MIDDLEWARE APPLICATIONS IN IOT

P.Eswari[1] and S.Balaji[2]

[1]II[nd] M.E CSE- Parisutham Institute of Technology and Science, Thanjavur.
[2] Assistant Professor - Parisutham Institute of Technology and Science, Thanjavur.
E-mail id – eswariselvam94@gmail.com,balagicse@gmail.com

Abstract— The Internet-of-Things (IoT) envisages a future in which digital and physical things or objects (e.g., smartphones, TVs, cars) can be connected by means of suitable information and communication technologies, to enable a range of applications and services. The IoT's characteristics, including an ultra large-scale network of things, device and network level heterogeneity, and large numbers of events generated spontaneously by these things, will make development of the diverse applications and services a very challenging task. In general, middleware can ease a development process by integrating heterogeneous computing and communications devices, and supporting interoperability within the diverse applications and services. Recently, there have been a number of proposals for IoT middleware. These proposals mostly addressed Wireless Sensor Networks (WSNs), a key component of IoT, but do not consider Radio-Frequency IDentification (RFID), Machine-to-Machine (M2M) communica-tions, and Supervisory Control and Data Acquisition (SCADA), other three core elements in the IoT vision. In this article, we outline a set of requirements for IoT middleware, and present a comprehensive review of the existing middleware solutions against those requirements. In addition, open research issues, challenges and future research directions are highlighted.

Index Terms--WSNs,RFID,MSM communication, SCADA, IoT characteristics, Middleware Requirements.

ACRONYMS

IoT	Internet of Things
M2M	Machine-to-Machine communications
RFID	Radio Frequency IDentification
SCADA	Supervisory Control and Data Acquisition
VM	Virtual Machine
WSN	Wireless Sensor Neywork

I. INTRODUCTION

With the advance of numerous technologies including sensors, actuators, embedded computing and cloud computing, and the emergence of a new generation of cheaper, smaller wireless devices, many objects or things in our daily lives are becoming wirelessly interoperable with attached miniature and low-powered or passive wireless devices (e.g., passive RFID tags). The Wireless World Research Forum predicts that by 2017, there will be 7 trillion wireless devices serving 7 billion people [1] (i.e., one thousand devices per person). This ultra large number of connected things or devices will form the IoT [2], [3].

By enabling easy access of, and interaction with, a wide variety of physical devices or things such as, home appliances, surveillance cameras, monitoring sensors, actuators, displays, vehicles, machines and so on, the IoT will foster the development of applications in many different domains, such as home automation, industrial automation, medical aids, mobile healthcare, elderly assistance, intelligent energy management and smart grids, automotive, traffic management, and many others [4]. These applications will make use of the potentially enormous amount and variety of data generated by such objects to provide new services to citizens, companies, and public administrations [3], [5]–[8].

In a ubiquitous computing environment like IoT, it is impractical to impose standards and make everyone comply. An ultra large-scale network of things and the large number of events that can be generated spontaneously by these things, along heterogeneous devices/technologies/applications of IoT bring new challenges in developing applications, and make the existing challenges in ubiquitous computing considerably more difficult [2], [3]. In this context, a middleware can offer common services for applications and ease application development by integrating heterogeneous computing and communications devices, and supporting interoperability within the diverse applications and services running on these devices. A number of operating systems have been developed [9]–[10] to support the development of IoT middleware solutions. In

general, these reside on the physical devices, and provide the necessary functionalities to enable service deployment. Complementary to middleware are programming language approaches [7], [8]. These approaches tackle some of the challenges (such as discovery, network disconnections, and group communication) posed by the IoT, but are limited in their support for others such as context-awareness (e.g., context-aware service discovery) and scalability.

Considering the importance of IoT in various domains, this article takes a holistic view of middleware for IoT and (1) identifies the key characteristics of IoT, and the requirements of IoT's middleware ,(2) based on the identified requirements, presents a comprehensive review of the existing middleware systems focusing on current, state-of-the-art re-search), and (3) outlines open research challenges.

II. BACKGROUND

A. IoT and its Characteristics

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) [6]. Also, the IoT can be viewed as either supporting consumers (human) or industrial

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 will be connected to the Internet, each providing data and information, and some, even services.

Sensor Networks (SNs), including wireless sensor networks (WSNs) and wireless sensor and actuator networks (WSANs), RFID, M2M communications and Supervisory Control and Data Acquisition (SCADA) are the essential components of IoT. According to the RFID community, IoT can be defined as, "The worldwide network of interconnected objects uniquely addressable based on standard communication protocols".

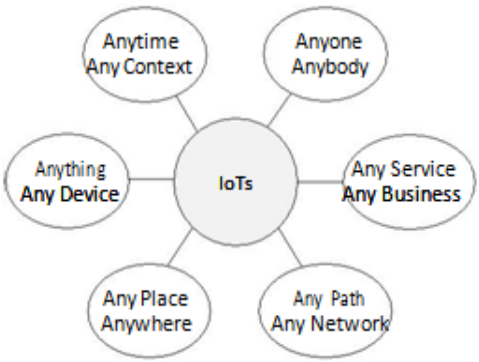

Fig.1 Definition of IoT

As described in more detail in this section, a number of the IoT's characteristics are inherited from one or more of these components. For instance, "resource-constrained" is inherited from RFID and SNs, and "intelligence" is inherited from WSNs and M2M. Other characteristics (e.g., ultra large-scale network, spontaneous interactions) are specific to the IoT.

Fig.2 Potential applications of IoT

B. Middleware in IoT and its Requiremens

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, level. 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.

- *Middleware Service Requirements*

Middleware service requirements for the IoT can be categorised as both functional and non-functional. Functional requirements capture the services or functions (e.g., abstractions, resource management)a middleware provides and non-functional requirements (e.g., reliability, security, availability) capture QoS support or performance issues.

In this section, no attempt is made to capture domain or application-specific requirements, as the focus is on generic or common functional ones.

- *Architectural Requirements*

The architectural requirements included in this section are designed to support application developers. They include requirements for programming abstractions, and other implementation-level concerns.

Providing an API for application developers is an important functional requirement for any middleware. For the application or service developer, high-level programming interfaces need to isolate the development of the applications or services from the operations provided by the underlying, heterogeneous IoT infrastructures. The level of abstraction, the programming paradigm, and the interface type all need to be considered when defining an API.

III. Overview Of Existing Work

Middleware in IoT is a very active research area. Many solutions have been proposed and impl implemented, especially in the last couple of years. These solutions are highly diverse in their design approaches (e.g., event-based, database), level of programming abstractions (e.g., local or node level, global or network level), and implementation domains (e.g., WSNs, RFID, M2M, and SCADA).

In this survey, the existing middleware solutions are grouped for discussion based on their design approaches, as below:

- Event-based
- Service-oriented
- Virtual Machine-Based Middleware
- Tuples spaces
- Database oriented

Some middleware use a combination of different design approaches. For instance, many service-oriented middleware's (e.g., SOCRADES, Servilla) also employ VMs in their design and development. Typically, hybrid approaches perform better than their individual design categories by taking the advantages of multiple approaches.

A. *Event-Based Middlewares*

In event-based middleware, components, applications, and all the other participants interact through events. Each event has a type, as well as a set of typed parameters whose specific values describe the specific change to the producer's state. Events are propagated from the sending application components (producers), to the receiving application components (consumers). An event system (event service), may consist of a potentially large number of application components (entities) that produce and consume events .Message-oriented middleware (MOM) is a type of event-based middleware. Generally, messages carry sender and receiver addresses and they are delivered by a particular subset of participants, whereas events are broadcast to all participants.

B. *Service-Oriented Middlewares*

The service-oriented design paradigm builds software or applications in the form of services. Service-oriented computing (SOC) is based on Service-Oriented Architecture (SOA) approaches and has been traditionally used in corporate IT systems. The characteristics of SOC, such as technology neutrality, loose coupling, service reusability, service composability, service discoverability, are also potentially beneficial to IoT applications.

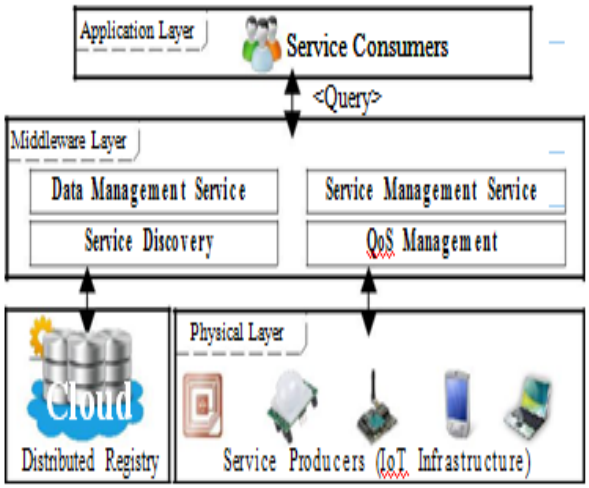

Fig.3 General design model for a Service-Oriented Middleware

C. *Virtual Machine-Based Middleware*

Virtual machine (VM) oriented middleware design provides programming support for a safe execution environment for user applications by virtualizing the infrastructure. The applications

are divided into small separate modules, which are injected and distributed throughout the network. This approach addresses architectural requirements such as high-level programming abstractions, self-management and adaptively, while supporting transparency in distributed heterogeneous IoT infrastructures , VMs can be divided into two categories: (i) Middleware Level VMs (VMs are placed between the OS and applications) and System Level VMs (substitute or replace the entire OS. Level VMs add capabilities (e.g., concurrency) to the underlying OSs System Level VMs free up resources that would otherwise be consumed by the OS.

D. Agent-Based Middlewares

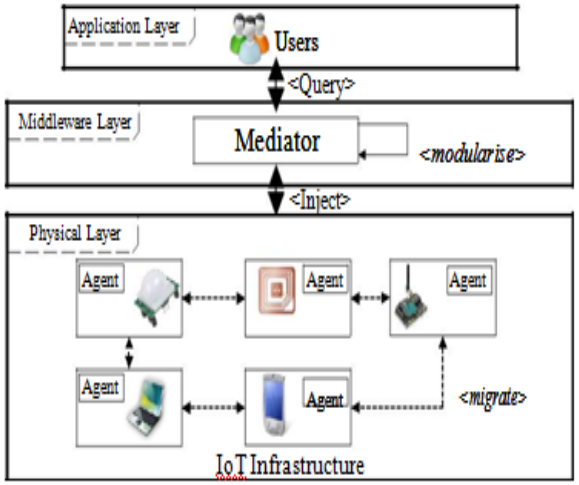

Fig.4 General design model for an agent-based middleware

In the agent-based approach to middleware, applications are divided into modular programs to facilitate injection and distribution through the network using mobile agents. While migrating from one node to another, agents maintain their execution state. Previous research in this area has presented a number of advantages for using mobile agents in generic distributed systems. In the context of the IoT middleware requirements, these are: resource management (network load reduction and network latency reduction), code management (asynchronous and autonomous execution and protocol encapsulation), availability and reliability (robustness and fault-tolerance), adaptiveness and heterogeneity. Moreover, an agent can engage in dialogues with other soft-ware agents to proactively gather data and update only parts of the application. Additionally, agent-based approaches consider resource-constrained devices.

E. Tuple-Space Middleware

In tuple-space middlewares, each member of the infrastructure holds a local tuple space structure. A tuple space is a data repository that can be accessed concurrently. All the tuple spaces form a federated tuple space Fig. 9) on a gateway. This approach suits mobile devices in an IoT infrastructure, as they can transiently share data within gateway connectivity constraints.

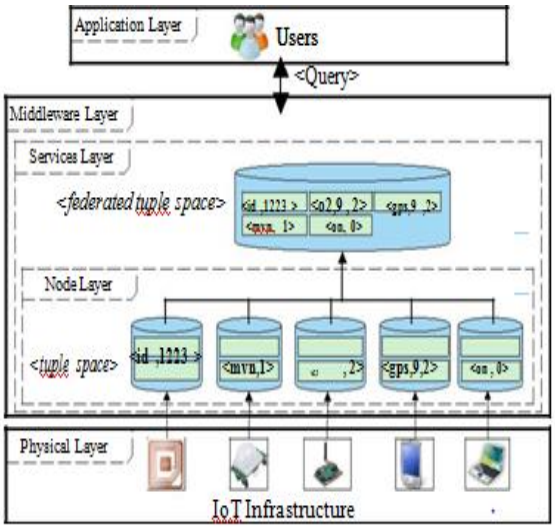

Fig.5 General design model for an tuple-space middleware

Applications communicate by writing tuples in a federated tuple space, and by reading them through specifying the pattern of the data they are interested in.

F. Database-Oriented Middlewares

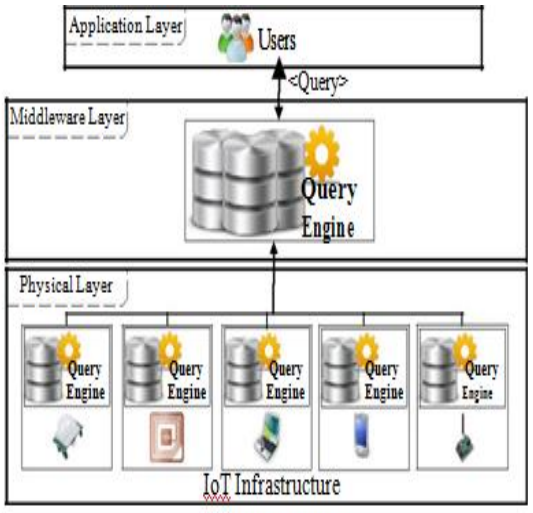

Fig.6 General design model for an database-oriented middleware

In database-oriented middleware, a sensor network is viewed as a virtual relational

database. An application can query the database using an SQL-like query language, which enables the formulation of complex queries Research in this area has been focused on developing a distributed database approach to interoperating systems.

IV. Open Research Challenges And Future Work

Although the middlewares presented herein address many issues and requirements in IoT, there are still some open re-search 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. This survey indicates that there have been significant advances in addressing many challenges for middleware in an IoT environment, with the following open challenges remaining.

Resource Discovery: The dynamic and ultra large-scale nature of the IoT infrastructure invalidates centralized resource registries and discovery approaches. However, deciding between purely distributed and hybrid solutions is complicated. A trade-off is necessary between registry distribution and the number of registries. Fewer registries provide consistent and fast discovery of resources under normal circumstances, but will not scale well when there is a large number of service discovery queries in IoT applications. Probabilistic resource (e.g., service) registries and discovery can be scalable, though may not work well in applications (e.g., mission critical applications) that need guaranteed discovery of resources with high accuracy. Further research is necessary for improved and highly accurate probabilistic models to make them suitable for diverse applications of IoT.

Resource Management: Frequent resource conflicts occur in IoT applications that share resources (e.g., actuators). Conflict resolution will be required to resolve conflicts in resource allocation among multiple concurrent services or applications. This is not considered in most existing middleware solutions, except ubiSOAP. There is clearly significant scope for future work in this area. Agent-based cooperative approach for conflict resolution could be a good starting point for autonomous conflict management.

Data Management: A vast amount of raw data continuously collected needs to be converted into usable knowledge, which implies aggregated and filtered data. Most of the surveyed middlewares offer support for data aggregation, but do not consider data filtering. Data filtering is likely to be found in application-specific approaches since the middleware is tailored for a specific application or group of applications. Moreover, no approach offers data compression. This remains an important issue for research since many IoT devices are resource-constrained and transmission of data is more expensive than local processing

Event Management: A large number of events are generated proactively and reactively in IoT. Because of this, it is expected that middleware components may become bottlenecks in the system. Most of the middleware surveyed cannot handle or have not been tested against this requirement. Also, events can be primitive (i.e., simple) or complex. Most middlewares statically pre-define how an event is handled. Further work should consider complex events and how to handle unknown events. Moreover, the work presented does not consider the difference between discrete (e.g., a door opens, switch on a light) and continuous events (e.g., driving a car).

Code Management: Re-programmability is one of the major challenges not only in IoT, but also in software development. Updates or changes in business logic should be supported by any IoT component. Agent-based, virtual machine-based and application-specific middlewares offer support for code management. Many do not distinguish between business logic code (i.e., application code) or firmware code. Moreover, none handles both cases. Many middlewares considered only homogeneous devices, though virtual machine approaches address this issue through migration and allocation of interpreted code, rather than compiled code. However, reducing the size of the interpreted code compared with the compiled code is still a challenge.

V. Summary And Future Work

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. In doing this, the key

characteristics of IoT and the requirements of IoT's middleware are identified. Based on these requirements, a comprehensive survey of these middleware systems focusing on current, state-of-the-art research has been presented. Finally, open research issues, challenges and recommended possible future research directions are outlined.

This survey categorises the existing middlewares according to their design approaches: event-based, service-oriented, agent-based, tuple-space, VM-based, database-oriented, and application-specific. Each category has many middleware proposals, which are presented accordingly. Most of these proposals have been reviewed and summarised in terms their supported functional, non-functional, and architectural requirements. The summaries show that each middleware fully or partially supports two or more of the listed requirements from each requirement type .

REFERENCES

[1] A.Gavras, A. Karila, S. Fdida, M. May, and M. Potts, "Future internet research and experimentation: the FIRE initiative," Computer Communication Review, vol. 37, no. 3, pp. 89–92, 2007.

[2] K. Paridel, E. Bainomugisha, Y. Vanrompay, Y. Berbers, and W. D. Meuter, "Middleware for the internet of things, design goals and challenges," Electronic Communications of the EASST, vol. 28, 2010.

[3] P. Bellavista, G. Cardone, A. Corradi, and L. Foschini, "Convergence of manet and wsn in iot urban scenarios," Sensors Journal, IEEE, vol. 13, no. 10, pp. 3558–3567, Oct 2013.

[4] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things: A vision, architectural elements, and future directions," Future Generation Computer Systems, vol. 29, no. 7, pp. 1645 – 1660, 2013.

[5] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," Computer networks, vol. 54, no. 15, pp. 2787–2805, 2010.

[6] D. Le-Phuoc,A. Polleres, M. Hauswirth, G. Tummarello, and C. Mor-bidoni, "Rapid prototyping of semantic mash-ups through semantic web pipes," in Proceedings of the 18th International Conference on World Wide Web. New York, NY, USA: ACM, 2009, pp. 581–590.

[7] Dohr, R. Modre-Opsrian, M. Drobics, D. Hayn, and G. Schreier, "The internet of things for ambient assisted living," in Information Technology: New Generations (ITNG), 2010 Seventh International Conference on, April 2010, pp. 804–809.

[8] Katasonov, O. Kaykova, O. Khriyenko, S. Nikitin, and V. Y. Terziyan, "Smart semantic middleware for the internet of things." INSTICC Press, 2008, pp. 169–178.

[9] Y. Ni, U. Kremer, A. Stere, and L. Iftode, "Programming ad-hoc networks of mobile and resource-constrained devices," SIGPLAN Not., vol. 40, no. 6, pp. 249–260, Jun. 2005.

[10] H. Zhou, The Internet of Things in the Cloud: A Middleware Perspec-tive,