IJIREEICE

ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

International Journal of Innovative Research in
Electrical, Electronics, Instrumentation and Control Engineering

ISO 3297:2007 Certified

Vol. 4, Issue 11, November 2016

Open Source Middleware for Internet of Things

Jyothi T

Assistant Professor, Department of ISE, GSSSIETW, Mysuru, India

Abstract: Internet of Things (IoT) has been recognized as a part of future internet and ubiquitous computing. It creates a true ubiquitous or smart environment. It demands a complex distributed architecture with numerous diverse components, including the end devices and application and association with their context. A key technology in the realization of IoT systems is middleware, which is usually described as a software system designed to be the intermediary between IoT devices and applications. The middleware for IoT acts as a bond joining the heterogeneous domains of applications communicating over heterogeneous interfaces. This paper presents the current gap and future directions in this field by a comprehensive review of the existing middleware systems for IoT and the Open Source Middleware Tools for the Internet of Things.

Keywords: Internet of Things, Middleware Requirements, Open source middleware.

I. INTRODUCTION

The Internet of Things (IoT) provides the ability for human and computers to learn and interact from billions of things that include sensors, actuators, services, and other Internet connected objects. The realization of IoT systems will enable seamless integration of the cyber-world with our physical world and will fundamentally change and empower human interaction with the world. A key technology in the realization of IoT systems is middleware, which is usually described as a software system designed to be the intermediary between IoT devices and applications.

Development of middleware.[1] & [2] in the domain of IoT is an active area of research. There have been a lot of researches towards building up this middleware addressing interoperability across heterogeneous devices serving diverse domains of applications, adaptation, context awareness, device discovery and management, scalability, managing a large data volumes and, privacy, security aspects of the said IoT environment.

Therefore there is a strong need to understand how the existing IoT-middleware systems work and address the different requirements of ubiquity as well as IoT.

In this article focus has been given to study the existing IoT-middlewares, understanding its functional components and categorizing and comparing them as per the various features along with the open sources middlewares for IoT. The remainder of this article is organized as follows.

First, the related work in IoT-middleware is presented, followed by descriptions of the essential functional blocks and the system architecture of the IoT-middleware system. The feature wise classification of theirs along with the different interfaces and syntax and semantics strategies are described in detail followed by the available open source middlewares for IoT.

II. BACKGROUND

Research into the IoT is still in its early stage, and a standard definition of the IoT is not yet available. IoT can be viewed from three perspectives: Internet-oriented, things-oriented (sensors or smart things) and semantic-oriented (knowledge) [3].
The definition of “things” in the IoT vision is very wide and includes a variety of physical elements. These include personal objects we carry around such as smart phones, tablets and digital cameras. It also includes elements in our environments (e.g. home, vehicle or work), industries (e.g., machines, motor, robot) as well as things fitted with tags (e.g., RFID), which become connected via a gateway device (e.g., a smart phone). Based on this view of “things”, an enormous number of devices [7] will be connected to the Internet, each providing data and information, and some, even services.

A. MIDDLEWARE

Generally, a middleware abstracts the complexities of the system or hardware, allowing the application developer to focus all his effort on the task to be solved, without the distraction of orthogonal concerns at the system or hardware level. Such complexities may be related to communication concerns or to more general computation. A middleware provides a software layer between applications, the operating system and the network communications layers, which facilitates and coordinates some aspect of cooperative processing. From the computing perspective, a middleware provides a layer between application software and system software. In the IoT, there is likely to be considerable heterogeneity in both the communication technologies in use, and also the system level technologies, and a middleware should support both perspectives as necessary.

Fig 1 presents the relationships between the IoT’s middle-ware requirements and its infrastructural and application characteristics.

IJIREEICE

ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

International Journal of Innovative Research in
Electrical, Electronics, Instrumentation and Control Engineering

ISO 3297:2007 Certified

Vol. 4, Issue 11, November 2016

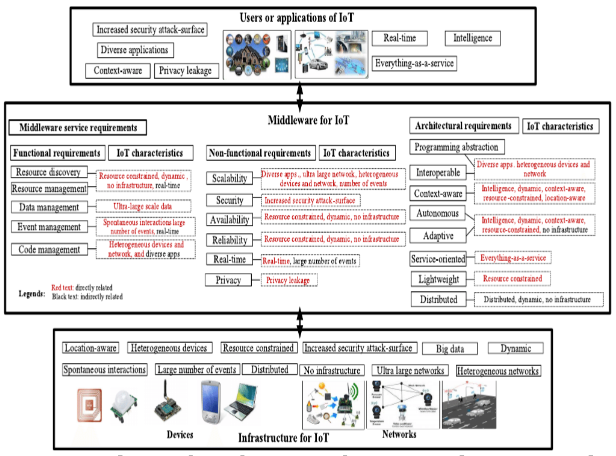

Fig 1: Relationships between the IoT applications and infrastructure and its middleware requirements (Research Gate)

As shown in this figure, most of the requirements are directly related to one or more characteristics of the IoT. A few of them are also indirectly linked (black text) to one or more characteristics of the IoT. For instance, the real-time behavior requirement is directly related to the application's real-time characteristics and indirectly to the large number of events. Also, a few of the middleware requirements (e.g., resource discovery and resource management) jointly capture the same set of IoT characteristics.

B. CLASSIFICATION OF IOT MIDDLEWARES

All the listed middlewares support device discovery and management [8]. Context aware functionality is supported by HYDRA, UBIWARE, UBIROAD and SMEPP. On the other hand, SOCRADES, SMEPP, GSN, UBIROAD and HYDRA are some examples of middleware implementing security and user privacy in their architecture. Based on platform portability, syntactic resolution, HYDRA, SMEPP and ASPIRE are OSGi compliant, UBIROAD usesJAVA and XML, UBISOAP uses J2SE and J2ME, GSN uses XML and SQL, SIRENA and SOCRADES use DPWS while SOCRADES also uses SAP NetWeaver [4] platform and ISMB uses any JAVA compliant platform.

Table 1. IoT-middleware comparison.

IoT Middleware	Features of Middleware				
	Device Management	Interoperation	Platform Portability	Context Awareness	Security and Privacy
HYDRA	✓	✓	✓	✓	✓
ISMB	✓	x	✓	x	x
ASPIRE	✓	x	✓	x	x
UBIWARE	✓	x	✓	✓	x
UBISOAP	✓	✓	✓	x	x
UBIROAD	✓	✓	✓	✓	✓
GSN	✓	x	✓	x	✓
SMEPP	✓	x	✓	✓	✓
SOCRADES	✓	✓	✓	x	✓
SIRENA	✓	✓	✓	x	✓
WHEREX	✓	✓	✓	x	x

Where X [5] is developed using J2EE architecture and is integrated with Oracle Application Server 10g.It also uses Rhino rule engine which is implementation of Java Script.

III. OPEN SOURCE MIDDLEWARES

A. AllJoyn

AllJoyn is a collaborative open source software framework that allows devices to communicate with other devices around them. AllJoyn framework is flexible, promotes proximal network and cloud connection is optional. A simple example would be a motion sensor letting a light bulb know no one is in the room it is lighting, so it can shut itself off. The system uses the Client–server model to organize itself. For example, a light could be a "producer" (server) and a switch a "consumer" (client).

Each "producer" on the network has an XML file called introspection that is used to advertise the device's abilities and what it can be asked to do. Microsoft has added a technology called Device System Bridge that allows devices using home or building protocols such as Z-Wave and BACnet to appear on an AllJoyn network. The system also has technology for audio streaming to multiple device sinks in a synchronized way.

AllJoyn provides bunch of services that can be integrated with its core.

- **Onboarding Service:** Provides a consistent way to bring (onboard) a new device onto Wi-Fi network.
- **Configuration Service:** Allows one to configure certain attributes of a device, such as its friendly name, default language, passcode etc.
- **Notification Service:** Allows text-based, audio and image (view URLs) notifications to be sent and received by other devices on the network.
- **Control Panel Service:** Allows devices to advertise a virtual control panel to be controlled remotely.
- **Home Appliances & Entertainment (HAE) Service:** Allows a common way of monitoring and managing HAE category devices, regardless of device manufacturers.
- **Lighting Service Framework (LSF)** provides an open and common way of communicating for AllJoyn-based connected lighting products, regardless of manufacturer.

B. OpenRemote

OpenRemote is software integration platform for residential and commercial building automation. OpenRemote platform is automation protocol agnostic, operates on off-the-shelf hardware and is freely available under an Open Source license. OpenRemote's architecture enables fully autonomous and user-independent intelligent buildings. End-user control interfaces are available for iOS and Android devices, and for devices with modern web browsers. User interface design, installation management and configuration can be handled remotely with OpenRemote cloud-based design tools.

Integrate:	AMX, KNX, Lutron, Z-Wave, 1-Wire, EnOcean, xPL, Insteon, X10, Infrared, Russound, GlobalCache, IRTrans, XBMC, VLC, panStamps, Denon AVR, FreeBox, MythTV, and more.
Design:	Customize control interfaces for each device, individualize user interfaces for each user
Control:	Control panels for Android and iOS devices, mobile web browsers, Desktop PCs
Automate:	Intelligent buildings with automated rules, scripts and events
Manage:	Remote updates, user interface changes, system diagnosis, import tools, device discovery
Cross-platform:	Install on Windows, Linux, Mac, Raspberry Pi, Alix, Synology, ReadyNAS, QNAP and others.

OpenRemote is an open source project, with the ambition to overcome the challenges of integration between many different protocols and solutions available for home automation, and offer visualization tools. OpenRemote Inc. was created, to enable the sponsorship of the OpenRemote open source project – in the vein of JBoss. OpenRemote follows a Professional Open Source methodology. It means that top contributors usually end up participating in the company, first as contributors, then as consultant as business develops, then as full-time employees and owners.

In any automation project there are two roles: the 'technicians' and the 'user interface designer'. The designer is primarily interested to define the use cases, and translate this into the behavior as well as UI of the system. For B2B projects, these roles are filled in by installers and UX/UI designers. In the ultimate B2C product these roles can be automated and both handled by the end-user. OpenRemote has chosen for a set of cloud-based configuration tools, with a clear distinction between the technical integration and UI design. These support both roles. The advantage of cloud-based tools is the possibility of remote support, both in project configuration as well as updates and maintenance.

A second choice they made was to have the integration and automation logic of devices and sensors, organized by a local runtime controller. The rationale is based on organizing an intrinsically stable and responsive system, meaning independence at the lowest level possible with the least dependency of higher level systems. An internet connection is only required for communication to (sub) systems outside the own network, or during configuration of the system. The presence of non IP based wired or wireless protocols, is another reason of requiring a local controller.
The third choice made was to use an object model to describe devices, commands and data, allowing for the programming of rules (using Drools), macros, commands, and designing a UI, independent of the underlying brand or protocol. This limits the system programming effort as the protocol specific programming will be eliminated.
An open-source middleware solution for the Internet of Things, OpenRemote allows you to integrate any device — regardless of brand or protocol — and design any user interface for iOS, Android or web browsers.

Key Features:
- Integrates a variety of protocols
- Customized solutions to suit your needs
- From single accounts to fully-branded solutions
- Cloud-based design tools

Using OpenRemote's cloud-based design tools for developing completely customized solutions, upgrades are streamlined, i.e devices are literally future-proof.

C. KAA
Kaa IoT Platform introduces standardized methods for enabling integration and interoperation across connected products. The Kaa IoT Platform is licensed under Apache 2.0, including the server and client components. Kaa is 100% free to use in open source or proprietary software with no royalties or fees. Kaa is designed to be robust, flexible, and easy to use. Out of the box Kaa enhances your connected products with a variety of functions.

- Profiling and grouping
- Events
- Log collection
- Notifications delivery
- Data distribution
- Transport abstraction

Kaa is a highly flexible open source platform for building, managing, and integrating connected software in the space of Internet of Things. Kaa provides a standardized approach for integration and interoperation across connected products. In addition, Kaa's powerful back-end functionality greatly speeds up product development, allowing vendors to concentrate on maximizing their product's unique value to the consumer.

D. Mango
Mango is a modular web-application framework. It takes a list of middleware, and an application and compiles them into a single http server object. The middleware and apps are written in a functional style, which keeps everything very self-contained. Mango aims to make building reusable modules of HTTP functionality as easy as possible by enforcing a simple and unified API for web frameworks, web apps, middleware.

IJIREEICE
ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526
International Journal of Innovative Research in
Electrical, Electronics, Instrumentation and Control Engineering
ISO 3297:2007 Certified
Vol. 4, Issue 11, November 2016

Key features of mango are

1. Data Acquisition: Mango can receive data across multiple protocols in sub-second intervals. Protocols such as BACnet, Modbus, SNMP, and others are built in and ready to use.
2. Real Time Data Monitoring: Data can be viewed easily on the Mango Watch List, Point Details Page, or custom HTML pages. From any modern browser real time and historical data can be viewed.
3. High Performance Database: Mango Automation Enterprise has a built-in high performance NoSQL database optimised for historical data. It greatly reduces system loads and allows for huge data sets to be quickly stored, accessed, and archived.
4. Logic and Automation: With Mango you have the power to write scripts to control equipment, calculate new data points, and crunch numbers live, in real time.
5. Security: All communications with Mango can be secured with SSL (Secure Socket Layer), ensuring the privacy of information.
6. Cross Platform: Mango can be installed on Windows, Linux, or Mac making it one of the most powerful fully cross platform application of its type. Furthermore, we can access Mango from mobile devices using just the native browser.
7. Graphic Dashboards: Graphic Views offers a simple and easy way to use images, graphics, and animations to create dashboards and HMIs. With drag and drop simplicity, our page is online quickly. JSP Pages allow us to write custom JSP pages using html and JavaScript that use your real time and historical Mango data. This allows for complete customization of dashboards, mobile apps, HMIs, and GUIs.
8. Internal Performance Monitoring: Keeping a large system running at peak performance requires good feedback on the internal process. Mango has excellent internal tools for measuring and tuning internal performance and capturing errors that help you insure long term top performance.
9. Open Source Components: The Mango core and many modules are Open Source. This provides an opportunity for in-depth education for third party developers, contributions from partner companies, and general transparency on the quality of the code.

E. OpenIoT

OpenIoT is a generic middleware platform for Internet-of-Things applications, which allows you to link together Internet-connected devices and semantic Web services via a friendly user interface, working either in Cloud Computing environments or with a local server. This platform is available as a Virtual Development Kit, providing a complete cloud solution for the Internet of Things which allows you to easily get up and running getting information from sensor clouds and connecting this information with Web services without worrying about exactly what different sensors are being used. The OpenIoT middleware enables the easy scalability of sensor networks and the addition of new, cost-effective sensors in an intrinsically flexible framework, and aims to provide a complete middleware for Internet-of-Things applications, connected sensors and wireless sensor networks. OpenIoT is building a novel platform for IoT applications, funded by the European Union, which includes powerful capabilities such as the ability to compose (dynamically and on-demand) non-trivial IoT services using a cloud-based and utility-based paradigm.

With an aim to facilitate open access to a wide range of technologies for Internet-connected sensors and other objects exposed as “services”, the creators claim that OpenIOT is the first open-source project to provide the means for setting up, managing and using a sensor cloud in this way. With the ability to support large-scale deployments by co-scheduling access from thousands of simultaneous users to millions of sensors and actuators, OpenIoT will be well placed for all IoT-based solutions of all sizes, and it will have a small number of its own open (public data) sensing services for anyone to send queries to. The OpenIoT project explores efficient ways to use and manage cloud environments for IoT entities and resources, such as sensors, actuators and smart devices, and the management of utility-based, pay-as-you-go business models for IoT networks and services.The platform will provide instantiations of cloud-based and utility-based IoT sensor and data management services, using the OpenIoT adaptive middleware framework for deploying and providing IoT services in cloud environments to enable the concept of “sensors as a service” business models for commercial IoT applications. OpenIoT supports flexible configuration and deployment of algorithms for collecting and filtering the large volumes of data that are collected by networks of Internet-connected objects, and processing and detecting those events that are determined to be particularly interesting and relevant to application or business outcomes. As OpenIoT is a completely open-source project, and all its source code is available for download – developers and end-users can examine and openly use the OpenIoT platform. You can use the OpenIoT source code to create innovative services, to extend OpenIoT with new sensor wrappers, or to improve the OpenIoT platform itself. Furthermore, OpenIoT also aims to provide the capacity for semantically annotating sensor data, according to the W3C Semantic Sensor Networks specification, streaming the data collected from various sensors to a cloud computing infrastructure, dynamically discovering and querying sensors and their data, composing and delivering IoT services that comprise data from multiple sensors and visualising IoT data using many different options such as maps and graphs.

An example application area where OpenIoT has been targeted is the improvement of efficiency in industrial operations such as manufacturing and agriculture. The OpenIoT platform can be used for intelligent sensing in manufacturing environments where it offers rapid integration of data from sensors and other devices in the manufacturing environment, dynamic and intelligent

ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

discovery of new sensors in factories, and analysis of data collected from the factory floor. The OpenIoT platform enables the dynamic selection of sensors along with the nearly-real-time fusion of sensor data in order to deliver any manufacturing indicators that are required – not just sets of inflexible, pre-configured indicators. This can increase the agility of decision-making and of the manufacturing process. One example of this agricultural application – where farmers and researchers can benefit from an instantaneous crop performance analysis platform that is powered by OpenIoT, using a wide range of distributed remote sensors gathering various types of data in order to build models that predict crop yields.

F. Nimbits

Nimbits is a PaaS that can be downloaded on a Raspberry Pi, Web Server, Amazon EC2, or Google App Engine. The platform is used for developing hardware and software solutions that can connect to the cloud or to each other, logging and retrieving large amounts of data from physical devices, triggering events or alerts, or initiating complex analysis.

Key Features:

- Download Nimbits servers on chips, servers or the cloud
- Open-source platform
- Event triggers and alerts
- Record and process geo and time-stamped data
- Build provided for Google App Engine and Linux Systems
- Compatible with most J2EE servers (Apache Tomcat, Jetty Server)

IV.CHALLENGES & RESEARCH WORK

Although the middlewares presented herein address many issues and requirements in IoT, there are still some open research challenges. In particular, research is needed in the area of dynamic heterogeneous resource discovery and composition, scalability, reliability, interoperability, context-awareness, security and privacy with IoT middleware. Importantly, most current middlewares address WSNs, while other perspectives (e.g., M2M, RFID, and SCADA) are rarely addressed. Even though there have been significant advances in addressing many challenges for middleware in an IoT environment, the following open challenges remains [6].

1. Challenges related to Functional Requirements like Resource Discovery, Resource Management, Data Management, Event Management and Code Management.
2. Challenges related to Non-Functional Requirements like Scalability, Reliability, Availability and Popularity.
3. Challenges related to Architectural Requirements like Programming Abstraction, Interoperability and Service-based requirements

V. CONCLUSION

Middleware is necessary to ease the development of the diverse applications and services in IoT. Many proposals have focused on this problem. The proposals are diverse and involve various middleware design approaches and support different requirements. This paper puts these works into perspective and presents a holistic view of the field. Finally, open research issues, challenges and recommended possible future research directions are outlined. The current state-of-the-art of the middleware for IoT explores different approaches to support some of the functionalities to operate in IoT domain. But none covers the full set of functionalities to meet the requirement of IoT-middleware as analysed here for any smart or ubiquitous environment. Middlewares have several short comings or open issues. They are available for respective domains separately. There exists no generic middleware which can be applicable across all possible smart environments. It has been observed from this study that to resolve scalability issues IPv6 is proposed but not yet resolved completely. Support for context detection and processing have not been achieved fully. Support of semantic modelling and managing of data volumes also fall in the open issues, particularly handling the crowd sourcing of diverse domain. There is a scope for research work in making a generic IoT-middleware system, which is applicable across all domains by making all the functional components reusable and can be added as add-on to the middleware system.
Although the existing middleware solutions address many requirements associated with middleware in IoTs, some requirements and related research issues remain relatively unexplored, such as scalable and dynamic resource discovery and composition, system-wide scalability, reliability, security and privacy, interoperability, integration of intelligence and context-awareness. There is significant scope for future work in these areas.

REFERENCES

[1] Liu, D. -L. Y. F., Liang, Y. -D.: A Survey of the Internet of Things. In: The 2010 International Conference on Electronic-Business Intelligence (ICEBI) (2010)
[2] Atzori, L., Iera, A., Morabito, G.: The Internet of Things: A Survey. In: Computer Networks, vol. 54, issue 15, pp. 2787—2805. (2010)
[3] L. Atzori, A. Iera, and G. Morabito, “The internet of things: A survey,” Computer networks, vol. 54, no. 15, pp. 2787–2805, 2010.
[4] SAP NETWEAVER, http://www.sap.com/platform/netweaver/components/index.epx
[5] http://wendang.baidu.com/view/ad7040a1b0717fd5360cdc8a.html
[6] Middleware for Internet of Things: a Survey, M.A. Razzaque, Marija Milojevic-Jevric, Andrei Palade, Siobh´an Clarke, IEEE INTERNET OF THINGS JOURNAL Vol. 3 Issue 1, pp 70-95, Feb 2016
[7] Jyothi T, Software Defined Network for Efficient Transmission in Wireless Networks,International Journal of Science, Engineering and Technology Research (IJSETR) Volume 5, Issue 10, October 2016
[8] Soma Bandyopadhyay, Munmun Sengupta, Souvik Maiti and Subhajit Dutta, “Role of middleware for Internet of Things: A study, International Journal of Computer Science & Engineering Survey (IJCSES) Vol.2, No.3, August 2011