# A Survey of Application Layer Protocols of Internet of Things

**Nawab bibi[1], Faiza Iqbal[2†], Salwa Muhammad Akhtar[1], Rabia Anwar[1], Shamshad bibi[3]**
nawab.bibi123@gmail.com, faiza.iqbal@se.uol.edu.pk,salwaakhtar0728@gmail.com, rabia.anwar.554@gmail.com
[1]Department of Computer Science & Information Technology,
[2]Department of Software Engineering, The University of Lahore, Lahore, Pakistan.
[3]Minhaj University, Lahore

**Abstract**
The technological advancements of the last two decades directed the era of the Internet of Things (IoT). IoT enables billions of devices to connect through the internet and share their information and resources on a global level. These devices can be anything, from smartphones to embedded sensors. The main purpose of IoT is to make devices capable of achieving the desired goal with minimal to no human intervention. Although it hascome as a social and economic blessing, it still brought forward many security risks. This paper focuses on providing a survey of the most commonly used application layer protocols in the IoT domain, namely, Constrained Application Protocol (CoAP), Message Queuing Telemetry Transport (MQTT), Advanced Message Queuing Protocol (AMQP), and Extensible Messaging and Presence Protocol (XMPP). MQTT, AMQP, and XMPP use TCP for device-to-device communication, while CoAP utilizes UDP to achieve this purpose. MQTT and AMQP are based on a publish/subscribe model, CoAP uses the request/reply model for its structuring. In addition to this, the quality of service provision of MQTT, AMQP, and CoAP is not very high, especially when the deliverance of messages is concerned. The selection of protocols for each application is very a tedious task.This survey discusses the architectures, advantages, disadvantages, and applications of each of these protocols. The main contribution of this work is to describe each of the aforementioned application protocols in detail as well as providing their thorough comparative analysis. This survey will be helpful to the developers in selecting the protocol ideal for their system and/or application.

***Key words:***
*Internet of Things (IoT), Application layer protocols, MQTT, CoAP, XMPP, AMQP.*

## 1. Introduction

Technological growth helps bring ease into our life. One of the major technologies used nowadays is the Internet of things (IoT). IoT is built upon the notion that any device recognized on the internet can access the data available on the internet at anytime from anywhere. IoT allows the connection of billions of devices with each other and the internet. These IoT devices are connected to the internet and extend the intelligent computing and communication capabilities to numerous everyday devices such as navigation systems in cars, washing machines, refrigerators, coffee makers, alarm clocks, etc. It connects these devices with the internet, using microcontrollers, sensors, and actuators, thus forming a digital world. The main purpose of IoT is to make devices capable of achieving the desired goal with minimal to no human intervention. In 2011, the number of IoT devices became more than the total number of people on the planet [1]. By 2025, it is expected that IoT will be able to connect all the devices we use in our everyday life with the digital world [2].

IoT has been deployed for different applications which make our life easier, such as traffic management, home control, and automation, industrial automation, healthcare, battlefield, etc. However, the connection of numerous heterogeneous devices with each other and the internet bring forth several challenges and issues that hinder the implementation of IoT concepts in our everyday routine. Some of these challenges are privacy, security, data integrity and interoperability [3][4]. Communication of IoT devices with each other is essential for the stability of the Internet of Things (IoT) paradigm. These IoT devices require different communication protocols to tackle the interoperability issue. Various models have been proposed that allow devices to connect and share data. The most commonly used architecture of IoT has four layers. At the top is the application layer, then there is the service layer, the network layer and the sensor layer is the last. The applications layer is where the information collected by IoT devices is processed and displayed. The service layer is responsible for ensuring that the applications, tools, security, and infrastructure integrate effectively with existing systems. The network layer is responsible to support connections between devices. The sensor layer deals with end components of IoT and is used to sense and obtain the data of devices [5][6]. Figure 1 gives an overview of the IoT architecture.

For IoT devices to communicate with each other various protocols are required at each layer. Protocols are the rules which are responsible for governing the communication of devices in a network. These protocols are responsible for governing the movement of information from one device (source) to another(destination) using internet as the medium. OSI and TCP/IP protocol model exist as a standard protocol which are adapted when data is sent from one communicating device to another. The main difference between OSI/TCPIP model and the IoT architecture is that the former is responsible for characterizing and

Manuscript received November 5, 2021
Manuscript revised November 20, 2021
https://doi.org/10.22937/IJCSNS.2021.21.11.41

standardizing the communication functions of a system regardless of the technology it uses, while the latter is a general architecture that can be used as a reference for any type of IoT application [7].

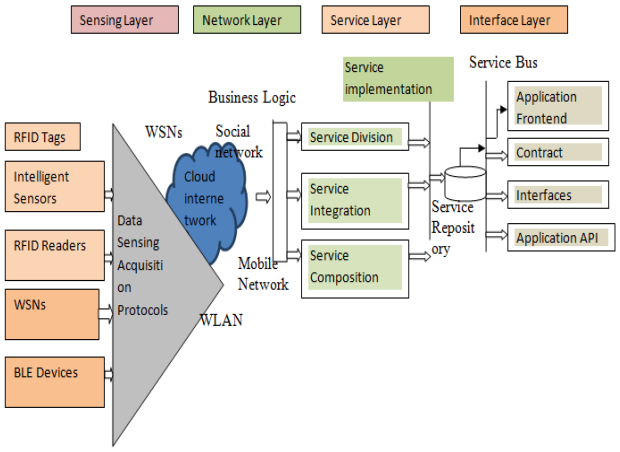


**Fig. 1** Internet of Things (IoT) Architecture.

The application layer of IOT relates with the end user and consists of software applications with their own application layer protocols. The existing protocols of the application layer were not able to perform efficiently with the IoT devices. These protocols include ZigBee, which is used for bi-directional communication between devices within the range of 10-100 m using short-range communication standards such as Bluetooth and Wi-Fi [8].With the advent of IoT era, some additional protocols were also needed to manage various challenges of IoT devices. Some of these challenges are as follows:

- Limited resources

IoT devices have limited resources at their disposal. Most of them have limited power life, limited processing power, and limited storage capacity which hinders their ability to fully accomplish their tasks.

- The heterogeneity of resources and devices

IoT connects heterogeneous devices. These devices perform different functions such as sensing the environment, performing control and actuation mechanisms, monitoring activities of agents within an environment. In order to process the heterogeneous data, collected and processedby these different devices, system requires some new application layer protocols.

- Security and Privacy

When IoT devices are sharing data with each other, they present the malicious users with numerous opportunities to hack the systems. This is extremely harmful for various life-critical systems such as an attack which disables the brakes of an automated car, or an attack on an automated insulin pump system, can severely risk human life. Similarly, an attack on an oil well, energy grid or water supply can be disastrous [9].

Constrained Application Protocol (CoAP) allows the functioning of low-power devices with minimal computational capability in the IoT environment. Message Queuing Telemetry Transport (MQTT) protocol is a message protocol based on publish/subscribe architecture, developed for the operation with low computational and space constrained devices. Advanced Message Queuing Protocol (AMQP)is a publish/subscribe model which allows interoperability and communication between different devices regardless of the language each device is using. Extensible Messaging and Presence Protocol (XMPP) is an instant messaging (IM) communication protocol, based on XML which used for multi-party communication. MQTT, AMQP and XMPP use TCP for device-to-device communication, while CoAP utilizes UDP to establish this communication. MQTT and AMQP are based on a publish/subscribe model whereas CoAP is based on request/reply model. In addition, the quality of service provision of MQTT, AMQP and CoAP is not very high, especially when the deliverance of messages is concerned [1].

The main contribution of this work is to describe each of aforementioned application protocols in detail as well as providing their thorough comparative analysis. Section II of this survey discusses research motivation of this research. Section III explains MQTT protocol, its architecture, applications, advantages and disadvantages. Section IV describes CoAP Protocol, its architecture, applications, advantages and disadvantages. Section V elaborates the XMPP protocol, its architecture, application, advantages and disadvantages. Section VI explains AMQP Protocol, its architecture, application, advantages and disadvantages. Section VII discusses the comparison of all these protocols. Finally, section VIII concludes this paper and summarizes contributions of this work.

## 2. Research motivation

The IoT concepts have helped in achieving a high level of technological advancements in many domains such as industries, smart environments, health care and everyday individual life. The devices in IoT have several constraints such slow processing capability, small storage capacity, short power life, etc. These devices are connected to each other using wired and wireless technology. However, most of the focus has been shifted to wireless communication. There are a number of communication protocols which are used to connect these resource bounded IoT devices with each other and to the internet using wireless technology.

During the literature review phase of this study, it has been discovered that thorough comparative analysis between existing application layer protocols is required. The main motivation of this work is to fill this research gap. The paper discusses, in detail, the architecture and working mechanism of the application protocols of IoT along with their advantages and disadvantages.

## 3. Message Queuing Telemetry Transport (MQTT) Protocol

### 3.1 Introduction

MQTT protocol allows the message to be sent and received, without the sender or the receiver knowing who is sending or receiving the information. This level of simplicity reduces the size of the message, resulting in reducing the demands on the network and the devices from which MQTT messages come from the client [10]. It is lightweight in nature andis based on the publish-subscribe modelfor allowing communication between numerous devices in a network.The Message Queuing (MQ) in MQTT is an open standard protocol [10] for delivering messages between devices. The Telemetry Transport (TT) in MQTT is used for transportation of different things [10]. The MQTT protocol is responsible for the transmission of data between devices in an IoT environment.

MQTT has three main components, a publisher, a subscriber and a broker. In MQTT, there is no need for the publisher (server) and subscriber (clients) to know the identity of each other. MQTT allows processing to be done by the server, therefore, it works efficiently with resource bounded IoT devices, with less processing and storage capacity. MQTT is easy to use and is flexible in nature, as it is able to manage both small and large devices [11].

Any MQTT connection has two agents the clients and the broker who acts as a server. The clients are the devices participating in the communication. Using publish-subscribe model, the publisher (server) sends the messages and the subscriber (client) requests a message to obtain the information in that message. The broker allows the connection of a client with the requested server. Here the client can be anything, a sensor, RFID tag or a mobile device, etc [12].

### 3.2 Architectural Detail

Figure 2 illustrates the MQTT protocol architecture. It mainly consists of three components, namely, subscriber, publisher, and broker. Subscriber is the client, publisher is the server, while broker is the message broker, also known as, integration broker or interface engine. A message broker acts as an intermediary computer program module, and is responsible for sending the message or specific topics which are requested by the subscriber to the publisher. In other words, it allows the translation of the formal messaging protocol of used by the subscriber to the formal messaging protocol of used by the publisher [13]. An interested device registers itself as a subscriber with the broker. When the subscriber sends a request for a certain type of information/data, the broker connects that subscriber with the relevant publisher, as a result of which communication between them begins. Authorization protocols are used by the broker to ensure that both the involved parties are legitimate [14]. To summarize the MQTT protocol architecture, a loose coupling between the information provider (publisher) and the information receiver (subscriber) is obtained using MQTT by introducing a broker between the two [10].

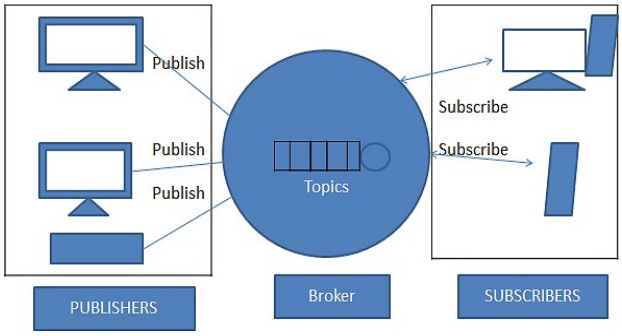


**Fig. 2** MQTT Protocol Architecture

Figure 2 illustrates that device 1 published a topic. Device 2 is subscribed to the topic which device 1 has published. So, device 2 receives the message [16]. Now-a-days many brokers use MQTT protocol for message delivering purposes. Among these Mosquitto is quite popular and uses the poll system call to deal with a large number of client-server devices. Mosquitto broker does three tasks. It, first, receives all the published and subscribed messages, then filters the received messages and finally publishes the messages to the registered clients.

Mosquito broker uses MQTT protocol. Due to its lightweight nature it is able to carry messages from one device to another, regardless of whether the devices involved are low power single board computers or full servers. Mosquito broker is a project by Eclipse. It has a free downloadable version for windows and Linux operating systems, thus making it compatible with most of the computer systems [15].

Mosquitto can be implanted using Apache ActiveMQ, which makes a thread pool and allots a single thread to manage numerous corresponding connections. Apache ActiveMQ is an open source message broker. It is just like mosquito broker which is able to send message from sender

to receiver. However, because of the message queue (MQ) capability of Apache Active MQ, it is able to connect more than one client to the server, by queuing the messages. Just like mosquito message broker it also implements MQTT. This allows the sending of message between two devices without them to be simultaneously available [17]. The major drawback of this technique is that it requires heavy context switching while handling numerous connections in parallel [13]. A comparison of Apache ActiveMQ [17] broker and Mosquitto [15] broker has been summarized in Table 1.

**Table 1**: Margin specifications

| Feature | Apache ActiveMQ | Mosquitto |
|---|---|---|
| Open source | Apache 2.0 | EPL/EDL |
| Windows Support | Yes | Yes |
| MQTT Implementation | Yes | Yes |
| MQTT Version | 3.1 | 3.1.1, 5.0 |
| Latest version | 5.15 | 1.6.2 |
| Clustering | Yes | No |

Figure 3 describes MQTT topic architecture. It describes the working of incoming messages and the response it generates. First a device specifies what topic it would like to publish. Topics are represented using strings separated by slashes"/". Each forward slash indicates a topic level e.g. Home/Office/Lamp [16]. Device 1 publishes the message "on/off", with the topic "home/office/lamp". Once the client subscribes, it will be able to turn the lamp on or off automatically.

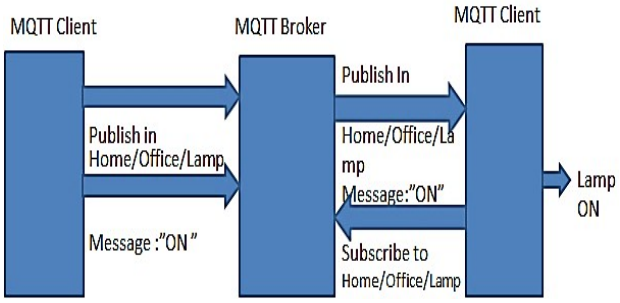


**Fig. 3** MQTT Topic Architecture [16]

## 3.3 Applications of MQTT

Health-care domain is the most prominent research field nowadays. With the numerous technological and medical achievements, a large number of medical issues have been resolved. However, even after all this, it is imperative to focus on improving this field as this field is directly related to the quality of human life. Due to this reason, many researchers and scientists from various fields are focusing their work to improve the quality of life and increase the average life expectancy of human life, by implementing their work on the health-care domain.

With the increasing population, providing health-care facilities to each and every individual around the world is becomes a tedious task. However, provision of health-care facility is a basic right of every human, and therefore, efforts must be made to provide this facility to the masses. Technology, particularly IoT, plays a vital role in connecting various medical devices, such as heart rate monitoring devices, body temperature monitoring devices, etc. embedded in geographically distributed smart environments, with each other to ensure the provision of medical facilities in a timely manner. These medical devices need reliable and efficient communication protocol to communicate with each other and other systems. MQTT is such a protocol. It helps in transferring messages from one device to another.

IoT-based healthcare applications [18] focus on physical sensors embedded in the environment around the patient, to provide them with continuous medical care. The mobile device is aggregation platform, such that the mobile device receives all the data collected by the sensors. The proposed framework [18] uses Raspberry Pi with Arduino for implementing and managing the concerned sensors. Other elements of the proposed work include Bio-impedance meter, Wi-Fi dongles, Battery and energy management system.

Other than health-care domain MQTT is also used in various other domains such as the industrial sector and in numerous web services. In a study [10], the authors have used MQTT in the industrial sector. Here, in point-of-sale situations, where sales information is sent and price updates are received. Major problem occurs when point-of-sale request is sent over a slow network. This issue can be dealt with using MQTT.

## 3.4 Advantages and Disadvantages of MQTT

This section describes advantages and disadvantages of MQTT. author has focused on the following of the MQTT. One of the major advantage of MQTT is that its lightweight Due to its lightweight nature, it is ideal for remote monitoring. The MQTT protocol is used in Industrial Internet of Things (IIoT) to help the proper functioning of Supervisory Control and Data Acquisition (SCADA) system. MQTT brings many powerful benefits to your process like reduction of network bandwidth consumption and development time. It provides improved scalability as the available bandwidth is maximized. MQTT allows the reduction of update rates to seconds thus remote sensing and control can be done in an efficient manner. It provides

secure permission-based security. MQTT brings many benefits to IIoT, in which sensors and devices are used particularly to enhance industrial processes. It makes distribution of information more efficient and provides three QoS levels [31].

There are some disadvantages of MQTT. As MQTT operates over TCP, therefore it requires more handshaking to increase the communication links exchanging messages, resulting in increasing wake-up and communication times. Moreover, since the broker is centralized thus it entails scalability issues [19].

## 4. Constrained Application (COAP) Protocol

### 4.1 Introduction

Constrained Application Protocol (CoAP), a session layer protocol, was first introduced by the IETF CoRE Working Group, which started the standardization of CoAP in March 2010. CoAP is a transfer protocol for web particularly optimized for constrained nodes, i.e., nodes or devices having limited memory and processing power, as well as resource-constrained networks, e.g., low power networks, in both IoT and M2M applications. CoAP is based on REST architecture. CoAP uses a subset of HTTP functionalities. These HTTP functionalities are redesigned as per the resource constraints of numerous IoT devices. Protocols can be made specifically for IoT applications, by modifying HTTP mechanisms.

HTTP is mainly concerned with the Transmission Control Protocol (TCP). TCP has many issues such as the inappropriate flow control mechanism and high overhead. In addition to this, TCP does not have multicast support for mobility. In contrast to HTTP, CoAP uses the User Datagram Protocol (UDP), due to which it has significantly lower overhead and improved multicast support. This is very helpful as many light weight applications in IoT have high overhead and power consumption. CoAP was designed to allow devices to use REST services to meet the device's power constraints. REST is the interface between the clients and the servers, and uses UDP protocol [20].

### 4.2 Architectural details

Figure 4 describes the architecture and functionality of CoAP in detail. Numerous devices such as the CCTV camera, RFID sensor, smartphone, etc., all act as CoAP client. The information generated by these clients on each particular day is sent to COAP server. After receiving this information, the CoAP server sends it to the Rest CoAP Proxy. The firewall connection is established for communication between CoAP environment and rest Internet.

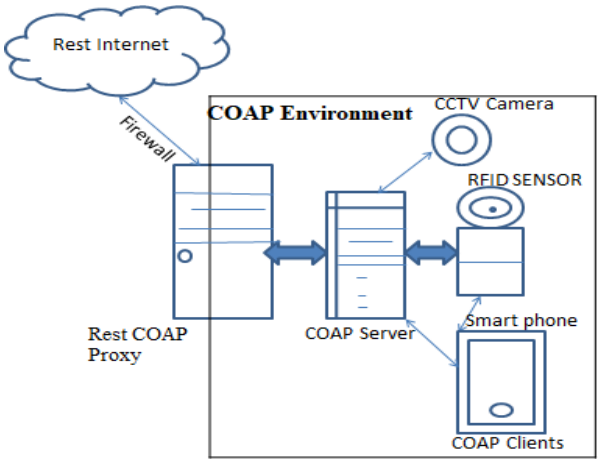


**Fig. 4** COAP Architecture.

The CoAP model is somewhat like the client/server model. Figure 5 explains that CoAP uses a two layers' structure [21]. The bottom of the two layers is the Message layer that has been designed to deal with UDP and asynchronous switching. While the top layer, which is the request/response layer, is concerned with the communication method and deals with the request/response message.

The abstract layering of CoAP includes Application layer, UDP layer, Requests/ Response and Messages layers [22]. CoAP can be merged with the application layer if the session layer lies between the transport layer and the application layer. At the transport layer, the UDP protocol is being used while different applications are being run in the application layer. COAP is organized in two sub-layers: (1) Messaging sub-layer which is the lower sub-layer and supports the four message types; Confirmable (CON), Non-Confirmable (NCON), Acknowledgement (ACK) and Rest(RST) message. (2) Request/response sub-layer is the upper layer and supports piggy-back, separate responses, Get request with a separate response and non-Confirmable request and response [6].

| Application | |
|---|---|
| Request/Response Layer | HTTP |
| Message Layer | TCP |
| Transport-UDP | IP |

**Fig. 5** Abstract Layer CoAP and HTTP Structural Model

The messages shared between clients and server are mainly concerned about reliability in three dimensions [21] i) reliability in sharing, ii) reliability of the network, iii) reliability in communication. The request/response sub-layer of CoAP is concerned with the following REST

messages; a) Confirmable: Retransmit until either acknowledgment arrives with the same message ID or timeout occurs in which the recipient fails to process the message and sends RST with the response. b) Non-confirmable modes represent the reliable and unreliable transmissions, respectively.

There exist other modes for request/response handling [6]. These include: Piggy-backed: When the client sends a request with a CON/NON, the server will send its response directly after receiving the message, i.e., within the acknowledgment message (ACK). ACK shall be received immediately with corresponding token number and message. Separate Response: The client sends the CON type message to the server and incase, the server is unable to respond immediately, an empty ACK is sent back to the client. After a certain time, the server sends the response in the form of a CON message with data. The ACK is then sent to the server from the client. A NON-type message is sent from the client to the server which has not given an ACK response. The server can in response send a NON-type in turn [6].

### 4.3 Applications of COAP

The main application of CoAP is in smart homes. The elements used in smart homes are mostly cheap and lightweight. Thus, CoAP is considered as the best option for smart home environments. The smart home environment provides handling and over-looking the energy consumption of home devices. This could result in warning against accidents before they occur, allowing devices to be remotely controlled, and reducing energy consumption dynamically. Using CoAP efficient exchange of information among nodes can be achieved. In system networking, data collection nodes consist of one proxy, smart socket and wireless data collection module. At the same time, this system will monitor the environment situation. If an abnormality occurs, the system will analyze it and switch off the relevant equipment [23].

### 4.4 Advantages and Disadvantages of COAP

This section describes advantages and disadvantages of CoAP. The main advantage of COAP is that it has reduced power requirements. It allows longer batteries life for IoT devices as it uses UDP. UDP requires minimum overhead for communications allowing faster wake up times and extended hypnotic states. It has smaller packet size which leads to faster communication cycles, thus resulting in longer battery life. Another key aspect of COAP is in security. When DTLS is used with UDP, the communication is encrypted and secure. However, some additional overhead is required to implement this. COAP presents asynchronous communication option. Clients can request to observe a device by setting a flag. The server (IoT device) can then stream state changes to the client as they happen. Either side can cancel the observe request. COAP supports IPv6. It was designed from the beginning to support IPv6, to allow multicasting. It allows resource discovery. Servers can provide a list of resources and media types. The client can then review and discover what is available [24].

There are also some disadvantages to CoAP. Due to the use of UDP with COAP, it results in message unreliability. UDP does not guarantee the delivery of datagrams. Even though, CoAP provides a way to request a confirmation acknowledgment to confirm the message was received. This does not verify that it was received in its entirety and decoded properly. CoAP is still in the developing phase. Standards are still maturing, however, it is likely to mature quickly as it is more popular. COAP presents some NAT issues. Network Address Translation (NAT) devices are commonly used in cloud environments. CoAP can have problems communicating with devices behind a NAT since the IP can be dynamic overtime. CoAP is unencrypted by default. This makes it natively unsecure and additional steps must be taken to make sure communication is not open to hackers [24].

## 5. Extensible Messaging and Presence Protocol (XMPP)

### 5.1 Introduction

Extensible Messaging and Presence Protocol (XMPP) is an instant messaging (IM) standard or a communication protocol, based on XML, used for multi-party communication [14]. XMPP allows the communication between two devices using the internet as the medium, regardless of the operating system of the devices. In other words, XMPP provides a real-time exchange of structured data. It provides IM applications with many advantages. It uses authentication mechanisms to ensure the reliability of the involved parties. It provides access control using the authentication mechanism and maintain data privacy. Encryption is done on hop-to-hop basis i.e. during transmission as well as end-to-end base i.e. at sender side. It allows other protocols to be compatible with it [25].

It is designed for close real-time applications and, thus, efficiently supports low-latency compact messages. It does not guarantee QoS. Moreover, XML messages have high overhead resulting in more power consumption of devices, which is major drawback for many IoT devices and applications. To overcome this, XMPP can be used. Even though it is rarely used in IoT domain but it has attracted the attention of the researchers in this domain due to its ability to improve the architecture to support IoT applications [20].

### 5.2 Architectural Detail

XMPP uses client-server architecture in which the client requests the connection to a server for exchanging messages. As the client-server model is decentralized by design, therefore, there is no central server in XMPP and there may be several servers running within one system or network. The XMPP client-server architecture is shown in Figure 6.


**Fig. 6** A simple XMPP architecture with two clients [7]

XMPP allows the users or the devices the ability to discover the available services along with the relevant information of these services, which may be residing within or across a network. Even though, with its decentralized client-server architecture, XMPP can ensure high scalability by allowing the specification of extension protocols (XEPs). One of these protocols, XEP-0174, can be used as an extension protocol, which enables server-less messaging without any infrastructure. Other XMPP extension protocols include, XEP -0045, which is used for efficient m:n communication for multi-user chats, XEP- 0030 for discovering a service and XEP-0060 for providing the user with the publish/subscribe functionality [5]. In XMPP, the exchange of data between devices is obtained via XML structured data, called "XML stanzas". An XML stanza is somewhat a piece of code having three major components; message, presence and iq (info/query). The structure of the XMPP Stanza is shown in Figure 7 [14].

XMPP uses TCP protocol for communication purposes. TLS is used for channel encryption and SASL is used for providing authentication to the overall system. The layering of XMPP includes Transport Control Protocol (TCP), Transport Layer Security (TLS), Simple Authentication and Security Layer (SASL) and Extensible Messaging and Presence Protocol (XMPP). An XMPP Client establishes an XML Stream with a server, after authenticating itself through SASL negotiation. An XMPP server verifies client authentication and then allows the client access to the XMPP network. XMPP allows two possible data communication paths; client-to-server and server-to-server. If both devices are clients, then a third device is required that acts as a server and allows them to communicate with each other after passing them through the TLS and SASL authentication channels [22].

### 5.3 Applications of XMPP

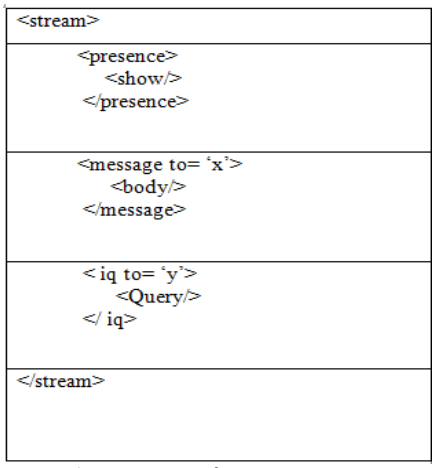


**Fig. 7** Structure of XMPP Stanza

VIRTUS uses publish/subscribe model and XMPP to allow devices to determine whether a message has successfully reached the destination or has failed to do so. It also helps the devices find out whether a subscriber is online or offline during message transmission. VIRTUS uses XMPP to allow the involved devices to share data with each other dynamically at runtime, without having to restart the entire IoT environment. In VIRTUS all the involved elements including sensors, algorithms and interfaces have been developed as modules allowing the building of many different versions of VIRTUS by reusing same modules in different compositions. In VIRTUS, each Module has an XMPP account. This account allows the system to know which modules are available at what time and also helps in the communication of modules with each other, achieved via the presence mechanism of the XMPP. XMPP Extension Protocols (XEPs) forms the basis of the VIRTUS architecture, these include; XEP-0050: this protocol is responsible for supporting communication between two XMPP devices. XEP-0004: this protocol is used for receiving responses from the destination. XEP-0030: this protocol is used to discover which devices are on the network. XEP-0060: this protocol is used to decouple the senders and receivers [26].

The sender publishes the events on an information node, and the receiver can subscribe itself to that node. This can also be done when the receiver is offline when the data is published. This comes very handy when developing e-health applications, where critical data must be retained even during poor connectivity. VIRTUS has the following three modules: XMPP server: for allowing access to XMPP services. Manager: for connecting different modules. Gateway: to provide an interface between an external software and a local instance. VIRTUS modules can be used to drive a sensor, implement ad-hoc application and connect to database [26].

### 5.4 Advantages and Disadvantages of XMPP

This section describes advantages and disadvantages of XMPP. One of the main advantage of the XMPP is that its free, open-source, easy to understand and implement protocol. It supports Google-Talk which can be accessed by any instant message (IM) supplier by using XMPP protocol. XMPP server can be used and run by anyone to manage the real-time messaging requirements of individuals as well as organizations. XMPP can also be used in network management, file sharing, gaming conventions, and remote monitoring [29].

There exist some disadvantages of XMPP. Practically there is no official support for XMPP clients or servers. Due to the distributed and decentralized nature of XMPP, data being transferred using XMPP protocol is repeated. Moreover, the overhead is extremely large in XMPP [30].

## 6. Advance Message Queuing Protocol (AMQP)

### 6.1 Introduction

Advanced Message Queuing Protocol (AMQP) is the international standard (ISO/IEC 19464), standardized by OASIS, developed for providing services like message orientation, queuing, point-to-point routing, publish/subscribe, reliability, and security. AMQP is a publish/subscribe model based on a reliable and efficient messaging queue and is used not only in business industry but also in commercial industry. The publish/subscribe approach of AMQP makes it highly extensibility. AMQP allows interoperability and communication between different devices regardless of the language each device is using. AMQP allows applications to share data among each other. AMQP thrives to provide applications/devices with reliability, security, and performance. It ensures reliability by using the message delivery guarantees which includes 1) At most once, 2) At least once, 3) Exactly once delivery.

### 6.2 Architectural Detail

Figure 8 shows the AMQP architecture along with its three major parts; Publisher(s), Consumer(s) and Intermediary/Server(s). Each of these three parts can be more than one in a single AMQP system and arranged on autonomous hosts. Distributors and purchasers talk to one another through message lines bound to trades inside the dealers. AMQP provides the message conveyance. Other parts of an AMQP framework are include 1) Distributor/Maker which is an application that develops messages with AMQP, 2) Directing Key Buyer is an application that gets the messages from at-least one distributers, 3) Message Line is an information structure that stores messages in memory or on plate messages and are put away in conveyance succession request. A Message Line is made by a buyer and is utilized solely by that customer, 4) Customer Trade is a coordinating and directing motor which acknowledges messages from distributors and duplicates the messages.

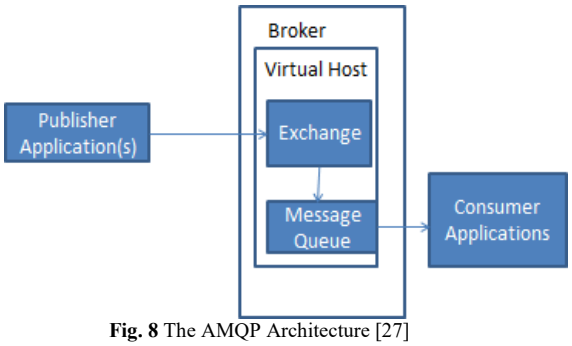


**Fig. 8** The AMQP Architecture [27]

A shopper makes a message line and uses the authoritative to connect the line with a trade. A Coupling ordinarily is related to a content string known as a Coupling Key. Trade course message by coordinating the messages 'directing keys against the lines' coupling keys. Trades are arranged into sorts dependent on the sort of key coordinating they perform of the Trade types that AMQP bolsters, we assess the accompanying: direct, fan-out, and Theme. Virtual Host: A client characterized namespace that gatherings and recognizes a lot of trades, message lines, and canister dings. Representative/Server: A server or daemon program that contains at least one virtual hosts, trades, message lines, and ties [27]. By characterizing a wire-level convention, AMQP usage can interoperate with one another. Correspondences are taken care of by two principle parts. Trades are utilized to course the messages to fitting lines. Steering among trades and message lines depends on some pre-characterized standards and conditions. Messages can be put away in message lines and after that be sent to collectors. Past this sort of point-to-point correspondence, AMQP likewise bolsters the distribute/buy in the interchanges model. AMQP characterizes a layer of informing over its vehicle layer. In this layer, AMQP characterizes two kinds of messages: uncovered back rubs that are provided by the sender and commented on messages that are seen at the beneficiary.

### 6.3 Applications of AMQP

AMQP has been used by authors in the stock market domain. Figure 9 represents the proposed application which has the defined components that make up its architecture. Gateway manages the buying and selling requests and delivers them the order queue of the matching engine via Qpid broke. Matching engine reads the orders from the order queue and compares the prices in the buy orders list with the prices in

the sell orders list. Once a match in both the lists, the matching engine moves forward to start the trading process. There are some messages which are sent by the matching engine to gateway, these are Trade confirmations, Order confirmations, Price quotes and Order authentication.

During this entire process, both the gateway and the matching engine send the performance measurement messages to the statistics reporter's queue via Qpid. A sampling interval from every 10,000 orders is used for taking the measurements. Statistics reporter receives the measurement messages from gateway and matching engine and compiles them to a file [27].

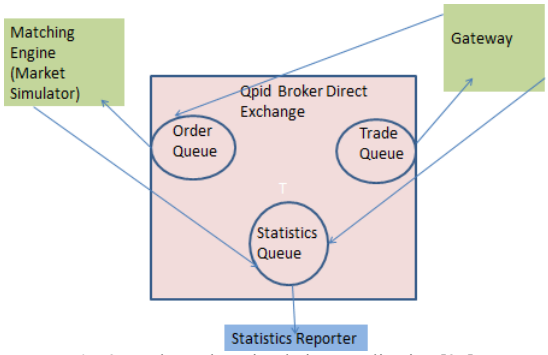


**Fig. 9** Stock Market Simulation Application [27]

### 6.4 Advantages and Disadvantages of AMQP

This section lists advantages and disadvantages of AMQP. One of the key advantage of AMQP is that it is very lightweight therefore it is very fast, consumes less CPU time and is easy to configure and run. Another advantage is the use of event based networking which can control lots of incoming connections thus supporting many clients using various languages. Moreover, AMQP has great symbolic logic for a work queue with its 3 states i.e. ready, reserved and buried. It supports delaying a job before it appears ready when submitted. It also supports TTR (time-to-run) semantics - so if a job takes more than the defined TTR, it will be available to other consumers even if the original consumer didn't finish yet. AMQP is very easy text-based protocol which inspired by the textual Mem cache protocol. Due to this protocol, clients are very easy and straight forward. Transactional are strained job reaches only 1 consumer. If the connection is lost or the consumer returns it, only then it will be available to other consumers to be quiet. If consumers finished the job, it will delete it. AMQP also guides intruding message [28].

There are also some disadvantages of AMQP. It handles redundancy at the client side like Mem cached so if a server goes down and there is no access to its disks, clients may lose their jobs. In AMQP, persistency model is a bit complex so for large queues, very large files can acquire results. These large files are already allocated but may never get cleared up. AMQP does not presents any safety model and doesn't run on Windows. It can, sometimes, also result in dealer lock in because there are no other such servers supporting the beans talked protocol [28].

## 7. Comparison and Evaluation

Table 2 represents summarized comparison of application protocols after conducting a thorough literature review. The survey focuses on the most commonly used application layer protocols along with their architecture, advantages, disadvantages, and their practical applications. Even though all of them are application layer protocols in IoT used for communication between clients and servers, they still have vast differences. MQTT is a lightweight protocol which runs over TCP/IP and uses publish/subscribe approach to send and receive messages. AMQP supports asynchronous messaging and allows for encrypted and secure data transmission. XMPP is based on XML and helps to transmit data in near-real-time. CoAP is a message transfer protocol which runs on UDP and is specifically designed to support communication between constrained devices over constrained networks. With the help of this survey it became clear that each application has its own set of requirements that can be achieved using different protocols and mechanisms, based on the scenario.

**Table 2:** Comparison between COAP, MQTT, AMQP, XMPP

| Sr.No | Parameter | CoAP | MQTT | AMQP | XMPP |
|---|---|---|---|---|---|
| 1 | Design Goal | To keep message overhead small and thus limit- ing the need of fragmentation | To reduce network bandwidth | To pass business messages between applications, and connect them on different platforms | To enable users to connect to multiple instant messaging systems |
| 2 | Types of applications supported | Machine-to-machine applications | Machine-to-machine and mobile applications | Business applications | Impractical or m-to-m communication |
| 3 | Mode of message exchange | Both synchronous and asynchoronous | Asynchronous | Asynchoronous | Both synchoronous and asynchoronous |
| 4 | Responsive/ Real-time | Real–time protocol | Real–time protocol | Not a Real–time protocol | Real–time protocol |
| 5 | Reliability | Built-in reliable mechanism | Reliability of message delivery through different QoS levels | Store-and-forward feature ensure reliability | TCP ensure reliability |
| 6 | Security | DTLS | SSL/TLS | SSL/TLS | SSL/TLS |
| 7 | Quality of service (QoS) | Provides QoS | Provides QoS | Provides QoS | No QoS option |
| 8 | Architecture | Request/Response | Publish/subscribe | Publish/Subscribe | Publish/ Sub-scribe and Re-quest/Response |
| 9 | Transport | UDP | TCP | TCP | TCP |
| 10 | Power Consumption | Decrease power consumption | Decrease power Consumption | More consumption than MQTT | Increase power consumption |
| 11 | Applicability | Web services | Small size, small power mobile applications | Business and commercial platforms | Chatting and message exchange |
| 12 | Network band-width | Increase network bandwidth | Decrease network bandwidth | Decrease network bandwidth | High consumption of N/W bandwidth |

The survey discovered the importance of publish subscribe protocol which can be considered as a great choice when dealing with applications that either generate, processes or analyzes a large amount of data, a common trait shared by all currently available IOT applications.

## 8. Conclusion

This survey discusses in great detail four of the most commonly used application layer protocols, namely, CoAP, XMPP, AMQP and MQTT. This survey provides a detailed explanation of each of these protocols and covers the architecture and working mechanism of these protocols, their physical implementations or applications and their advantages and disadvantages. This work also briefly discusses the history of these protocols. After a thorough literature review it was concluded that publish/subscribe protocols are very effective when it comes to handling large volumes of data, which is a very common phenomenon in IoT domain due to the interconnection of billions and trillions of devices constantly generating and sharing data with each other. The selection of protocols for each application is very a tedious task. This work will be able to help developers in the selection of the optimal protocol for their applications.

## References


[1] Karagiannis,V., Chatzimisios,P., Vazquez-Gallego,F., & Alonso-Zarate,J.:*A Survey on Application Layer Protocols for the Internet of Things.* Transaction on IoT and Cloud Computing, 1(1), 9-18 (2015)

[2] Guner, A., Kurtel, K., &Celikkan, U. (2017, October). A message broker based architecture for context aware IoT application development. In International Conference on Computer Science and Engineering (UBMK) (pp. 233-238). Antalya, Turkey: IEEE.

[3] Suresh, P., Daniel, J. V., Parthasarathy, V., &Aswathy, R. H. (2014, November). A state of the art review on the Internet of Things (IoT) history, technology and fields of deployment. In International conference on science engineering and management research (ICSEMR) (pp. 1-8). Chennai, India: IEEE.

*[4]* Alseady, S., Baz, A., Alsubait, T., Alarabi, L.,& Alhakami, H.:*Towards Security Challenges to Internet-of-Things: Big Data, Networks, and Applications.* IJCSNS International Journal of Computer Science and Network Security, vol.20(11), pp. 131–141.(2020)

*[5]* Bendel, S., Springer, T., Schuster, D., Schill, A., Ackermann,R., & Ameling,M.. *A Service Infrastructure for the Internet of Things based on XMPP. In* IEEE international conference on pervasive computing and communications workshops (PERCOM Workshops) (pp. 385-388). San Diego, USA: IEEE. (2013)

[6] Mitra,J. (2018). *Internet of Things – A review of the Architecture of Networking and Communication Protocols.* International Journal of Scientific Development and Research (IJSDR), 3(5), 229-245.

[7] Bassole, D., Kabore, K. K., Traore, Y., Sie, O., & Sta, H. B. (2019). Design and implementation of secure communication protocols for Internet of Things systems. 2019 IEEE International Smart Cities Conference (ISC2), Casablanca, Morocco, 2019, pp. 112-117.

[8] Babu, B.S., Srikanth, K., Ramanjaneyulu, T., & Narayana, I.L. (2016). IoT for Healthcare. International Journal of Science and Research (IJSR), 5(2), 2319-7064.

[9] Colitti, W., Steenhaut, K., & De-Caro, N. (2011). Integrating Wireless Sensor Networks with the Web. In Proc. Extending the Internet to Low power and Lossy Networks (IP+SN) (pp. 32-36). Chicago, IL: IEEE.

[10] Lampkin, L., Leong, W.T., Olivera, L., Rawat, S., Subrahmanyam, N., & Xiang, R. (2012). Building Smarter Planet Solutions with MQTT and IBM WebSphere MQ Telemetry Austin, TX: IBM Redbooks

[11] Deschambault, O., Gherbi, A., &Légaré, C. (2017). Efficient implementation of the MQTT protocol for embedded systems. JIPS (Journal of Information Processing Systems), 13(1), 26-39.

[12] Kashyap, M., Sharma, V., & Gupta, N. (2018). Taking MQTT and NodeMCU to IOT: Communication in Internet of Things. In Procedia Comput. Sci. (vol.132, pp.1611–1618). Amsterdam,NL: Elsevier.

[13] Pipatsakulroj, W., Visoottiviseth, V., &Takano, R. (2017). muMQ: A lightweight and scalable MQTT broker.In IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN) (pp. 1-6). Osaka, Japan: IEEE.

[14] Al-Fuqah, A., Guizani, M., Mohammadi, M., Aledhari, M., & Ayyash, M. (2015). Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. IEEE Communication Surveys Tutorials ,17(4), 2347-2376.

[15] da Silva, A. C. F., Breitenbücher, U., Képes, K., Kopp, O., &Leymann, F. (2016, November). OpenTOSCA for IoT: automating the deployment of IoT applications based on the mosquitto message broker. In Proceedings of the 6th International Conference on the Internet of Things (pp. 181-182). Stuttgart, Germany: ACM.

[16] Santos, R. (2017). What is MQTT and how it works. Retrieved from: https://randomnerdtutorials.com/what-is-mqtt-and-how-it-works

[17] Snyder, B., Bosanac, D., & Davies, R. (2011). Active MQ in Action. New York, USA: Manning Publications.

[18] Jamin, A., Fasquel, J.B., Lhommeau, M., Cornet, E., Abadie-Lacourtoisie, S., Henni, S., & Leftheriotis, G. (2016). An Aggregation Plateform for IoT-Based Healthcare: Illustration for Bioimpedancemetry, Temperature and Fatigue Level Monitoring. Internet of Things Technologies for HealthCare, 1(2), 125–130.

[19] Minteer, A. (2017). Analytics for the Internet of Things (IoT). Retrieved from: https://www.oreilly.com/library/view/ analytics-for the/ 9781787120730 /e86ff73d-7e8c-4eda-9890-0ceebbadcf78.xhtml

[20] Salman, T., & Jain, R. (2016). Networking Protocols and Standards for Internet of Things. Retrieved from: https://www.cse.wustl.edu/jain/cse570-15/ftp/iotprot/index.html

[21] Rahman, R.A., & Shah, B. (2016). Security analysis of IoT protocols: A focus in CoAP. In 3rd MEC International Conference on Big Data and Smart City (ICBDSC) (pp. 1-7). Muscat, Oman: IEEE.

[22] N˘astase, L. (2017). Security in the Internet of Things: A Survey on Application Layer Protocols. In 21st International Conference on Control Systems and Computer Science (CSCS) (pp. 659-666). Bucharest, RO: IEEE.

[23] Chen, X. (2014). Constrained Application Protocol for Internet of Things. Retrieved from: https://www.cse.wustl.edu/jain/cse57414/ftp/coap/index.html

[24] Minteer, A. (2019). Analytics for the Internet of Things (IoT). Retrieved from https://learning.oreilly.com/library/view/analytics-for-the/9781787120730/ee557386-c97f-48ee82c8-625b495fffba.xhtml.

[25] Bellavista, P., & Zanni, A. (2016). Towards better scalability for IoT-cloud interactions via combined exploitation of MQTT and CoAP. In IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI) (pp. 1-6). Bologna, Italy: IEEE.

[26] Bazzani, M., Conzon,D., Scalera,A., & Trainito,C.I. (2012). Enabling the IoT paradigm in e-health solutions through the VIRTUS middleware. In IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications (pp. 1954-1959). Washington, DC: IEEE.

[27] Subramoni, H., Marsh,G., Narravula,S. ,PingLai, & Panda,D.K. (2008). Design and evaluation of benchmarks for financial applications using Advanced Message Queuing Protocol (AMQP) over InfiniBand. In Workshop on High Performance Computational Finance (pp. 01-08). Austin, TX: IEEE.

[28] Sharma, C., & Gondhi, D. N. K. (2018). Communication Protocol Stack for Constrained IoT Systems. In 3rd International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU) (pp. 01-06). Nainital, Uttarakhand: IEEE

[29] Nie, P. (2006). An open standard for instant messaging: eXtensible Messaging and Presence Protocol (XMPP). In Seminar on Internet working, (pp. 1-6). Helsinki, Finland: Helsinki University of Technology.

[30] Malik, M.I, McAteer, I.N., Hannay, P., Syed, N.F., & Zubair, B. (2018). XMPP architecture and security challenges in an IoT ecosystem. In proceedings of the 16th Australian Information Security Management Conference (pp. 62-73).

[31] Veeramanikandan, M., Sanakarana, S. (2019). Publish/subscribe based multitier edge computational model in internet of things for latency reduction. Journal of Parallel and Distributed Computing, 12(7), 18-27.