Open Source Internet of Things Platforms

Senthil Kumaran. S, Balakannan S. P.

Abstract: ***Internet of Things (IoT) is rapidly growing and stresses every field of computer science. With the proliferation of heterogeneous devices and things that are connected and on boarded on the internet [1], standards play an important role in order to keep the IoT eco-system growing and inter operate with each other. Formulating standards and keeping them open is important. It is not enough to keep the standards open but also the software that enables these standards should be open, in order to cater to the demanding needs of IoT. Many IoT vendors, organizations and researchers are advocating and encouraging the use of Free Open Source Software [2] in IoT deployments at all levels. When many devices and things are connected there should be an easy way to establish the environment and monitor the entire IoT system at a central location for ease of management, which are the expectations of the IoT platforms that will be discussed in this paper. This paper surveys the different strictly Open Source Software [2] platforms that are available for IoT and proposes some common features that should be part of any IoT platform.***

Keywords: internet of things, iot, survey, platforms, open source, foss

I. INTRODUCTION

Internet of Things (IoT) connects heterogeneous devices or things via internet. These connected devices communicates with each other and also generates information from the sensors attached to them and sends it back to edge network equipment [3] and a larger cloud. Depending on the nature of decision that has to be taken, it could be decided in the following places:

1) Locally on the device with the available resources in the device
2) On an edge computing equipment with larger resource availability and data available from a locality of neighboring devices
3) On a cloud environment where larger analysis could be done based on huge data that is available from the past and complex algorithms could be run to decide further action with the abundance of different computing resources on the cloud.

In a situation where a large IoT eco-system has to be established with huge number of devices connected over the internet and data has to flow in and out for different types of analysis, a central system or dashboard comes handy to manage the entire IoT deployment, which is the purpose of these IoT platforms that we will see in this paper.

Rest of this paper is organized as follows: Section II describes the basic elements that should be part of these IoT platforms. Section III introduces the various Open Source Software based IoT platforms that are available as of this writing with some user base (based on whether it is been talked up on the internet or not) and describes their features and characteristics briefly. Section IV identifies some must have features in IoT platforms that will help to get most out of these platforms and provide long term association for the users of these platforms.

II. DESIRABLE FEATURES OF IOT PLATFORMS

Though there are many IoT platforms available, this paper mainly focuses on Open Source Software based IoT platforms. The IoT platforms considered in this paper are not just offered in the cloud or as Software as a Service (SaaS) [4], which does not satisfy the Open Source criteria. These IoT platforms should be based on any available Open Source license [5] and the source code of these platforms should be accessible and modifiable. It should be possible to deploy these platforms in local test beds or internal cloud environments. This will help in order to evaluate these platforms, before making a decision to adopt it at a large scale either on private deployments or on public clouds.

III. EXISTING OPEN SOURCE IOT PLATFORMS

A. DeviceHive

DeviceHive [6] is a flexible and highly scalable IoT platform that is released under Apache 2.0 Open Source license [7]. It has multiple deployment options using docker [8] which enables it to be deployed in public, private, hybrid or community cloud environments [9] both for large and small organizations. DeviceHive supports connecting devices with the help of REST API [10], WebSockets [11] or MQTT [12]. It offers libraries in various programming languages such as Java, Node.js and Python for device integration. In order to visualize data DeviceHive can fetch real time data from any data source and allows to create different dashboards based on the users preferences. It supports various big data solutions such as ElasticSearch, Apache Spark, Cassandra and Kafka for real-time data processing. Apache Spark Streaming support enables batch analytics and machine learning based on the received data.

Revised Manuscript Received on December 16, 2019.
***** Correspondence Author
Senthil Kumaran. S*, Department of Information Technology, Kalasalingam Academy of Research and Education, Krishnankoil. TamilNadu, India.. Email: sk@stylesen.org
Balakannan. S.P, Department of Information Technology, Kalasalingam Academy of Research and Education, Krishnankoil. TamilNadu, India.. Email: balakannansp@gmail.com

Table- I: Comparison of Open Source IoT Platforms

IoT PLATFORM	DEVICE MANAGEMENT	DEVICE INTEGRATION	PROTOCOLS	ANALYTICS	VISUALIZATION	LICENSE
DeviceHive	Yes	REST, MQTT	REST, WebSockets, MQTT	Apache Spark (Real-Time)	Yes	Apache 2.0
DSA	No	REST	HTTP	Splunk	No	Apache 2.0
Kaa	Yes	SDK, REST	MQTT, CoAP, XMPP, TCP, HTTP	Real Time using Kaa, Apache Cassandra and Apache Zappelin	Yes	Apache 2.0
MainFlux	Yes	REST	SDK	HTTP REST, MQTT, WebSockets, CoAP	Yes	Apache 2.0
SiteWhere	Yes	REST, Mule, AnyPoint	MQTT, AMQP, Stomp, WebSockets and direct socket connections	Apache Spark (Real-Time)	No	CPAL 1.0
ThingsBoard	Yes	REST	MQTT, CoAP and HTTP	Yes	Yes	Apache 2.0
Zetta	Yes	REST, WebSockets, HTTP	MQTT, CoAP, XMPP, TCP, HTTP, etc.	Yes	Yes	MIT
Node-RED	Yes	REST	MQTT, CoAP, XMPP, TCP, HTTP, etc.	No	Yes	Apache 2.0
Thinger.io	Yes	REST Clients	HTTP, etc.	Yes	Yes	MIT

B. Distributed Services Architecture (DSA)

DSA [13] is a distributed architecture that enables IoT. It comprises of three components namely, Distributed Service Broker (DSBroker), Distributed Service Link (DSLink) and nodeAPI all of which are released under Apache 2.0 Open Source license. There could be more than one implementation of each of the DSA component adhering to the DSA, where specific implementations choose their own license, but at least one implementation exists with an Apache 2.0 license. The DSA unifies devices, services and applications into a real time data model that is structured and adaptable. The complete architecture is de-centralized and individual vendors choose to implement the DSLinks in order to onboard support for their products in the DSA. Each DSA module is light weight and hence could be deployed to constrained devices [14]. The real time data that flows into a DSA system from the connected devices and services, could be used to create and visualize various kinds of dashboards and analytics.

C. Kaa

The Kaa project [15] provides an IoT platform that is released under Apache 2.0 Open Source license [7]. It is an efficient platform that caters to different use cases and could be deployed in any kind of cloud environment. With the Kaa platform any device could be connected and real time data can be collected. The collected data could also be analyzed. Devices of any type ranging from class 0, class 1, class 2 [16] to even larger devices can be integrated with the provided SDK that gives Java, C++ and C libraries. Both structured and unstructured data [17] could be analyzed with Kaa. It offers connectivity via both UDP [18] and TCP [19], along with other IoT specific connectivity technologies. A dashboard for real time device monitoring is available in Kaa. Remote devices could be provisioned and configured, along with providing Over the Air (OTA) updates [20] whenever required. It also supports various data analytics methods in order to analyze user behavior and provide targeted notifications to end users.

D. MainFlux

MainFlux [21] is released under Apache 2.0 Open Source License [7] that could be deployed both on premises and on the cloud. MainFlux is based on a microservices based architecture [22] that is highly responsive and scalable. It provides APIs exposed via HTTP RESTful, MQTT, Web-Socket and CoAP [23]. Device integrations are available via SDKs that supports programming languages such as C, C++, JavaScript, Go and Python. Device provisioning, management and Over the Air (OTA) updates are available in MainFlux. Deployment of MainFlux is available via Docker images which makes it easy and scalable too. It also provides encrypted communication of data between different endpoints using TLS and DTLS. The dashboard provided is highly intuitive and offers much functionality for event management and analytics.

E. SiteWhere

SiteWhere [24] is released under CPAL 1.0 Open Source License [25]. SiteWhere is an IoT application enablement platform that helps in acquiring, storing and analyzing data obtained from devices at a massive scale. The platform is based on a microservices architecture [22] which provides reliable and scalable service. It integrates technologies such as Apache Kafka and docker for highly scalable requirements of IoT deployments. Device management is run as a separate microservice and offers various functionalities. The major components that comprise SiteWhere are as follows:

- Instance Management
- Configuration Management with Apache ZooKeeper
- Event Processing with Apache Kafka
- Inter-Microservice Communication with GRPC
- Global Microservices
- Multitenant Microservices

F. ThingsBoard.io

ThingsBoard [26] is an IoT platform that offers data collection, processing, visualization and device management. ThingsBoard is released under Apache 2.0 Open Source license [7]. Various IoT protocols such as MQTT, CoAP and HTTP are supported. ThingsBoard could be deployed both on premise and on the cloud. It provides a very good dashboard through which devices and other assets could be provisioned and managed

with ease. The collected data could be visualized using custom widgets and dashboards could be built based on that, which can then be shared to interested parties. The dashboards created can deliver real time data visualization. It can also trigger events based on the data analyzed and send notifications. There is a rule engine that helps in writing flexible rules to operate with the incoming data.

G. Zetta

Zetta [27] is built using Node.js [28] based on a API intensive architecture. The usage of Node.js to build Zetta makes it cross platform. Zetta is released under MIT Open Source license [29]. Zetta combines all the popular API based methodologies such as REST APIs, WebSockets and reactive programming [30] in order to create the IoT platform. It best suits IoT applications that are data intensive and requires real time responses. It supports all device protocols which are mediated to HTTP in order to communicate with the device and the server. It offers a robust query mechanism with which we can query different aspects of the platforms and the devices connected to it. The query mechanism also enables an easy way to notify based on an event that occurs for a matching query, for example, adding and removing of devices can be notified if a matching query exists for the same. For all the devices that gets connected to Zetta, there is a consistent interface provided. Zetta architecture has the following components:

- Zetta Server
- Scouts
- Drivers
- Server extensions
- Registry

H. Node-RED

Node-RED [31] is an event driven flow based IoT platform built on Node.js, thus making it cross platform. Node-RED is released under Apache 2.0 Open Source license. The RED in Node-RED stands for Rapid Event Developer. It is web browser based platform, where flows are created for different events on the web browser. It provides a WYSIWYG editor [32] kind of an environment for the IoT event creation in the web browser. The resulting flow is stored or generated as JSON [33] files. These JSON files can then be shared for collaborating with fellow developers on an IoT project. Node-RED follows an highly event-driven non-blocking architecture for the IoT application. The flows created in the web browser are then deployed on the devices and other runtimes, with a click of a button making it much user friendly.

I. Thinger.io

Thinger.io [34] is a client-server based IoT platform, where the server predominantly lives in the cloud. All the client libraries of Thinger.io are released under the MIT Open Source license [29]. Local deployments of Thinger.io server are possible with the help of the docker images that are provided. The client libraries are provided for different platforms which are listed below:

1) SigFox
2) Linux
3) Arduino
4) Android
5) ARM Mbed

Thinger.io provides a seamless integration of devices that uses the above platforms with minimal effects.

IV. GENERIC FEATURES

An ideal IoT platform should offer end to end solution in order to establish an IoT eco-system. There are some common features that are essential for any IoT platform, which is discussed in this section. Sensors are the lowest level components of an IoT eco-system that are connected to some device or are capable of sending data directly to concerned systems. The IoT platform should offer the following features for device interaction:

- Provisioning
- Configuration
- Monitoring
- Management
- Updation

The devices available in an IoT deployment sends huge amount of data accumulated via the sensors attached to it to an IoT platform. Handling these data easily is a prime feature required in an IoT platform. Following are some of the data related features that should be an integral part of the IoT platform:

- Acquisition / Collection mechanisms
- Storage
- Analytics
- Visualization
- Dashboard - both readymade and customizable
- Provision to send data to different systems for analyzing
- Notifications or Events based on data analysis

Just like any other deployment, an IoT platform should provide some key aspects in order to be considered for production use which are listed below:

- Scalability
- Reliability
- Robustness
- Customization
- Low Operational Expense
- Protocol Support
- Hardware Support
- Security

An easy to deploy mechanism is a win for any IoT platform at different deployment medium such as on-premises or on the cloud. It shouldn't complicate the deployment scenario and make the users spend more time on debugging the platform than to concentrate on the IoT solution they are working with. Providing a simple and intuitive dashboard to manage and monitor the vitals of such a complex IoT platform is important for larger adoption of the platform itself.

V. CONCLUSIONS

There are many Open Source Software based IoT platforms available as of today, out of which we have chosen the ones that offers a complete solution to manage, monitor and analyze devices and the data generated from these devices. Table I, summarizes the Open

Source IoT platforms and their features discussed on this survey paper. When IoT platforms are offered as Open Source Software, it becomes easy to evaluate these platforms locally and there is an option to enhance or modify to the needs at hand, before starting to invest on huge deployments. Though this is not an exhaustive list of all the available Open Source IoT platforms, it takes the most prominent ones with an active Open Source community around them. Thus, this paper helps in order to know some of the IoT platforms available and that are offered as Open Source projects, to help researchers and others who are actively looking for this specific requirement.

REFERENCES

1. Gartner report on connected things. https://www.gartner.com/en/newsroom/press-releases/2017-02-07-gartner-says-8-billion-connected-things-will-be-in-use-in-2017-up-31-percent-from-2016
2. Open Source Software. https://en.wikipedia.org/wiki/Opensourcesoftware
3. Edge computing. https://en.wikipedia.org/wiki/Edge_computing
4. Software as a Service https://en.wikipedia.org/wiki/Software_as_a_service
5. Open Source Software License. https://opensource.org/licenses
6. Device Hive. https://devicehive.com
7. Apache 2.0 License. https://www.apache.org/licenses/LICENSE-2.0
8. Docker. https://docs.docker.com/
9. Sumit Goyal,"Public vs Private vs Hybrid vs Community -Cloud Computing: A Critical Review", IJCNIS, vol.6, no.3, pp.20 - 29, 2014. DOI: 10.5815/ijcnis.2014.03.03
10. Fielding, Roy Thomas. Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation, University of California, Irvine, 2000.
11. I. Fette, A. Melnikov RFC6455, The WebSocket Protocol, 2011.
12. MQTT Version 5.0. Edited by Andrew Banks, Ed Briggs, Ken Borgendale, and Rahul Gupta. 07 March 2019. OASIS Standard. https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html
13. Distributed Services Architecture http://iot-dsa.org/
14. S. Bandyopadhyay and A. Bhattacharyya, "Lightweight Internet protocols for web enablement of sensors using constrained gateway devices," 2013 International Conference on Computing, Networking and Communications (ICNC), San Diego, CA, 2013, pp. 334-340. doi: 10.1109/ICCNC.2013.6504105
15. Kaa Project https://www.kaaproject.org/
16. C. Bormann, M. Ersue, A. Keranen RFC7228, Terminology for Constrained-Node Networks Abstract 2014.
17. Structured vs Unstructured data. https://www.datamation.com/big-data/structured-vs-unstructured-data.html
18. J. Postel RFC768, User Datagram Protocol, 1980.
19. Vinton Cerf, Yogen Dalal, Carl Sunshine RFC675, SPECIFICATION OF INTERNET TRANSMISSION CONTROL PROGRAM, 1974.
20. Halder, Subir & Ghosal, Amrita & Conti, Mauro, 2019. Secure Over-The-Air Software Updates in Connected Vehicles: A Survey.
21. MainFlux https://www.mainflux.com/
22. Microservices. https://en.wikipedia.org/wiki/Microservices
23. Z. Shelby, K. Hartke, C. Bormann IETF RFC7252, The Constrained Application Protocol (CoAP), 2014.
24. SiteWhere http://www.sitewhere.org/
25. Common Public Attribution License Version 1.0. https://opensource.org/licenses/CPAL-1.0
26. ThingsBoard https://thingsboard.io/
27. Zetta. https://www.zettajs.org/
28. Node.js https://nodejs.org/en/
29. MIT License. https://opensource.org/licenses/MIT
30. Reactive Programming. https://en.wikipedia.org/wiki/Reactive_programming
31. Node-RED. https://nodered.org
32. WYSIWYG. https://en.wikipedia.org/wiki/WYSIWYG
33. T. Bray, Ed. IETF RFC8259, The JavaScript Object Notation (JSON) Data Interchange Format, 2017.
34. Thinger.io https://thinger.io/

AUTHORS PROFILE

Senthil Kumaran S. received his M.S. degree from Birla Institute of Technology and Science, Pilani in the year 2010. He has received his bachelors degree from the Department of Computer Science and Engineering, Thiagarajar College of Engineering, Madurai in the year 2006. He is a committer and Project Management Committee (PMC) member at Apache Software Foundation, Debian Maintainer and Apache Subversion developer. Currently he is a research scholar in the Department of Information Technology, Kalasalingam University, Tamilnadu, India.

Balakannan S.P received his Ph.D. degree from the Department of Electronics and Information Engineering at Chonbuk National University, South Korea (2010). He has received his master degree (5 years integrated) from the Department of Computer Science and Engineering, Bharathiar University, India, in the year 2003. He has worked as a Project Assistant in Indian Institute of Technology (IIT), Kharagpur, India from 2003 to 2006. Currently, he is working as Assistant Professor in the Department of Information Technology, Kalasalingam University, Tamilnadu, India. His areas of interest include Wireless Network, Network Coding, Cloud & Green Computing, Cryptography, and Mobile Communication.