INTELLIGENT AUTOMATION & SOFT COMPUTING, 2017
http://dx.doi.org/10.1080/10798587.2017.1290328

Middleware for Internet of Things: Survey and Challenges

Samia Allaoua Chelloug[a] and Mohamed A. El-Zawawy[b,c]

[a]Department of Information Technology, College of Computer and Information Sciences, Princess Nourah bint AbdulRahman University, Riyadh, Kingdom of Saudi Arabia; [b]College of Computer and Information Sciences, Al Imam Mohammad Ibn Saud Islamic University, Riyadh, Kingdom of Saudi Arabia; [c]Department of Mathematics, Faculty of Science, Cairo University, Giza, Egypt

ABSTRACT

The Internet of things (IoT) applications span many potential fields. Furthermore, smart homes, smart cities, smart vehicular networks, and healthcare are very attractive and intelligent applications. In most of these applications, the system consists of smart objects that are equipped by sensors and Radio Frequency Identification (RFID) and may rely on other technological computing and paradigm solutions such as M2 M (machine to machine) computing, Wifi, Wimax, LTE, cloud computing, etc. Thus, the IoT vision foresees that we can shift from traditional sensor networks to pervasive systems, which deliver intelligent automation by running services on objects. Actually, a significant attention has been given to designing a middleware that supports many features; heterogeneity, mobility, scalability, multiplicity, and security. This papers reviews the-state-of-the-art techniques for IoT middleware systems and reveals an interesting classification for these systems into service and agent-oriented systems. Therefore two visions have emerged to provide the IoT middleware systems: Via designing the middleware for IoT system as an eco-system of services or as an eco-system of agents. The most common feature of the two approaches is the ability to overcome heterogeneity issues. However, the agent approach provides context awareness and intelligent elements. The review presented in this paper includes a detailed comparison between the IoT middleware approaches. The paper also explores challenges that form directions for future research on IoT middleware systems. Some of the challenges arise, because some crucial features are not provided (or at most partially provided) by the existing middleware systems, while others have not been yet tackled by current research in IoT.

KEYWORDS

IOT; Middleware; Sensor network; RFID; Service oriented middleware; Agent oriented middleware

1. Introduction

IoT is a concept that emerged recently from the intersection of multiple technologies and computing paradigms to allow a variety of things that are uniquely identified to be effectively present in a certain environment. IoT concept addresses this challenge by allowing things to collect and exchange data through a wired or a wireless infrastructure that may be governed by many communication protocols. Moreover, the IoT combines two ideas. The first one focuses on a network-oriented style, however, the second one pushes towards on generic objects. So, the aim of the IoT is to create a network of intelligent objects that are able to take context-based decisions and adapt themselves to the surrounding environment (Atzori, Iera, & Morabito, 2010). IoT is a reality; it has been reported in (Internet of caring things, 2014; The Internet of Things: making the most of the Second Digital Revolution, 2014) that approximately 14 billion of objects are connected to the Internet. The industry trends lead to the believe that a massive growth in IoT deployment; hence the number of connected objects will reach 100 billion objects that will include devices rather than pcs, smartphones, and tables.

One of the essential components of each IoT system is the middleware. This component plays a vital role as it provides an infrastructure supporting communication between heterogeneous devices, abstraction of different applications, service discovery, mobility of the things, and security and privacy. Designing a middleware for IoT is an active research field; variety of designs and approaches has been proposed to realize the middleware concept (Atzori et al., 2010; Perera, Zaslavsky, Christen, & Georgakopoulos, 2014; Bandyopadhyay, Munmun Sengupta, Souvik Maiti, & Subhajit Dutta, 2011). Many of these approaches professionally utilize the service-oriented technology. Other approaches investigate semantic web to solve the syntax and semantic conflicts. However, recent research work implements the middleware as a set of interacting agents that act according to their rules.

This paper surveys existing techniques for implementing an IoT middleware. In this context we provide a classification to the existing techniques and a comparison between most important existing techniques. Equally important, the paper reviews current challenges and suggests directions for future work that are related to the middleware systems.

This paper is organized as follows: Section 2 presents basic concepts behind IoT. Section 3 surveys the middleware techniques illustrating main concepts and requirements to satisfy the IoT features. This also includes introducing and discussing a new classification for middleware techniques, service-oriented approaches and agent-oriented approaches. Section 4 presents potential challenges that have not yet been addressed in the existing literature for middleware for IoT. Finally Section 5 concludes the paper.

CONTACT Samia Allaoua Chelloug SAChelloug@pnu.edu.sa

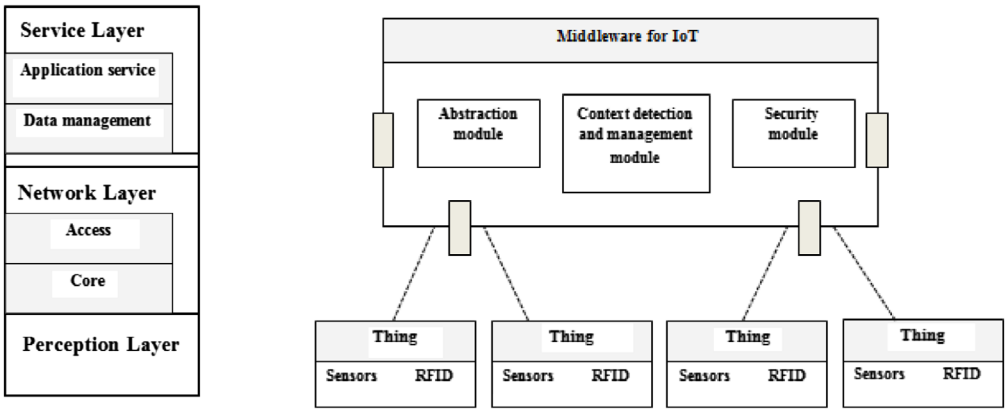

Figure 1. IoT Architecture and Middleware.

2. IoT Concepts

IoT has been attracting much interest of researchers in the last few years. This is so as it enables a set of things/objects to be:

- Pervasive by sensing data.
- Connected through wired or wireless networks.
- Identified via a unique address.
- Cooperative with other things to create new applications or services (Vermesan & Friess, 2013).

In this sense, IoT creates a worldwide network of interconnected objects that should be uniquely addressable. Computers, smartphones, vehicles, homes appliances, cameras are examples of such things/objects (Atzori et al., 2010). For example, smart refrigerators trace and report the availability and expiration dates of food items. They also rely on IoT network to place an order to grocery shops once a certain limit of the supply of food items is reached (Kopetz, 2011).

IoT is the integration of many technologies. Some of them help to acquire and process contextual information, while others improve security and privacy. Namely, sensor networks and RFID technologies play a major role for IoT systems. More specifically, RFID includes a tag that is equipped with an antenna for object identification. A sensor network based on RFID provides not only the possibility to identify objects, but also to track their behaviour or measure some parameters of the environment (Atzori et al., 2010; Jia, Feng, Fan, & Lei, 2012). On the other hand, cloud computing is also exploited in some IoT systems to create contents and applications for the users. The motivation behind integrating IoT and cloud computing is to enhance an IoT environment by taking benefit from the storage and processing capacities of cloud computing. This helps to avoid sensors' constraints. The cloud computing may also benefit from IoT by providing services to smart objects (G. Suciu, Vulpe, Todoran, Cropotova, J. Suciu, Suciu, Vulpe, Todoran, Cropotova, & Suciu, 2013; Botta, de Donato, Persico, Pescape, 2014). The work in (Li, Vögler, Claessens, & Dustdar, 2013) provides a virtual vertical architecture where each customer can adapt its own solution to its environment. (Li et al., 2013) propose to develop an IoT Paas (Dillon, Wu, & Chang, 2010) that delivers IoT services in a scalable manner. Moreover, architecture introduced in (He, Yan, & Da Xu, 2014) focuses on vehicular networks and relies on cloud computing for delivering real-time services. Beside the services offered by a service oriented system, the architecture in (He et al., 2014) proposed new services to achieve the cloud aims; network and data processing, data storage, and network management. The research of (He et al., 2014) developed also two models that enable an intelligent framework for parking service and a vehicular data mining cloud service respectively for guiding drivers and avoiding dangerous roads. IoT applications are numerous. For example, intelligent cars, trains, roads, and trails would be equipped by sensors and tags and communicate with traffic control sites. IoT can be used for smart homes and offices. It enables controlling the room heating and changing the room lightning according to the time and day (Atzori et al., 2010; Jia et al., 2012; Kopetz, 2011).

A simplified architecture of IoT is depicted in Figure 1. The architecture has two main layers consisting of two sub-layers each. The perception layer includes all technologies that allow perceiving and collecting data. The network layer takes care of transporting data in a transparent manner using the suitable communication standards including Wifi, Wimax, GPRS, and WSN. Rather than the data management sub-layer of the service layer, which treats complex and uncertain data structures, the application service sub-layer handles transforming data into content and providing an interface to user application (Jia et al., 2012). The data management sub-layer is also called the middleware layer and it represents the most critical layer of the architecture.

3. Middleware for IoT

The idea of IoT is to have a large number of different devices producing enormous amounts of data. Therefore there is a need for software (IoT middleware) that is to coordinate between components of IoT (applications exploiting the hardware and data). Hence, the role of IoT middleware is to facilitate the interaction between a multitude of diverse devices and data. This is to be done in a way that makes it easy to produce a new IoT having a single core code working on different kinds of devices or data formats.

In other words, the middleware is needed for the following reasons:

- It provides an abstraction by resolving the syntax and the semantic of sensor data.
- It is difficult to define common standards among a set of diverse devices (Soma Bandyopadhyay et al., 2011).

Three main functional components are required for IoT middleware:

- Interface protocol for providing interoperability and resolving the syntax and semantics.
- Central management: It is responsible for device discovery, and context detection and management.
- Application abstraction: Which provides the interface with local and remote applications (Atzori et al., 2010; Soma Bandyopadhyay et al., 2011).

It is important to mention that most middleware systems ensure the device management functionality. However, only some of them are designed to handle context awareness issues. Figure 1 illustrates a simplified scheme for the required components of an IoT middleware to allow interaction among a set of things. The scheme shows that the middleware should be equipped with interfaces to make the communication possible among the things. The middleware has three main modules taking care of abstraction, context-detection and management, and security. The abstraction module provides an abstraction for things and applications. The context detection and management module is meant to ensure the context awareness and mobility management features. However, the security module provides the mechanisms of authentication, privacy, and security.

There are many issues that make developing an IoT middleware not an easy task. These issues include:

- Interoperability: The middleware should allow heterogeneous devices to collaborate together.
- Scalability: A large number of devices should be supported by the middleware.
- Unfixed infrastructure: Mobile objects are expected to publish their location and their resources.
- Multiplicity: The middleware should allow a kind of optimization such that the best service will be selected among many ones.
- Security: This is a tricky issue as the IoT system combines hardware devices and networks. Therefore IoT may face cloud attacks. On the other hand, some devices require different security mechanisms due to their limited energy (Chaqfeh & Mohamed, 2012; Gil, Ferrández, Mora-Mora, & Peral, 2016).

Adapting specific communication protocols is another issue that must be tackled before deploying an IoT middleware system. Paper of (Azzara, Bocchino, Pagano, Pellerano, & Petracca, 2013) listed and discussed the main protocol and paradigm solutions for IoT middleware systems.

- IEEE 802.15.4: Its main feature concerns energy savings by altering the devices from the active to the idle sate and vice versa depending on the medium status (Institute of Electrical & Electronics Engineers, 2006).
- IETF 6LoWPan: Allows sending IPV6 datagrams in IEEE 802.15.4 based networks (Mulligan, 2007).
- Routing protocol for low-power and lossy networks: It builds an optimized graph to reach a specific destination based on the links and nodes properties (Internet Engineering Task Force (IETF), 2012).
- IeTf constrained application protocol: It is an application protocol that is intended for use by resource constrained devices. It allows mapping with HTTP to provide M2 M interactions. In fact, CoAP (constrained application protocol) interoperates with many HTTP client or HTTP server (Internet Engineering Task Force (IETF), 2014).
- Efficient XML interchange: It is an extension of XML and allows data representation and exchange between resource constrained devices (W3C, 2014).
- Restful web services: It enables the use of representational state transfer for web services called using their URIs. The restful web services representation provides the opportunity to specify the constraints of the web services to obtain a specific property (Alarcon & Wilde, 2010).

After reviewing a huge amount of related literature, we came to the conclusion that most middleware systems can be classified as service-oriented middleware (Hong, 2012) or agent-oriented middleware. Some middleware did not follow the service or the agent approach. Furthermore, we point out that the work in (Hong, 2012) presents an IoT middleware that is based on web services where each IoT resource is identified via a URI and the interaction within the resources is done through the web browser. (Hong, 2012) has also established a comparison between his proposed resource oriented middleware that has been tested using Zigbee and the service oriented approach. The conclusion was that resource oriented approach is suitable for dynamic environments. However from our point of view, the resource oriented approach is just an extension of the service oriented approach.

The middleware systems presented in (Jayaraman, Perera, Georgakopoulos, & Zaslavsky, 2014) is limited to constrained mobile devices. In this case, sensors collect data only if a consumer makes a request. Hence, sensors use little energy. In addition, the proposed middleware system enables sensors to process data locally before transmitting it to the cloud. In this context the middleware provides a plugging for each sensor it is compatible with it.

The Virtus middleware (Bazzani, Conzon, Scalera, Spirito, & Trainito, 2012) was designed for e-health application and did not follow the service, nor the agent approaches. It incorporates different technologies: Java to support portability, OSGI (OSGi Alliance, 2007) to allow a modular and a dynamic solution, XMPP open-protocol that is based on XML to ensure real time communications among heterogeneous sensors and actuators. XMPP (Internet Engineering Task Force (IETF), 2004) is required to exchange messages and events and check if each message reaches its destination. The main advantage of Virtus concerns the possibility of publishing data and retrieving it once the status of the receiver is online.

A middleware was designed in (Azzara et al., 2013) for the European Project Intelligent Cooperative Sensing for Improved Efficiency (ICSI). This system applies the software engineering approach to propose a coupe of components realizing the goals of IoT middleware system; the event and the configuration managers. It is based on restful web services and a network operating system. The core advantage of the proposed middleware in (Azzara et al., 2013) concerns its flexibility and therefore the ability to use it for different applications.

In addition, we indicate that some research works classify sensor networks middleware as IoT middleware systems. Furthermore, the Mires (Souto et al., 2006) middleware supports the communication between sensor applications based on publish-subscribe mechanisms.

TinyDB is also a sensor network middleware, which provides mechanisms to handle (Madden, Hellerstein, & Hong, 2003):

- Data readings of different sensors.
- Query generation and propagation.
- Memory management sensors.
- Topology management for efficient routing.

Senceive (Hermann & Dargie, 2008) was also developed for sensor networks. The separation between the sensing and application functionalities is the main feature of Senceive that provides a high abstraction.

Bearing in mind features of sensor networks, we came to the conclusion that the IoT environment should support extra properties related to intelligence and identification. The IoT environment may also integrate other technologies and protocols that are not used by sensor networks. This brings to attention, the big issue and debate of how to extend sensor networks middleware systems to handle the IoT scenario and an open challenge concerns the enhancement of sensor networks middleware systems to handle IoT scenario. So, in the next subsection we review IoT middleware that falls in our two classes of middleware systems.

3.1. Service-oriented IoT Middleware

Service-oriented architectures (SOAs) (Endrei et al., 2004) have been exploited to address some challenges of IoT middleware. A service oriented architecture is used to manage many services by incorporating a service provider that is intended to host one or some services, a service consumer that represents any application, and a register of services. A SOA is able to support three important functionalities (Hachem, Pathak, & Issarny, 2014; Issarny et al., 2011):

- Service discovery: It allows announcing new services and it also performs a search for the best services that satisfy a certain request.
- Composition: This functionality should be enabled once the system cannot discover an appropriate service to handle a specific request.
- Access: It provides the interaction with the discovered services.

Based on a SOA, a complex system is decomposed into an ecosystem that consists of simple components. In order to alleviate the problem of interoperability, much research effort has been expanded on designing a middleware for IoT via modifying or enhancing the SOA. We will present in details related work that proved and demonstrated that SOA is an efficient way to build an IoT middleware.

Mobile Internet of Things (MOBIOT) (Hachem et al., 2014; Issarny et al., 2011) solves the network topology issues to avoid the situation of requesting a service that becomes suddenly unavailable. The MOBIOT middleware is based on a set of Nasa's sweet ontologies to specify the IoT system (sensors, actuators, etc.) and their relationships. The main feature of MOBIOT is that the registration and lookup services are probabilistic. This controls the ability of a new thing from registering its services. MOBIOT computes the probability of presence of a mobile thing within a sensing coverage. MOBIOT has been implemented, but its performance has not been compared to existing service oriented middleware for IoT.

The work in (Teixeira, Hachem, Issarny, & Georgantas, 2011) proposed a middleware solution that aims at improving the service discovery in an IoT environment, which is characterized by inaccurate data and may face many data availability and conflict resolution problems. A mathematical model is used to estimate the conflict resolution and makes it easy to connect to a server to find services that match the required attributes. In a smart way, the proposed system of (Teixeira et al., 2011) utilizes semantic concepts and provides an ontology for sensors, actuators, and any other physical unit.

It is worth noting that the semantic approach that has been introduced in (Song, Cárdenas, & Masuoka, 2010) has many advantages. It enables interoperability, because the service requestors can understand the available services of the providers. It also allows context awareness by reducing the search space for service discovery and composition functionalities, and improves security and privacy decisions.

WiseMid middleware (Domingues, Damaso, & Rosa, 2010) is a service-oriented middleware that assumes a sensor network that is integrated to the Internet, so, it jointly considers two features; IP communication and energy saving. The middleware itself tolerates different mechanisms for energy saving:

- Aggregation: The aggregation service avoids the network traffic overload by processing correlated or redundant data.
- Reply storage timeout: This service is very useful since it stops sending the messages that contain the same parameters as the first message, which has already been sent.
- Atomic type conversion: This service removes bytes from messages by converting their type.
- Invocation asynchronous patterns: Four patterns of asynchronous communication are provided to avoid any wasting time that consumes sensors' energy.

Each WiseMid interface service is specified through the Interface Definition Language (IDL) (W3C, 1997). Each interface is simply described by its name and the operation handled by the service. More specifically, IDL specifies the input/output parameters of the operations and the exceptions that may trigger at any time. From an architectural perspective, WiseMid is composed of three layers:

- Common services layer: It specifies generic services that are required by any application (such as aggregation, grouping, and naming).
- Distribution service: Ensures remote request/reply interactions among services.
- Service infrastructure: Consists of the server and client request handlers.

The power consumption of WiseMid sensors was evaluated and the performance of the middleware was proved to be good. However, the main disadvantage of WiseMid is that it does not provide any semantic interoperability among services (Domingues et al., 2010).

The basic idea of Hydra (Eisenhauer, Rosengren, & Antolin, 2009) is to model each IoT device as a service. The system of (Eisenhauer et al., 2009) associates a semantic to each service using an ontology language like the Web Ontology Language (OWL). In particular, each device is a semantic web service that allows interoperability. Hydra annotates also new devices using a device development kit. Commands (such as get, start, stop, and current power consumption) and specific services are performed by each device. Hydra supports many operating systems (including TinyOs, Linux, and Windows) and many physicals layers (such as Zigbee and Bluetooth). Hydra

Table 1. Comparison between IoT Middleware Systems.

Approach Features	Mobiot	WiseMid	Hydra	(Gama et al., 2012)	(Zhou et al., 2013)	Socrades	UbiWare	(Fortino and Russo (2013).	(Fortino et al., 2013)	(Yang et al., 2012)	(Milagaia, 2008)
Syntax and semantic resolution	Provided	Not provided	Provided	Not provided	Provided	Not provided	Provided	Not provided	Not provided	Partially provided	Provided
Context detection	Partially provided	Partially provided	Not provided	Partially provided	Not provided	Partially provided	Provided	Partially provided	Partially provided	Provided	Provided
Application abstraction	Not provided	Partially provided	Partially provided	Partially provided	Provided	Not provided	Partially provided	Not provided	Partially provided	Partially provided	Not provided
Scalability	Provided	Not provided	Not provided	Not provided	Not provided	Not provided	Provided	Not Provided	Not provided	Not provided	Not provided
Mobility	Provided	Not provided	Not provided	Not provided	Not provided	Not provided	Not provided	Not provided	Not provided	Not provided	Not provided
Inaccurate data	Provided	Partially provided	Not provided	Partially provided	Not provided	Not provided	Not provided	Not provided	Not provided	Not provided	Not provided
Security	Partially provided	Not provided	Not provided	Not provided	Not provided	Partially provided	Not provided	Not provided	Not provided	Not provided	Not provided
Intelligence and reasoning	Partially provided	Not provided	Not provided	Not provided	Partially provide	Not provided	Partially provided	Partially provided	Not provided	Partially provided	Not provided

architecture is divided into two subsystems; application elements and device ones. The device elements include a semantic, a service, a network, and a security layer. The application elements include the same layers as the device elements, but it incorporates additional components; schedule, ontology, event, and diagnosis managers (Zarghami, 2013). It is worth mentioning that a diagnosis manager that ensures error detection and provides recovery solutions is integrated to Hydra. The diagnosis manager relies on the QoS manager, which negotiates the QoS parameters with other services. The diagnosis manager is based on OWL ontology and Semantic Web Rule Language (SWRL). Hydra is state based and reports errors and warnings (Zhang & Hansen, 2008).

The work in (Gama, Touseau, & Donsez, 2012) introduced a generic service-oriented middleware that is extensible, and adaptable. This middleware is based on Java technology to define the main interfaces for starting, stopping, and configuring the readers as well as configuring the destination that should receive the reports. The middleware of (Gama et al., 2012) also focuses on an IoT environment that includes clients, servers, sensors, RFID readers and other intermediate network devices. The middleware allows RFID events to be sent to a specific destination. The proposed architecture contains a collection and a filtering component that are responsible for collecting and relaying RFID events.

The contribution in (Zhou, Fan, & Ma, 2013) is based on a service oriented middleware that annotates user demands, web services, and data resources that are classified using domain ontologies. The annotation avoids any ambiguity and enables to attach a semantic such that the system can reason about the suitable service. The accuracy of the classification has been measured through an experiment that demonstrated the performance of the middleware. The advantage of such middleware is that for new applications, one would to only replace the domain ontologies.

The main feature of Socrades (de Souza et al., 2008) is to integrate web services enabled devices with enterprise applications such ERP. The Socrades middleware is based on the following services:

- Brokered access to devices: Provides the communication between web services and servers via an intermediate party.
- Service discovery: Is distributed and relies on UDP multicast.
- Device Supervision: Provides the required static and dynamic information about each physical device.
- Service lifecycle management: Ensures updating some services when necessary.
- Crosslayer service catalogue: Is responsible of composition and discovery of a set of relevant ERP services.
- Security support: Controls the communication of services, devices and the communication between them. It also controls the access and supports security and confidentiality.

3.2. Agent-oriented IoT Middleware

Other IoT middleware research has adopted another strategy to build a middleware for IoT. The use of the agent technology is an effective way to design decentralized systems that hold partial failures, mobility, coordination, and negotiation. Despite its

advantages, the multi-agent approach presents some obstacles for the middleware level: IoT includes heterogeneous devices that may use different protocols and different data formats and standards. So, ensuring the interoperability property with a multi-agent system becomes a tricky problem.

The idea of Ubiware core middleware (Katasonov, Kaykova, & Khriyenko, 2008; Nagy et al., 2009) is to integrate a software agent to every IoT resource. The software agent will be responsible for controlling the state of the resource. The agent will communicate with other agents. It should also be able to discover other agents. The structure of the Ubicore (reference) agent is based on three layers; the behaviour engine in Java, a declarative middle-layer that stores the agents' beliefs and a set of atomic behaviours, which represent Java components that act as sensors, actuators. The Ubicore middleware aimed also at addressing the problem of the huge number of rules and beliefs in an IoT environment. So, Ubicore developed a solution called Semantic Agent Programming Languages (SAPL) (Katasonov & Terziyan, 2008) that ensures the understanding of the semantics of the rules. SAPL provides a semantic data model and reasoning, because it is based on RDF. In addition, SAPL uses the same storage for data and code. Further, using SAPL, any rule can add or remove other rules upon its execution. In SAPL, everything is modeled as a basic semantic statement or a set of linked semantic statements. The statements are made of contexts and/or objects. The Ubicore middleware has been implemented and demonstrated for a smart service desk application. Ubicore middleware has been able to perform some automation for an operator's service desk by integrating heterogeneous components: Humans, customers' equipment's, and databases. Ubicore collects and reports all information concerning the customers and allows them to report any problem.

The research in (Fortino & Russo, 2013) proposed to tackle the complexity of an IoT environment by integrating cloud computing and multi-agents systems. The core motivations are related to the facts that:

- The cloud provides powerful storage devices and enables sharing resources and applications.
- Each agent is capable of taking a decision proactively.

The proposed architecture in (de Souza et al., 2008) is based on JADE (Bellifemine, Poggi, & Rimassa, 2001), which is a framework for developing multi-agent systems. In addition, the architecture relies on programming languages for sensor networks. It includes user agents that track humans' behaviour and object agents for modeling the behaviour of Internet objects. The developed architecture of (de Souza et al., 2008) deals also with the communication issues and provides the required interfaces using ACL (Agent Communication language) (FIPA Architecture Board, 2000).

The idea of the middleware suggested in (Fortino, Guerrieri, Lacopo, Lucia, & Russo, 2013) is similar to that presented in (FIPA Architecture Board, 2000). However, the novelty of the work proposed in (Fortino et al., 2013) is related to the event driven feature of the implemented agent oriented middleware. This includes a behaviour component that formalizes the objects' behaviours as a set of tasks that can be either proactive or reactive. The management of the events is handled by the dispatcher. There is also a communication component that manages the communication among the Internet objects. The sensors and actuators (associated with every object) are managed by the device management component of the architecture. There is also a knowledge base, which is controlled by the KB management component for describing the objects. Each is represented by its type and its source. In particular, an event may carry a piece of information, a request, an error, or a log-in issue. The event may be generated either by an internal or external software or trigged by the device itself. According to (Zhang & Hansen, 2008), this middleware has been implemented using JADE platform. Also (Zhang & Hansen, 2008) investigates many different scenarios of a smart environment for supporting the working environment of office users.

Finally, we point out that the authors of (Yang, Wang, Liu, & Wang, 2012) have presented a middleware for IoT that is based jointly on web service and multi-agent systems. The developed middleware acts at the network layer to hide the heterogeneity in term of protocols and formats of the exchanged messages. Each Internet object is considered as a web service and two different agent layers are provided; the agent layer transforms any communication protocol to a web service. However, the gateway agent lay generates web service interfaces. The middleware enables to test and maintain a link and discover other agents. Simulation results of (Yang et al., 2012) demonstrated that the performance of the proposed middleware is good in terms of the throughput and the time consumed by the agents for the communication.

In this context, DPWS (Device Profile for web services) (Milagaia, 2008) aims to merge the service and agent oriented approaches. It is based on a stack that handles the discovery process of services, the announcement of events, the assignment of an address to a specific device, the association of a policy to a web service, and other optional security mechanisms. The middleware includes some agents for managing the server and others for managing clients. The agents can look up for others by sending probe messages and upon the detection of an event, a message is sent to the subscriber. The middleware was tested and proved to be successful in connecting entities of a virtual production line.

One of the main results of our survey is the construction of a concise comparison (Table 1) between the above presented approaches. We qualitatively captured and summarized the features of each approach using the terms "provided", "partially provided" and "not provided". We use the term "partially provided" to express the situation when the concerned technique solves one or more aspects of the concerned feature but it did fully cover it. The features indicated in Table 1 are partially provided for one or more of the following reasons:

- Syntax and semantic resolution: Absence of the semantic aspect and/or the syntax did not allow modeling all required scenarios of an IoT system.
- Context resolution: Ignorance of events or other devices' status.
- Application abstraction: The way of applying the concerned middleware for different applications is not specified.
- Inaccurate data: Lack of fusion of data.
- Security: Just some security issues are handled.
- Intelligence and reasoning: Lack of reasoning about new situations that are not specified.

A precise investigation of Table 1 shows that all desired criteria of a robust middleware system (syntax and semantic resolution, context awareness, application abstraction,

scalability, mobility, processing of inaccurate data, security, and intelligence) are still not fully provided by a single middleware system. Therefore so many research challenges are still open. In particular, scalability, mobility, inaccuracy of data, security and intelligence are severe features that should be handled by next generation middleware systems.

4. Challenges

Deep investigation of the existing techniques of IoT in general and that of IoT middleware systems in particular derived us to believe that these are very active and live research areas. Our survey of IoT middleware systems resulted in concluding the following list of open problems, challenges, and issues:

- Developing and effectively checking/testing models to provide support for the complexity of an IoT middleware? The most related work to this direction is the work (Reetz, Kümper, Lehmann, & Tönjes, 2012), which develops a framework for service testing in an IoT dynamic environment. This framework is able to generate a specific test using finite state machines.
- Developing a self-adaptive middleware for IoT that takes into account the network conditions and links reliability is still a challenge.
- Building a self-healing middleware for critical applications is a good research topic. The middleware should recognize a failure and isolate it even though there is not a rule that handles the error. In other words, the middleware should be intelligent.
- An interesting open problem is how to combine the characteristics of service and agent oriented approaches in one technique. The challenge concerns the enrichment of service oriented with some multi-agent characteristics or vice versa. This may result in a flexible, reconfigurable, and interoperable middleware. The work presented in (Yang et al., 2012; Milagaia, 2008) is a step towards the integration of service and agent oriented approaches, but still the integration is not well specified and there still work needed to enhance the syntax and semantic resolution as well as the reasoning aspect.
- Memory management is an important issue in IoT. The idea of IoT makes it not clear whether traditional memory management techniques are convenient or suitable for architectures of IoT. One scenario for example is that we have many devices in a small spaces (communicating with each other). In this case is it convenient to have each of the devices having its own private memory or is it more convenient to have a shared memory? Or to have part of each memory shared with others devices? Such questions need to be answered.
- The real-time aspects of some IoT applications have not been addressed in the developed middleware.
- The languages OWL and Semantic Web Rule Language (SWRL) that are used by some middleware for IoT are not generic enough. Developing a programming language for main commands of communications between heterogeneous devices composing IoT is a promising direction for future work. Such programming model has to be associated with semantics for its commands. Also the commands of the language are to be designed in a way that makes them general enough to allow heterogeneous devices to communicate. However the generality of the commands have to accommodate the specific nature of IoT. It seems that there is no research at all in this direction, which may make good work in this direction very welcome to the IoT community. One way of designing the language commands is to have the commands reacting to and setting values of group of variables in the local and global memories of different devices, (new way to achieve this (we believe). Therefore, we think this direction ends up with a communication language rather than a programming language (or may be a mix of both of them).
- The available middleware did not specify the relationship between the security layer and the other ones.

5. Conclusion

This paper discussed the main middleware infrastructures for IoT. The paper presented a classification of the existing IoT middleware systems. The literature review led us to conclude most recent research work adopted the service-oriented approach for abstracting the complexity of the middleware design and providing useful interfaces for the application layer. Among our findings is that it is common for many researches in the field to report the need to integrate semantic web in a service-oriented style to support the heterogeneity of the things. On the other hand, the "agent-oriented" style proves useful to achieve the same job while allowing self-configuration, self-healing, and reasoning. Our investigation reveals and discusses many challenges and open problems that are still to be addressed in this area. One major challenge concerns the integration of the service and agent-oriented approaches. Another problem arises by the fact that the IoT specification languages are not generic enough to support different applications. Other issues are that the self-healing and self-configuration properties are not fully, nicely, and neatly realized. Finally there is no standard checking model for middleware systems, which makes comparing them is not a precise job.

Disclosure Statement

No potential conflict of interest was reported by the authors.

Notes on contributors

Samia Allaoua Chelloug received an Engineering degree in Computer Science in 2003, a Master's degree in Computer Science in 2006, and a Ph.D. degree in Networking in 2013, all from University of Constantine, Algeria. From 2006 to 2013 she was an Assistant Professor in the Faculty of Computer Science in Constantine University. In August 2013, she joined the Department of Networks and Communication Systems at Princess Nourah bint AbdulRahman University (Riyadh, Kingdom of Saudi Arabia) where she is presently an Assistant Professor. Her current research interests include wireless sensor networks, body area networks, cloud computing, cognitive radio, mobile wireless networks, vehicular networks, Internet of Things, and pervasive computing. Dr. Samia Allaoua Chelloug has published 15 papers in journals and conference proceedings and she has reviewed some ICCC15 conference papers.

Mohamed A. El-Zawawy received a Ph.D. in Computer Science from the University of Birmingham in 2007, an M.Sc. in Computational Sciences in 2002 from Cairo University and a BSc. in Computer Science in 1999 from Cairo University. Dr .El-Zawawy is an associate professor of Computer Science Dept. at Faculty of Science, Cairo University Since 2014. Currently, Dr. El-Zawawy is on a sabbatical from Cairo University to College of Computer and Information Sciences, Al Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh, Kingdom of Saudi Arabia. During the period 2007–2014 Dr .El-Zawawy held the position of an Assistant Professor of Computer Science at Faculty of Science, Cairo University. During the year 2009, he held the position of an extra-ordinary senior research at the Institute of Cybernetics, Tallinn University of Technology, Estonia, and worked as a teaching assistant at Cairo University from 1999 to 2003 and later at Birmingham University from 2003 to 2007. Dr. El-Zawawy is interested in static analysis, shape analysis, type systems, semantics of programming languages, internet of things, and networks.

References

Alarcon, R., & Wilde, E. (2010). RESTler: Crawling RESTful services. *In proceedings of the 19th International conference on World Wide Web* (pp. 1051-1052). Raleigh, North Carolina. DOI: 10.1145/1772690.1772799

Atzori, L., Iera, A., & Morabito, G. (2010). The internet of things: A survey. *Computer Networks, 54*, 2787–2805. Doi:10.1016/j.comnet.2010.05.010.

Azzara, A., Bocchino, S., Pagano, P., Pellerano, G., & Petracca, M. (2013). Middleware solutions in WSN: The IoT oriented approach in the ICSI project. *In Proceedings of IEEE International Conference on Software Telecommunications and Computer Networks* (pp. 1-6). DOI: 10.1109/SoftCOM.2013.6671886

Bandyopadhyay, S., Munmun Sengupta, M., Souvik Maiti, S., & Subhajit Dutta, S. (2011). Role of middleware for internet of things: A study. *International Journal of Computer Science and Engineering Survey, 2*, 94–105. Doi:10.5121/ijcses.2011.2307.

Bazzani, M., Conzon, D., Scalera, A., Spirito, M., & Trainito, C. (2012). *Enabling the IoT paradigm in e-health solutions through the VIRTUS middleware. IEEE 11th International conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. USA, 1954–1959, DOI: 10.1109/TrustCom.2012.144

Bellifemine, F., Poggi, A., & Rimassa, G. (2001). Developing multi-agent systems with a FIPA-compliant agent framework. *Software Practice and Experience*, John Wiley and Sons, Ldt., *31*, 103–128, http://www.emse.fr/~boissier/enseignement/maop14/courses/readings/FIPA-JADE.pdf

Botta, A., de Donato, W., Persico, V., & Pescape, A. (2014). On the integration of cloud computing and internet of things. *In 2014 International Conference on Future Internet of Things and Cloud (FiCloud)* (pp. 23–30). DOI: 10.1109/FiCloud.2014.14

Chaqfeh, M., & Mohamed, N. (2012). Challenges in middleware solutions for the internet of things. *In proceedings of The 2012 International Conference on Collaboration Technologies and Systems (CTS 2012)*. Denver, Colorado, USA, https://pdfs.semanticscholar.org/15bc/ef45aaee19326844a183fea6a4aabd56d0f9.pdf

de Souza, LMS., Spiess, P., Guinard, D., Köhler, M., Karnouskos, S., & Savio, D. (2008). SOCRADES: A web service based shop floor integration infrastructure. In C. Floerkemeier (Ed.), *proceedings of Internet of Things 2008 Conference* (pp. 50–67). Zurich, Switzerland. DOI: 10.1007/978-3-540-78731-0_4

Dillon, T., Wu, C., & Chang, E. (2010). Cloud computing: Issues and challenges. *In the proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications* (pp. 27-33). DOI: 10.1109/AINA.2010.187

Domingues, JPO., Damaso, AVL., & Rosa, NS. (2010). WISeMid: Middleware for integrating wireless sensor networks and the internet, In proceedings of IFIP International Conference on Distributed Applications and Interoperable Systems, 70-83, DOI: 10.1007/978-3-642-13645-0_6.

Eisenhauer,M., Rosengren, P., & Antolin, P. (2009). A development platform for integrating wireless devices and sensors into ambient intelligence systems. *In proceedings of 6th IEEE Annual Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks Workshops*, (pp. 367-373). DOI: 10.1109/SAHCNW.2009.5172913

Endrei, M., Ang, J., Arsanjani, A., Chua, S., Comte, P., Krogkahl, P., … Newling, T. (2004). *Patterns: Service-oriented architecture and web services*. New York, NY: International Technical Support Organization, IBM red books. ISBN 073845317x.

FIPA Architecture Board. (2000). *FIPA agent management support for mobility specification*. Geneva, Switzerland: Foundation for Intelligent Physical Agents. http://www.fipa.org/specs/fipa00087/PC00087B.pdf

Fortino, G., Guerrieri, A., Lacopo, M., Lucia, M., & Russo, W. (2013). An agent-based middleware for cooperating smart objects. *In communications in Computer and Information Science* (pp. 387–398). DOI: 10.1007/978-3-642-38061-7_36

Fortino, G., & Russo, W. (2013). Towards a Cloud-assisted and Agent-oriented Architecture for the Internet of Things. *In proceedings of the 14th workshop from Objects to Agents (WOA 2013)* (pp. 60–65). Turin. http://ceur-ws.org/Vol-1099/paper15.pdf

Gama, K., Touseau, L., & Donsez, D. (2012). Combining heterogeneous service technologies for building an Internet of Things middleware. *Computer Communications, 35*(4), 405–417. Doi:10.1016/j.comcom.2011.11.003.

Gil, Pablo, Ferrández, Youcef, Mora-Mora, Markus, & Peral, J. (2016). Internet of things: A review of surveys based on context aware intelligent services. *Sensors* 2016, 16, 1-2, DOI: 10.3390/s16071069

Hachem, S., Pathak, A., & Issarny, V. (2014). Service-Oriented Middleware for the Mobile Internet of Things: A Scalable Solution. *IEEE GLOBECOM: Global Communications Conference* (pp. 1-6). Austin, United States. https://hal.inria.fr/hal-01057530/file/GBC14.pdf

He, W., Yan, G., & Da XU, L. (2014). Developing Vehicular Data Cloud Services in the IoT Environment. *IEEE Transactions on Industrial Informatics, 10*, 1587–1595. Doi:10.1109/TII.2014.2299233.

Hermann, C., & Dargie, W. (2008). Senceive： A Middleware for a Wireless Sensor Network. *Advanced Information Networking and Applications* (pp. 612-619). Okinawa. DOI: 10.1109/AINA.2008.34

Hong, Y. (2012). A resource-oriented middleware framework for heterogeneous internet of things. *In Proceedings of the conference on Cloud and Service Computing* (pp. 12–16). Doi:10.1109/CSC.2012.10.

Internet Engineering Task Force (IETF). (2004). Extensible Messaging and Presence Protocol (XMPP): Core. RFC 6120, https://tools.ietf.org/html/rfc6120

Internet Engineering Task Force (IETF). (2012). RPL: IPv6 routing protocol for low power and lossy networks. RCF 6550, https://tools.ietf.org/html/rfc6550

Internet Engineering Task Force (IETF). (2014). Constrained Application Protocol (CoAP). RFC 7252. https://tools.ietf.org/html/rfc7252

Institute of Electrical and Electronics Engineers. (2006). IEEE Std 802.15.4-2006,Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs). https://standards.ieee.org/findstds/standard/802.15.4-2006.html

Internet of caring things. (2014). Trend report. http://trendwatching.com/x/wpcontent/uploads/2014/04/2014-04-IoCT.pdf

Issarny, V., Georgantas, N., Hachem, S., Zarras, Apostolos, Vassiliadist, P., Autili, M., … Hamida, Amira Ben (2011). Service-oriented middleware for the future internet: State of the art and research directions. *Journal of Internet Services and Applications, 2*, 23–45. Doi:10.1007/s13174-011-0021-3.

Jayaraman, PP., Perera, C., Georgakopoulos, C., & Zaslavsky, A. (2014). MOSDEN: A scalable mobile collaborative platform for opportunistic sensing applications. *Transactions on Collaborative Computing, 14* 1), 1–16, https://arxiv.org/pdf/1405.5867v1.pdf

Jia, X., Feng, Q., Fan, T., & Lei, Q. (2012). RFID technology and its applications in Internet of Things (IoT). *in Proceedings of International Conference on Consumer Electronics, Communications and Networks (CECNet)* (pp. 1282–1285). Yichang. DOI: 10.1109/CECNet.2012.6201508

Katasonov, A., Kaykova, O., & Khriyenko, O. (2008). Smart semantic middleware for the internet of things. *In proceedings of the 5th International Conference on Informatics in Control, Automation and Robotics* (pp. 169-178). Funchal, Madeira, Portugal. http://www.mit.jyu.fi/ai/papers/ICINCO-2008.pdf

Katasonov, A., & Terziyan, VY. (2008). Semantic Agent Programming Language (S-APL): A Middleware Platform for the Semantic Web. *Semantic Computing, 2008 IEEE International Conference on Semantic Computing* (pp. 504-511). DOI: 10.1109/ICSC.2008.82

Kopetz, H. (2011). *Real-time systems: Design principles for distributed embedded applications*. Boston, USA: Springer. ISBN-13: 978–1441982360

Li, F., Vögler, M., Claessens, M., & Dustdar, S. (2013). Efficient and scalable IoT service delivery on cloud. *IEEE CLOUD, 2013*, 740–747. Doi:10.1109/CLOUD.2013.64.

Madden, S., Hellerstein, J., & Hong, W. (2003). Tinydb: In-network query processing in tinyos. Technical report. http://telegraph.cs.berkeley.edu/tinydb/tinydb.pdf

Milagaia, R. (2008). DPWS middleware to support agent-based manufacturing control and simulation. Robotics and integrated manufacturing (Master thesis). Monte de Caparica, Universidade Nova de Lisboa.

Mulligan, G. (2007). The 6LoWPAN architecture. *In Proceedings of the 4th workshop on Embedded Networked Sensors* (pp. 78-82). 10.1145/1278972.1278992

Nagy, M., Katasonov, A., Khriyenko O., Nikitin, S., Szydlowski, M., & Terziyan, V. (2009). Chapter 14 in automation & control - theory and practice (247–273), challenges of middleware for the internet of things, ISBN 978-953-307-039-1, DOI: 10.5772/7869

OSGi Alliance. (2007). About the OSGi service platform, technical whitepaper. http://www.osgi.org/documents/collateral/OSGiTechnicalWhitePaper.pdf.

Perera, C., Zaslavsky, AB., Christen, P., & Georgakopoulos, D. (2014). Context aware computing for the internet of things: A survey. *IEEE Communications Surveys & Tutorials, 16*, 414–454. Doi:10.1109/SURV.2013.042313.00197.

Reetz, ES., Kümper, D., Lehmann, A., & Tönjes, R. (2012). Test driven life cycle management for internet of things based services: A semantic approach. *In proceedings of the 4th International Conference on Advances in System Testing and Validation Lifecycle* (pp. 21–27). Lisbon. https://pdfs.semanticscholar.org/2e75/f816f2c73929504d1107581175ceecdb5450.pdf

Song, Z., Cárdenas, AA., & Masuoka. R. (2010). Semantic middleware for the Internet of Things. *In Proceedings of Internet of things –IoT* (pp. 1-8). DOI: 10.1109/IOT.2010.5678448

Souto, E., Guimarães, G., Vasconcelos, G., Vieira, M., Rosa, Nelson, & Ferraz, C. (2006). Mires, a publish/subscribe middleware for sensor networks. *Personal and Ubiquitous Computing, 10*, 37–44. Doi:10.1007/s00779-005-0038-3.

Suciu, G., Vulpe, A., Todoran, G., Cropotova, J., & Suciu, V. (2013). Cloud computing and internet of things for smart city deployments. *In proceedings of the 7th International Conference Challenges of the Knowledge Society (CKS 2013)* (pp. 1409–1416).

Teixeira, T., Hachem, S., Issarny, V., & Georgantas, N. (2011). Service oriented middleware for the internet of things: A perspective. *ServiceWave '11, LNCS, 6994*, 220–229.

The Internet of Things: making the most of the Second Digital Revolution. (2014). UK government chief scientific adviser, (Ref: GS/14/1230).

Vermesan, O., & Friess, P. (2013). Internet of things -converging technologies for smart environments and integrated ecosystems. Aalborg, Denmark: River Publisher. ISBN: 978-87-92982-73-5.

W3C. (1997). Web Interface Definition Language (WIDL). http://www.w3.org/TR/NOTE-widl

W3C. (2014). Efficient XML interchange (EXI) format. http://www.w3.org/TR/exi/

Yang, Y., Wang, Z., Liu, Q., & Wang, L. (2012). Building a pervasive SOA based IOT communication middleware using runtime compilation and reflection. *Journal of Computational Information Systems, 8*, 643–654.

Zarghami, S. (2013). Middleware for internet of things (Master Thesis). University of Twenty.

Zhang, W., & Hansen, KM. (2008). An OWL/SWRL based diagnosis approach in a pervasive middleware. *In proceedings of the 20th International Conference on Software Engineering and Knowledge Engineering (SEKE'2008)* (pp. 893-898) http://www.hydramiddleware.eu/hydra_papers/An_OWL-SWRL_based_Diagnosis_Approach_in_a_Pervasive_Middleware.pdf

Zhou, M., Fan, H., & Ma, Y. (2013). Semantic annotation method of IOT middleware. *In proceedings of the International Conference on Intelligent Control and Information Processing (ICICIP)* (pp. 495-498). China. DOI: 10.1109/ICICIP.2013.6568125