Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 7.056

IJCSMC, Vol. 9, Issue. 6, June 2020, pg.77 – 81

Comparative Study of IoT Protocols and Classification of Cloud Platforms

Saumya Mathkar[1]; Nagaraj Vernekar[2]; Gautam Jotkar[3]
[1]Dept. of Computer Science Engineering, Goa College of Engineering, Ponda, Goa, India
[2]Dept. of Computer Science Engineering, Goa College of Engineering, Ponda, Goa, India
[3]Research and Development Department, Energy Automation Siemens Ltd., Verna, Goa, India
[1] saumyamathkar2@gmail.com; [2] nv@gec.ac.in; [3] gautam.jotkar@siemens.com

Abstract— Nowadays with evolving technologies, it is possible to connect real world with virtual world by using some intelligent sensors or physical object. Internet of Things (IoT) is one of such technology. The rise of interest in IoT is leading to a new era in which all objects, devices and sensors that are connected over the internet. This will change the pattern of how we collect and analyse data to make our systems more efficient. This will in turn improve quality of mankind. Using IoT the user will be able to communicate and control physical devices. This survey paper describes and compares some major protocols like MQTT, AMQP, CoAP, XMPP, HTTP/HTTPS, RESTFUL SERVICES and WEBSOCKETS. Comparison of IoT protocols is done based on parameters like what type of transport, security, Architectural style it provides and how it works in Low power and lossy network. We have mentioned whether the protocol support Publisher/Subscriber mode or Request/response mode for communication. We have also discussed some major cloud platforms that are used in IoT like AWS, AZURE, firebase cloud computing. We have made comparative study on cloud platform based on its computing architecture, services, load balancing techniques and security.

Keywords— Internet of things (IoT), MQTT, AMQP, CoAP, XMPP, HTTP/HTTPS, RESTFUL SERVICES, WEBSOCKETS, AWS, AZURE and firebase cloud computing

I. INTRODUCTION

Machine to Machine communication is a pretty old idea (ie. M2M) but IoT is not. Many companies are trying to shift from M2M to IoT. IoT technologies allow things, or devices to act smartly. In IoT devices can connect remotely to the internet using WIFI or GSM module. In IoT connection, end points use a special set of rules and regulations to communicate with other end points in a network. In this paper, we will discuss about the protocols of IoT which are working at different layers. We will also discuss the cloud computing platforms that are used in IoT.

II. PROTOCOLS IN IOT

A. Message Queue Telemetry Transport (MQTT)

Message queue telemetry transport (MQTT) is asynchronous publish/subscribe protocol. MQTT is a light weight, open and easy to implement. MQTT protocol use TCP/IP or other protocol that gives ordered, and bidirectional connections. This protocol supports low bandwidth and high latency networks. MQTT is widely used protocol in various platforms to connect things in IoT. MQTT play major role in IoT as it is widely used as a messaging protocol between things and servers.

Publish/Subscribe protocols convene better requirement of IoT than request/response protocol. In Publish/Subscribe, clients don't have to request for updates. The bandwidth of the network decreases and the need for using computational resources is dropping.

MQTT works as follows –

Broker in MQTT [7] contains topics. In MQTT the client can be publisher or subscriber. As a publisher client can publish or send information to the broker at specific topic or as a subscriber client can receives automatic messages on every new update in the topic he subscribed. MQTT protocol is designed in such a way that we can sparingly use bandwidth and battery. MQTT is presently used in Facebook Messenger [19].

MQTT has less overhead compared to TCP based protocols [9]. To provide security, MQTT broker may require username/password authentication. TLS/SSL (secure sockets layer) handles this authentication. When we compare MQTT to UDP-based CoAP, we can see that MQTT has higher overhead compared to CoAP. But, due to absence of TCPs retransmission mechanism, loss of packet is more in CoAP. For low packet losses MQTT experiences lower delays than CoAP. Furthermore delays and packet loss are based on the QoS. When the level of QoS is advanced delays and packet loss degrades increases.

B. Constrained Application Protocol (CoAP))

Constrained Application Protocol, (CoAP) is a synchronous request-response protocol [6]. It is a messaging protocol and uses REST (Representational State Transfer) architecture. Most of IoT devices are resources constrained (i.e., low computing capability and small storage) we cannot use HTTP in IoT due to its complexity. To overcome the HTTP complexity issue, CoAP was proposed to modify some HTTP functions to meet the requirements for IoT. CoAP was implemented with the help of the HTTP methods [4]. To maintain the overall design lightweight, CoAP runs over UDP. POST, GET, DELETE AND PUT are the HTTP commands used in CoAP. These commands are used to offer resource-oriented communications between client/server architecture.

CoAP is known as request/response protocol, it utilizes synchronous as well as asynchronous responses. This UDP-based protocol is designed to decrease bandwidth requirements and eliminate the TCP overhead [3]. Additionally, CoAP supports multicast as well as unicast, compared to TCP. CoAP is running on the UDP which is not reliable. However, CoAP has incorporated its own mechanisms for providing reliability. In each packet, 2 bits inside the header states the message type and the required QoS levels. Altogether there are 4 types of messages :

1. Confirmable: "request message" that needs an acknowledgement (ACK).

2. Non-Confirmable: "request message" that does not require acknowledgement.

3.Acknowledgment: It confirms the response of confirmable message.

4.Reset: It confirms the response of a message that could not be further processed.

There is a Stop-and-Wait retransmission mechanism for confirmable messages. In this mechanism, 16-bit header field in each CoAP packet called Message ID which is unique ID and it is used for detecting duplicates. CoAP's HTTP Mapping allows clients to use resources on HTTP servers. This is done by reverse proxy which is used to translate the HTTP Status codes into Response codes [2]. CoAP was created for the M2M and IoT communications but it does not include any security features. To make CoAP secure another protocol ie Datagram Transport Layer Security (DTLS) is proposed. DTLS provides data integrity, authentication, cryptographic algorithms, confidentiality and automatic key management [5]. UDP transmission is secured by DTLS, but it was not proposed for IoT. DTLS protocol does not support multicasting mechanism [24], which is a main advantage of CoAP when we compare to other protocols. DTLS handshakes [8] require additional packets that occupy additional computational resources, increase the network traffic and shorten the lifespan of mobile devices that run on an essential part of the IoT like batteries. Being designed for the IoT, CoAP is compatible with HTTP. CoAP meets web requirements like simplicity, minimized overheads and multicasting.

C. Extensible Messaging and Presence Protocol (XMPP)

Extensible Messaging and Presence Protocol (XMPP) was mainly designed for message exchanging and chatting. But being an older protocol, it failed to offer the necessary services for the latest applications. Because of this, Google stopped supporting the XMPP standard. However, recently XMPP has regained a lot of attention as a communication protocol in IoT. XMPP protocol inherits the features of XML protocol. Because of this, XMPP has great scalability, addressing, and security capabilities. XMPP can be used in video streaming, multiparty chatting and voice.

The three roles that are included in XMPP are as follows: client, server, and gateway. Message routing and Functionality of link management can be achieved by a server. Stable communication among all heterogeneous system is supported by gateway. Using TCP/IP protocol Client can connect to the server. Client can transmit context based on XML streaming protocol. XMPP can be operated in IoT as it supports the object to object transmission with XML-based text messages. XMPP runs over TCP/IP and provides request/response (synchronous) and publish/subscribe (asynchronous) messaging systems. XMPP designed for real-time transmission that is why it supports lower latency message exchange and message footprint [11]. XMPP has TLS/SSL security which is built in the core of the specification.

As XMPP supports the publish/subscribe architecture which widely used in IoT in compared to request/response approach of CoAPs. XMPP has already provided the protocol which is supported by all over the Internet with regard to the MQTT [12]. XMPP uses extensible Markup Language (XML) messages that results in overhead and necessitate XML parsing that require extra computational ability which increases power consumption.

D. Advanced Message Queuing Protocol (AMQP)

Advanced Message Queuing Protocol (AMQP) is used to provide message service (routing, queuing, security and reliability etc.)[15]. It focuses mainly on the message-oriented environments and can be considered as a message-oriented middleware protocol. AMQP protocol provides publish/subscribe (asynchronous) communication with messaging. The main advantage AMQP is its store-and-forward feature. This feature provides reliability even when there is network disruptions [15]. AMQP provides Security by using TLS/SSL protocols over TCP [17]. Success rate of AMQP is directly proportional to its bandwidth. Comparing AMQP with REST, AMQP can send more messages per second [14].

E. The Hypertext Transfer Protocol (HTTP)/ The Hypertext Transfer Protocol Secure (HTTPS)

Hypertext Transfer Protocol (HTTP) is an application protocol for collaborative, distributed, hypermedia information systems. In IoT technology based on communication HTTP is applied for transmitting large number of packets. HTTP protocol sends many small packets to the server to get connected. This may lead to usage of high traffic which may cause high network resource utilization and may lead to network delay. HTTP works on TCP/IP which provides a reliable communication. The whole communication process is completed by means of persistent establishment of connection in the network. During the communication some problems causes due to overhead of protocol in IoT. Normally all HTTP calls are stateless. It provides authentication every time it as it is connected to IP or URL to do Rest API calls so the session is not saved. So, after getting the response the device closes the connection. Therefore, IoT causes serious overhead in network communication.

HTTPS was designed Netscape Communication in 1994. Originally, to provide security HTTPS was used with the SSL protocol. But after a time, SSL evolved into Transport Layer Security (TLS). Protocol overheads similar to network resources and large delay in HTTP are reduced by MQTT.

F. WEBSOCKET

Websocket protocol was developed as part of the HTML 5. Websocket is neither publish/subscribe nor a request/response protocol [10]. In Websocket to establish a Websocket session, first client initializes a handshake with a server. The handshake is similar to HTTP protocol, web servers can handle Websocket sessions and HTTP connections through the same port [20]. The HTTP headers are eliminated during a session. Clients and servers can swap messages in full-duplex asynchronous connection. The session is close when it is no longer needed from either server or client side. To reduce the transmission overhead, Websocket was created by allowing full-duplex real-time communications. There is also a Websocket sub-protocol called Websocket Application Messaging Protocol (WAMP). WAMP provides publish/subscribe messaging systems. Websocket runs over the TCP protocol. It does not implement reliability mechanisms. If required, the sessions can be secured using the Websocket over TLS/SSL. Websocket is not suitable for resource constrained devices like other previous protocol. Client/server-based architecture of websocket does not suitable for IoT applications. However, it is designed for real-time communication, it minimizes overhead, it is secure. It can provide efficient messaging systems by using WAMP. Thus, it can compete any other protocol running over TCP.

G. RESTFUL SERVICES

Representational State Transfer (REST) is not really a protocol but an architectural style [14]. REST uses the HTTP methods like GET, POST, PUT, and DELETE to provide a resource-oriented messaging system. In this all events are performed with request/response (synchronous) HTTP commands. It uses the built-in HTTP accept header. This header indicates the data format. The content type can be XML or JavaScript object notation (JSON) and depends on the HTTP server and its configuration. REST plays an vital part in IoT because it is widely used by all the commercial M2M cloud platforms. Moreover, it can be implemented easily in tablet applications and Smartphone because it only uses HTTP library that is available for all the Operating Systems. The HTTP features can be completely used in the REST together with authentication, cashing and content type negotiation [13]. RESTful make use of reliable and secure HTTP. It uses TLS/SSL for security. However, most of M2M platform provide unique authentication keys for authentication instead of using HTTP request. To achieve security these keys, have to be in the request header. Even if REST is widely used in M2M platforms, it is not likely to become a leading protocol due to the fact that, it has complex implementation. It uses HTTP that means it is not compatible with constrained-communication devices. The additional overhead associated to request/response protocols affect battery usage. It also continuously polls for values especially when there are no new updates and the overhead becomes useless. Issues can be avoided if we use a publish/subscribe protocol such as MQTT or XMPP.

TABLE I.

COMPARISON BETWEEN IOT PROTOCOLS

Protocols	**CoAP**	**MQTT**	**XMPP**	**AMQP**	**HTTP/REST**	**Web-socket**
transport	UDP	TCP	TCP	TCP	TCP	TCP
Publisher/subscriber	No	Yes	Yes	Yes	No	Yes (with wamp)
Request/response	Yes	No	Yes	No	Yes	No
security	DTLS	SSL	SSL	SSL	SSL	SSL
QoS	Yes	Yes	No	Yes	No	Yes

Open source	Yes	Yes	Yes	Yes	Yes	Yes
Architectural style	P2P	Broker	P2P	P2P, Broker	P2P	P2P
Low power and lossy network	Excellent	Fair	Fair	Fair	Fair	Fair

III.COMPARATIVE STUDY OF CLOUD PLATFORM

In this section we will briefly discuss about cloud providers. We will classify AWS, Azure and firebase cloud in terms of its architecture, load balancing, services, security and programming framework.

A. *Cloud Architecture*

The Cloud architecture is a design of software applications which uses internet accessible services. Cloud Computing architecture comprises of many cloud components, which are loosely coupled.

1. *Services Software as a Services (SaaS)* – Mostly in SaaS we do not require to download and install the application, we can directly run it through the web browser. SaaS is also called as cloud application service. Some important examples are Salesforce Dropbox, Google Apps, Hubspot, Slack, Cisco WebEx.

2. *Platform as a Services (PaaS)* – It is partially similar to SaaS, but only the dissimilarity is PaaS gives a platform for creation of software. It is also known as cloud platform services. When we compare this to SaaS, we can access software on the internet in SaaS because it is independent of any platform. Example: Force.com, Windows Azure, OpenShift, Magento Commerce Cloud.

3. *Infrastructure as a Service (IaaS)* – It is also called as cloud infrastructure services. It is responsible for managing applications data, runtime environments and middleware. Example: Amazon Web Services (AWS) EC2, Google Compute Engine (GCE), and Cisco Metapod.

4. *Backend as a Service (BaaS)* – It provides a backend for applications (mostly mobile). They offer tools and applications for various computer languages to integrate with their backend. They also offer other services like Analytics, storage, Push notifications, dashboards, social integration. It is similar to SaaS, but BaaS is typically targeted at developers, whereas SaaS is targeted at end users. Example- firebase cloud.

B. *Load balancing*

Cloud load balancing is the method of resource distribution and workloads in a computing environment. It allows managing application by allocation of resources within various networks, computers, or servers. Cloud load balancing involves hosting the traffic in workload and demands that exist over the Internet. Load balancing is operated to implement failover (continuation of service after the failure or breakdown of components). These components are observed continuously. When one component does not respond, the load balancer is informed and it stops sending traffic to it. Load balancing also enables other features such as scalability.

C. *Security*

Cloud security provides a set of procedures, policies, controls and technologies to secure cloud-based systems, infrastructure and data. The security measures are implemented to protect data, secure users privacy by providing authentication rules for users or devices and to support regulatory compliance.

D. *Cloud classification based on services*

TABLE I.
CLOUD CLASSIFICATION

	Amazon web service	**Azure**	**Firebase cloud computing**
Computing architecture	Elastic compute cloud(EC2) allows uploading XEN virtual machine images to the infrastructure and gives client APIs to instantiate and manages them. Public cloud	An inter-scale cloud services platform hosted in Microsoft data centers, which provides an OS and a set of developer services that can be used individually or together	Inherits Google Cloud Messaging (GSM's) Infrastructure. But simplifies the client side development
Load balancing	Service allows users to balance incoming requests and traffic across multiple EC2 instances. Round robin load balancing, HAProxy	Built –in hardware load balancing	Optimize how much of your database's capacity is in use processing requests at any given time (reflected in **Load** or **io/database_load** metrics).

Service	IaaS, Xen images	Paas	BaaS
Security	Type II (SAS70 Type II) certification, firewall, X.509 certificate, SSL protected API, access control list	Tokenservice (sts) creates Security Assertion Markup Langauge token according to rule	Firebase support for GDPR and CCPA ISO and SOC compliance,
Programming framework	Amazon machine Image(AMI), Amazon Mapreduce framework	Microsoft.NET	_

IV. CONCLUSIONS

In this paper, we have presented some IoT protocols that have gained lot of importance in IoT. Among them, we have identified CoAP as the only one that runs over UDP, thus making it the most lightweight, followed by Websocket that significantly reduces the communications overhead. The communication and computational ability of the devices involved must be also considered when choosing the most appropriate protocol. If battery consumption and constrained communication are not an issue, RESTful services can be easily implemented and interact with the Internet using the worldwide HTTP. On the contrary, MQTT is not that commonly used like HTTP but has proved that it can be efficiently used for battery devices. There are various factors that influence the selection of IoT protocols, the most important among this are the communication, computational ability of the devices and battery consumption. When we move towards cloud platforms, clouds are intended to offer services to outside users. We have classified the cloud based on its features provided. Both Azure and AWS Pricing models offer pay as you go structure [20]. Azure charges, per minute basis whereas AWS charges you on hourly basis. Azure offers more flexibility for short term subscription plans [1],[16]. In case of certain services, Azure tends to be costlier than AWS when the architecture starts scaling up. Google's Firebase provides a cohesive set of services that are simple and easy to understand. Pricing is clear and there is excellent support for Android and iOS [18]. In AWS calculating the total cost to run your solution is not a straightforward. Microsoft have gone some way to help simplify pricing and management for their services.

REFERENCES

[1] Shukurte Luma-Osmani, florim Idrizi, Shpresa Ademi, Ramzan Fetai,"Above the clouds: a brief overview of Microsoft Azure environment and applications", 5-6 2018.

[2] Maria Rita Palattella, Nicola Accettura, Xavier Vilajosana, Thomas Watteyne, Luigi Alfredo Grieco, Gennaro Boggia, Mischa Dohler, "Standardized Protocol Stack for the Internet of (Important) Things", Communications Surveys & Tutorials IEEE 15(3), 2013.

[3] Sye Loong Keoh, Sandeep S. Kumar, Hannes Tschofenig, "Securing the Internet of Things: A Standardization Perspective", Internet of Things Journal IEEE (Volume: 1, Issue: 3), June 2014.

[4] Angelo P. Castellani, Mattia Gheda, Nicola Bui, Michele Rossi, Michele Zorzi, "Web Services for the Internet of Things through CoAP and EXI", IEEE International Conference on Communications Workshops (ICC), 5-9 June 2011.

[5] Thamer A. Alghamdi, Aboubaker Lasebae, Mahdi Aiash, "Security Analysis of the Constrained Application Protocol in the Internet of Things", Second International Conference on Future Generation Communication Technology (FGCT), 12-14 Nov. 2013.

[6] Mr. Bhavin M. Mehta1, Mr. Nishay Madhani2, Mrs. Radhika Patwardhan3, "Firebase: A Platform for your Web and Mobile Applications", December 2017.

[7] Shinho Lee, Hyeonwoo Kim, Dong-kweon Hong, Hongtaek Ju, "Correlation Analysis of MQTT Loss and Delay According to QoS Level", International Conference on Information Networking (ICOIN), 28-30 Jan.

[8] Shahid Raza, Hossein Shafagh, Kasun Hewage, Ren Hummen, Thiemo Voigt, "Lithe: Lightweight Secure CoAP for the Internet of Things", Sensors Journal, IEEE 13(10), Oct. 2013.

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

[10] Victoria Pimentel, Bradford G. Nickerson, "Communicating and Displaying Real-Time Data with WebSocket", Internet Computing IEEE 16(4), July-Aug. 2012.

[11] Sven Bendel, Thomas pringer, Daniel Schuster, Alexander Schill, Ralf Ackermann, Michael Ameling, "A Service Infrastructure for the Internet of Things based on XMPP", IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), 18-22 March 2013.

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

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

[14] Joel L. Fernandes, Ivo C. Lopes, Joel J. P. C.Rodrigues, Sana Ullah, "Performance Evaluation of RESTful Web Services and AMQP Protocol", Fifth International Conference on Ubiquitous and Future Networks (ICUFN), 2-5 July 2013.

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

[16] https://www.quora.com/Which-is-better-for-job-opportunities-AWS-or-Azure

[17] http://en.wikipedia.org/wiki/Advanced Message Queuing Protocol, cited 28 Jul 2014.

[18] https://blogs.endjin.com/2016/08/aws-vs-azure-vs-google-cloud-platform-mobile-service

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

[20] https://www.edureka.co/blog/aws-vs-azure/

