Ponte Message Broker Bridge Configuration
Using MQTT and CoAP Protocol
for Interoperability of IoT

Madhavi Dave¹, Mohanish Patel¹, Jyotika Doshi², and Harshal Arolkar²
¹ Indus University, Ahmedabad, India
madvhi.mca@gmail.com
² GLS University, Ahmedabad, India

Abstract. Internet of Things is emerging technology from few years where autonomous devices and technology communicate without human intervention. For IoT, many smart objects are interconnected and communicate through infrastructure of internet that is the base of a global network. The major challenge of IoT automation is to inter-connect heterogeneous devices, sensors and real-time application. Research has demonstrated recent development for a varied scope of solutions and devices of IoT eco-system. However, each interoperable technology provides its proprietary infrastructure and mechanism to communicate, which leads to IoT interoperability issue. Many research organizations, IoT industry and standardization institutes are working for empowering interoperability of resources between autonomous IoT devices. These efforts are categorized according to its implementation and usage considerations. The interoperable communication mechanism also depends on communication protocols and its communication models used at application layer. Major devices use MQTT, CoAP, RestAPI, AMQP and HTTP protocols at application layer for communication. This paper is intended to describe interoperability types and its impact on seamless communication. As a solution of interoperability issue, message bridge can be implemented between application layer protocols of request-response and publish-subscribe communication models. This paper demonstrate message bridge approach based on Ponte, which is open source IoT project designed by Eclipse for intercommunication between application layer protocols. The paper also demonstrates working of Ponte with the resolution of open issue related to message retention for cross protocol communication.

Keywords: Internet of Things · IoT protocol · Application layer protocol · Interoperability · MQTT · CoAP · HTTP · Message bridge · Ponte

1 Introduction

Internet of Things is a technology with connected devices to the internet having advantage of receiving and sending information over the internet and perform task depending on the information. Progressing towards recent technologies, it has become possible
to interconnect any device to other device using IoT [1]. This has become possible due to wireless networks and mobile devices with the help of which, we can control ample of devices from remote locations. Internet is the driving element for IoT enabled devices to have direct communication between machine-to-machine, thus coalescing digital and physical world together. Connecting each device is still a challenge to IoT as every IoT device has certain protocols and technologies upon which it is designed. It is difficult to cope with the growing requirements of humans as well as modern devices. Protocols used in majority of IoT devices which fall under Machine-to-Machine (M2M) category. With the help of these predefined protocols, IoT devices can establish seamless connection. To decide what kind of action a device should take, methods based on respective protocols are used. Majority of the event processing are based on Publish-Subscribe and Request-Response method [2], these methods are central part of MQTT and CoAP respectively. MQTT and CoAP are playing a major role in connecting devices to the internet. For achieving vision of connecting every device to the internet it must use HTTP for communication. WWW is fully based on HTTP protocol that opened a path to the internet for millions of people. The Internet of things has come so far by developing in so many areas and it has so many applications that its diversity can be an obstacle in its growth [16]. Each day many devices require seamless connection to the Internet, on the parallel side these IoT things are manufactured by different thousand manufacturers. These manufacturers are coming up with their own protocols, thus making it difficult to make devices communicate with each other. The need for Interoperability emerges due to heterogeneous devices and its manufacturer specific features.

2 IoT Interoperability

Interoperability can be defined as enabling devices and systems to communicate with each other regardless of the technical dependencies of their manufacturers. The Oxford Dictionary defines the term Interoperability as “able to operate in conjunction”. The definition of IoT says that it is global platform for communication between self-configurable devices which have identities, use smart interfaces and connected to network without interruption. The devices used in IoT systems are constrained by limited memory and processing capacity. The domains using such devices and communication among them are medical, transport, business, agriculture, health care, infrastructure, etc. There are more than 300 IoT platforms for intercommunication in market and many more are coming by each passing day [17]. The well-known IoT platforms providers are AWSIoT (Amazon), HomeKit (Apple), Watson (IBM), Brillo (Google), AzureIoT (Microsoft) and Jasper (Cisco) [22].

According to McKinsey analysis, missing interoperability is threat to IoT market. It is stated by author that, 40% of potential benefits can be get by interoperability among IoT devices [3]. The case in which interoperability is not provided for communication the devices must bound to single system provided by manufacturers. It restricts the system for using the defined communication mechanism by the providers [4]. Thus it becomes difficult for users or small scale providers to take services of heterogeneous devices and integrate them to make an information network. Such issues make interoperability a biggest challenge of IoT along with standardization, scalability and security [18]. The
standardization is needed for seamless communication technology among heterogeneous IoT things. Interoperability is raising day by day in IoT market due to heterogeneity of devices [21]. As it is described in Fig. 1, IoT interoperability is having four types as listed below [6],

![Fig. 1. Interoperability in Internet of Things [6]](image)

**Device Level Interoperability**: It deals with hardware and software features of heterogeneous device which have diversity based of proprietors and thus have interconnection issues.

**Network Level Interoperability**: The network on which devices are performing communication are distinct in nature. There may be IoT device, mobile device or any desktop computer as “Thing” and it may use different technologies to connect to the network.

**Platform Level Interoperability**: It arise due to variety of platforms available for providing service and application for communication.

**Protocol Level Interoperability**: As there are heterogeneous devices and each may have different proprietor. The communication protocol and technology used may varies from device to device.

**Data Level Interoperability**: The data gathered by sensors are going to be stored either at device or at cloud for the system. The data filtration functions handle and store data in their unique formats.

**Syntactical Interoperability**: The message format used by communication devices to interconnect may uses different formats according to technical design of the device.

**Semantic Interoperability**: In case of M2M (Machine-to-Machine) communication, both intercommunicating devices must follow same units. This system will not work efficiently when communication will be obstructed by varied semantics of device.

### 3 IoT Interoperability Challenges

The interoperability in IoT communication can be achieved when the two devices are compatible with each other [7]. The world of Internet of Things is facing high degree of
heterogeneity because of hardware, protocols and technologies used by devices. Even for one IoT system, there could be many interleaved communication technologies. Also, for one communication protocol, there could be number of communication application and design strategies. To handle such diversity in many aspects, communication within constrained devices is a big challenge. With increasing demand in future, the integration of wide variety of constrained devices will become necessary to improve [13].

The security and privacy of information and network must incorporate with basic principles like authentication, authorization, confidentiality, integrity and availability [8]. IoT covers many domains of real life and applicable to global economy. Thus, IoT security and privacy problems need most attention to be resolved. The developers and users of IoT devices and frameworks have committed for secure solution and guarantee for application used with privacy concerns. Likewise, an aggregate solution for security can also find optimal solution for scalability problem [4].

In IoT culture, many of the IoT device manufacturer are coming up with devices and services. An environment of proprietary in IoT technologies constraint growth value for users and industry. Though complete interoperability for all products and services is not possible, users may be diffident to buy IoT products and services in case of integration issues, ownership problems, and concern over proprietary for IoT device, service or platform [19].

4 Interoperability Solution

IoT has varied range of communication protocols such as MQTT, CoAP, AMQP, and HTTP at application layer [2]. Among the mentioned protocols, MQTT and AMQP use Publish-Subscribe communication model while CoAP and HTTP are based on Client-Server architecture. The intercommunication among different communication creates issues as the message format is following different patterns for fundamentally different protocols [4]. For interoperable functionality, the communication process described in Fig. 2 is followed. Thus solution for interoperability comprised of an intermediator platform who helps in converting messages from one to other format. There are many brokers, gateways and interoperable devices available for resolving the issue but most of them are having proprietary constraints [20].

![Image](image_url)
The interoperability can be achieved by various technologies of networking like virtual networking, fog computing and IP based networking solutions. It can also be implemented by open API, semantic web technology and service oriented architecture (SOA). These technologies are based on basic interoperability approaches like frameworks, platforms and open projects [21]. Each of the interoperable solution can be implemented in suitable IoT system with compatible set of protocols. The interoperability approaches are mentioned in detail as follows [22].

4.1 IoT Framework

The framework of ICT (Information and Communication Technology) system ensures to integrate necessary entities for dependency and reliability. IoT framework is expected to provide support to constrained devices having M2M inter-connectivity. IoT framework refers to the technology which is reliable and provide appropriate infrastructure for timely communication between constrained machines with proper analysis of data. Many enterprises have offered framework solutions in recent years which helps in connecting public or private network to the cloud. The key issue in success of framework is its robustness for implementing connection among devices of various manufacturers. The known IoT open source frameworks are oneM2M, HyperCat, OMA LWM2M and OpenHAB.

4.2 IoT Platform

IoT platform provides hardware and software support to build on the system. Platform follows some basic rules of implemented framework and supports the top up application features for allocated task. There are numerous IoT platforms are available in market and it’s almost impossible to find out any best solutions among all available platforms. These platforms are proprietary and thus hold its communication model and data privately. User of the system need to have detailed knowledge for intercommunication among different platforms. In many cases, cross platform APIs are used for data integration of different platforms. It becomes difficult when the platform is not open source and does not allow any API to access its data without proprietary privileges. The interoperability can be achieved when data are communicated or integrated using cross-platform or cross-domain application. The examples of popular IoT platforms are Kaa and EclipseIoT, rest platforms are having proprietary rights and not available as open source platform.

4.3 IoT Project

IoT projects are the bridge, gateway or adaptor provided for communicating data, signals or specifications. The project works as the middle-ware between two or more different platforms and can be extended with the help of plugins. The major technical aspect of IoT project depends on underlying protocol used for specific communication model. Thus, the bridge or gateway can be designed for different communication protocols at sender device and receiver device. This middle-ware can be a dedicated hardware or embedded with software, which must be installed in any resource rich device. It
has limited scalability based on underlying protocols used for interoperability. As the number of protocol increases in bridge or gateway, so is the complexity. For making it more usable, the standardization of protocols and its seamless integration is required. The few well-known IoT projects with open source are Ponte, OpenIoT and FiWare. Among the available open source projects for interoperability, this paper focuses on Ponte for configuration of broker bridge for widely used application layer protocols MQTT and CoAP.

5 Message Broker Bridge: Ponte

Ponte is Mobile-to-Mobile bridge framework designed under Eclipse Technology Project [10]. It can be used as bridge between devices for different communication protocol to interoperate. Ponte has reduced complexity of interoperability by bridging M2M protocols like HTTP, MQTT and CoAP together. Real power of Ponte is used, when connected devices are based on constrained protocol though it connects them to the rest of the world. Ponte has been integrated with wide number of Data Storage Engines like MongoDB, LevelDB and Redis which becomes handy for developers to analyze and store data [2]. It is a full package of many publish/subscribe brokers like RabbitMQTT, Mosquito, ZeroMQ and also AMQP including Redis and MongoDB [4]. Ponte is a build over on QEST and Mosca, yet another contribution by creator of Ponte and was rewritten in Javascript [7]. Most of IoT communication occurs in publish/subscribe communication model, thus Ponte is implemented using pub/sub model as described in Fig. 3 [10].

- RabbitMQ with implementations of the AMQP protocol
- Redis - the key/value store by antirez
- Mosquito with implementations of the MQTT protocol
- MongoDB - documental NoSQL web apps are built.

Fig. 3. Ponte architecture [10]
5.1 Case Study: Home Automation

For the explanation of the system, the home automation system is taken as example. In this case study, the communication is explained in both the ways, i.e. CoAP to MQTT and MQTT to CoAP by considering a scenario as described below.

**Accessing Light from Remote Location:** In the case of accessing lights from remote location, the services of web are used. The data can be sent over internet/web with the help of HTTP or REST resources. The web request can be made by application, which sends signal for accessing resource at home. The application can be considered as CoAP client, which is sending request from remote location for switching the light on at home. The request would reach CoAP server via medium of internet/web, which resides locally. The received datagram is in CoAP format and it cannot be understood by local devices (light). Here, Ponte receives the datagram sent by CoAP server and convert in MQTT packet format so the local devices can act upon it. The MQTT packet would be sent to MQTT broker, which is responsible for publishing the messages. The MQTT broker creates queue and distribute the message topic wise. The subscribed node (light) will be forwarded the message. Thus the message received by final MQTT node, which is light, in this case, is switched on/off as per the given message by remote user.

**Check Lights Status from Remote Location:** For checking the light’s status from remote location, the event for light on/off can be driven from local source. In this scenario when the light will be on/off, the application of remote user can inform about the change in light’s status. Here when the local user send command for changing state of light, the MQTT broker receives the command. The broker publishes the command to the connected devices which have subscribed on the topic “light”. Thus, the devices which have light may change its state. Along with giving command to subscribed device, the broker also sends message to Ponte bridge. The message would be received in MQTT format and convert into URI format for CoAP server. The CoAP server forwards message to remote user in web response format by using web connection. The remote user can check the status of light change (on/off) using the mobile application.

5.2 Service Model Implementation

For implementing service model, home automation system has taken into consideration. As this domain is highly in progress and interests’ innovation. It also needs to fulfill the commercial requirement for IoT system, which is mobility, security and reliability [9]. For demonstration of service model, the command can be sent and received by MQTT, CoAP or HTTP using the message broker bridge of Ponte. Ponte is used as a solution for the problem of integration among various devices, which has implementation of standard protocols like MQTT, CoAP and HTTP. For supporting more protocols, the specific protocol adapters can be used. The Ponte bridge can store different message semantics in non-relational database format. In MQTT, user can propose QoS level as per the requirement of the system unlike CoAP and HTTP. For the same implementation in Ponte, the message persistence can be achieved with clustering and distributed database. Thus, the
user subscribes and receive high volume of data with proper availability as per config-
ures QoS. The system using multiprotocol tends to use API for remote operations using
CoAP and HTTP. Ponte message broker bridge integrate such characteristics of various
protocols and make the communication possible with MQTT’s publish-subscribe model
and CoAP/HTTP’s request-response model. The service model for communication using
Ponte Bridge is shown in Fig. 4.

Ponte is an application based on node manager, thus it requires node.js to imple-
ment as a base. Node manager can be installed by NVM (Node Version Manager) as it
allows user to install recommended version from available multiple versions of node.js
and NPM. Installing Node will automatically install NPM on the computer using any
operating system. Ponte works on node version 0.12, 4.3.1, 5.0 to any latest node version
till 12.4.0. It is stated in [10] that Ponte is not working with node version 5.7 but it is
implemented to communicate with all the three protocols. For installing Ponte, the two
options are available as follow,

$ npm install ponte -g or
$ npm install ponte bunyan -g

First command will install Ponte with default configuration settings, while second
command demonstrates that bunyan is a functionality which outputs log messages and
provides an easy logger interface for reading ongoing process in Ponte. To start Ponte
in terminal,

$ ponte -v or $ ponte -v | bunyan
After Ponte will be started and terminal will show following logger,

```
[2019-11-25T07:28:24.485Z] INFO:ponte/30351 on mak: server started red→(service=MQTT, mqtt=1883)
[2019-11-25T07:28:24.491Z] INFO:ponte/30351 on mak: server started red→(service=MQTT, mqtt=3001)
[2019-11-25T07:28:24.492Z] INFO:ponte/30351 on mak: server started red→(service=MQTT, mqtt=5681)
```

Default port for MQTT will remain same every time as Ponte is a direct implementation of MQTT. Port number for CoAP and HTTP can be changed as shown in the logger information, default port number for CoAP and HTTP defined in Ponte is 3000 and 5683 by default. It can be done by changing port number in server.js file found at /usr/lib/node modules/ponte/lib location.

### 5.3 MQTT to HTTP

In order to publish from MQTT client to HTTP client it is necessary that messages published from MQTT protocol to be in retained mode, thus `-r` should be added at last of mosquitto pub command like shown below,

```
$ mosquitto_pub -d -t LED -m ON –r
```

To receive in HTTP GET method is to be used as follow,

```
$ curl http:/localhost:3001/resources/LED
```

### 5.4 CoAP to MQTT

Same as HTTP in order to receive message in CoAP retain flag should be set true by adding `-r` at last while publishing message from MQTT client (Fig. 5). To receive in CoAP observe switch should be turned on.

```
$ mosquitto_pub -d -t LED -m ON –r
$ coap -o coap:/localhost/r/LED
```

### 5.5 CoAP to HTTP

Ponte can use PUT method to send messages from CoAP to MQTT or HTTP. CoAP client named, CoAP-CLI can be used with PUT method for the communication.

```
$ echo -n OFF \ coap put coap:/localhost/r/LED
```
To receive messages in MQTT client mosquitto sub can be used.

$ mosquitto_sub -d -t LED

6 Research Findings

The Ponte message bridge is lacking in the areas of IP addressing, message retention, Host Identity protocol and such other 235 issues [14]. Due to open issues, the use of Ponte has shrunk though it has least overheads for communication. The issue of IP addressing and message retention is solved as demonstrated in service model section. Ponte is also having less delay and failure compare to other interoperability solution. The payload size for Ponte depends on the sender and receiver technology, i.e. the supported packet size of CoAP and MQTT.

6.1 Open Issue Resolution: Message Retention

In Open project of Ponte, user of Ponte bridge has pointed that the message send by any protocol is not preserved for view [11]. The issue can be resolved by making changes in configuration file through set of commands. A retain flag is set to true so broker can store last received message on a particular topic. As a broker it will only store one message per topic that is last received. Broker will publish last receive message as soon as client subscribes to that particular topic. When retained flag is set to false or -r is not added to the end of the command then protocols which are working on GET/PUT method such as CoAP and HTTP will not receive messages published from MQTT client unless until
first message received is from CoAP and HTTP i.e. if retained flag is set to false CoAP and HTTP clients have to send their message first in order to receive messages from MQTT.

7 Conclusion

The interoperability of IoT devices is been the major issue since inception of technology and with the growing market of IoT devices, the issue needs more attention. Although there are many researches going on for IoT interoperability, no solution is able to provide justified result for the issue. The categories of interoperability are described with available open source solutions. All categories have different approaches and techniques based on implementation domain. It is concluded that among existing solutions, IoT frameworks are having high overload for any M2M constrained device. In addition, the IoT platforms described are having proprietary rights reserved by the organization, which have designed it. Ponte is most suitable among described IoT projects, which is open source Eclipse project for IoT interoperability. It works with MQTT, CoAP and HTTP (application layer) protocols for intercommunication. The implementation and working of Eclipse Ponte is described which is based on Node Version Manager. Ponte is an open source solution with few open issues for development. An open issue of message retention is resolved by adding elements in configuration file, which can help in achieving interoperability among application layer communication protocols. For future work, the issue of security and Host Identifier Protocol is required to resolve for better interoperability among IoT devices.

References

1. Ashton, K.: Internet of Things. RFiD J. 22, 97–114 (2009)
2. Salman, T., Jain, R.: Networking protocols and standards for Internet of Things. CSE (November 2015)
3. Manyika, J., et al.: The Internet of Things: Mapping the Value Beyond the Hype. McKinsey Global Institute, New York (2015)
4. Macaulay, T.: RIoT Control: Understanding and Managing Risks and the Internet of Things. Morgan Kaufmann, Burlington (2016)
5. Gubbi, J., Buyya, R., Marusic, S., Palaniswami, M.: Internet of Things (IoT): a vision, architectural elements, and future directions. Future Gener. Comput. Syst. 29, 1645–1660 (2013)
6. Gaedke, M.: Interoperability in Internet of Things: taxonomies and open challenges. Mob. Netw. Appl. 24(3), 796–809 (2019)
7. Ding, L., Shi, P., Liu, B.: The clustering of internet, Internet of Things and social network. In: 3rd International Symposium on Knowledge Acquisition and Modeling (KAM10), pp. 417–420, Wuhan, China (November 2010)
8. Suo, H., Wan, J., Zou, C., Liu, J.: Security in the Internet of Things: a review. In: International Conference on Computer Science and Electronics Engineering, ICCSEE, pp. 648–651 (2012)
9. Zamfir, S., Balan, T., Sandu, F., Costache, C.: Mobile communication solutions for the services in the Internet of Things. In: Borangiu, T., Drăgoicea, M., Nővoa, H. (eds.) IESS 2016. LNBIP, vol. 247, pp. 619–632. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-32689-4_47
10. Ponte - Connecting Things to Developers. https://www.eclipse.org/ponte/
11. Eclipse Technology Project. https://projects.eclipse.org/projects/technology/ponte/
12. Ponte - M2M Bridge Framework for REST Developers. https://www.eclipse.org/proposals/technology.ponte
13. Konduru, V.R., Bharamagoudra, M.R.: Challenges and solutions of interoperability on IoT. In: International Conference on Smart Technologies for Smart Nation. IEEE (2017). ISBN: 978-1-5386-0569-1
14. Open Issues of Ponte Message Broker. https://bugs.eclipse.org/bugs/show_bug.cgi?id=536595
15. Stallings, W.: Fundamentals of Modern Networking: SDN, NFV, QoE, IoT and Cloud. Pearson Publication, London (2016)
16. Suresh, P., Daniel, J., Aswathy, R., Parthasarathy, V.: A state of the art review on the Internet of Things (IoT) history, technology and fields of deployment. IEEE (March 2014). ISBN: 978-1-4799-7613-3/14
17. IoT-From Research and Innovation to Market Deployment. RiverPublisher (2014). ISBN: 978-87-93102-4-1
18. Vermesan, O., Friess, P.: Internet of Things-Converging Technologies for Smart Environment and Integrated Ecosystem. IEEE, Piscataway (2016)
19. Al-Fuqaha, A., Aledhari, K., Mohsen, G., Ammar, R., Mehdi, M.: Toward better horizontal integration among IoT services. IEEE Commun. Mag. 53(9), 7279 (2015)
20. Bellavista, P., Zanni, A.: 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) (September 2016). ISBN: 978-1-5090-1131-5/16
21. Noura, M., Atiquzzaman, M., Gaedke, M.: Interoperability in Internet of Things: taxonomies and open challenges. Mob. Netw. Appl. 24, 796–809 (2018)
22. Paolo, B., Zanni, A.: Scalability of kura-extended gateways via MQTT-CoAP integration and hierarchical optimizations. Internet of Things. IEEE (October 2016)