Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2015, Article ID 139735, 17 pages
http://dx.doi.org/10.1155/2015/139735

Research Article

Architecting the IoT Paradigm: A Middleware for Autonomous Distributed Sensor Networks

George Eleftherakis,[1] Dimitrios Pappas,[2] Thomas Lagkas,[1] Konstantinos Rousis,[2] and Ognen Paunovski[2]

[1] *Computer Science Department, The University of Sheffield International Faculty, CITY College, 54622 Thessaloniki, Greece*
[2] *South-East European Research Centre (SSERC), The University of Sheffield International Faculty, CITY College, 54622 Thessaloniki, Greece*

Correspondence should be addressed to Thomas Lagkas; tlagkas@city.academic.gr

Received 31 May 2015; Revised 6 November 2015; Accepted 17 November 2015

Academic Editor: Davide Brunelli

Actualizing Internet of Things undoubtedly constitutes a major challenge of modern computing and is a promising next step in realizing the unification of all seamlessly interacting entities, either human users or participating machines, under a shared, coherent architecture. While it has now become common belief that the related solutions should be based on compatible network infrastructure employing widely accepted communication schemes, the specifics of the intermediate system that would act as global interface for all involved "things" are yet to be determined. A rising trend to define such machine-based entities is through cyber-physical systems, in terms of collaborating elements with physical input and output. Certainly, sensor networks constitute the most representative realization of such systems. Taking these issues and opportunities under consideration, this work proposes a bioinspired distributed architecture for an Internet of Things that exhibits self-organization properties to enable efficient interaction between entities modeled as cyber-physical systems, mainly focusing on sensor networks. Furthermore, a middleware has been implemented according to the proposed architecture, which serves the role of the backbone of this network as a multiagent and autonomous distributed system. The evaluation results demonstrate the self-optimization properties of the introduced scheme and indicate global network convergence.

1. Introduction

The extensive use of the Internet and its phenomenal penetration worldwide, along with the massive trends of social networking mobile computing, have created a demand for unprecedentedly complex applications. Heterogeneous platforms and services often require seamless integration with each other and intuitive sharing of information. Such demands were undoubtedly a decisive factor on the success of decentralized architectures during the last decades. The need to connect everything on the Internet under a common vision of the Internet of Things (IoT), a network of uniquely identifiable and interacting objects, the introduction of Cyber-Physical Systems (CPS), and the ambition for a "smart" planet, necessitates architectures that enable scalable, autonomous, and robust solutions which should be capable of exhibiting extremely dynamic behaviors.

Modern sensor networks require a flexibility that will allow them to operate in a decentralized manner and often without requiring a well-defined infrastructure. There is an increased demand for solutions that work in extremely dynamic environments and enable creation of an infrastructure of a sensor network in a dynamic and autonomous way, especially in situations that well-defined and well-designed infrastructures do not exist or are not preferable. Studies, as well as simple observation of natural systems, have shown how self-* properties can lead to seamless ways of forming highly robust and dynamic systems. This dictated a research trend over the last decades towards nature-inspired solutions in artificial systems and especially artificial distributed networks.

This work proposes an architecture for building a self-organizing overlay network of CPS, with a particular focus on sensor networks. This architecture is heavily inspired by nature, and specifically the fact that innovative design of individual nodes (microscopic level) can lead to the emergence of desired global properties (macroscopic level). It aims to offer an IoT solution of interacting CPS in the form of a middleware that facilitates interaction and interconnection of things in a distributed manner, providing scalability, self-adaptation, and self-organization. The applicability of the proposed architecture is validated with a realistic design and an implementation solution that could support real-world scenarios.

The following section presents the state of the art on architectural paradigms of the IoT, different approaches on CPS, and autonomous systems. In Section 3 the proposed architecture is described and justified in different levels of abstraction. Section 4 details the design of the proposed architecture and offers a thorough description of the proposed implementation solution. Finally, the preliminary results and the evaluation strategy are presented in Section 5, after which we conclude the paper and present future plans for the middleware.

2. Background and Related Work

In this section, we discuss the current status of those issues that are taken into account for the formulation of the proposed solution. Specifically, the important architectural aspects of IoT are provided, key characteristics and approaches for CPS and autonomous systems are presented, and we also elaborate on concerns and solutions for modern sensor-based architectures.

2.1. Architectural Issues in the Internet of Things. The vision of the IoT represents an assortment of interconnected, interacting objects, which could be an overlay network that utilizes the Internet itself but is not limited to this solution alone. Although identifying each network peer can be challenging, it is not the primary concern when realizing the IoT. According to Ashton, the most crucial aspect is the interaction among humans and objects or among objects alone [1]. As the type of interactions is dependent on the type of network and the needs it was designed to fulfill, aiming for a single architecture that fits every scenario is practically impossible and it leads to a multitude of standardization issues [2–4].

The types of interactions within a network may define the functionality of each peer, as well as the purpose of the network itself, but the virtual representation of these objects is of utmost importance in realizing the vision of an IoT. Such representations have an architectural impact on ADS and they also affect the node representation in the network, as well as the manner in which information flowing between humans and devices is being represented [5].

An important architectural differentiation is the approach taken towards communication, with the two general categories being *synchronous* and *asynchronous communication*. In an IoT implementation, a synchronous communication would introduce a large overhead which could defeat its purpose of guaranteeing delivery in a timely fashion [6, 7]. Thus, an asynchronous mode of communication is deemed more appropriate for such scenarios.

In a network with the size and interconnection characteristics of an IoT, it is expected that an immense amount of information is being generated which should be collected only when this is requested or when a real-world object is capable of providing that data [7, 8]. This implies that there is no need for processing unless an event is triggered, although other solutions could be implemented such as utilizing agents and agent-based messaging. Equally important to such implementations is how the messages are processed and prioritized; if they are processed in a first-come, first-served manner, that would simulate an undesired synchronous communication model. Apart from an event-driven approach with a properly structured messaging protocol, the agent-based programming paradigm has a lot to offer to the IoT. Specifically, the capacity for intelligence that agents inherently possess enables the opportunity for achieving autonomy and self-management for complex systems such as those that serve as the backbone of an IoT implementation [9–11].

Finding solutions to IoT issues in the agent-based programming paradigm hints at the possibility of opportunities to be found in other popular programming paradigms. In particular, the most relevant one may be that of Service-Oriented Architecture (SOA). A SOA-compliant middleware would be eligible to provide any valuable information or functionality possessed by the real-world objects to end-users in the form of services. The most important property, in terms of value to the IoT, that can be found in software complying with a SOA is, unarguably, interoperability. Interconnecting heterogeneous devices and allowing for interoperability through a SOA are possible through the use of standardized protocols and semantics [7, 9, 12]. Nonetheless, developing a SOA for a middleware which aims to realize an IoT is posing various research challenges including heterogeneity, mobility, scalability, adaptiveness, adaptability, awareness, security, and privacy [13].

One of the important architectural concerns pertains to the problem of discovery strategies for services in a SOA-compliant middleware. The two fundamental aspects of service discovery have been shown to correspond to the following: (a) how services are organized within a system and (b) how they are located within a system. The former is shown to be best achieved by storing information about services in the form of a localized registry (i.e., a repository on each node), which describes what services can be provided by this node [13]. Notwithstanding, the crucial issue of locating them inside the network still remains; to this end, the state of the art dictates two particular approaches: (a) centralized repository of registries and (b) distributed discovery.

Centralized approaches are most commonly adopted by the industry, including well-established solutions such as EPCglobal, Afilias, ID@URI, and the BRIDGE project [8]. A centralized implementation necessitates the service consumer to connect to a remote registry, with an interaction in the manner of the client-server paradigm and a user querying a database. Centralized solutions are undoubtedly practical but there are several concerns that need to be taken under consideration. Specifically, the larger the repository becomes the more expensive (resource-wise) it is for a single administrator to manage. At the same time, service availability becomes more problematic, since it constitutes a single point of failure and attack.

Distributed discovery, on the other hand, is much more diverse and a single best approach is lacking. Different strategies have been developed to solve different problems with various trade-offs. Biologically inspired [14], traditional P2P [8], DNS-based solutions [15], and Web-based solutions [16] are few of the most popular approaches.

A recurring theme on most approaches for realizing an IoT architecture is the use of Web services. Inherently, the design of Web services has the power to provide unique identification to the service providers and exceptional, if not unparalleled, ease of access to the service consumers (i.e., the things in both cases). Their immense popularity showcases how they are virtually everywhere in this era of cloud computing and attests to their power to contribute to a different perspective at the vision of the IoT, that of the Web of Things (WoT). Once more, it is essential to stress the difference between the IoT and the WoT: the former is about interconnection and interaction of the devices and users in general (which could be the Internet itself or any other type of network), whereas the latter focuses on achieving these goals through the World Wide Web (WWW) and the Web services in particular (which is always an overlay above the Internet) [17].

Accessing a Web service is fairly uncomplicated as long as its URI is known, along with the type of the requests it can respond to. This holds true whether this is a RESTful service (adhering to the Representational State Transfer architectural constraints) or a Web Services Description Language (WSDL) service utilizing the Simple Object Access Protocol (SOAP), the two predominant approaches to the realization of Web services [18, 19]. Therefore, the virtual representation of any device or user comes in the form of either a service consumer or a service provider, bringing ever closer the fields of IoT and SOA, along with the formidable advantages the latter has to offer. In fact, interoperability is achieved seamlessly through the Web services and SOA, although there still exists the challenge of abstracting the functionality of the things as services.

2.2. Cyber-Physical and Autonomous Distributed Systems. With the advances in the field of wireless sensor networks and the popularity they enjoy nowadays, not discounting a heavy investment in that field towards encompassing IoT for the much-discussed benefits, Cyber-Physical Systems (CPS) and robotics or sensors have come to be almost synonymous terms for several practitioners and researchers [20]. On the other hand, some claim that the term of CPS indicates the next step in the evolution of wireless sensor networks [21]. Despite wireless sensor networks being the prevalent CPS family, significant work has begun towards most other families and generally most devices that have integrated with embedded systems [22]. The modern smart car, for example, is one of the most frequent examples of a CPS.

Architecturally, CPS are composed of varied subsystems, or parts, each of which can be represented at an abstract level by a component, thereby proving the meaningful application of component-based software engineering (CBSE) on this particular area. The Dependable Emergent Ensembles of Components (DEECo) is an example middleware that has evolved into a fully fledged solution for CPS design by improving interaction among subsystems with minor strain and effort on development, all the while being applicable to a variety of scenarios [23]. Programming Temporally Integrated Distributed Embedded Systems (PTIDES) model is another example, a programming model that succeeds in simplifying and enhancing the development of distributed, real-time CPS by resorting heavily to software components and model-based design [24]. To summarize, the literature signifies the capacity of SOA to solve most issues concerning the exchange of information among distinct and even heterogeneous interconnected CPS systems.

All approaches presented so far require from the system to operate within its life cycle to its fullest potential with minimal to virtually no human intervention at all, which requirement actually matches the concept of autonomy in computing. It was initially introduced by IBM in 2001 in an attempt to integrate heterogeneous devices and environments [25]. As a means of evaluating whether a system was exhibiting autonomy or not, it was decided that the system should possess several properties known as self-*: self-configuration, self-healing, self-optimization, and self-protection [26, 27].

MAS are often found in the center of discussions about autonomy and particularly for autonomous ADS. This is attributed to the fact that their interactions have been shown to lead to the emergence of beneficial properties in these systems, such as robustness, scalability, and adaptability by using partial views, feedback, and self-evaluation functions [28].

Services and a SOA are also capable of providing some degree of autonomy to ADS. Services can handle the task of communication or information sharing, and focusing on lightweight services with descriptions based on standards will definitely contribute to enhancing the awareness aspect of autonomy [29, 30]. In addition to borrowing from the field of SOA, principles fundamental to the CBSE can be utilized for ADS towards autonomy, offering as much as they do to CPS, if not more.

Additionally, architectures such as the Organic Grid, where agents attempt to colonize resources they discover in the ADS much like an array of biological organisms would (humans included), achieve self-organization and hence a satisfying degree of autonomy [31]. On the other end of the spectrum, a system may exhibit emergent properties that differ from those envisioned by its designers and that could prove to be harmful to the system. Thus, modeling such solutions requires careful and extensive evaluation of the final system [32].

2.3. Sensor Network Architectures. It is apparent that architectural efforts on sensor networks face the same challenges of identification, distribution, and autonomy that were previously discussed. The emergence of unified entities with behavioral patterns arising from the individual participating devices has inspired similar architectural ideas for sensor networks. A representative example is BiSNET [33], a sensor network architecture inspired by bee movement patterns, which includes a middleware and multiple agents that behave according to biological principles.

From an architectural point of view, the most crucial and well-researched part is the middleware as it establishes the roles and the connectivity aspects of the whole system. A flexible, plug-and-play type of middleware has been proposed in [34], named Global Sensor Networks (GSN). It specializes in sensors and on how users can access their information. The GSN middleware utilizes a directory for interconnecting the sensors and users accessing the network, while it employs the IEEE 1451 standard for drawing data from the sensors. GSN nodes communicate directly with each other in a P2P fashion, without any intelligence behind the query processing [34]. Consequently, an implementation through the GSN middleware will not lead to a system that can adapt to changes to its expected operational environment.

Another middleware with an interesting architectural approach on interconnecting different types of sensors is SenaaS, which utilizes the adapter software design pattern [6]. Examples of middleware that exploit the agent-based paradigm for realizing sensor network architectures include Flexeo, an intelligent wireless sensor network [35], and Cognitive Office, a middleware for intelligent environments [10]. Both of them offer a decentralized approach to intelligence [11]. Introducing, however, intelligence in the lower layers of the considered architecture is highly questionable, as it could potentially hinder flexibility. This criticism stems from the fact that a flexible middleware for an IoT model should be agnostic of what things should or are capable of doing. Rather, such a middleware should focus on facilitating adaptive interconnection (encompassing all of its aspects) of machines and users, in the meantime achieving adaptiveness through intelligent agents in the middle layers.

Undoubtedly, wireless sensor networks have given rise to the usability and effectiveness of agents in distributed systems. In particular, the employment of mobile devices in such networks has strengthened the arguments for the efficacy of mobile software agents in dynamic architectures [36]. This type of agents are autonomous and able to adapt to the environment. In contrast, a stationary agent provides to the sensor node it is associated with information originated from its adjacent nodes. They can be embedded in the sensing devices executing their process and then moving between network sensors. The agent functionality heavily depends on the specific architecture. For instance, in hierarchical sensor networks, the agents can be created in higher order nodes, whereas in flat topologies typically the sink node is the one that instantiates the agents. In a standard scenario, the mobile agents traverse the sensor network collecting monitoring information from each one of the involved devices and eventually delivers it to the sink. In that manner, significant portion of the available bandwidth and energy can be conserved; however, there are important issues related to sensor hardware limitations and other significant practical constraints.

Typically, such implementations employ agent-based middleware solutions to realize communications inside a sensor network. One of the most known middleware frameworks for wireless sensor networks is Agilla [37]. Agilla can be run on TinyOS in order to control mobile agents over a sensor network. The agents are programmed to act towards the achievement of a common goal while their movement is regulated by the middleware. Another similar middleware for wireless sensor networks is SensorWare [38]. Although SensorWare does not employ agents, it provides similar capabilities through the use of scripts. AgentScape is an agent-based, multilayer, sensor network architecture introduced in [39]. Agents are situated in predefined locations and can communicate with external systems. Finally, Deluge [40] and Impala [41] are representative examples of platforms which allow code distribution over wireless sensor networks. However, they assume that some code is installed in advance to all participating nodes as they are not agent-based.

The employed data sampling technique can have a significant impact in the sensor network architecture as well. Efficiently scheduling the collection of information from sensors is crucial for the effectiveness and the lifetime of the network. A well-designed architecture facilitates the data collection process, allowing regular updates while keeping the usage of resources to a minimum. A widely adopted idea is the use of a portion of the available sensors, since there are often overlaps and not all nodes are always required. The backcasting algorithm proposed in [42] is based on this concept. SORA (Self Organizing Resource Allocation) [43] is adopting principles of market-oriented programming to define a penalty scheme for data forwarding, that way reducing resource and energy consumption. A combination of adaptive sampling and effective routing is proposed in USAC (Utility-Based Sensing and Communication model) [44], in an effort to save energy by adapting observations.

A major concern in such adaptive systems regards the entity that is actually performing the decision making. An efficient and flexible solution should allow individual entities to make local decisions on the sampling schedule. In [45], for example, binary integer programming techniques are employed to assign tasks to nodes, allowing them to reach decisions based on their observations.

2.4. Discussion on Findings. The IoT focuses primarily on the interconnection of and interactions among the entities connected to it, whether they are human end-users or machines. Event-driven, service-oriented approaches to interaction, along with an overlay network of uniquely identifiable entities, are essential to the vision of an IoT. Furthermore, centralized approaches to discovery of services are preferred in the industry due to high performance; nonetheless, they have difficulty supporting volatile, ad hoc networks and overcoming disaster scenarios. The entities in an IoT nowadays are expected to be CPS, such as the sensor networks, and designing such systems hints at the need for the following two: (a) properties to be found in the field of software components that solve CPS subsystem design issues and (b) a SOA for supporting their interactions. Meanwhile, infusing autonomy in that distributed system, which forms this overlay network actualizing the IoT vision, can be achieved by drawing inspiration from nature for self-organization and the utilization of the agent-based programming paradigm for autonomous operation.

Modern sensor networks have been shown to incorporate in part several of the aforementioned solutions and programming paradigms in order to achieve their goals. While the majority of solutions are tailored to a specific scenario, or at least a specific family of them, and they successfully address these issues, they are nonetheless not designed with an eye towards a future IoT application with the capacity to support numerous scenarios. These solutions also do not focus on the need for delivering a system capable of autonomous operation, achieved through minimized human intervention and attributed to self-organization properties. Finally, such systems attempt to facilitate effortless, ad hoc connectivity of the devices, albeit primarily relying on centralized approaches for interconnection and resource discovery.

Addressing the aforementioned needs and exploring the alternatives, we propose an architecture that aspires to use bioinspired concepts to support autonomy through self-adaptation and self-organization in the field of ADS. Consequently, we incorporate that model into the design of a middleware capable of seamlessly interconnecting humans and heterogeneous machines, which evidently has the capacity to facilitate interactions among them. In an attempt to provide a universal, easily adoptable solution, we ascertain that the middleware possesses properties in its architecture that allow for a substantially high degree of extensibility and adaptability, when incorporated into an end-system supporting the desired vision of an IoT.

3. Proposed Architecture

The Emergent Distributed Bio-Organization (EDBO) is a model that was conceived during long term work on harnessing emergent phenomena in artificial distributed systems (ADS). Initial work on understanding the micro-macro causal links, along with ADS experience gained through a simpler version of EDBO, has led to the development of a disciplined framework for engineering emergence, and EDBO as an abstract distributed system model aiming to engineer emergent properties at the macroscopic level [14].

EDBO is based on the concept of focusing on properties and interactions at the microscopic level in order to allow desired properties to emerge at the global, or macroscopic, level. Previous bioinspired approaches tackle specific problems with limited scope while EDBO attempts a holistic approach while it addresses multiple operational issues of distributed systems.

The main problem that EDBO attempts to solve is the discovery of resources in an unstructured, fully decentralized, network, under varying conditions. The rationale of introducing emergent properties in a distributed system is to improve its operational efficiency and to eliminate the need for human configuration and management. EDBO is a continuation of previous work in distributed systems where the focus was on understanding how relationships among network nodes affect resource discovery overall [46]. Although it is a generic case study which represents distributed systems, more concrete studies, like file sharing networks, high performance computing, decentralized schemes of Web services, or sensor and monitoring networks, could potentially benefit from the results of this work.

While there is a diverse set of objectives that contemporary distributed systems are expected to satisfy, not all of them can be appropriately treated as macroscopic properties which could emerge from local interactions. The main objectives of EDBO are to achieve scalability, robustness, and availability in a distributed system without explicit engineering. Instead, by following an experiment-driven framework, various properties and behaviors are introduced as different hypotheses of achieving the aforementioned global objectives. Emergent properties can be utilized in the organization and maintenance of the network as well as the basis for improving the discovery of resources within the network.

The EDBO model is described at length in the following section, along with the biologically inspired concepts and properties that were employed as enabling mechanisms of the desired emergent behaviors.

3.1. Emergent Distributed Bioorganization. The scope of EDBO is limited to the top layer of the distributed systems paradigm. A generic unstructured distributed system model has been devised which will maintain a high level of availability, scalability, and robustness, under different operational conditions. This is achieved by introducing biologically inspired properties in the EDBO model as well as employing agent-based techniques in order to guide emergent phenomena in an ADS. The conceptual structure followed in the model is built in terms of a distributed application that provides resources to external users, rather than a distributed system that is composed of users (peers).

EDBO nodes are represented by agents referred to as *BioBots* (Figure 1(b)) which use two-way logical connections (relationships) to form an overlay network. Each BioBot has a limited number of relationships to other BioBots which are managed autonomously. In terms of functionality, a BioBot serves as a wrapper for a set of resources (abstracting data, functionality, and services) which are provided to user requests (queries).

A BioBot represents the core routing component which facilitates the propagation of queries through the network in an autonomous manner. BioBot behavior is based on several bioinspired heuristic mechanisms (elaborated in Section 3.2) that guide its decision making. The heuristics rely on the BioBot's internal state, the available relationship meta-data, and the current state of the environment, which in terms of the EDBO paradigm is referred to as *BioSpace*. This environment layer acts as a middleware between the BioBots and the underlying operating system and physical infrastructure (see Figure 1(a)), with different middleware instances running on different servers being interconnected in order to facilitate

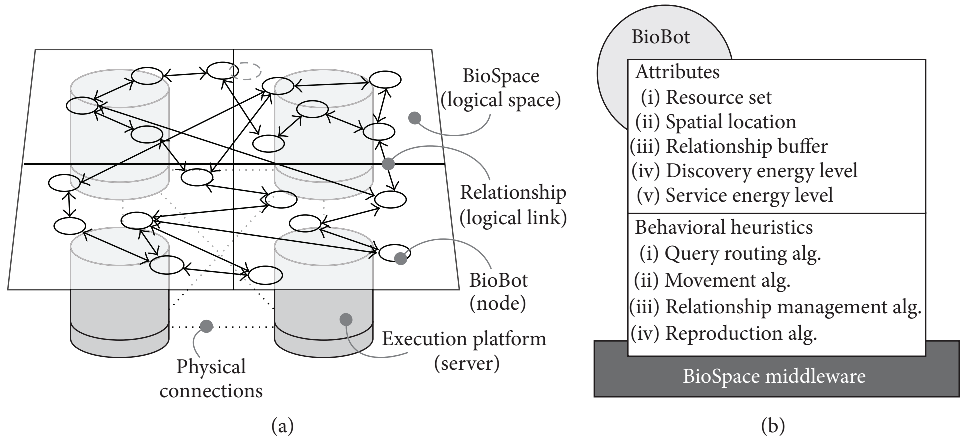


Figure 1: (a) Simplified overview of the BioSpace; (b) BioBot's core attributes and heuristics.

a unified environment and a singular spatial universe. In addition, the BioSpace facilitates several functional operations such as communication, offspring creation, and movement service.

3.2. EDBO: Biological Principles. EDBO utilizes a range of microscopic properties and interactions (at the BioBot level) in order to achieve global-level properties such as organization and optimization.

Autonomous Decision-Making. Autonomy is one of the fundamental principles of nature; biological entities usually act without explicit central or external control. Autonomous decision making in EDBO is realized through several algorithms that use local interaction and information to guide agent behavior.

Death and Birth Events through Energy Maximization. Energy can be seen as a major life force in nature, enabling biological entities to perform a variety of tasks. In EDBO, the concept of energy is portrayed as a BioBot attribute. More specifically, the model proposes two different energy types:

(i) *Discovery energy* is the primary indicator of BioBots' usefulness in facilitating the resource discovery process. The discovery energy is awarded to the BioBots that facilitate a successful query match. BioBots can spend discovery energy while performing various activities such as moving, exchanging messages, and reproducing.

(ii) *Service energy* denotes how popular or successful is a resource in the system. Service energy is allocated to each resource individually when the resource matches a query. Service energy is consumed each time query matching occurs.

In the natural world, biological organisms aspire to maximize the energy gain and minimize the energy loss as a vital principle for survival. In a similar fashion, energy maximization is a core principle of the EDBO model. BioBots which fail to maximize their energy and reach a critical low are dying (i.e., removed from the network), while successful agents are rewarded with the ability to reproduce. The latter can happen through single-parent replication, which results in identical copies, or through binary reproduction in which both parents' properties are combined to produce a new mixture of service and discovery characteristics.

Adaptation through Ad Hoc Selection. This is directly inspired by natural selection and fitness-based evolution. BioBot fitness is expressed through energy levels which clearly represent how fit or unfit an entity is, at any point during its life cycle. Death and birth events allow the network to continuously select the most successful BioBots according to the current network conditions.

3.3. EDBO: Operational Emergence. The main challenge in the EDBO model is to devise an effective and efficient way of discovering resources in an initially unstructured and fully decentralized network. Query matching is implemented through simple keyword comparison. Intuitively, query routing appears to be the deciding factor for the success of resource discovery. However, our hypothesis is that providing an appropriate network organization and resource population size is as important. The rationale behind this is that by imposing ad hoc overlay organization and adjustment of the number of BioBot/resource instances, the system will be able to maintain acceptable levels of query match success rate without proactive (well-informed) routing strategies that often impose high communication overheads. Towards this direction, the EDBO platform aims to utilize emergent behaviors in two main aspects of operation: the availability and scalability of resources and the organization and optimization of the network.

The scalability and availability of resources are an important aspect in ADS, since the query load imposed by the users often exhibits extreme fluctuations. Additionally, some of the resources are in higher demand (popular) than others (unpopular). In order to cope with these requirements, EDBO utilizes energy distribution as the main feedback mechanism that regulates the size of the BioBot population.

Since the amount of energy entering the system is regulated by the number of successfully resolved queries, in situations where there is a high query load the amount of available energy in the system will be increased. This allows for a larger BioBot reproduction rate which essentially scales up the number of available resource providers. Moreover, by using energy driven selection of parents (see Section 3.2) popular resources have higher chances of being reproduced. A decrease in the query load leads to an overall energy decrease, which in turn increases the BioBot death rate and reduces the number of birth events. Thus, the size of the BioBot population scales down.

Structured overlays are typically performing better than unstructured ones. In the EDBO this is tackled via an emergent overlay organization and connection (relationship) optimization. This is mainly inspired by the supernode architecture [47] which enables application of discovery mechanisms to improve discovery performance overall. In the supernode architecture, the overlay structure is logically divided into two layers: the global layer which is composed out of supernodes that enable global connectivity and the local cluster layer which facilitates local interconnection. In order to facilitate the ad hoc formation of supernode structures through local BioBot decisions, EDBO incorporates flexible relationship buffer size, different relationship acquisition strategies, and energy metadata. The selection of the superbots is based on the BioBots' discovery energy level, which in turn determines the total number of relationships (slots) that a BioBot can establish. BioBots with high discovery energy and a large number of relationships are granted the superbot status. Such BioBots allocate part of their connections to other superbots, while the rest of the connection slots are used to facilitate the connectivity with the nodes in the local cluster. BioBots in the local cluster optimize their relationships through evaluation of partners' spatial distance in order to ensure tight connectivity in the local cluster.

3.4. Realizing the EDBO Architecture for CPS. The proposed architecture aims to serve as a main infrastructure that will enable any authorized consumer to perceive the required sensor or other types of data as if connected to the nervous system of an organism. It provides a solution that enables utilization of emergent behaviors to achieve availability and scalability of resources and the organization and optimization of the network. The proposed middleware is a realization of the above-described research prototype network (EDBO) that achieves several self* properties using the bioinspired solution described above. Thus, it is composed of BioBots (logical nodes) that are realized in a BioSpace (middleware). BioBots are capable of communicating with CPS if in range, enabling bridging of the desired CPS and the middleware.

EDBO for CPS facilitates a plug and play solution which allows the addition of such a system to the network at anytime. By adding a new CPS (provider) to the proposed middleware, which is based on the EDBO architecture, a service will be automatically provided and discovery will be enabled in a fully decentralized manner. Consumers realized as compatible clients will be able to discover and then consume all the provided data from the connected CPS without the need of any central control. The abstract proposed architecture is depicted in Figure 2.

The following section details the design followed towards implementing the EDBO for CPS architecture as a viable solution for real-world wireless sensor networks.

4. Middleware Design

In order to fulfill its goal as a middleware, the design of the EDBO for CPS implementation allows for several desired properties to be present in the final system. To begin with, it is essential that an ADS, which can account for all the functionality and properties, must be developed as a multiagent system to enable autonomy and realize the ideas presented by the EDBO model (the BioSpace, the BioBot, and their interactions). Meanwhile, the core principles of component-based software engineering have to be realized for the middleware to be able to accommodate any future development; those are composability, substitutability, reusability, and extensibility [48]. Finally, it was discussed how a SOA is essential to tackle most of the common architectural and implementation issues that are expected of an IoT-enabling middleware.

Naturally, a suitable framework for the development of such a system is essential, with the following traits being of paramount importance: offering the ability to design with SOA principles, encompassing the much-desired properties of software components and, of course, providing the tools to develop a full-fledged MAS. The Jadex Active Components (JAC) framework has been chosen to this end, offering all of the above and additionally solving issues such as simulation support, security, as well as offering fast prototyping [49]. Finally, it must be mentioned that the nonfunctional properties expected of the EDBO middleware have been considered during its design. Specifically, software patterns such as the strategy, factory, observer, and adapters have been used heavily in order for the software to support a high degree of extensibility and at the same time maintain the highest possible degree of applicability to assorted scenarios.

The following subsections will detail how the proposed architecture elements depicted in Figure 2 were designed as a middleware by virtue of the chosen framework for its development. Firstly, details of the capabilities of the JAC framework are presented, followed by the design details of the BioSpace and BioBot abstractions as entities in a multiagent, distributed system. The section continues with a presentation of the conceptual architecture of the middleware and our approach towards solving the IoT issues, eventually concluding with a short presentation of the manner in which the various middleware components operate once deployed. A package overview of all individual components of the system is illustrated in Figure 3.

4.1. The Jadex Active Components Framework. The JAC project started out as Jadex, an extension over the JADE framework that aimed to utilize the BDI agent model and offer an effective and efficient implementation of it over JADE. Over time, it has grown to become a complete, separate

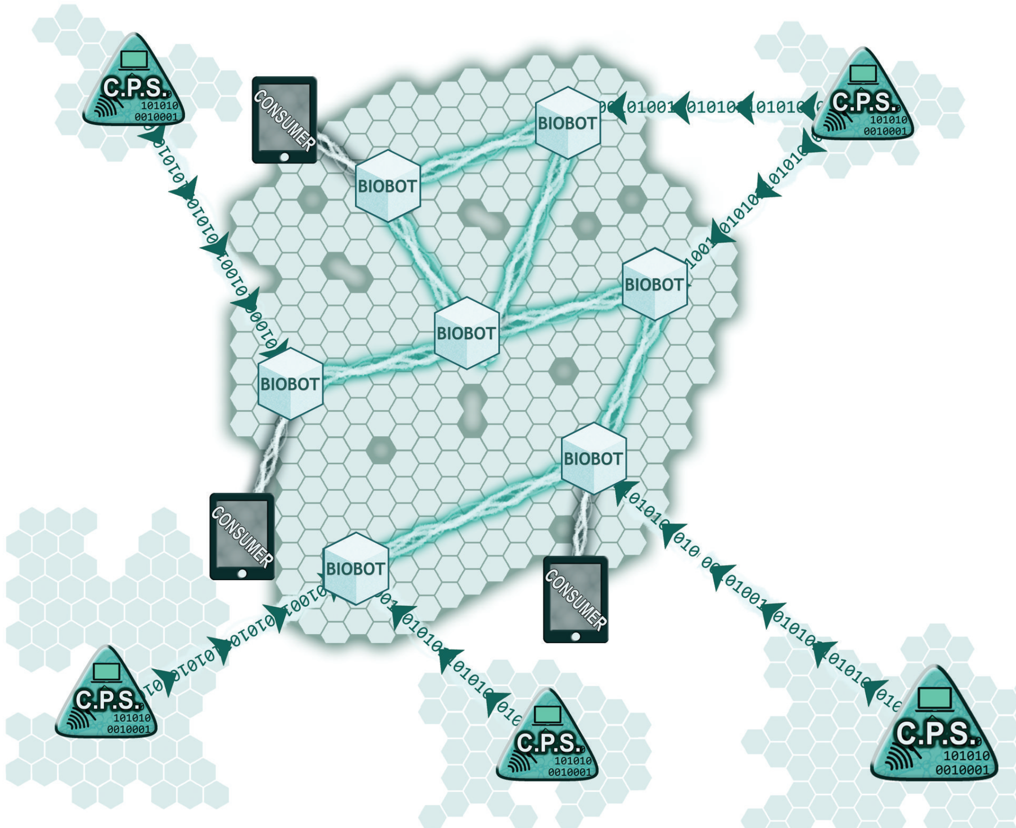


Figure 2: An abstract representation of the proposed architecture.

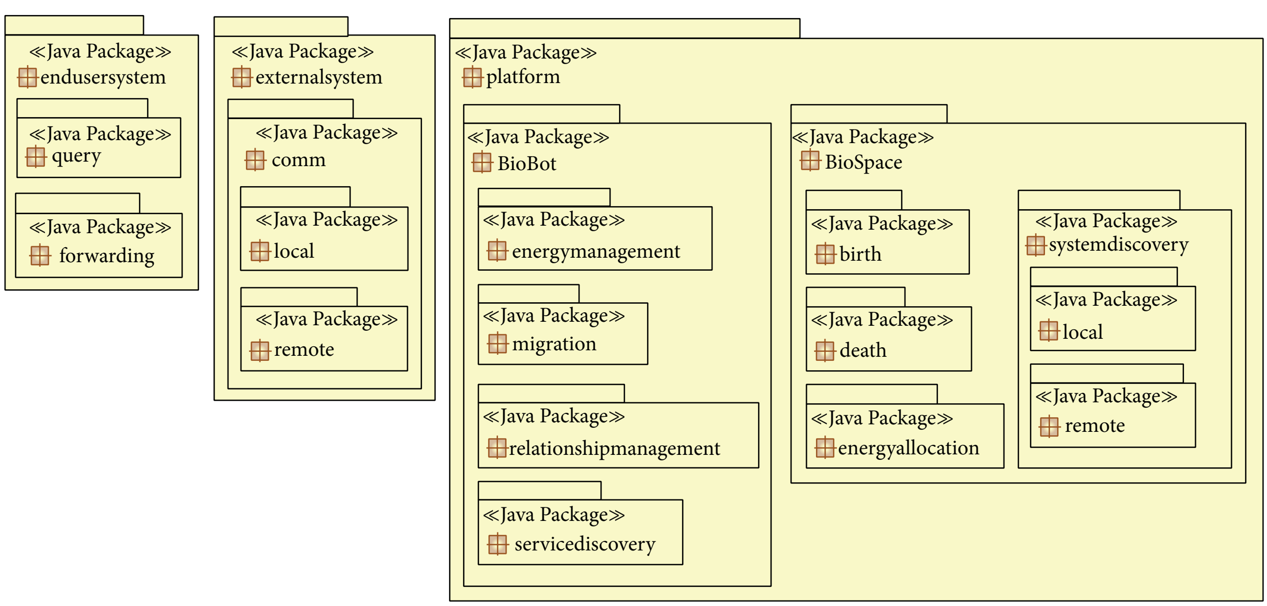


Figure 3: EDBO middleware package overview. The end-user system running at the nodes consuming services, the external system at the nodes providing services, and the platform on the nodes forming the EDBO overlay network.

framework for creating a MAS, or an ADS, that can be deployed on a number of different devices, provided that they can run a JVM, including a specialized, lightweight version of the framework that can be used for deployment of systems on Android devices. The framework does not offer only BDI agent implementation; in addition, it supports Plain Old Java Object (POJO) programming for any type of agent (e.g., model-based, reflex agents).

The main strength of the framework lies in the fact that it is created with a SOA in mind, with the categorization of implemented services as either required (an agent cannot function without this service available somewhere in the ADS) or provided (functionality that the agent possesses), thus simplifying the translation of the EDBO model to system architecture. Meanwhile, JAC offers asynchronous agent interaction using the future paradigm (see Figure 4) to facilitate invocation of agent provided services, which practically eliminates the need for the utilization of an ACL and messaging to simulate asynchronous agent interaction. This interaction is realized in an event-driven manner and hence covers the needs of an architecture that can support the IoT.

The most integral part of the JAC architecture is the Active Component (AC), which refers to the agent implementation encapsulated as a software component through several properties. Furthermore, the additional properties of the AC come from the SOA aspect of the framework and essentially constitute a separate module that attaches to the agent and is responsible for handling any service-related property or functionality the AC is qualified for. Finally, apart from any traditional services offered through an ADS developed with JAC, the framework is capable of exposing any service implemented for an AC as either a WSDL or RESTful service. The added value of this capability is the realization of an IoT as a WoT, with the latter functioning as either a wrapper of the system or complementing its provided services.

4.2. The BioSpace. In the model, the BioSpace is described as a logical, 2D space where the BioBots live and die. An important distinction that must be made at this point is that the model regards it as a single entity offering services to the BioBots (e.g., energy allocation and migration), but when translating the model into software the risk of employing centralized solutions is inevitable. This is why the software running on each system has been designed in such a manner that it can still provide BioSpace services to the BioBots on that particular system, without the need to resort to a centralized mechanism (e.g., a database with BioBot locations). This is achieved primarily by the abstraction of the migration service to merely provide the functionality of migrating (i.e., changing current BioBot logical space location), while the BioBots store that location individually instead.

The JAC framework classifies the services in the system as either required or provided. A BioBot consuming the services provided by the BioSpace should have to specify those services as required. But in a system adhering to the SOA principles, it is essential that these services are provided by someone; in the case of a JAC the provider is always an AC. Therefore, a very basic AC has been created with

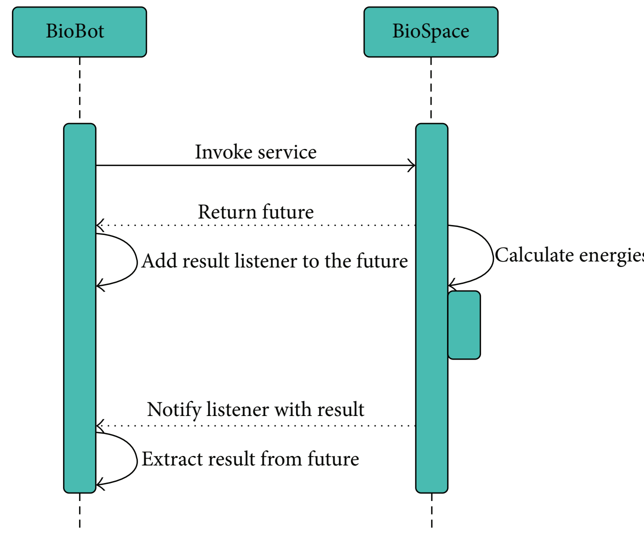


Figure 4: Sequence diagram depicting the future paradigm at work using the example of the Energy Allocation Service.

the sole purpose of offering the BioSpace services (specified as provided). In JAC services also have a scope, which refers to where they will be available in the ADS (e.g., platform, local network, globally). A platform scope is used to this end so that it is only accessible by the BioBots living on the current machine.

An example of this design can be seen in the following scenario: a BioBot is born and the BioSpace has to allocate service and discovery energies to it.

(1) The BioBot AC invokes the Energy Allocation Service of the BioSpace AC (assigns a listener to a future event).

(2) The BioSpace AC receives the invocation (creates the future event) and determines how much energy to allocate, and when ready it communicates it to the BioBot AC (sets the result of the future and notifies the listener).

(3) The BioBot AC retrieves the energy levels sent to it by the BioSpace AC (extracts the result wrapped within the future event) and begins its life functions.

Figure 4 provides an illustration of this process involving the JAC future paradigm (service-oriented, event-driven communication).

4.3. The BioBot. The EDBO model describes the BioBot as a very basic, reflexive agent: one that can simply react to queries it receives regarding service discovery (either from users or other BioBots) and relationship or reproduction requests by other BioBots. Its functionality is hence limited and it is clear that the three aforementioned services are specified as provided services, while the ones provided by the BioSpace AC are required by the BioBot AC to ensure that a BioBot cannot be born without a BioSpace to live in. The state of the BioBot, which it monitors as an agent, is distributed among the distinct services it provides. Specifically, monitoring of

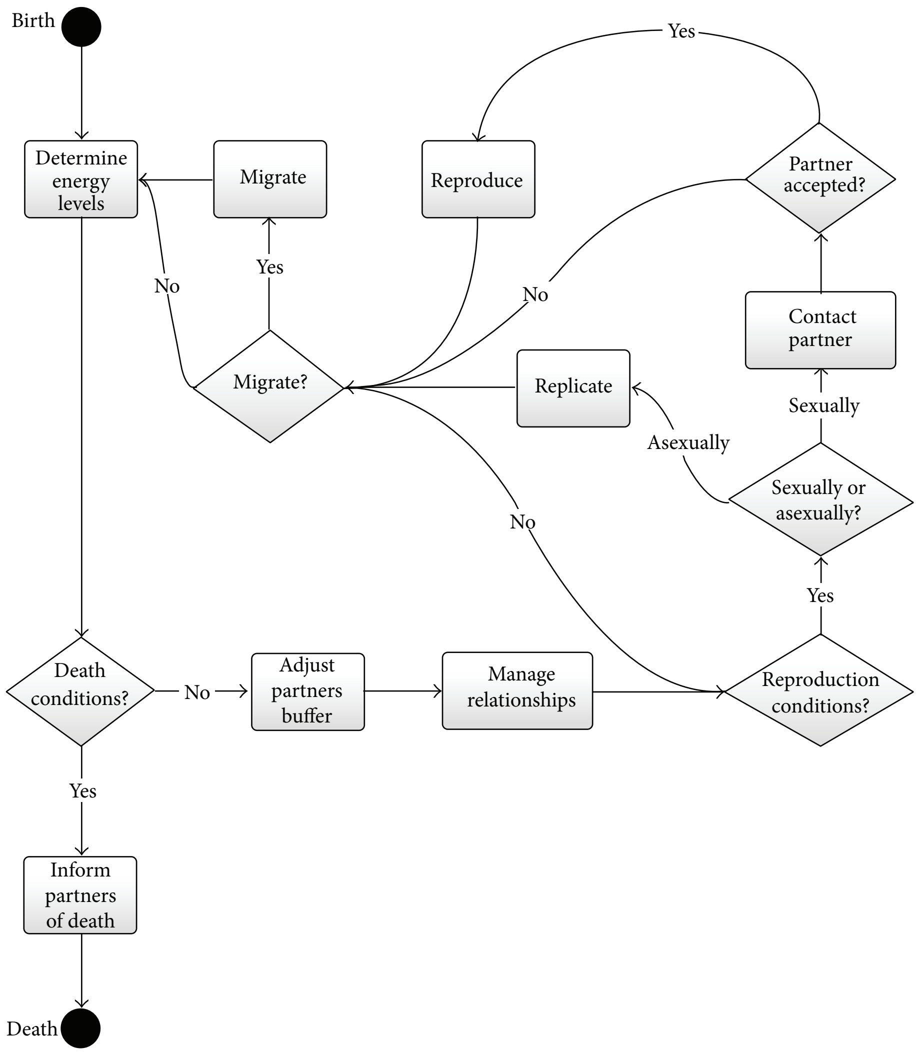


Figure 5: Flowchart demonstrating the life cycle of the BioBot.

energies is handled by the Energy Management service, monitoring of location by the migration service, monitoring of relationships and metadata on the partners by the Relationship Management service, and keywords related to the services the BioBot is aware of through the service discovery service. The observer software pattern has been applied to these services and the BioBot itself so that monitoring of its state can be carried out elegantly in an event-driven manner, in addition to having the capacity to be tailored to the administrator's needs.

Additionally, the JAC framework specifies three functions that can be used to abstract the behavior of an AC being created (the BioBot being born), its main process as expected of an agent (BioBot life cycle), with the last one describing what happens when the AC is being destroyed (BioBot dying). The life cycle of the BioBot, how it performs its life functions as designed for the middleware, is presented in Figure 5. The BioBot is designed to ask the BioSpace for energy allocation and introduction to another BioBot when it is born, ending relationships before dying and monitoring service levels and taking several actions regarding its bioinspired functions (migration, reproduction, replication, and relationships). As a deviation from the BioBot agent described in the model, the BioBot AC is capable of skipping one of its life functions: processing incoming messages and acting on them. This is attributed to the SOA design of the AC in the framework and the event-driven manner of AC interactions thanks to the future paradigm.

BioBot interactions take place as described above for the BioBot and BioSpace AC, and they are facilitated through two different services specified as provided services in the design: the Relationship Management service and the Service Discovery service. The former service is used to find a partner to keep for migration towards possible reproduction and forwarding queries, an action that the latter service is responsible for. This Relationship Management service has been designed with the strategy software pattern so that it is trivial to configure how BioBots choose their partners

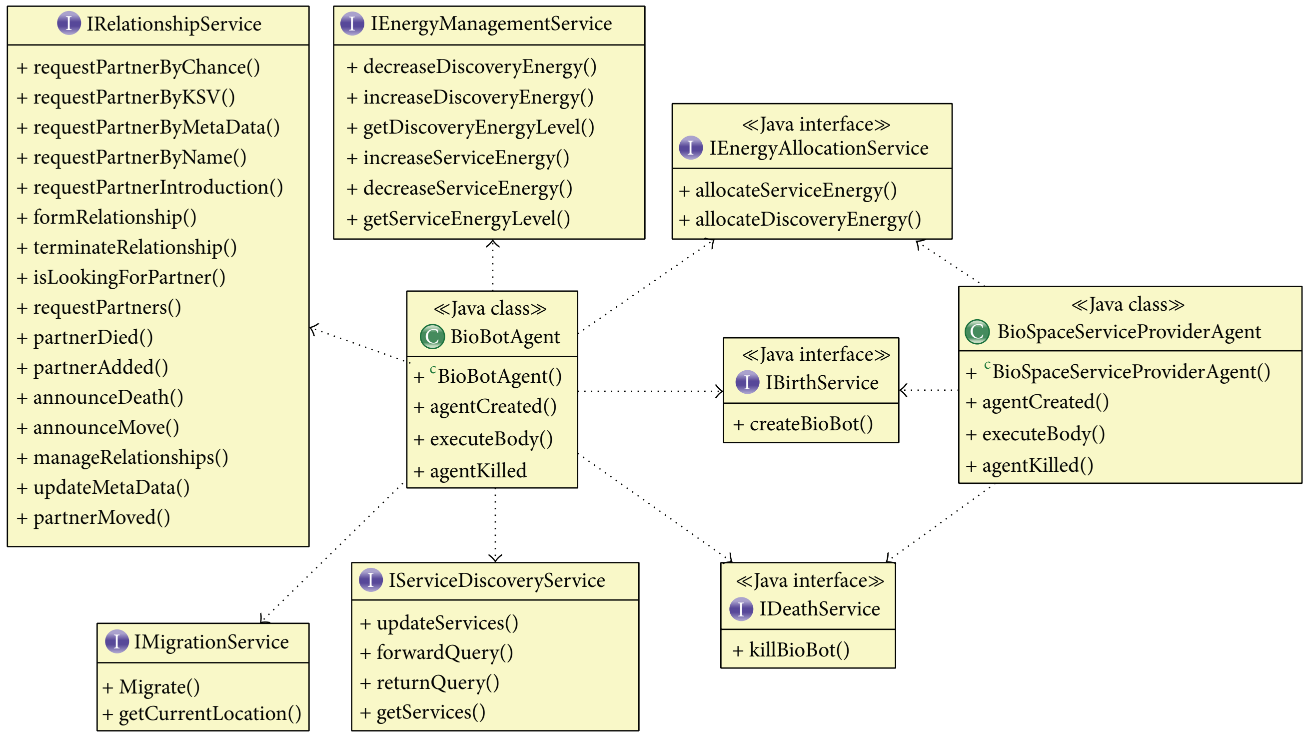


Figure 6: The BioBot and BioSpace agents, including their provided and required services.

according to the model: randomly, using keyword similarity or favoring better metadata. This allows for the effortless integration of better, more optimized strategies in the future.

BioBots are introduced to each other through either the BioSpace or another BioBot, which essentially means that the BioBot keeps a proxy of its partner so that they can communicate to them directly when needed (framework concept of AC External Access). In order to obtain this proxy, which for newly born BioBots is usually provided by the BioSpace AC, the design relies on the JAC framework capacity for awareness of other computer systems. This awareness mechanism is the core of the system that realizes it as an ADS, thanks to the various methods it provides for discovering systems connected to the overlay network: Broadcast, Multicast and IP Scanning for local networks, and Registry, Message and Relay for global networks. Awareness settings can be enabled and disabled at runtime, and many can be used together at the same time providing an excellent solution to ad hoc connectivity and reactivity to disaster scenarios. An overview of the BioBot agent and the services it provides, as well as the one it requires from the BioSpace, is illustrated in Figure 6.

4.4. Addressing IoT Concerns. At the very core of the IoT lie the issues of the unique identification of things, their interconnection, and interactions, as well as their virtualization. The latter solves the problems of device heterogeneity due to the fact that the virtual representation of the system is expected to be uniform. Naturally, the EDBO model has not been designed with all these issues in mind, providing

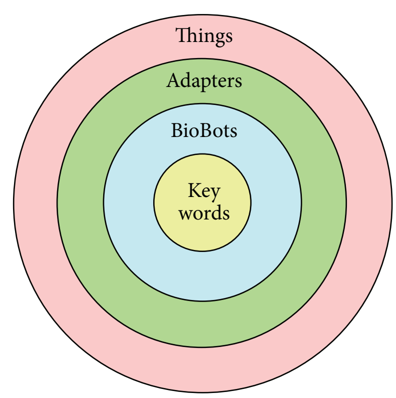


Figure 7: The layered architecture of the EDBO middleware.

the mere abstraction of a keyword to represent the functionality a thing possesses. Through its layered architecture (Figure 7) and the exploitation of the mechanisms available by the JAC framework, the design of the EDBO middleware provides the means to leverage the aforementioned issues. Each layer of this architecture will be discussed in this section.

The outermost layer named *Things* represents the software running on the end-point, which can be a single device, an end-user, or even a complex CPS. Two systems have been developed to support different roles for end-point devices: one for "things" offering a service and one for a human end-user attempting to search for a service. The design is modular and loosely-coupled; therefore, the appropriate libraries can be used in any system that aspires to provide a different type of user interface with the core of the middleware. This layer fully supports a WoT implementation because the services, the ones which realize this interface, have been exposed as RESTful services and can thereby be accessed by any system that is capable of RESTful communication. This can be a simple mobile phone app (the JAC framework provides extremely lightweight libraries for deployment of the system on Android devices), a Web application, or a server gathering data from end-point devices, among many other possibilities. The API of this functionality is detailed in the documentation of the middleware. As such, the ultimate purpose of this layer is to serve as the first step in achieving the interaction among "things," one of the core issues IoT implementations ought to address.

In the onion-layered architecture of the EDBO middleware, the layer *Things* communicates directly with the layer *Adapters*. Whereas the former aimed to facilitate interaction at a higher level, the latter aims to address the problem of the lower-level interoperability, taking into account the expected high degree of heterogeneity among "things." When a user or a system initiates an action, the information provided by the varied forms of end-point software on the outer layer has to be communicated to the inner layer of *BioBots*. Nonetheless, the BioBots can only understand a predefined format for data communicated to them; therefore, translation is needed. This second layer provides the means to solve this translation problem and achieve interoperability. The adapter software design pattern has been employed to this end, thus allowing developers to extend the provided classes and create their own adapters as needed for translating the information from their devices or CPS into an EDBO service. Already provided are sample adapters that can translate data to and from Windows INI, XML, and JSON formats. Essentially, the sheer flexibility of adapter approach provides the means to achieve the much needed virtualization of a human user or a system in any IoT scenario.

Having solved the problem of interoperability and interaction among "things" with the 2 outer layers, the third layer attempts to solve the interconnection of things. This layer is now provided by the core of the EDBO model: the bioinspired approach that creates the overlay network of the end-point systems and facilitates their communication. The BioBots are always aware of who their partners are, what services they can offer, and which partners to contact to try and discover a service they cannot offer themselves. The Service Discovery and Relationship services shoulder the majority of the workload to this end, which is solving the problem of interconnection in an IoT implementation, in conjunction with the awareness mechanisms that the JAC framework provides. The uniqueness in the case of the EDBO middleware is that the contribution of this layer is not only limited to the interconnection of devices, rather it extends to the provision of autonomous operation via self-organization, not a strict requirement for an IoT application but an extremely beneficial property.

The innermost layer is that of *Keywords*, the abstraction currently used to represent how the BioBots determine if the search term supplied by a thing matches a term they are aware of. This layer does not represent some form of a centralized repository of services, rather it aspires to provide some sort of very primitive semantics to the middleware. Future work is planned to replace this layer with enhanced semantics, if possible with an extended ontology. The unique identification of the things connected to the network is achieved in part thanks to this layer, but also in part due to how the External Access proxies of platforms connected to the rest of the network are handled by the underlying awareness mechanisms of the JAC framework.

4.5. Middleware Operation. The operation of the EDBO middleware in a sample deployment scenario can provide an insight into some of its capabilities. The walkthrough presented herein assumes a scenario in the domain of health care for elderly patients. In this scenario, the patient possesses a health sensor platform to record some data that that their personal doctor would like to monitor and the history of which should be recorded in the hospital server for processing by expert systems. Figure 8 illustrates how these actors and systems are interconnected with the EDBO middleware deployment.

The two application servers run the core of the middleware, each with a BioSpace and BioBots to create the overlay network (link 1, Figure 8) that end-point systems can connect to and offer or consume services. The health sensor platform runs software created by the manufacturer; however, they have incorporated the System Discovery libraries from EDBO to connect with BioBots (link 2, Figure 8) and they have also implemented an adapter to translate the data produced by their platform (e.g., the XML adapter) into EDBO services. The android phone possessed by the doctor runs a simple, lightweight app that periodically checks for the data of their patients by using the RESTful API of a specific BioBot (link 4, Figure 8). The JSON adapter is used to translate the data from the EDBO service into the format that the app can process to produce alerts as needed. Finally, the server runs the already provided EDBO External System EDBO software to connect with the BioBots (link 3, Figure 8) and hence acts as a service consumer in this case by getting the data from the health sensor platform at regular intervals. It implements a new adapter that parses EDBO service data and maps them onto fields for a MySQL database.

This scenario depicts only two EDBO middleware systems to serve as the backbone of the overlay network on nodes, but this could be extended to feature any number of platforms needed to satisfy the area coverage needs of the specific IoT scenario. Whenever a new EDBO middleware platform is instantiated on an application server, it will attempt to use all of the allowed platform awareness mechanisms available in order to find an entry point to the network (e.g., local area broadcast and IP message passing but not IP scanning). Once it has formed a relationship with another BioBot, it can be conversed with during service discovery and it can now offer any services that have been published on that application server.

Having showcased the formulation of an IoT with the two outer layers of the EDBO architecture above (Things and Adapters, Figure 7), the service discovery capabilities

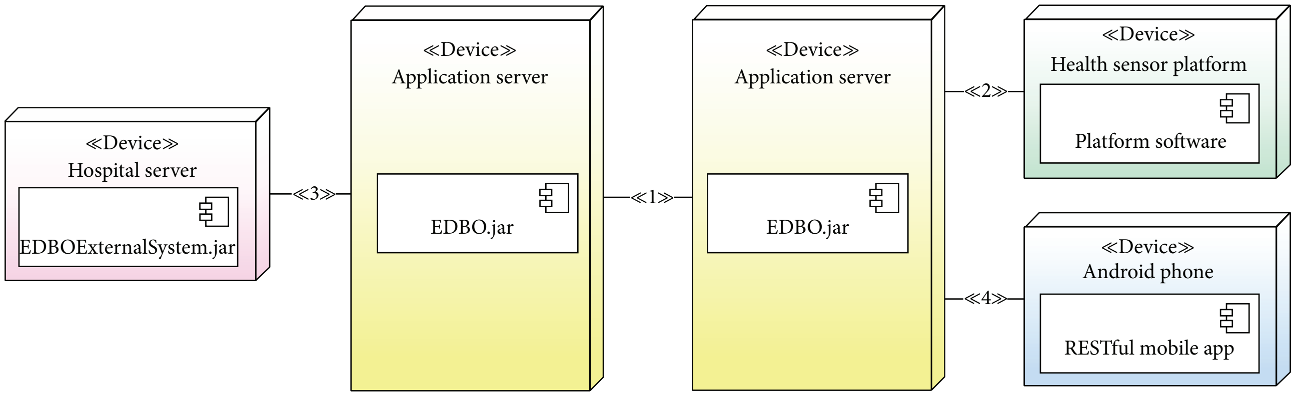


Figure 8: A sample deployment scenario for the EDBO middleware.

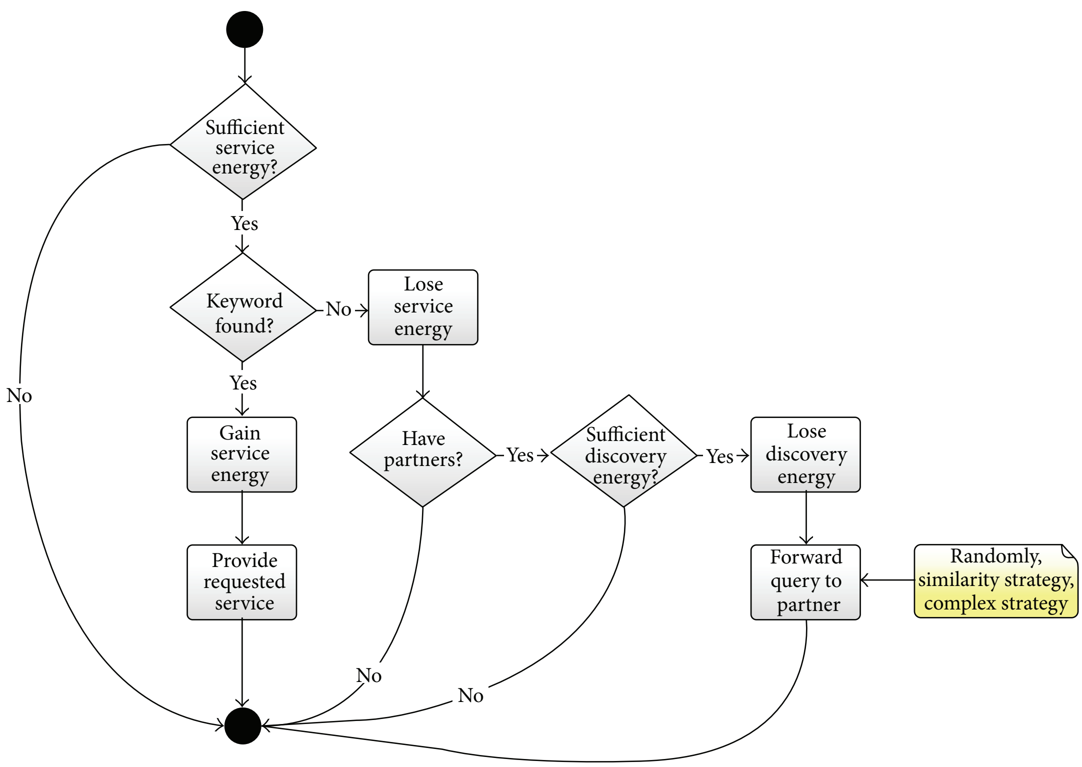


Figure 9: Flowchart depicting the decentralized query forwarding mechanism.

of the middleware with the inner layers remain to be seen (BioBots, Keywords, Figure 7). This process takes place in the case at hand when the mobile phone app is planned to update the latest data on the patients being monitored (e.g., every 10 minutes) and when the server collects data for processing (e.g., every 1 hour). In the case of the android app, the BioBot will receive the query on the RESTful wrapper of the service, process it, and immediately reply with the requested service data since they are readily available. On the other hand, the hospital server sends the request to the platform it is connected to but the BioBot receiving that query cannot satisfy it itself. Consequently, it forwards the query to their partner who can provide the requested data. The flow of this discovery process as the BioBots perform it in each case is outlined in Figure 9. The energy level fluctuations presented there trigger the BioBot life cycle decision making process (Figure 5) and lead to self-organization.

5. Middleware Evaluation

The EDBO model promises self-adaptivity and self-optimization in an ADS, properties that need to be verified during the transition from model to software. In order to ascertain that the EDBO middleware demonstrates the capabilities offered by the model, extensive testing has taken place continuously throughout its iterative development process, which was in turn followed by an elementary evaluation of its capabilities. Both of these tasks have been planned since the beginning of the development process and have been facilitated by the functionality offered by the JAC framework, and specifically through the Jadex Control Center (JCC) software offering a test center and through the integrated logging mechanisms for the ACs.

The approach towards testing has been carried out in two ways: (a) unit testing for each component developed and (b) integration testing when it was encompassed in the rest of the system. The JCC, through its test center, offers a frontend for executing JUnit-like, component-based test cases for each AC that has been developed (in this case the BioBot and BioSpace agents). As the main guideline for developing these test cases served a formal model of the BioBot implemented as an X-machine, which has been developed as part of the EDBO model. The X-machine represents all possible states and transitions among them taking place throughout the lifecycle of the BioBot and offers a formal testing methodology. Therefore, it was elementary to map these states and transitions onto the JUnit test cases needed to verify and validate BioBot behavior through its functions and the functions of its services (both provided and required). The BioSpace service provider also had JUnit test cases covering its functionality, although no formal model existed to guide the process (no life functions to perform).

The evaluation of the middleware has been focused on investigating the capabilities of the middleware towards realizing a self-adaptive, self-optimizing ADS. The optimal approach to determining the behavior of an ADS developed with the EDBO middleware was deigned to be the runtime evaluation in the form of a case study comparing results of a system with no self-organization capabilities to one that possesses them. Past results on the evaluation of the model have shown that the most optimal strategy out of the three currently formulated and implemented is that of the complex processing of the metadata. This is the case where BioBots tend to form meaningful relationships in the long term (self-optimization) and are capable of handling user behavior fluctuations by reevaluating current relationships and reproducing or replicating (self-adaptive). Consequently, the middleware was configured to use two different strategies for the same case and compare the results: the complex strategy and the random strategy.

Evaluation was carried out through the facilities provided by the JAC framework. Specifically, each AC has an inherent logging mechanism that can be activated and its entries can be viewed at real-time during execution via the component viewer offered by the JCC. Additionally, the design of the system enforces the observer pattern and hence a custom implementation of the abstract observer provided with the middleware offers the means to record all important data from the state changes and interactions of each BioBot. A parser has been developed to read and review the resulting logs, producing information on the data sought after to determine the properties of the system. The data gathered regarded response accuracy (how many queries were satisfied on average) and response delay (average number of hops for each query). These values were measured at the very beginning of runtime and compared to a few moments later (short-term adaptivity) and nearing the end of runtime (long-term optimization), in order to determine if the desired properties emerged in the system. As such, these metrics constitute the Key Performance Indicators (KPIs) for the purposes of this quantitative evaluation: determining efficiency of the system over the passage of time (totally unstructured origins, adaptation, and organized system in the end).

Table 1: Response accuracy and delay regarding the selected service for the runtime evaluation scenario using the random and complex (metadata) strategies.

Time frame	Avg. response accuracy		Avg. response delay (hops)	
	Random	Metadata	Random	Metadata
After 5 minutes	60.25%	64.09%	2.921	2.896
After 20 minutes	57.92%	71.23%	3.057	2.842
After 50 minutes	58.51%	89.56%	3.013	2.439

In order to determine how the system can adapt to user behavior, the evaluation scenario required the development of a querying function that sends various keywords to the BioBots, emphasizing a select keyword at the beginning (to establish a baseline), again after a little time during runtime and near the end again. Naturally, this evaluation has been carried out as a demonstration of the system using an elementary case study that could simulate the following scenario: an ADS of weather sensors. The system is deployed over several regions and the end-users can check the weather for each region. As an example, after the system has just been deployed and initialized, a hailstorm affecting crops in one of the regions prompts users trying to check on the status of their crops to request the data from that region primarily. The system thus ought to adapt to this sudden need and try to prepare for future needs. After this initial influx of requests for this region, users start to check on other regions for a significant time period to determine if they were also affected. After this interval of normalized activity, users decide to check again on the region suffering from the hailstorm to determine its current state. At this point, the system should display its self-organization capabilities by having evolved in such a manner that requests towards this “popular” query search can be satisfied faster and with higher accuracy.

The results gathered from the evaluation can be found in Table 1. Positive results on response accuracy demonstrate that the system manages to adapt to user behavior in the short-term, which may become even more significant in the long-term as the system optimizes itself to serve more queries similar to the ones it received at the beginning and the first half of its runtime. Delay results did not provide much information regarding short-term capabilities of the system, which may be attributed to the low initial energy levels of BioBots restricting more partners. Nonetheless, there were some changes towards the end of the evaluation hinting at the possibility of self-organization considering that metric, too.

As expected, the random strategy produced no positive results whatsoever regarding self-organization. On the contrary, it appears that random partner selection leads to BioBots having “bad” relationships and wasting energy on forwarding queries aimlessly. Unlike the evaluation results of the model, no superbots have been observed, which is attributed to the limited resources available for evaluating an ADS at runtime as compared to simulation results. Finally, a criticism of the model, or at least for its initial configurations regarding energy levels, stemmed from the fact that several BioBots that could not satisfy queries (or forward them) died after some time, which prohibited access to services they offered; in one case, the single BioBot offering a particular service died, effectively losing all access to that service in the network.

Overall, the system demonstrates the capacity to address the issue of load balancing and, moreover, it can do so without any human intervention, proving the self-organization and self-adaptivity capabilities of the system. There is a differentiation here with the traditional approaches to load-balancing that similar solutions employ, where several factors are measured over time during network operation and then settings are adjusted to account for expected load balance depending on network size and connected users, specific hardware settings of nodes such as remaining energy [50], or knowledge of the general structure of the network such as the very efficient SAAS-RWSNs [51]. In such cases this happens when these values reach the appropriate point that the new settings have to be applied, and it further demands the existence of a centralized mechanism that keeps all these data of all network nodes so that they can be accessed by every node. Contrary to this process, the EDBO middleware is capable of adapting to the changes in its operational environment in a continuous rather than in a discrete manner, and also without any knowledge on network structure either. The advantage in this case would be the example of 990 users connected at the same time experiencing a slow response because the next optimization is to take place when the number goes above 1000; in the case of the EDBO, the system should have adapted gradually to better support these users already.

6. Conclusions

The vision of an Internet of Things is a promising approach to bringing together the two worlds of cyber and physical, especially imagining the gains in accessibility and applicability of sensor networks connected to it. By harnessing the power of component and agent-based services, as well as object-oriented approaches, we designed an extensible middleware capable of realizing a wide range of scenarios with that vision in mind. Its strength lies in the bioinspired model at the core of its design, one that is capable of offering autonomous operation to such an artificial distributed system of interconnected and interacting cyber or physical entities, whether machines or humans. Self-organization features have emerged in the system thanks to short-term adaptivity to user behavior, as well as long-term self-optimization for expected user requests.

Future work to the EDBO middleware is planned to add features and functionality and to optimize certain aspects of its operations. As a first step, work is underway on developing a data annotation framework to enable the seamless integration of CPS to the core architecture of BioBots (the innermost layer of our architecture, Figure 7 earlier). Discarding the primitive keyword matching functionality, the successful integration of semantics will provide a more refined architecture that allows for better automation of the introduction of new sensor providers to the system. Several standards for data representation and communication for varied devices are being investigated, in an effort to meet this goal and enhance the virtualization aspect, and hence the interoperability, of the things communicating via the EDBO middleware. We also aim to work on establishing a security infrastructure that goes beyond the basic security issues already solved in the current realization of our proposed distributed architecture.

The issue of BioBots dying and losing access to services will be investigated through multiple future evaluations on various case studies and assorted size networks. This one issue has already sparked research into alternatives to the fundamental biofunction of death the BioBots possess and its conversion into temporary dormancy. The product of the evaluation work is expected to produce a policy for optimal energy level configurations for the BioBots and offer further insight into the scalability in IoT applications. Furthermore, the results of these evaluations will be compared to results of similar systems in specific applications in order to develop a better understanding of the overall potential of EDBO and the areas that require further optimization, its standing among similar systems, and the possible integration of ideas from those systems with the EDBO ecosystem.

We currently apply the proposed solution in this paper as the core architecture for health monitoring [52], aiming to evaluate it in a real case study and at the same time refine it in order to enable seamless, and automate as much as possible, integration of sensors to the system. Research is also expanding to the domain of computational acoustic scene analysis, so as to prove the capacity of the system to support different and more demanding applications, but also as an opportunity to investigate the capacity for mapping BioBot life functions onto real-life actions for connected devices and CPS. One example could be the mapping of BioBot migration onto sensor motor actions towards seeking a potentially better place for receiving more accurate auditory data. Another is the example of adjusting sensor energy levels in accordance with BioBot energy levels to fine-tune system autonomy through prolonged battery life. Finally, the introduction of new types of energies will be investigated, due to the research on EDBO that already indicates the potential emergence of more self-* properties that could benefit future IoT applications.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

The authors are cofinanced by Greece and the European Union (European Social Fund) O.P. “Human Resources Development,” NSFR 2007–2013.

References

[1] K. Ashton, "That 'Internet of Things' thing: in the real world, things matter more than ideas," *RFID Journal*, vol. 22, pp. 97–114, 2009.

[2] F. Mattern and C. Floerkemeier, "From the internet of computers to the Internet of Things," in *From Active Data Management to Event-Based Systems and More*, vol. 6462 of *Lecture Notes in Computer Science*, pp. 242–259, Springer, Berlin, Germany, 2010.

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

[4] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of things: vision, applications and research challenges," *Ad Hoc Networks*, vol. 10, no. 7, pp. 1497–1516, 2012.

[5] P. Barnaghi, W. Wang, C. Henson, and K. Taylor, "Semantics for the internet of things: early progress and back to the future," *International Journal on Semantic Web and Information Systems*, vol. 8, no. 1, pp. 1–21, 2012.

[6] S. Alam, M. M. R. Chowdhury, and J. Noll, "SenaaS: an event-driven sensor virtualization approach for internet of Things cloud," in *Proceedings of the 1st IEEE International Conference on Networked Embedded Systems for Enterprise Applications (NESEA '10)*, pp. 1–6, IEEE, Suzhou, China, November 2010.

[7] D. Uckelmann, M. Harrison, and F. Michahelles, "An architectural approach towards the future Internet of Things," in *Architecting the Internet of Things*, pp. 1–24, Springer, Berlin, Germany, 2011.

[8] S. Evdokimov, B. Fabian, S. Kunz, and N. Schoenemann, "Comparison of discovery service architectures for the internet of things," in *Proceedings of the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC '10)*, pp. 237–244, IEEE, Newport Beach, Calif, USA, June 2010.

[9] A. Katasonov, O. Kaykova, O. Khriyenko, S. Nikitin, and V. Y. Terziyan, "Smart semantic middleware for the internet of things," in *Proceedings of the 5th International Conference on Informatics in Control, Automation and Robotics (ICINCO-ICSO '08)*, vol. 8, pp. 169–178, Funchal, Portugal, May 2008.

[10] L. Roalter, M. Kranz, and A. Möller, "A middleware for intelligent environments and the Internet of Things," in *Ubiquitous Intelligence and Computing*, vol. 6406 of *Lecture Notes in Computer Science*, pp. 267–281, Springer, Berlin, Germany, 2010.

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

[12] L. Tan and N. Wang, "Future Internet: the internet of things," in *Proceedings of the 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE '10)*, pp. V5376–V5380, Chengdu, China, August 2010.

[13] T. Erl, *Service-Oriented Architecture: Concepts, Technology, and Design*, Pearson Education India, 2005.

[14] G. Eleftherakis, O. Paunovski, K. Rousis, and A. J. Cowling, "Emergent distributed bio-organization: a framework for achieving emergent properties in unstructured distributed systems," in *Intelligent Distributed Computing VI*, G. Fortino, C. Badica, M. Malgeri, and R. Unland, Eds., vol. 446 of *Studies in Computational Intelligence*, pp. 23–28, Springer, Berlin, Germany, 2013.

[15] R. Klauck and M. Kirsche, "Bonjour Contiki: a case study of a DNS-based discovery service for the Internet of Things," in *Ad-hoc, Mobile, and Wireless Networks*, vol. 7363 of *Lecture Notes in Computer Science*, pp. 316–329, Springer, Berlin, Germany, 2012.

[16] D. Guinard, V. Trifa, S. Karnouskos, P. Spiess, and D. Savio, "Interacting with the SOA-based internet of things: discovery, query, selection, and on-demand provisioning of web services," *IEEE Transactions on Services Computing*, vol. 3, no. 3, pp. 223–235, 2010.

[17] D. Zeng, S. Guo, and Z. Cheng, "The web of things: a survey," *Journal of Communications*, vol. 6, no. 6, pp. 424–438, 2011.

[18] Z. Shelby, "Embedded web services," *IEEE Wireless Communications*, vol. 17, no. 6, pp. 52–57, 2010.

[19] D. Guinard, I. Ion, and S. Mayer, "In search of an Internet of Things service architecture: REST or WS-*? A developers' perspective," in *Mobile and Ubiquitous Systems: Computing, Networking, and Services*, vol. 104 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pp. 326–337, Springer, Berlin, Germany, 2012.

[20] E. A. Lee, "Cyber physical systems: design challenges," in *Proceedings of the 11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing (ISORC '08)*, pp. 363–369, IEEE, Orlando, Fla, USA, May 2008.

[21] F.-J. Wu, Y.-F. Kao, and Y.-C. Tseng, "From wireless sensor networks towards cyber physical systems," *Pervasive and Mobile Computing*, vol. 7, no. 4, pp. 397–413, 2011.

[22] E. A. Lee, "Computing foundations and practice for cyber-physical systems: a preliminary report," Tech. Rep. UCB/EECS-2007-72, University of California, Berkeley, Berkeley, Calif, USA, 2007.

[23] R. Al Ali, T. Bures, I. Gerostathopoulos et al., "DEECo: an ecosystem for cyber-physical systems," in *Proceedings of the 36th International Conference on Software Engineering*, pp. 610–611, ACM, Hyderabad, India, June 2014.

[24] J. C. Eidson, E. A. Lee, S. Matic, S. A. Seshia, and J. Zou, "Distributed real-time software for cyber-physical systems," *Proceedings of the IEEE*, vol. 100, no. 1, pp. 45–59, 2012.

[25] P. Horn, "Autonomic computing: IBM's perspective on the state of information technology," *Computing Systems*, vol. 15, pp. 1–40, 2001.

[26] A. G. Ganek and T. A. Corbi, "The dawning of the autonomic computing era," *IBM Systems Journal*, vol. 42, no. 1, pp. 5–18, 2003.

[27] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[28] B. Biskupski, J. Dowling, and J. Sacha, "Properties and mechanisms of self-organizing MANET and P2P systems," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 2, no. 1, article 1, 2007.

[29] D. Kurzyniec, T. Wrzosek, D. Drzewiecki, and V. Sunderam, "Towards selforganizing distributed computing frameworks: the H_2O approach," *Parallel Processing Letters*, vol. 13, no. 2, pp. 273–290, 2003.

[30] E. Di Nitto, C. Ghezzi, A. Metzger, M. Papazoglou, and K. Pohl, "A journey to highly dynamic, self-adaptive service-based applications," *Automated Software Engineering*, vol. 15, no. 3-4, pp. 313–341, 2008.

[31] A. J. Chakravarti, G. Baumgartner, and M. Lauria, "The organic grid: self-organizing computation on a peer-to-peer network,"

IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans., vol. 35, no. 3, pp. 373–384, 2005.

[32] H. Schmeck, "Organic computing—a new vision for distributed embedded systems," in *Proceedings of the 8th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC '05)*, pp. 201–203, IEEE, May 2005.

[33] P. Boonma, P. Champrasert, and J. Suzuki, "BiSNET: a biologically-inspired architecture for wireless sensor networks," in *Proceedings of the International Conference on Autonomic and Autonomous Systems (ICAS '06)*, 54, IEEE, Silicon Valley, Calif, USA, July 2006.

[34] K. Aberer, M. Hauswirth, and A. Salehi, "Middleware support for the 'Internet of Things'," in *5th GI/ITG KuVS Fachgespräch "Drahtlose Sensornetze"*, LSIR-CONF-2009-017, Stuttgart, Germany, 2006.

[35] J. I. Vazquez, A. Almeida, I. Doamo, X. Laiseca, and P. Orduña, "Flexeo: an architecture for integrating wireless sensor networks into the Internet of Things," in *3rd Symposium of Ubiquitous Computing and Ambient Intelligence 2008*, vol. 51 of *Advances in Soft Computing*, pp. 219–228, Springer, Berlin, Germany, 2009.

[36] O. Dagdeviren, I. Korkmaz, F. Tekbacak, and K. Erciyes, "A survey of agent technologies for wireless sensor networks," *IETE Technical Review*, vol. 28, no. 2, pp. 168–184, 2011.

[37] C.-L. Fok, G.-C. Roman, and C. Lu, "Agilla: a mobile agent middleware for self-adaptive wireless sensor networks," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 4, no. 3, article 16, 2009.

[38] A. Boulis, C.-C. Han, R. Shea, and M. B. Srivastava, "SensorWare: programming sensor networks beyond code update and querying," *Pervasive and Mobile Computing*, vol. 3, no. 4, pp. 386–412, 2007.

[39] T. Harman, J. Padget, and M. Warnier, "A multi-layered semantics-ready sensor architecture," in *Proceedings of the 3rd International Workshop on Agent Technology for Sensor Networks (ATSN '09)*, Budapest, Hungary, May 2009.

[40] J. W. Hui and D. Culler, "The dynamic behavior of a data dissemination protocol for network programming at scale," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, pp. 81–94, ACM, November 2004.

[41] T. Liu and M. Martonosi, "Impala, a middleware system for managing autonomic, parallel sensor systems," in *Proceedings of the 9th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP '03)*, vol. 38, pp. 107–118, San Diego, Calif, USA, June 2003.

[42] R. Willett, A. Martin, and R. Nowak, "Backcasting: adaptive sampling for sensor networks," in *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks (IPSN '04)*, pp. 124–133, ACM, Berkeley, Calif, USA, April 2004.

[43] G. Mainland, D. C. Parkes, and M. Welsh, "Decentralized, adaptive resource allocation for sensor networks," in *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*, pp. 315–328, USENIX Association, 2005.

[44] P. Padhy, R. K. Dash, K. Martinez, and N. R. Jennings, "A utility-based sensing and communication model for a glacial sensor network," in *Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS '06)*, pp. 1353–1360, ACM, May 2006.

[45] J. Kho, A. Rogers, and N. R. Jennings, "Decentralised adaptive sampling of wireless sensor networks," in *Proceedings of the 1st International Workshop on Agent Technology for Sensor Networks*, pp. 55–62, Honolulu, Hawaii, USA, May 2007.

[46] O. Paunovski, G. Eleftherakis, K. Dimopoulos, and T. Cowling, "Evaluation of a selective distributed discovery strategy in a fully decentralized biologically inspired environment," *Information Sciences*, vol. 180, no. 10, pp. 1865–1875, 2010.

[47] B. Beverly Yang and H. Garcia-Molina, "Designing a super-peer network," in *Proceedings of the 19th International Conference on Data Engineering*, U. Dayal, K. Ramamritham, and T. M. Vijayaraman, Eds., pp. 49–60, IEEE Computer Society, Bangalore, India, March 2003.

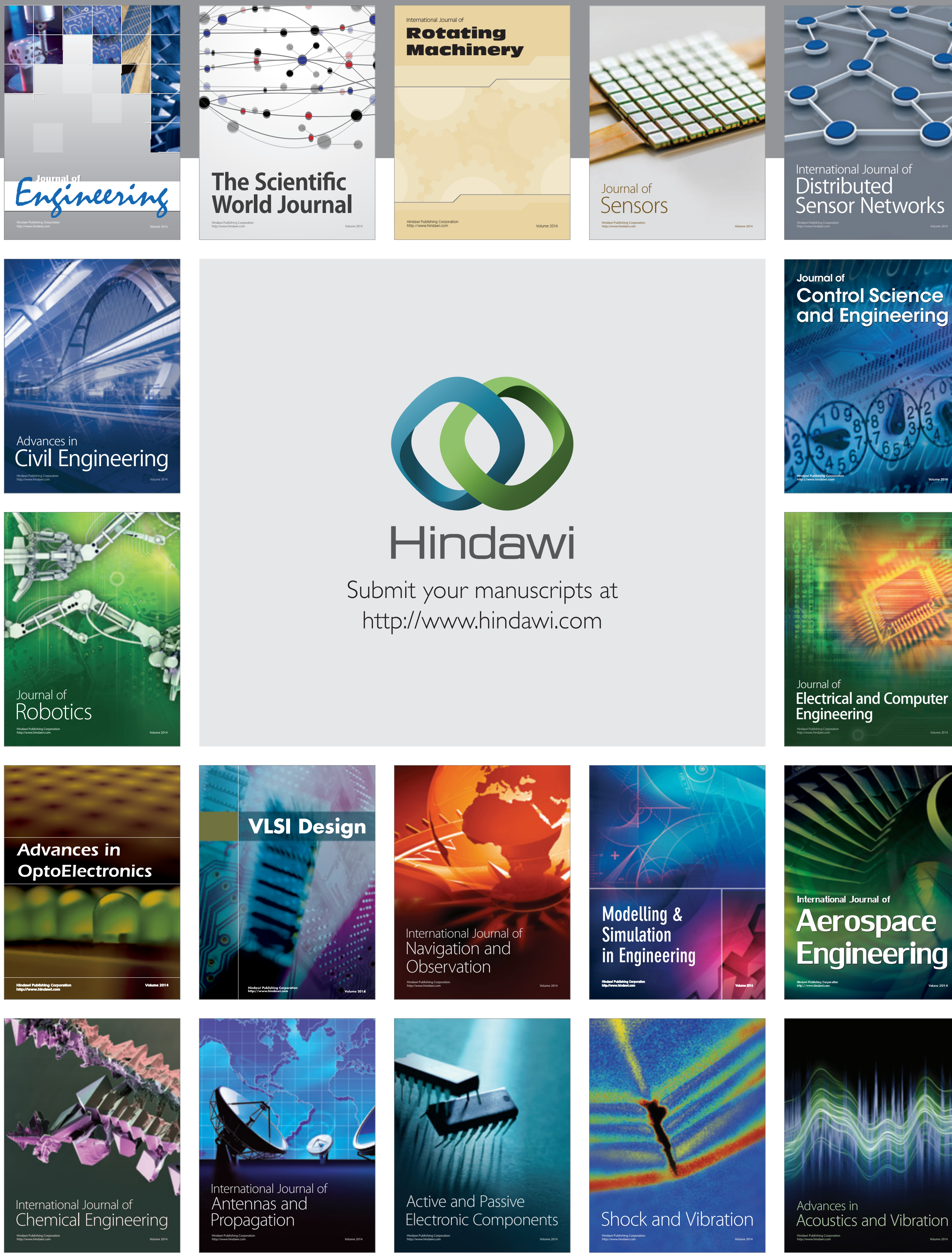
[48] I. Crnkovic, J. Stafford, and C. Szyperski, "Software components beyond programming: from routines to services," *IEEE Software*, vol. 28, no. 3, pp. 22–26, 2011.

[49] L. Braubach and A. Pokahr, "Developing distributed systems with active components and Jadex," *Scalable Computing: Practice and Experience*, vol. 13, no. 2, pp. 100–119, 2012.

[50] Z. Zhang, Y. Wang, F. Song, and W. Zhang, "An energy-balanced mechanism for hierarchical routing in wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 2015, Article ID 123521, 10 pages, 2015.

[51] K. Kim, C. Ha, and C. Ok, "Network structure-aware ant-based routing in large-scale wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 2015, Article ID 521784, 16 pages, 2015.

[52] A. Basholli, T. Lagkas, G. Eleftherakis, and P. A. Bath, "Wireless monitoring systems for enhancing national health services in developing regions," in *Proceedings of the International Conference on Health Informatics (HEALTHINF '14)*, M. Bienkiewicz, C. Verdier, G. Plantier, T. Schultz, A. L. N. Fred, and H. Gamboa, Eds., pp. 511–516, ESEO, SciTePress, March 2014.


Journal of Engineering
The Scientific World Journal
International Journal of Rotating Machinery
Journal of Sensors
International Journal of Distributed Sensor Networks
Advances in Civil Engineering
Hindawi
Submit your manuscripts at
http://www.hindawi.com
Journal of Control Science and Engineering
Journal of Robotics
Journal of Electrical and Computer Engineering
Advances in OptoElectronics
VLSI Design
International Journal of Navigation and Observation
Modelling & Simulation in Engineering
International Journal of Aerospace Engineering
International Journal of Chemical Engineering
International Journal of Antennas and Propagation
Active and Passive Electronic Components
Shock and Vibration
Advances in Acoustics and Vibration