SOCA
DOI 10.1007/s11761-016-0198-1

ORIGINAL RESEARCH PAPER

Decentralized service discovery and selection in Internet of Things applications based on artificial potential fields

Elli Rapti[1] · Anthony Karageorgos[2] · Catherine Houstis[3] · Elias Houstis[1]

Received: 9 March 2016 / Revised: 4 July 2016 / Accepted: 3 August 2016

Abstract The Internet of Things (IoT) vision involves a future Internet integrated with real-world objects that can commonly offer their functionality trough services. In such pervasive environments of IoT networks, locating and invoking suitable services is quite challenging and traditional service discovery and selection approaches have been proven inadequate. In this paper, taking inspiration from natural metaphors, a decentralized service discovery and selection model is proposed. The model is based on artificial potential fields (APFs) which are formed upon each user service request and become active at points where services can be provided. Such points are termed as service provision nodes (SPNs). The strength of each APF depends on the percentage of requested services that can be provided by the respective SPN, as well as on SPN service load and availability with the aim to balance service load among SPNs. Service discovery and selection is then driven by artificial forces applied among user service requests and SPNs. Simulation results indicate that the proposed approach maintains satisfactory performance and scalability as the number of SPNs in an IoT network increase and efficient load balancing of the requested services among the SPNs in comparison with other approaches.

Keywords Decentralized service discovery · Service selection · Software agents · Artificial potential fields · Internet of Things

✉ Anthony Karageorgos
karageorgos@computer.org

Elli Rapti
erapti@ireteth.certh.gr

Catherine Houstis
houstis@e-ce.uth.gr

Elias Houstis
elias.houstis@gmail.com

1 Institute for Research and Technology Thessaly, CERTH, Vólos, Greece

2 Applied Informatics Laboratory, Technological Educational Institute of Thessaly, Karditsa, Greece

3 Department of Electrical and Computer Engineering, University of Thessaly, Vólos, Greece

1 Introduction

The advent of ubiquitous wireless connectivity in conjunction with the ever increasing deployment of pervasive computing technologies has changed the landscape of information and communication technologies [1]. An example is the Internet of Things (IoT) metaphor which is defined as "a world-wide network of interconnected objects uniquely addressable based on standard communication protocols" [2]. IoT refers to integration of large numbers of real-world objects, that is "things", in the Internet and aims to simplify high-level interactions of the physical world to resemble interactions taking place in virtual electronic worlds [3].

In contrast to traditional Internet, IoT includes an additional sensing layer which enables integration of both intelligent, such as laptops and smartphones, and non-intelligent or weakly-intelligent devices [4], such as radio-frequency identification (RFID) tags and sensors [5]. This sensing layer enables detection of the physical status of things, which in turn can drive immediate response to changes in the real world. Such a fully interactive and responsive network offers immense potential for citizens, consumers and businesses [6].

The number of interconnected things rapidly grows toward millions and even billions [7], and therefore, issues concerning heterogeneity and interoperability have arised. As a result, service-orientation has emerged as a promising solution to tackle IoT network complexity. In service-oriented IoT settings, each thing offers its functionality through standard services, which can be termed IoT services. This approach enables dynamic discovery and invocation of IoT functionality, as well as on-demand formation of new composite functionality through IoT service composition [8].

The majority of service-oriented approaches available in the literature commonly adopt centralized service organization [1,3,8–11]. In centralized organization, services connect to an appropriate server, termed "registry", and register their information details such as a description of their functionality and their network invocation address. The registry can then be searched based on the required criteria, and the selected services can be located and invoked using the recorderd network invocation address. A typical example is universal description, discovery and integration (UDDI) which has been recognized as the most popular discovery mechanism for Web services [12]. In pervasive IoT environments however, where users typically access resources and IoT functionality through mobile devices, relying on centralized service organization approaches proves to be inefficient [13]. Mobile devices impose limitations to service provision due to limited memory and processing power and to restricted input capabilities [14]. Since service registration is a complex process [15], it would require extra power and additional bandwidth from mobile devices, limiting thus further their service provision capabilities. Furthermore, centralized service registries may easily suffer from problems such as performance bottlenecks and vulnerability to failures as the number of service consumers and service requests increase in open service-oriented environments [16].

To overcome the shortcomings of centralized service discovery approaches, many research efforts focused on decentralized service discovery [16–19], since it offers better flexibility, robustness, scalability and fault–tolerance [20]. Some approaches [19,21,22] adopt structured peer-to-peer-based (P2P) discovery methods to achieve even data distribution and efficient query routing by controlling the topology and imposing constraints on the data distribution through the use of distributed hashing tables (DHT). Although using structured P2P can potentially improve the scalability of service discovery, directly applying DHT-based P2P approaches to decentralized service discovery can be insufficient for guaranteeing the availability of published service descriptions. This can lead to serious problems in open and dynamic service-oriented environments where nodes join and leave the network dynamically [16].

Considerable research has been devoted to adoption of nature-inspired computing paradigms, such as physical, chemical, biological and social, for development of decentralized service oriented systems [7,22,23], taking advantage of their inherent capability to accommodate spatiality, self-adaptivity, and evolvability [24]. Among nature-inspired approaches used for service discovery and composition, artificial potential fields (APFs) have attracted attention recently [23,25]. APFs use observations from physics where complex, stable motions that have many properties arise from relatively simple laws and have been extensively studied for robot navigation [26–28] and routing in wireless sensor networks [29–32]. In addition to spatiality, self-adaptivity and evolvability, APFs are computationally inexpensive [33], which renders them suitable for the resource constrained devices of IoT networks (i.e., sensors, smartphones), while the reduction in computational complexity makes their use tractable as the number of devices in an IoT network grows larger.

In this paper, a decentralized service discovery model which performs emergent selection of atomic services based on artificial potential fields (APFs) is presented. The proposed model is an extension of the work described in [34] and focuses on selecting a suitable set of services in an IoT network satisfying a particular user request. The selected services can belong to the same or different network nodes (i.e., SPNs). Taking inspiration from electrically charged particles, APFs generated by SPNs apply forces to user service requests aiming to lead them to the most suitable SPNs. The generated APFs are based on the compatibility between user service requests and the services offered by SPNs, as well as on SPN availability. The main contribution of this work is a novel service discovery and selection model where the discovery process considers: (a) the number of services of user service requests that can be satisfied by each SPN and (b) the number of requests waiting to be served in each SPN, in order to ensure load balancing throughout all SPNs in the IoT network.

The remainder of the paper is structured as follows: Sect. 2 provides an overview of the related literature. Section 3 presents a brief introduction on APFs, while Sect. 4 presents the formulation of the decentralized service discovery and selection problem. Section 5 introduces the proposed approach for service discovery and selection based on APFs, followed by Sect. 6 which presents a discussion of the simulation results. Finally, Sect. 7 summarizes the main contributions of this work.

2 Related work

The traditional centralized client/server model has been adopted for service discovery since the beginning of service-oriented computing. UDDI [12] has been recognized as the most popular discovery mechanism for Web services and has been adopted by various software, such as Windows Server 2003 by Windows. However, in the pervasive environment of IoT networks, users are typically mobile and access resources (information, services, etc.) through mobile devices. Subsequently, peers join and leave the network dynamically causing frequent information update, which produces large system overhead [35]. Therefore, relying on existing centralized infrastructures is not efficient as they inherently suffer from poor performance in complicated open environments that demand high scalability [19].

To address poor performance of centralized architectures in open environments, such as IoT networks, decentralized service-oriented approaches have been widely studied in the literature. Chakraborty et al. [36] propose a distributed, decentralized and fault–tolerant architecture for dynamically reactive service composition in pervasive environments, which is based on a peer-to-peer model. A central role is assigned to a distributed broker, that can be executed in any node of the network, to manage the composition. However, the proposed model does not consider alternatives when part of the requested resources is missing. Furthermore, the proposed architecture involves a fault tolerance system based on the use of checkpoints, which is quite expensive in terms of both cache and computational overhead for mobile devices. To achieve dynamic discovery and selection of QoS-based services, Yu et al. [37] propose heuristic algorithms that can be used to find near optimal solutions more efficiently than exact solutions using a broker-based architecture. Despite the significant improvement achieved by the proposed algorithms compared to finding exact solutions, the proposed algorithms do not scale efficiently with respect to an increasing number of web services. In addition, their implementation in the highly distributed environment of IoT networks would incur significant communication costs. Gu et al. [35] propose SpiderNet, a quality-aware service composition middleware, which includes a heuristic algorithm that selects service instances in a hop-by-hop manner and finds suitable service invocation paths while observing multiple end-to-end QoS constraints. However, SpiderNet requires strict definition of the services to be composed, that is, it supports only static service composition, which is not generally possible in dynamic environments. Luo et al. [38] propose a heuristic algorithm to select candidate services for a QoS-aware service composition in distributed grid environments. As the performance of the proposed algorithm has not been tested in large scale distributed grid environments, it is not clear how it would scale in the pervasive environment of IoT networks. Al-Oqily and Karmouch [39] present a decentralized and self-organizing service composition algorithm based on a selective flooding routing strategy. Routing decision is based on peer observation on queries and result feedbacks. By combining service discovery and composition into one step, the system achieves more efficiency and less latency. However, it is not clear how it would scale in networks of different sizes.

Regarding decentralized service discovery, a range of industrial standards have emerged over the last years. Sun microsystems developed Jini [40] which provides a service discovery framework based on JavaSpace [41] and Java RMI (remote method invocation). Services register with lookup servers which act as directories and can be queried to determine suitable services that match a required set of criteria. Other approaches include UPnP [42], service location Protocol [43], and secure service discovery protocol [44], which differ in the expressiveness of their service description or whether they provide a push-based or pull-based discovery. However, these systems were not designed for wireless environments and are therefore not suitable for the dynamic and infrastructure-less IoT networks.

In addition, decentralized service discovery has often been based on Peer-to-peer (P2P) platforms since P2P overlay networks provide suitable infrastructures for routing and data localization in decentralized and self-organized environments. In that view, peers act not only as nodes providing routing and data location services, but also as servers providing service access. Examples of such P2P approaches are CAN [45], Pastry [46] and Chord [21]. However, P2P service discovery approaches cannot generally guarantee the availability of published service descriptions, which can result in serious problems in cases where unexpected failure of service nodes cannot be avoided [19].

Toward the nature-inspired metaphor, Gharzouli and Boufaida [22] introduce a distributed service discovery approach based on epidemic algorithms which locates semantic web services in unstructured peer-to-peer networks. The proposed approach stores peer-to-peer composed web service descriptions and the respective composition methods, in distributed hash tables which increases network overhead and service search time. Furthermore, that approach lacks QoS selection criteria for the returned results. Lenders, May, and Plattner in [25] propose a service discovery protocol based on the analogy with an electrostatic field. The proposed approach selects a service instance based on two metrics: the hop count between service requester and service provider and the service capacity, which refers to the nominal capacity of a service instance. That service selection algorithm is fully automated and does not involve any interaction with users. The limitation of that approach is that it focuses only on the discovery of single services rather than sets of services.

3 Artificial potential fields

Artificial potential fields (APFs) is a nature-inspired technique initially proposed by [47] to solve the robot navigation problem in dynamic environments. Robots were modeled as moving particles inside attractive and repulsive APFs generated by goals and obstacles, respectively. The robots' motion was then dictated by an artificial force resulting from the negative gradient of the generated global potential field [48].

In this work, the notion of APFs is adopted to discover and select a set of services requested by a user in an IoT network in a completely decentralized manner without external user intervention. Our approach to select the requested services and thus determine how to forward user requests among the nodes of the network is inspired from physics and specifically from the electrostatic interaction between electrically charged particles as described by Coulomb's law. To draw the analogy, in this work, each user request is associated with a charge depending on the requested services and SPNs are associated with a charge depending on the provided services and their availability to serve new requests. The resulting force between the charges is used to route a user request among the SPNs of the IoT network so that all requested services are delivered to the user.

4 Problem formulation

An IoT network can be defined as a directed graph $G(V, E)$, in which V denotes the set of all service provision nodes v and E denotes the set of wireless links among SPNs that can communicate directly. We consider $V = V^P \cup V^S$, where V^P represents the set of all high-level devices (e.g., tablets, smartphones, RFID readers), and V^S represents the set of low-level devices (e.g., RFIDs, sensors). Services run on high-level devices, while low-level devices are only able to provide information to the network (e.g., current temperature). The set of links E between SPNs, which represents their capability of direct communication, is defined as $E = \{e_{ij} = (v_i, v_j) | i, j \in \mathbb{N}, i \neq j, v_i \in V^P, v_j \in V^P \cup V^S, d(v_i, v_j) \leq R\}$, where $d(v_i, v_j)$ denotes the distance between SPNs v_i and v_j, and R denotes the maximum communication distance among two SPNs.

To better represent the lack of centralized control, it is considered that two software agents reside in each SPN $v \in V^P$, a service agent $A^S(v)$ (as discussed in [49]) and a requester agent $A^R(v)$. Service agents can interact with the respective devices infrastructure to be able to provide their functionality, while requester agents handle explicit user requests for functionality.

Due to the fact that SPNs are wirelessly interconnected, each agent (requester or service) can establish communication relationships with different agents in their proximity, which is defined as its neighborhood. Thus, considering an agent (requester or service) in SPN $v_i \in V^P (i \in \mathbb{N})$, its neighborhood is defined as a set $N(v_i)$ which contains all agents in SPNs $v_j \in V^P$, $j \in \mathbb{N}$, with $d(v_i, v_j) \leq R$, where R is the maximum radius of communication between any two SPNs. That is: $N(v_i) = \{A^m(v_j) | m \in \{S, R\}, j \in \mathbb{N}, v_j \in V^P, d(v_i, v_j) \leq R\}$. Therefore, an agent communication graph is created over the IoT network, as shown in Fig. 1, termed agent communication network. A communication link, which represents the direct communication between two agents, is denoted as $L_{ij} = (A^m(v_i), A^m(v_j))$, where m is equal to S for a service agent or R for a requester agent, $i, j \in \mathbb{N}$ and $v_i, v_j \in V^P$. In the general case, the neighborhood of an agent changes dynamically along with the structure of the IoTs network, for example, when visitor mobile devices appear and disappear as in pervasive computing settings.

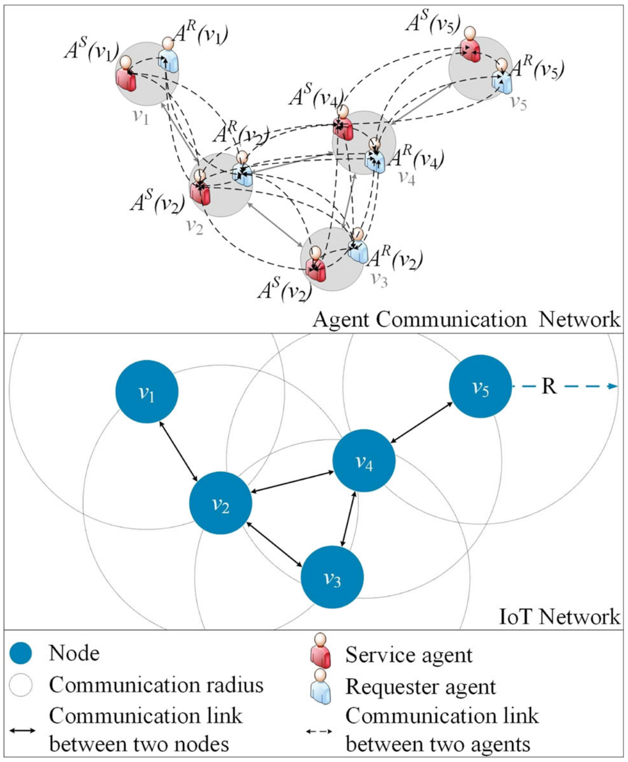

Fig. 1 Representation of an IoT network and the respective agent communication network

Each service agent $A^S(v_i), i \in \mathbb{N}$ and $v_i \in V^P$, has a service set $S(v_i)$ that represents the atomic encapsulated functionality it can provide to the users in the form of software services. That is: $S(v_i) = \{s_k^i | i \in \mathbb{N}, k = 1, 2, \ldots, N_i\}$ where N_i is the total number of services provided by $A^S(v_i)$. Each atomic service $s_k \in S(v_i)$ is defined as a 2-tuple including the input and output parameters of the respective service. That is: $s_k^i = (\text{input}_k^i, \text{output}_k^i), i \in \mathbb{N}, k = 1, 2, \ldots, N_i$. Each atomic service is considered of having only one input and one output parameter. For example, considering an atomic service s_1^1 on SPN v_1 that converts currency from GBP to USD, then its input parameter is the amount of money in GBP and its output the amount of money in USD, that is s_1^1 = (GBP,USD).

Each user request, denoted as $\text{Req}(v_i)(i \in \mathbb{N}, v_i \in V^P)$, handled by a requester agent $A^R(v_i)$, represents a personalized query for services providing a particular functionality. Each user request is defined as a set containing input and output parameter 2-tuples representing the requested functionality. That is: $\mathrm{Req}(v_i) = \{(\mathrm{input}_l^i, \mathrm{output}_l^i) | l = 1, 2, \ldots, N_A\}$, where N_A is the total number of 2-tuples requested. For example, considering a user on SPN v_1 requesting a currency converter from GBP to USD and Euro, then the respective user request would be formulated as $\mathrm{Req}(v_1) = \{(\mathrm{GBP,USD}), (\mathrm{GBP,EURO})\}$. User requests are provided to requester agents either explicitly by the user of the respective SPN or by other requester agents in their neighborhood.

For each user request $\mathrm{Req}\left(v_j\right)$ received by a requester agent $A^R(v_i)(i, j \in \mathbb{N}, v_i, v_j \in V^P)$, it initially defines the services that can be provided by service agent $A^S(v_i)$ with respect to $\mathrm{Req}\left(v_j\right)$, that is $\mathrm{Sel}\left(v_j\right) = S(v_i) \cap \mathrm{Req}\left(v_j\right)$, which represents the set of selected services from $A^S(v_i)$. Then, $A^R(v_i)$ modifies request $\mathrm{Req}\left(v_j\right)$ according to Eq. (1):

$$\mathrm{Req}'\left(v_j\right) = \mathrm{Req}\left(v_j\right) - \mathrm{Sel}\left(v_j\right) \tag{1}$$

and sends it toward the SPN of the most promising (i.e., one that is able to provide most of the requested input and output 2-tuples on time) service agent in a similar way as proposed in [22]. The process followed for the selection of the most promising SPN is described in detail in Sect. 4. In addition, each requester agent $A^R\left(v_j\right)$ $(i \in \mathbb{N}, v_i \in V^P)$ creates a request table where it stores the user requests it has received. Each user request $\mathrm{Req}\left(v_j\right)$ $(j \in \mathbb{N}, v_j \in V^P)$ is stored in the request table as a triple containing a user request identifier, denoted as ID, the SPN from which $A^R\left(v_j\right)$ received the request, denoted as PreNode, and the set $\mathrm{Sel}\left(v_j\right)$ of services selected from service agent $A^S\left(v_j\right)$.

In addition, each service agent $A^S(v_i), v_i \in V^P$ and $i \in \mathbb{N}$, has a request queue where user requests are placed by requester agents waiting to be served. The request queue $Q(v_i)$ of service agent $A^S(v_i)$ is defined as a sequence $Q(v_i) = (\mathrm{Sel}^l\left(v_j\right) | l \in \mathbb{N}, v_j \in V^P)$, which includes all user requests waiting to be served by $A^S(v_i)$. It is considered that each service agent can serve only one request at a time and that service provision time can generally vary depending on the input and output 2-tuples included in the submitted request.

In this work, the proper discovery and selection of atomic services in the context of a service composition is such that all of the requested 2-tuples for a given user request $\mathrm{Req}(v_i), v_i \in V^P$ and $i \in \mathbb{N}$, are satisfied. The final collection of selected services to be provided to the user, denoted as Col, includes all selected atomic services, that is $\mathrm{Col} = \{s_k^i | i, k \in \mathbb{N}\}$, such that $\mathrm{Col} = \mathrm{Req}(v_i)$, creating a service invocation graph. Service discovery and selection is achieved dynamically through the agents interactions including one or more services belonging to the same or different SPNs in the network. The set of selected services is provided to the user by the requester agent where he/she placed request through successive service invocations.

5 Service selection based on artificial potential fields

Exploiting the benefits arising from field-based approaches, the notion of APFs is adopted to achieve decentralized and dynamic discovery and selection of services in the context of a service composition in the pervasive environment of IoT networks. Each service agent is considered to generate APFs that exert attractive and repulsive forces over all user requests in their neighborhood. The services provided by service agents are considered as goals exerting attractive forces over requests, while the request queues of service agents are considered as obstacles exerting repulsive forces, respectively, based on the number of requests in each queue. The generated forces, attractive and repulsive, applied on a user request force the requester agent to send the user request to be served toward the SPN of the most promising service agent (i.e., one that is able to provide most of the requested services on time). User request transmission is considered to be taking place only along the communication links between the requester agents. The respective forces exerted on requester agents are described in detail in the following subsections.

5.1 Service satisfaction potential field

The service satisfaction potential field (SSF) aims to ensure that each user request will be satisfied. Considering requester agent $A^R(v_i)$ in SPN $v_i \in V^P$, $i \in \mathbb{N}$, carrying a request $\mathrm{Req}(v_i)$, the requester agent has to determine the number of 2-tuples included in the user request that the service agents residing in the current SPN and neighboring SPNs, $v_j \in V^P$ and $v_j \in N(v_i)$, $j \in \mathbb{N}$, are able to satisfy according to their available services.

Considering service agent $A^S\left(v_j\right)$ in SPN $v_j (j \in \mathbb{N}, v_j \in V^P)$, the charge responsible for the SSF, denoted as $\mathrm{SSCh}\left(A^S\left(v_j\right)\right)$, is a number representing the number of services that can be provided to user request $\mathrm{Req}(v_i)$ by $A^S\left(v_j\right)$. That is:

$$\mathrm{SSCh}\left(A^S\left(v_j\right)\right) = \left|S\left(v_j\right) \cap \mathrm{Req}(v_i)\right| \tag{2}$$

where $|\cdot|$ denotes the cardinality of the respective set.

Following the electric field analogy, where the electric field strength is equal to the quotient of the charge to the square of the distance multiplied by a constant k_0, the strength of the SSF, denoted as $P_S\left(A^S\left(v_j\right)\right)$ is a numerical value calculated according to Eq. (3):

$$P_S\left(A^S\left(v_j\right)\right) = \frac{k_S \cdot \mathrm{SSCh}\left(A^S\left(v_j\right)\right)}{r^2} \tag{3}$$

where r is the distance from charge $\mathrm{SSCh}\left(A^S\left(v_j\right)\right)$ and $k_S \geq 0$ is a constant representing the processing power SPN

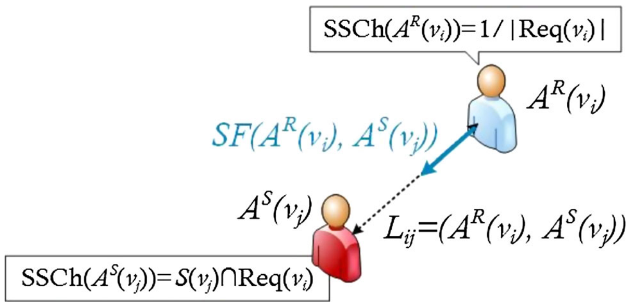
SSCh($A^R(v_i)$)=1/|Req(v_i)|
$A^R(v_i)$
SF($A^R(v_i)$, $A^S(v_j)$)
$A^S(v_j)$
L_{ij}=($A^R(v_i)$, $A^S(v_j)$)
SSCh($A^S(v_j)$)=$S(v_j)$∩Req(v_i)

Fig. 2 Service satisfaction force exerted on request Req(v_i) of requester agent $A^R(v_i)$ by $A^S(v_j)$ (attractive)

v_j. Although electric fields in nature extend infinitely in a continuous space, an IoT network is a discrete space of SPNs where distance is measured as the number of hops between SPNs. Since each SPN can only communicate with its neighbors that are one hop away, the SSF can only span to a distance equal to 1 for all neighboring SPNs. For simplicity, constant k_0 is set equal to 1, considering that all devices in the IoT network have the same processing power. Therefore, Eq. (4) is:

$$P_S\left(A^S\left(v_j\right)\right) = \text{SSCh}\left(A^S\left(v_j\right)\right) \tag{4}$$

Considering a requester agent $A^R(v_i)(i \in \mathbb{N}, v_i \in V^P)$ as a charged particle in the neighborhood of $A^S\left(v_j\right)$, that is $v_i \in N\left(v_j\right)$, its respective charge is a numerical value inversely proportional to the number of 2-tuples included in the user request, that is:

$$\text{SSCh}\left(A^R(v_i)\right) = \frac{1}{|\text{Req}(v_i)|} \tag{5}$$

The service satisfaction potential field of $A^S\left(v_j\right)$ exerts a force termed service satisfaction Force on $A^R(v_i)$ along the communication link $L_{ij} = \left(A^R(v_i), A^S\left(v_j\right)\right)$, as shown in Fig. 2. Following the electric field analogy, where, according to Coulombs law, the force exerted on a charge q in a field E is equal to their product, the magnitude of the service satisfaction force exerted on $A^R(v_i)$ by $A^S\left(v_j\right)$ is a numerical value calculated according to Eq. (6):

$$\text{SF}\left(A^R(v_i), A^S\left(v_j\right)\right) = \text{SSCh}\left(A^R(v_i)\right) \cdot P_S\left(A^S\left(v_j\right)\right) \tag{6}$$

The service satisfaction force is attractive, from $A^R(v_i)$ toward $A^S\left(v_j\right)$, with values in the interval [0, 1], where 0 represents the fact that no requested 2-tuples can be served and 1 that all requested 2-tuples can be served by $A^S\left(v_j\right)$.

5.2 Availability potential field

Considering only service satisfaction could lead to congestion problems in SPNs that would attract many requests, for example, due to high number of different services provided. To avoid congestion of service requests, decrease waiting time for requests to be served and achieve a better load balancing among all nodes in the IoT network, the availability potential field is introduced. The availability potential field (AF) aims to ensure a load balancing among all nodes in the network by ensuring a better disperse of requests among the SPNs. That, in turn, would lead to a reduction of the waiting time experienced in each SPN by a requester agent, as this solves the dual purpose of congestion avoidance and fast application completion [7].

Considering service agent $A^S\left(v_j\right)$ in SPN $v_j(j \in \mathbb{N}, v_j \in V^P)$, the charge producing the AF, denoted as AvCh$\left(A^S\left(v_j\right)\right)$, is a numerical value equal to the number of the selected services of user requests that are waiting to be served by $A^S\left(v_j\right)$ in $Q\left(A^S\left(v_j\right)\right)$. That is:

$$\text{AvCh}\left(A^S\left(v_j\right)\right) = |\text{Sel}^1(v_k) \cup \cup \text{Sel}^\text{T}(v_m)| \tag{7}$$

where, $|\cdot|$ denotes the cardinality of the respective set, $v_k, v_m \in V^P$ with $k, m \in \mathbb{N}$ and $\text{Sel}^t(v_m) \in Q\left(A^S\left(v_j\right)\right)$ with $t \in [1, T]$ and T the number of selected service sets waiting to be served by $A^S\left(v_j\right)$.

Following the electrical field analogy, the strength of the AF is a numerical value calculated according to Eq. (8):

$$P_A\left(A^S\left(v_j\right)\right) = \frac{k_A \cdot \text{AvCh}\left(A^S\left(v_j\right)\right)}{r^2} \tag{8}$$

where, $k_A = 1/k_S$ with $k_S \geq 0$ representing the processing power of node v_j. For simplicity, k_S is equal to 1, considering that all devices in the IoT network have the same processing power. Considering that each SPN can only communicate with its neighbors that are one hop away, AF can only span to a distance r equal to 1 for all neighboring SPNs. Therefore, $P_A\left(A^S\left(v_j\right)\right)$ is calculated as follows:

$$P_A\left(A^S\left(v_j\right)\right) = k_A \cdot \text{AvCh}\left(A^S\left(v_j\right)\right) \tag{9}$$

Considering a requester agent $A^R(v_i)(i \in \mathbb{N}, v_i \in V^P)$, as a charged particle in the neighborhood of $A^S\left(v_j\right)$, that is $v_i \in N\left(v_j\right)$, its respective charge is a numerical value inversely proportional to the sum of services that can be provided to satisfy user request Req(v_i) and the number of the selected services waiting to be served in $Q\left(A^S\left(v_j\right)\right)$. That is:

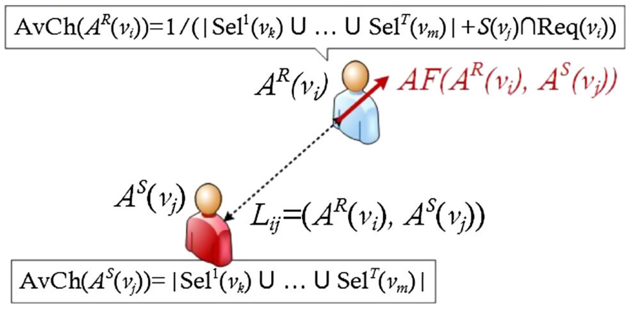
AvCh(A^R(v_i))=1/(|Sel^1(v_k) ∪ … ∪ Sel^T(v_m)|+S(v_j)∩Req(v_i))
A^R(v_i)
AF(A^R(v_i), A^S(v_j))
A^S(v_j)
L_ij=(A^R(v_i), A^S(v_j))
AvCh(A^S(v_j))= |Sel^1(v_k) ∪ … ∪ Sel^T(v_m)|

Fig. 3 Availability force exerted on request $\mathrm{Req}(v_i)$ of requester agent $A^R(v_i)$ by $A^S(v_j)$ (repulsive)

$$\mathrm{AvCh}\left(A^R(v_i)\right) = \frac{-1}{\left(|\mathrm{Sel}^1(v_k) \cup \cup \mathrm{Sel}^{\mathrm{T}}(v_m)| + |S(v_j) \cap \mathrm{Req}(v_i)|\right)} \quad (10)$$

with $v_k, v_m \in V^P$ and $k, m \in \mathbb{N}$.

The availability potential field of $A^S(v_j)$ exerts a force termed availability force on request $\mathrm{Req}(v_i)$ of $A^R(v_i)$ along the communication link $L_{ij} = \left(A^R(v_i), A^S(v_j)\right)$, as shown in Fig. 3. Considering the electrical analogy, the availability force exerted on $A^R(v_i)$ by $A^S(v_j)$ is a numerical value calculated according to Eq. (11):

$$\mathrm{AF}\left(A^R(v_i), A^S(v_j)\right) = \mathrm{AvCh}\left(A^R(v_i)\right) \cdot P_A\left(A^S(v_j)\right) \quad (11)$$

The availability force is repulsive, from $A^S(v_j)$ toward $A^R(v_i)$, with values in the interval $[-1, 0]$, where -1 represents high congestion and 0 represents low congestion in node v_j.

5.3 Resultant force

To ensure that a request $\mathrm{Req}(v_i)$ in node $v_i (i \in \mathbb{N}, v_i \in V^P)$ will be send toward the most promising SPN $v_j (j \in \mathbb{N}, v_j \in V^P)$ achieving the best balance between service satisfaction and availability, the service satisfaction force and the availability force defined above are linearly combined to get the resultant force applied on a requester agent $A^R(v_i)$ by service agent $A^S(v_j)$. The resultant force $F\left(A^R(v_i), A^S(v_j)\right)$ is calculated according to Eq. (12) below:

$$F\left(A^R(v_i), A^S(v_j)\right) = \alpha \cdot \mathrm{SF}\left(A^R(v_i), A^S(v_j)\right) + (1-\alpha) \cdot \mathrm{AF}\left(A^R(v_i), A^S(v_j)\right) \quad (12)$$

where, α is a parameter in the range of $[0, 1]$ representing the degree of influence of each field. The resultant force can be

Algorithm 1 Service Discovery and Selection

```
1:  for each $Req(v_j)$ do
2:      Calculate $Sel(v_j) = S(v_i) \cap Req(v_j)$
3:      Store $\langle ID, PreNode, Sel(v_j)\rangle$ in the request table
4:      Calculate $Req'(v_j) = Req(v_j) - Sel(v_j)$
5:      if $Req'(v_j) = \emptyset$ then
6:          Send response to $PreNode$
7:      if $Req'(v_j) \neq \emptyset$ then
8:          for each $A^S(v_k)$ in $N(v_i)$ do
9:              Calculate $SSF(A^R(v_i), A^S(v_k))$
10:             Calculate $AF(A^R(v_i), A^S(v_k))$
11:             Calculate $F(A^R(v_i), A^S(v_k))$
12:             Send $Req'(v_j)$ to $A^R(v_k)$ with highest
13:             $F(A^R(v_i), A^S(v_k))$
```

either attractive or repulsive toward $A^S(v_j)$, taking values in the interval $[-1, 1]$, depending on the value of α.

5.4 SPN selection

When the resultant force $F\left(A^R(v_i), A^S(v_j)\right)$ for each service agent in $v_j \in N(v_i) (i, j \in \mathbb{N}, v_i, v_j \in V^P)$ has been calculated, the requester agent $A^R(v_i)$ needs to determine the service agent that exerts the highest force toward which it will send request $\mathrm{Req}(v_i)$. Denoting $\mathrm{SelSPN}(v_i)$, a singleton including the SPN in which resides $A^S(v_j)$ exerting the highest resultant force on $A^R(v_i)$. That is $\mathrm{SelSPN}(v_i) = (v_j | F\left(A^R(v_i), A^S(v_j)\right) = \mathrm{Max}(F\left(A^R(v_i), A^S(v_j)\right)), v_i, v_j \in V^P, v_j \in N(v_i))$, where $\mathrm{Max}(F\left(A^R(v_i), A^S(v_j)\right))$ is a function calculating the highest resultant force exerted on $A^R(v_i)$ by an $A^S(v_j)$ residing on SPN $v_j \in N(v_i)$. This ensures that each user request will continue moving among nodes, allowing them to avoid being trapped in areas of local minima where user requests cannot be satisfied.

Algorithm 1 describes the process followed by each requester agent $A^R(v_i)$ upon receiving a user request $\mathrm{Req}(v_i)$.

6 Simulation experiments

6.1 Simulation settings

In order to evaluate the proposed decentralized service discovery and selection mechanism for the pervasive environment of IoT networks, several tests have been performed. A real-world audio–video virtual guide executed on mobile devices (e.g., smartphones, tablets) [50] is used as the experimental setup. The virtual guide is provided to the visitors of a cultural event and is able to create a customized tour for each individual visitor according to their requests.

Each visitor's device represents an SPN of the IoT network which is able to communicate wirelessly with other devices in their proximity. Each device is provided with a service

Table 1 Audio–video virtual guide services

Service	Input	Output
Video streaming	Video signal	Encoded video
Audio streaming	Audio signal	Encoded audio
Audio to text converter	Audio signal	Text
Text translator	Input text	Translated text

and a requester agent, which are able to provide and request multimedia services to and from the network, respectively. Each visitor is therefore able to submit a request for multimedia services to the IoT network through their device which will result in the generation of a customized tour through the composition of the discovered audio–video services. Examples of the audio–video virtual guide services used in our experiments are provided in Table 1.

A simulation tool was developed in Java using Mason multi-agent simulation toolkit [51] in order to validate the proposed approach. The experiments did not include evaluations based on execution time, as they tend to be hardware dependent and would be misleading as devices in IoT networks tend to be highly heterogeneous regarding their hardware specifications, ranging from low-level devices, such as RFIDs and sensors, to high level, such as smartphones and tablets.

The tests evaluate the performance of the service discovery and selection mechanism in the decentralized environment of the audio–video virtual guide using the notion of APFs. The tests include the following metrics:

– the average number of hops required to satisfy a user request,
– the number of requests placed in the queue of each service requester agent.

Each network included in the simulations is an undirected graph of varying size (100, 500 and 1000 SPNs). For each network configuration, 100 simulations were conducted. Thus, each value represents the average of 100 measurements. All requester agents in the network have the same probability of generating a user request, which is successfully satisfied when all individual services included in the user request are discovered in the network. A search is considered to have failed when a requester agent cannot forward the request to its neighbors, either because its neighborhood set N is empty or all requester agents in N already provide their services in the current search.

6.2 APF parameters

To analyze the influence of the service satisfaction potential field and the availability potential field in the discovery and selection mechanism proposed in this paper, the parameter

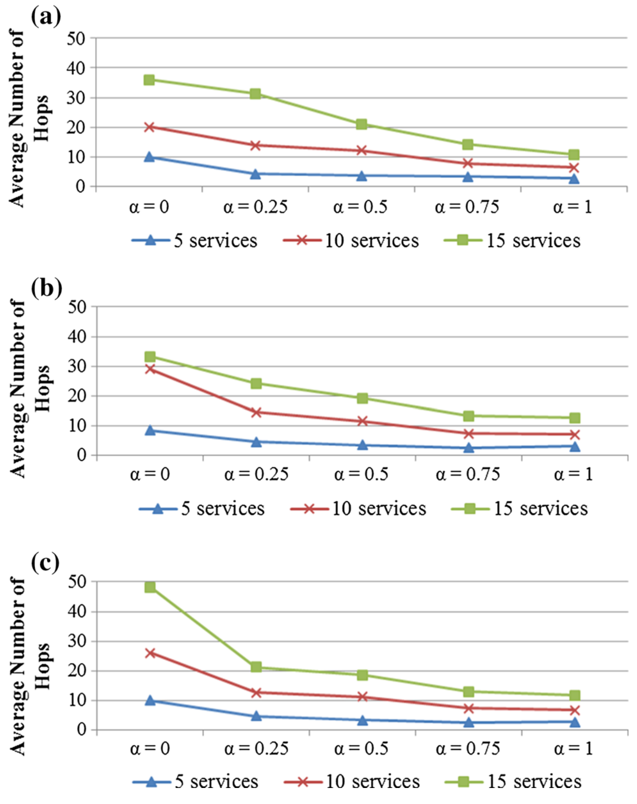

Fig. 4 Average number of hops required for the discovery of 5, 10 and 15 services included in a request for different α values and number of nodes. **a** 100 SPNs, **b** 500 SPNs, **c** 1000 SPNs

a took different values in the interval [0, 1]. Each requester agent in the network is considered to have an average of eight neighbors and the services are evenly distributed across all service agents in each network configuration. The networks consisted of 100, 500 and 1000 SPNs in order to evaluate the influence of the potential fields in environments of different scale. In addition, the behavior of the discovery and selection approach was tested for requests containing a varying number of services, namely 5, 10 and 15 different services.

A value of a equal to 0 means that the resultant force exerted on a requester agent is calculated considering only the availability potential field. This configuration would be similar to a random search as each user request would be sent toward the SPN of the service agent with the lowest queue, without considering the number of services contained in the request that can be provided by the respective service agent. On the other hand, a value of a equal to 1 means that the resultant force is calculated considering only the service satisfaction potential field. Therefore, a requester agent would forward the request toward the SPN of the service agent that would provide the highest number of services contained in the respective request. Every value in between 0 and 1 represents a respective influence of both fields in the selection of the most appropriate service agent. The results of the tests conducted are provided in Fig. 4 for each network configuration.

From the three diagrams of Fig. 4a–c, which correspond to networks consisting of 100, 500 and 1000 SPNs, respectively, it can be concluded that the approach has similar behavior despite the network size. For a value a equal to 0, a higher average number of hops is required in order to satisfy a user request containing a different number of requests. In particular, the average number of hops required is almost proportionate to the number of services requested. Therefore, a triple number of average hops is required for the discovery and selection of a triple number of services. As the value of a is gradually increased from 0 to 0.5, the average number of hops required is dramatically decreased by more than 50 % in most cases, which means that the average number of hops required is reduced by half. The average number of hops required for each request is further decreased by increasing the value of a from 0.5 to 1, however to a lesser extent than in the previous case analyzed before.

Even though, taking into consideration only the service satisfaction potent field for the calculation of the resultant force on a requester agent, that is $a = 1$, leads to a smaller average number of hops required for a user request to be satisfied, this has a negative impact to the even distribution of the requests among the nodes, which leads to congestion problems. From the results shown in Fig. 5, it can be concluded that a value of a equal to 1 creates sharper edges as shown in Fig. 5e on certain nodes, which means that a higher number of requests are placed on certain nodes. The best results are obtained for values of a equal to 0.25 and 0.5 where requests are more evenly distributed among all SPNs in the network achieving a better load balance.

Taking all of the above into account, it can be concluded that the best configuration is achieved for a value of a equal to 0.5. Each request is satisfied with a sufficiently low average number of hops and an even distribution of requests among all SPNs in the network is achieved.

6.3 Routing performance and scalability

In order to evaluate the routing performance of the proposed service discovery and selection approach, the average hops needed for a user request containing a certain number of services is compared for different network sizes. The number of required services per request was set to 5, 10 and 15. In each simulation, the user request was provided randomly to a SPN of the network. From the results obtained from the simulation experiments and which are shown in Fig. 6, it can be concluded that routing performance is highly dependable on the number of services included in the request. That is, there is a significant increase in the number of hops required for a request to be satisfied as the number of services included in the respective request increase in all three network settings. On the other hand, the approach achieves good scalability as network size does not affect routing performance. That is,

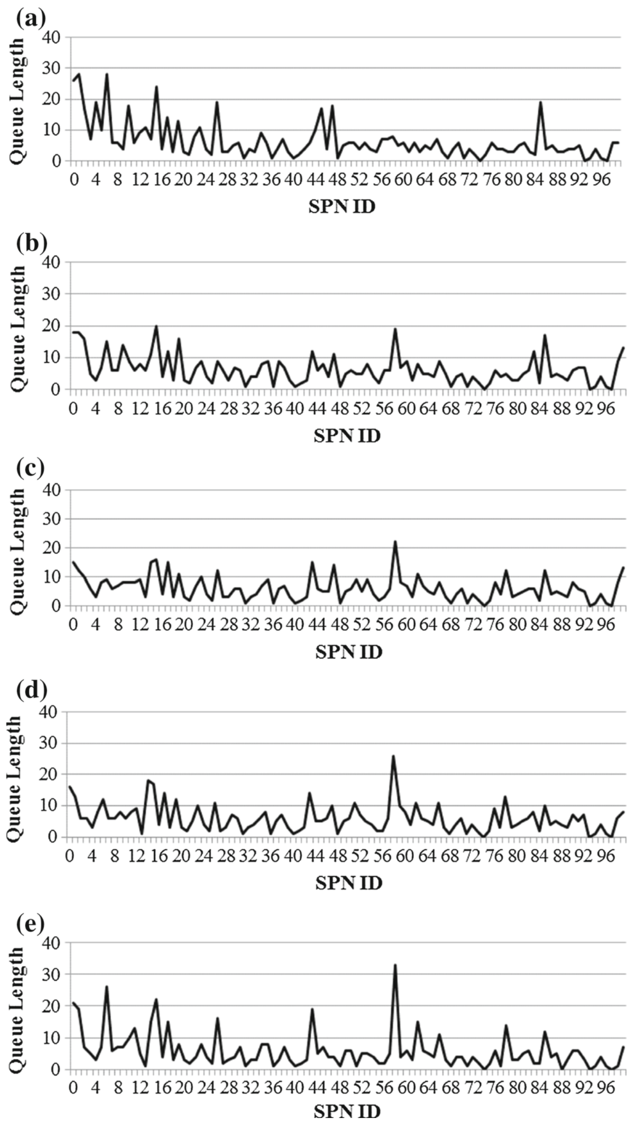

Fig. 5 Number of requests in the queue of each SPN after the submission of 100 requests in a network of 100 SPNs. **a** $\alpha = 0$. **b** $\alpha = 0.25$. **c** $\alpha = 0.5$. **d** $\alpha = 0.75$. **e** $\alpha = 1$

the average number of hops required for each request to be satisfied remains at the same level regardless of the number of SPNs in the network. In particular, as the number of services included in the request increases, the routing performance is slightly improved as the number of SPNs in the network increases.

In addition, to evaluate the routing performance of the proposed approach, the average hops required for a user request containing a certain number of services for different network sizes is compared to the results obtained using a flooding-based approach, where requests are forwarded to all neighboring SPNs. From the results provided in Fig. 7, while both approaches behave similarly regardless the number of SPNs in the IoT netwrok, the flooding-based approach

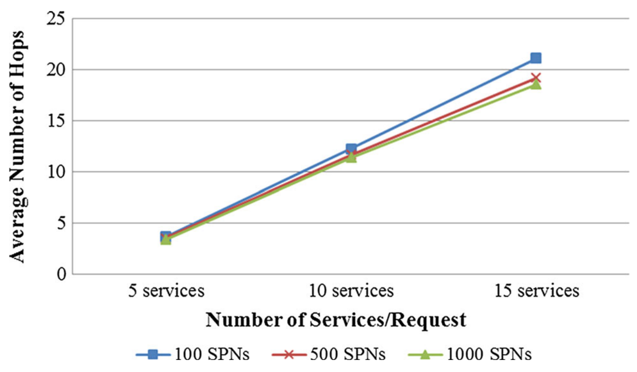

Fig. 6 Average number of hops required for the discovery of 5, 10 and 15 services included in a request with respect to the network size

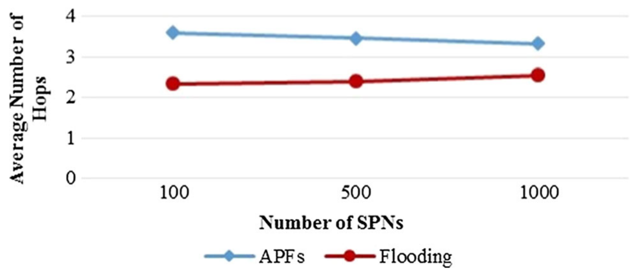

Fig. 7 Average number of hops required for the discovery and selection of five services/request with respect to the network size for the APF-based and flooding-based approaches

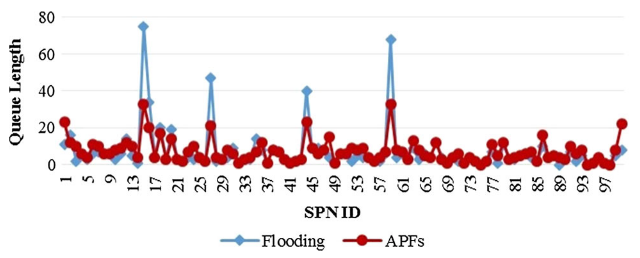

Fig. 8 Number of requests in the queue of each SPN after the submission of 100 requests in a network of 100 SPNs for the APF-based and flooding-based approaches

requires a lower number of hops for a request to be satisfied. However, the difference in the performance of both approaches is so low (≈1 hop for each network configuration) that it cannot overcome the huge amount of messages generated by flooding-based approaches in such unstructured networks.

6.4 Load balancing

To evaluate the ability of our request forwarding technique to distribute requests evenly among all SPNs of an IoT network, simulation experiments were conducted using as baseline a typical flooding protocol as a discovery mechanism. Figure 8 presents the distribution of 100 requests, each one including five services, in a network consisting of 100 SPNs using the proposed APF-based approach and the flooding-based approach. The graph shows the evident benefit of the APF-based approach in comparison with flooding in relation to load balancing. Requests are more evenly distributed among all SPNs in the first case, while in the second case more sharp edges appear on certain SPNs which means that more requests are gathered in those SPNs. Requester agents can therefore achieve a global load balancing in an IoT network based only on local information.

7 Discussion

In this paper, a decentralized service discovery and selection approach for IoT networks is presented. When a wireless device enters the network, it establishes communication relationships with other devices in their proximity, which consists its neighborhood. Service discovery and selection is then achieved through artificial potential fields exerting forces over user requests.

The proposed approach differs from existing approaches in several ways. From a structural point of view, the proposed approach is not based on hierarchical organization, which means that all software agents in the system are considered to be equal. In addition, it does not require an initial period to acquire knowledge about network settings by using flooding strategies that cause overhead. Moreover, as IoT environments are highly dynamic, service discovery and selection is not based on stored information from previous searches, since they would be unreliable. Therefore, each SPN only maintains information regarding current local network structure.

In contrast to centralized approaches utilizing global knowledge, the proposed approach is based only on the local view of each SPN in the network. Therefore, to achieve service discovery and selection requester, agents consider artificial forces exerted on them by artificial potential fields generated by their neighboring service agents. These forces exerted among requester and service agents enable each user request consisting of a different number of services to be satisfied in a small number of hops, while achieving load balancing between all SPNs.

Several simulation experiments have been performed including comparison with a flooding-based approach to evaluate the performance of the proposed approach. The results obtained are very promising since the inclusion of availability potential fields leads to even distribution of requests among SPNs in the network, without causing significant decrease in routing performance. Therefore, the proposed approach can achieve global load balancing by evenly distributing requests among all SPNs through local agent interactions.

8 Conclusion

In this paper, an approach for decentralized service discovery and selection in the pervasive environments of IoT networks is presented and is based on the notion of artificial potential fields. It has been shown how artificial potential fields can be mapped to the service discovery and selection problem in IoT networks to achieve global equilibrium through local interactions. Each device in the network, representing an SPN, provides its functionality through services which are handled internally by software agents. Upon receiving a user request, each agent in an SPN aims to locate other agents in the IoT network that offer certain services included in the request in order to satisfy the recived request and fulfill its goal.

Further research efforts will include the extension of the proposed work to consider multiple variables in generating artificial potential fields, such as hop count, as well as to incorporate dynamic adaptation to address changes in network topology. In addition, the integration of semantic matching of services would be highly beneficial since the mobile environment of IoT networks is characterized by heterogeneity of service interfaces. Last but not least, the performance of the proposed approach will be tested through an implementation of a real-world IoT application and it will be compared with both centralized and decentralized approaches.

Acknowledgments This research has been supported by GSRT (General Secretariat for Research and Technology), Program: "National Strategic Reference Framework 2007–13", Act: "Support for SMEs Groups for Research and Technological Development", Project title: "e-Furniture: Dynamic Networking of Furniture and Wood Product Manufacturing Enterprises".

References

1. Zambonelli F, Viroli M (2011) A survey on nature-inspired metaphors for pervasive service ecosystems. Int J Pervasive Comput Commun 7(3):186–204
2. Christin D, Reinhardt A, Mogre PS, Steinmetz R (2009) Wireless sensor networks and the internet of things: selected challenges. In: Proceedings of the 8th GI/ITG KuVS Fachgespräch Drahtlose sensornetze, pp 31–34
3. Teixeira T, Hachem S, Issarny V, Georgantas N (2011) Service oriented middleware for the internet of things: a perspective. In: Abramowicz W, Llorente I, Surridge M, Zisman A, Vayssiere J (eds) Towards a service-based internet. Springer, Berlin, Heidelberg, pp 220–229
4. Ma HD (2011) Internet of things: objectives and scientific challenges. J Comput Sci Technol 26(6):919–924
5. Atzori L, Iera A, Morabito G (2010) The internet of things: a survey. Comput Netw 54(15):2787–2805
6. Uckelmann D, Harrison M, Michahelles F (2011) An architectural approach towards the future internet of things. Springer, Berlin
7. Ahmed T, Tripathi A, Srivastava A (2014) Rain4service: an approach towards decentralized web service composition. In: 2014 IEEE international conference on services computing (SCC). IEEE, pp 267–274
8. Spiess P, Karnouskos S, Guinard D, Savio D, Baecker O, De Souza LMS, Trifa V (2009) Soa-based integration of the internet of things in enterprise services. In: IEEE international conference on web services, 2009. ICWS 2009. IEEE, pp 968–975
9. Csorba MJ, Meling H, Heegaard PE (2011) A bio-inspired method for distributed deployment of services. New Gener Comput 29(2):185–222
10. Chan NN, Gaaloul W, Tata S (2012) A recommender system based on historical usage data for web service discovery. SOCA 6(1):51–63
11. Chen H, Li S (2010) Src: a service registry on cloud providing behavior-aware and qos-aware service discovery. In: 2010 IEEE international conference on service-oriented computing and applications (SOCA). IEEE, pp 1–4
12. Clement L, Hately A, von Riegen C, Rogers T et al (2004) Uddi version 3.0. 2, uddi spec technical committee draft. OASIS open standards consortium
13. Papadopoulos P, Tianfield H, Moffat D, Barrie P (2013) Decentralized multi-agent service composition. Multiagent Grid Syst 9(1):45–100
14. Al-Masri E, Mahmoud QH (2006) A context-aware mobile service discovery and selection mechanism using artificial neural networks. In: Proceedings of the 8th international conference on electronic commerce: the new e-commerce: innovations for conquering current barriers, obstacles and limitations to conducting successful business on the internet. ACM, pp 594–598
15. Guinard D, Trifa V, Karnouskos S, Spiess P, Savio D (2010) Interacting with the soa-based internet of things: discovery, query, selection, and on-demand provisioning of web services. IEEE Trans Serv Comput 3(3):223–235
16. He Q, Yan J, Yang Y, Kowalczyk R, Jin H (2013) A decentralized service discovery approach on peer-to-peer networks. IEEE Trans Serv Comput 6(1):64–75
17. Sapkota B, Roman D, Kruk SR, Fensel D (2006) Distributed web service discovery architecture. In: Null. IEEE, p 136
18. Li Y, Zou F, Wu Z, Ma F (2004) Pwsd: a scalable web service discovery architecture based on peer-to-peer overlay network. In: Jeffrey Xu Yu, Xuemin Lin, Hongjun Lu, Yanchun Zhang (eds) Advanced web technologies and applications. Springer, Berlin, Heidelberg, pp 291–300
19. He Q, Yan J, Yang Y, Kowalczyk R, Jin H (2008) Chord4s: a p2p-based decentralised service discovery approach. In: IEEE international conference on services computing, 2008. SCC'08, vol 1. IEEE, pp 221–228
20. Bicchi A, Marigo A, Pappas G, Pardini M, Parlangeli G, Tomlin C, Sastry S (1998) Decentralized airtraffic management systems: performance and fault tolerance. In: Proceedings of IFAC international workshop on motion control, Grenoble. Citeseer, pp 279–284
21. Stoica I, Morris R, Karger D, Kaashoek MF, Balakrishnan H (2001) Chord: a scalable peer-to-peer lookup service for internet applications. ACM SIGCOMM Comput Commun Rev 31(4):149–160
22. Gharzouli M, Boufaida M (2011) Pm4sws: a p2p model for semantic web services discovery and composition. J Adv Inf Technol 2(1):15–26
23. Ahmed T, Mrissa M, Srivastava A (2014) Magel: a magneto-electric effect-inspired approach for web service composition. In: 2014 IEEE international conference on web services (ICWS). IEEE, pp 455–462
24. Mamei M, Zambonelli F (2006) Field-based coordination for pervasive multiagent systems. Springer Science & Business Media, Berlin
25. Lenders V, May M, Plattner B (2005) Service discovery in mobile ad hoc networks: a field theoretic approach. Pervasive Mobile Comput 1(3):343–370

26. Adeli H, Tabrizi MHN, Mazloomian A, Hajipour E, Jahed M (2011) Path planning for mobile robots using iterative artificial potential field method. Int J Comput Sci Issues 8(4):28–32
27. Jaradat MAK, Garibeh MH, Feilat EA (2012) Autonomous mobile robot dynamic motion planning using hybrid fuzzy potential field. Soft Comput 16(1):153–164
28. Masoud AA (2007) Decentralized self-organizing potential field-based control for individually motivated mobile agents in a cluttered environment: a vector-harmonic potential field approach. IEEE Trans Syst Man Cybern Part A: Syst Hum 37(3):372–390
29. Jiang HF, Qian JS, Sun YJ (2011) Virtual electrostatic field based multi-sink routing algorithm in WSN. J China Univ Min Technol 40(2):321–326
30. Xu YS, Ren FY (2009) Potential field based multi-strategy routing protocol in WSN. ZTE Commun 15(6):32–36
31. Jiang H, Sun Y, Sun R, Chen W, Ma S, Gao J (2014) A distributed energy optimized routing using virtual potential field in wireless sensor networks. Int J Distrib Sens Netw 2014:10
32. Kalantari M, Shayman M (2006) Design optimization of multi-sink sensor networks by analogy to electrostatic theory. In: IEEE wireless communications and networking conference, 2006. WCNC 2006, vol 1. IEEE, pp 431–438
33. Ruchti J, Senkbeil R, Carroll J, Dickinson J, Holt J, Biaz S (2011) Uav collision avoidance using artificial potential fields. Tech. rep. CSSE11-03, Computer Science and Software Engineering Department, Auburn University
34. Rapti E, Karageorgos A, Gerogiannis VC (2015) Decentralised service composition using potential fields in internet of things applications. Proc Comput Sci 52:700–706
35. Gu X, Nahrstedt K, Yu B (2004) Spidernet: an integrated peer-to-peer service composition framework. In: 2004. Proceedings 13th IEEE international symposium on high performance distributed computing. IEEE, pp 110–119
36. Chakraborty D, Perich F, Joshi A, Finin T, Yesha Y (2003) A reactive service composition architecture for pervasive computing environments. In: Cambyse GO (ed) Mobile and wireless communications. Springer, Berlin, Heidelberg, pp 53–60
37. Yu T, Zhang Y, Lin K-J (2007) Efficient algorithms for web services selection with end-to-end qos constraints. ACM Trans Web (TWEB) 1(1):6
38. Luo J-Z, Zhou J-Y, Wu Z-A (2009) An adaptive algorithm for qos-aware service composition in grid environments. SOCA 3(3):217–226
39. Al-Oqily I, Karmouch A (2011) A decentralized self-organizing service composition for autonomic entities. ACM Trans Auton Adapt Syst (TAAS) 6(1):7
40. Sun microsystems, jini architecture specification version 2.0 (2003). http://www.sun.com/software/jini/specs/
41. Waldo J (1998) Javaspaces specification 1.0. Sun Microsyst 29 Technical report, p 30
42. Universal Plug (2003) Play (upnp) forum. Microsoft Corporation
43. Veizades J, Guttman E, Perkins CE, Kaplan S (1997) Service location protocol. Internet Eng Task Force: RFC 2165, p 72. https://www.ietf.org/rfc/rfc2165.txt
44. Czerwinski SE, Zhao BY, Hodes TD, Joseph AD, Katz RH (1999) An architecture for a secure service discovery service. In: Proceedings of the 5th annual ACM/IEEE international conference on mobile computing and networking. ACM, pp 24–35
45. Ratnasamy S, Francis P, Handley M, Karp R, Shenker S (2001) A scalable content-addressable network, vol 31. ACM, New York
46. Rowstron A, Druschel P (2001) Pastry: scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In: Rachid G (ed) Middleware 2001. Springer, Berlin, Heidelberg, pp 329–350
47. Khatib O (1986) Real-time obstacle avoidance for manipulators and mobile robots. Int J Robot Res 5(1):90–98
48. Kim DH, Wang H, Shin S (2006) Decentralized control of autonomous swarm systems using artificial potential functions: analytical design guidelines. J Intell Rob Syst 45(4):369–394
49. Vallée M, Ramparany F, Vercouter L (2005) A multi-agent system for dynamic service composition in ambient intelligence environments. Citeseer, Grenoble
50. Puliafito A, Cucinotta A, Minnolo AL, Zaia A (2010) Making the internet of things a reality: the wherex solution. In: Daniel G, Antonio I, Giacomo M, Luigi A (eds) The internet of things. Springer, New York, pp 99–108
51. Luke S, Cioffi-Revilla C, Panait L, Sullivan K, Balan G (2005) Mason: a multiagent simulation environment. Simulation 81(7):517–527