMQP is an application layer messaging protocol developed by John O’Hara at JPMorgan Chase in London, the UK, in 2003 [24] and standardized by OASIS. Like MQTT, AMQP is a broker-based message protocol similar to the client-server model [25]. In this protocol, message transmission between two nodes maintains one-to-one communication. It follows asynchronous publish-subscribe architecture. AMQP assumes connection-oriented reliable transport protocol TCP handled by TLS/SSL [26]. Compared to the REST, AMQP can send many messages per second [27]. Unlike MQTT, the AMQP broker consists of two components: exchange and queue [10]. The exchange component of the broker is responsible for performing the routing functionality by forwarding messages to the appropriate message queue. Messages are stored in a message queue until received by the receiver. This mechanism works for both end-to-end and publisher-subscriber models. There are two types of messages named bare messages and annotated messages [28]. Publishers use bare messages, and subscribers use annotated messages. Still, AMQP is not widely used and suitable for IoT sensor devices because of limited memory. As a result, its use is still quite limited within the world of IoT. Like MQTT, AMQP has three QoS levels (QoS 0, QoS 1, and QoS 2) [29]. The message is delivered at most once for QoS 0, exactly once for QoS 1, and at least once for QoS 2.

Interoperability, reliability, and trustworthiness are the striking features of the AMQP protocol [30]. It delivers acknowledgment messages of the sent messages making it crucial for banking institutions. The main interest of AMQP's development is to use it in the financial industries [30]. In that continuity, for commercial purposes in server-based analytical environments i.e., in banking industries, non-bank financial institutions (NBFI), business messaging, insurance companies, and industrial environment applications use AMQP. American banking and financial service-providing company JPMorgan use AMQP [31]. Home automation and vehicle-to-vehicle communication are the sectors where AMQP is extensively used too. Microsoft Azure service bus and Azure IoT hub support communication also use AMQP [32].

3.3. Extensible messaging and presence protocol

XMPP is an extensible markup language (XML) language-based messaging transport connectivity open-source protocol introduced in 1999 by the Jabber software foundation (JSF) [33]. It is the most heavyweight protocol. The internet engineering task force (IETF) standardized XMPP a decade ago [34]. In XMPP, there are three types of XML stanzas [35]. These are message stanza, presence stanza, and IQ stanza. Notable features of XMPP are instant messaging, real-time entertainment, telepresence, chatting, and message exchange. It targets messaging applications over the internet. It supports both publish-subscribe (asynchronous) and request-response architecture. The publish/subscribe architecture allows multicast communication. Unlike MQTT and AMQP, XMPP does not provide QoS guarantees [36]. As QoS is not guaranteed, it doesn't suit M2M communications. As we know, CoAP generates lower overhead than MQTT [37] but in XMPP, XML messages create additional overhead due to headers and tag formats that increase power consumption. Openness, scalability, extensibility, and flexibility are the noticeable features of XMPP [6]. Unlike MQTT and CoAP, XMPP consumes more power [38].

Security, real-time communication, flexibility, easy understandability, and open standard are the notable merits of XMPP. Text-based communication, no QoS, higher bandwidth, overhead, and message size are the significant demerits of XMPP. Real-time communications like instant messaging, Group chat (Google chat, Facebook chat), multi-party chat, voice and video calls, telepresence, gaming, collaboration, offline messaging, voice mailing, content syndication, and vehicle tracking are the fields where we can use the XMPP.

3.4. Constrained application protocol

CoAP is an application layer messaging protocol introduced in 2010 by IETF [39]. It targets M2M communications and is designed for constrained-resource devices [40]. The devices which don't support HTTP can use CoAP protocol. Like XMPP, CoAP supports both publish-subscribe and request-response architecture. It assumes connectionless unreliable transport protocol user datagram protocol (UDP) [41]. CoAP relies on datagram TLS (DTLS) for security purposes [42]. Instead of using topics, CoAP uses a uniform resource identifier [43]. CoAP uses GET, PUT, POST, and DELETE methods for performing the create, retrieve, update, and delete operations [44]. In CoAP, request and response messages are marked as confirmable and non-confirmable, respectively [45]. CoAP supports many-to-many communication [46].

Like MQTT, CoAP protocols are lightweight and suitable for M2M communications. If the devices have limited control functionalities, low overhead, low latency, low bandwidth, low availability, and limited RAM for M2M communication, we can choose CoAP [37]. Data authenticity and integrity, cryptographic algorithm support, confidentiality, and automatic key management are the plus points of CoAP. Supported data formats for CoAP are XML, JavaScript object notation (JSON), and concise binary object representation (CBOR) [25]. In addition to the above, limited libraries, securities, and scarcity of available solution support
are the major drawbacks of CoAP. Smart home systems, mobile phones and microcontrollers, smart grids, and building automation are the sectors where we use CoAP [47].

3.5. Data distribution service

DDS is an application layer messaging protocol released and standardized by the object management group (OMG) in 2004 [48]. It is the first open interoperable middleware protocol. Mostly, it targets real-time M2M communications. It doesn't support broker based message protocol. To ensure the QoS, DDS maintains 23 levels and a variety of quality criteria [18]. Control reliability, volatility, liveliness, resource utilization, filtering and delivery, ownership, redundancy, security, durability, flexibility, urgency priority, timing deadlines, and the latency of the data are notable QoS factors of DDS. Like MQTT, DDS supports publish-subscribe and request-response architecture for real-time systems. It defines two sub-layers: data-centric publish-subscribe (DCPS), and the data-local reconstruction layer (DLRL) [32]. DCPS circulates information to the subscribers. DLRL acts as an interface to the DCPS functionalities [48]. DLRL is an optional layer that is shared among distributed objects. Distributed objects, military imaging and systems, hospital integration, cyber-physical, industrial parts of IoT, and wind firms are the sectors where we can use DDS.

3.6. Simple text-oriented messaging protocol

STOMP is a simple, text-based, lightweight, wire-format message communication protocol [49]. STOMP was introduced in 1999, and it is an open and bi-directional protocol. It is maintained by the programmers' community [50] and can perform one-to-many communications [16]. The main target of STOMP’s development was to use the Apache ActiveMQ system [9]. STOMP does not have any topics or queues like MQTT. Like the other data protocols of IoT, STOMP supports the publish-subscribe architecture [51]. STOMP has more similarities to HTTP [52]. Simplicity and easy understandability are the notable features of STOMP. Compared with AMQP, STOMP is a simple and flexible protocol. The client of the STOMP protocol can communicate with almost every available STOMP message broker. This feature provides easy and widespread messaging and a wide range of language bindings, platforms, and brokers. RabbitMQ message broker uses STOMP protocols to ease and scale the deployment of modern cloud services. So far as we know, there are three versions of STOMP. These are STOMP 1.0, STOMP 1.1, and STOMP 1.2. Among these versions of the STOMP protocol, STOMP 1.2 version is the latest version. STOMP 1.2 was released worldwide on 22nd October 2012. If we have the freedom to choose any protocol in the entire network system, we can choose STOMP arbitrarily and use STOMP as a publisher and receiver. Simple message queuing applications can use STOMP. A critical analysis of the messaging protocols for IoT systems, namely, MQTT, AMQP, CoAP, XMPP, DDS, and STOMP based on several criteria to introduce their characteristics comparatively, is presented in Table 1 (see in appendix).

4. CONCLUSION

Over the past two decades, one of the dominant trends in modern technology has been IoT, and the reliance on it has been steadily increasing. The main goal of the current study is to provide an impactful guideline for choosing an appropriate IoT messaging protocol based on different specifications and requirements. This research enhances our understanding of the protocol architecture, features, and other necessary specifications of the application layer of IoT messaging protocols. The most crucial finding from the study is that no specific protocol can be called the best for all circumstances. The observations of this study suggest that we have to choose a protocol based on the type of IoT deployment, scope, project cost, power consumption, geographic size, target user groups, purpose of use, and other relevant aspects. Due to the advancement of technology in the upcoming years and severe economic recessions and energy crises worldwide, we need to be careful about power consumption. By selecting the optimal IoT protocol, we can save reasonless time, and money wastage helps in successful project completion. Hopefully, this research will enhance understanding of various gradations, usage of IoT protocols, and optimal protocol selection.
APPENDIX

<table>
<thead>
<tr>
<th>Criteria</th>
<th>MQTT</th>
<th>XMPP</th>
<th>AMQP</th>
<th>CoAP</th>
<th>DDS</th>
<th>STOMP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header size</td>
<td>2 bytes</td>
<td>Undefined/no limit</td>
<td>8 bytes</td>
<td>Minimum 4 byte</td>
<td>16 bytes</td>
<td>Implementation dependent</td>
</tr>
<tr>
<td>Maximum length</td>
<td>5 bytes</td>
<td>Unknown</td>
<td>Variable width</td>
<td>Typically, 20 Bytes determined by the server capacity</td>
<td>Unknown</td>
<td></td>
</tr>
<tr>
<td>Encoding format</td>
<td>Binary (UTF-8)</td>
<td>Character (XML &amp; EXI)</td>
<td>Binary</td>
<td>Binary/text</td>
<td>Binary</td>
<td>Text (HTTP like text syntax)</td>
</tr>
<tr>
<td>Overhead</td>
<td>Low/minimal</td>
<td>Yes</td>
<td>Low/minimal</td>
<td>Minimum</td>
<td>Unknown</td>
<td>Yes</td>
</tr>
<tr>
<td>Usage of topics</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Message payload size</td>
<td>256 MB</td>
<td>Variable size</td>
<td>32-bit</td>
<td>Minimum 0 and maximum 2^16-1=65535</td>
<td>Unknown</td>
<td>Unknown</td>
</tr>
<tr>
<td>Security</td>
<td>TLS/SASL</td>
<td>TLS/SSL</td>
<td>TLS/SASL</td>
<td>DTLS</td>
<td>TLS</td>
<td>Unknown</td>
</tr>
<tr>
<td>Throughput (msg/sec)</td>
<td>75.5 (QoS 0) 14.3 (QoS 1) 7.8 (QoS 2)</td>
<td>Unknown</td>
<td>67</td>
<td>148.3</td>
<td>Unknown</td>
<td>Unknown</td>
</tr>
<tr>
<td>Telemetry message</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Consumption</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Memory</td>
<td>Limited</td>
<td>High</td>
<td>Unknown</td>
<td>Limitedmemory (10 KB)</td>
<td>Unknown</td>
<td>High</td>
</tr>
<tr>
<td>Power</td>
<td>Low</td>
<td>High</td>
<td>Unknown</td>
<td>Low/Reduced</td>
<td>Unknown</td>
<td>High</td>
</tr>
<tr>
<td>Microsoft</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Azure IoT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Suite support</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QoS Option</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>3</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>QoS Level</td>
<td>3</td>
<td>No</td>
<td>3</td>
<td>2</td>
<td>No</td>
<td>Nearly 23 No</td>
</tr>
</tbody>
</table>

REFERENCES


An extended review of the application layer messaging protocol of the internet of things (Ronok Bhowmik)


An extended review of the application layer messaging protocol of the internet of things (Ronok Bhowmik)

Ronok Bhowmik is currently employed as a Programmer in the Cyber Center at Noakhali Science and Technology University (NSTU), Noakhali, Bangladesh. He received a B.Sc. Engg. in Computer Science and Telecommunication Engineering (B.Sc. Engg. in CSTE) from the NSTU, Bangladesh in 2012. He completed an M.Sc. in Computer Science from the Department of Computer Science and Engineering of Jahangirnagar University (JU), Bangladesh in 2016. His research interests include the areas of software engineering, the internet of things (IoT), and machine learning. He can be contacted at email: ronokbhowmik@admin.nstu.edu.bd.

Md. Hasnat Riaz is working as an Assistant Professor in the Department of Computer Science and Telecommunication Engineering (CSTE), at Noakhali Science and Technology University (NSTU). He received a B.Sc. Engg. in Computer Science and Telecommunication Engineering (B.Sc. Engg. in CSTE) in 2012 and M.Sc. in Telecommunication Engineering in 2016 from the NSTU, Bangladesh. His research interests include cloud computing, mobile cloud computing, the internet of things (IoT), data science, and machine learning. He can be contacted at email: hasnat.cste@nstu.edu.bd.