A Bio-inspired Service Discovery and Selection Approach for IoT Applications

Elli Rapti, Catherine Houstis, Elias Houstis
Dept. of Electrical and Computer Engineering
University of Thessaly, Volos, Greece
erapti@ireteth.certh.gr, houstis@e-ce.uth.gr,
elias.houstis@gmail.com

Anthony Karageorgos
Dept. of Wood and Furniture Design and Technology,
Technological Educational Institute of Thessaly
Karditsa, Greece
karageorgos@computer.org

***Abstract*—Traditional service discovery and selection approaches which rely mostly on centralized architectures, have been proven inadequate in the pervasive environment of the Internet of Things (IoT). In such settings, where decentralization of decision-making is mandatory, bio-inspired computing paradigms have emerged due to their inherent capability to operate without any central control. In this paper, taking inspiration from the widely studied bio-inspired *Response Threshold Model,* a decentralized service discovery and selection model is proposed. Preliminary results indicate that the proposed approach exhibits efficient scalability and routing performance.**

Keywords- Response threshold model; decentralised service discovery; service selection; software agents; Internet of Things

I. INTRODUCTION

The Internet of Things (IoT) aims to bring together billions of real-world objects (i.e. things) by connecting them in an Internet-like structure, allowing them to communicate by exchanging information and to enable new forms of interaction among things and people [1]. Contrast to traditional Internet which generally does not have sensing ability, the IoT has the additional sensing layer, which enables the integration of non-intelligent or weakly-intelligent devices [2].

As the number of things connected in a network will rapidly grow larger [3], issues of heterogeneity, interoperability, unknown topology and availability arise. To this end, service-oriented computing has emerged as a promising solution as each thing provides its functionality through standard services [4]. Most service-oriented approaches in the literature adopt a centralised architecture ([4-7]) where each service connects to a server (called a registry) and provides information about itself, including a network address. However, in the pervasive environment of IoT, where users are mobile and typically access resources (such as information, services and sensors) through mobile devices, relying on existing centralised SOA approaches is not efficient [8]. Registering a service with one or more registries is a rather complex process [9] that requires both power and bandwidth [10]. In such pervasive environments decentralised approaches for service discovery and composition [3, 8, 11, 12] are promising as they offer flexibility, robustness, scalability and fault-tolerance to the system [13].

Towards the development of efficient decentralised SOA approaches for the IoT, this paper proposes a bio-inspired decentralized service discovery and selection model which takes advantage of the coordination mechanisms of biological societies. The proposed model is inspired from the Response Threshold Model (RTM), as introduced in [14], and performs emergent selection of atomic services. The remainder of this paper is structured as follows: in Section II the problem statement is formulated and the proposed solution approach is described. In Section III the results of simulation experiments are discussed and, finally, Section IV concludes the paper.

II. DECENTRALISED SERVICE DISCOVERY AND SELECTION BASED ON RTM

A. 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 (SPNs) 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 (such as tablets, smartphones and RFID readers), and V^S represents the set of low level devices (such as RFIDs and sensors). Services run on high level devices, while low level devices are only able to provide information to the IoT network, such as room temperature. Links between SPNs represent their capability of direct communication and are defined as $E = \{e_{ij} = (v_i, v_j) | i, j \in \mathbb{N}, i \neq j, v_i, v_j \in V^P d(v_i, v_j) \leq R\} \cup \{(v_i, v_j) | i, j \in \mathbb{N},\ i \neq j, v_i \in V^P, v_j \in V^S,\ d(v_i, v_j) \leq R\}$, 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 a software agent resides in each SPN $v \in V^P$, and is termed service requester agent $A(v)$. Service requester agents can interact with the infrastructure of the IoT device associated with the respective SPN and can provide their functionality through exposition of standard services, while also handle explicit user requests for functionality.

978-1-5090-2628-9/16 $31.00 © 2016 IEEE
DOI 10.1109/SCC.2016.126

Due to the fact that SPNs are wirelessly interconnected, each agent can establish communication relationships with other agents in its proximity, which is considered as its neighborhood. More specifically, considering an agent $A(v_i)$ in SPN $v_i \in V^P$ ($i \in \mathbb{N}$), its neighbourhood is defined as a set $N(v_i)$ which contains all agents $v_j \in V^P, j \in \mathbb{N}$, within a communication distance R. That is: $N(v_i) = \{A(v_j) | j \in \mathbb{N}, v_j \in V^P, d(v_i, v_j) \leq R\}$. 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 $L_{ij} = (A(v_i), A(v_j))$, where $i, j \in \mathbb{N}$ and $v_i, v_j \in V^P$. In the general case, the neighbourhood of an agent changes dynamically along with the structure of the IoT network, for example when visitor mobile devices appear and disappear in pervasive computing settings.

In addition, each service requester agent $A(v)$, $v \in V^P$, has a service registry denoted as a set $S(v)$ that represents the atomic encapsulated functionality it can provide to the users in the form of software services. That is: $S(v) = \{s_k | k = 1, 2, \dots, K\}$ where K is the total number of services provided by $A(v)$. Each atomic service $s \in S(v)$ is defined as a 2-tuple including the input and output parameters of the respective service. That is: $s = (input, output)$. For simplicity, each atomic service is considered as having only one input and one output parameter. For example, considering an atomic service s_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 $s_1 = (\text{GBP}, \text{USD})$.

Each user request, denoted as $\text{Req}(v)$ ($v \in V^P$), handled by a service requester agent $A(v)$, represents a personalised query for services providing a particular functionality. Each user request is defined as a set containing input and output parameter pairs representing the requested functionality. That is: $\text{Req}(v) = \{(input_l, output_l) | l = 1, 2, \dots, L\}$, where L is the total number of pairs requested. For example, considering a user on SPN v requesting a currency converter service from GBP to USD and Euro, the respective user request would be formulated as $\text{Req}(v) = \{(GBP, USD), (GBP, EURO)\}$. User requests are provided to service requester agents either explicitly by the user of the respective SPN or by other service requester agents in their neighborhood.

Each service requester agent $A(v)$ ($v \in V^P$) has a local search module that determines the services that can be provided directly by $A(v)$ for each user request $Req(v)$ received. That is $Sel(v) = S(v) \cap Req(v)$, which represents the set of selected services from $A(v)$. Then, user request $Req(v)$ is modified according to equation (1):

$$Req'(v) = Req(v) - Sel(v) \quad (1)$$

The new user requests along with the number of hops from the initiating node are then sent towards the most promising service requester agents as described in detail in the next section. In addition, each service requester agent $A(v)$ ($v \in V^P$) has a request table where it stores the user requests it has received. Each user request $Req(v)$ ($v \in V^P$) is stored in the request table as a triple containing a user request identifier, denoted as ID, the SPN from which $A(v)$ received the request, denoted as $PreNode$, the number of hops from the initiating node ($Hops$) and the set $Sel(v)$ of services selected from service agent $A(v)$, that is $\langle ID, PreNode, Hops, Sel(v) \rangle$.

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 $Req(v)$, $v \in V^P$, are satisfied with the lowest number of hops required. The final collection of selected services to be provided as a response to a user request, denoted as $Col(v)$, includes all selected atomic services, that is $Col(v) = \{s_k | k \in \mathbb{N}\}$, such that $Col(v) = Req(v)$, 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. Each $A(v)$ with $Req'(v) = \emptyset$ sends a response of search completion to the $PreNode$, as stored in its request table, denoted as $Res(v)$. The response of completion is a triple including the user request identifier, the number of hops and the collection of selected services, that is $Res(v) = \langle ID, Hops, Col(v) \rangle$.

The set of selected services are provided to the user by the requester agent where he/she placed the request, through successive service invocations.

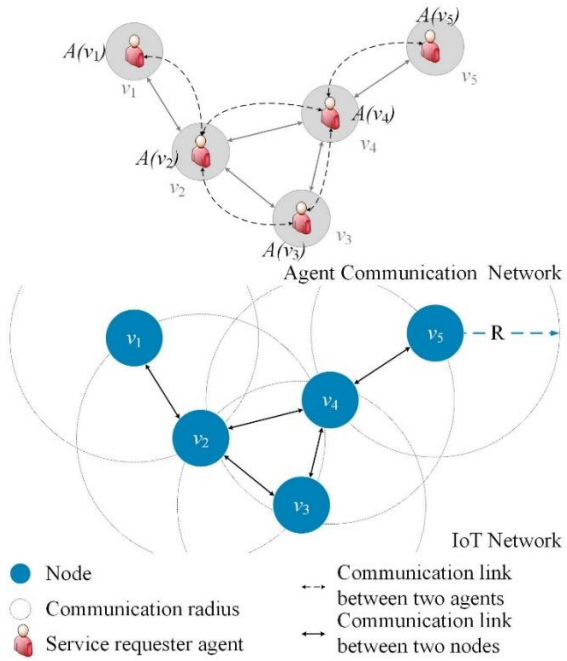

Figure 1. IoT network and the respective agent communication network

B. The Response Threshold Model

The underlying theoretical formulation of the proposed approach is the so called Response Threshold Model (RTM) [14]. RTM is inspired from decentralised insect colonies and assumes that individuals have an inherent threshold to respond to stimuli associated with specific elements and, in a group, individuals with the lowest threshold for an element stimulus will be attracted to it more often [15]. As a result, each individual responds with high probability to a given stimulus when its intensity exceeds the individual's threshold [16].

In this work, RTM is mapped to the service discovery and selection problem in IoT settings, taking advantage of the inherent capability of social insect colonies to operate without any central control. With this decentralized way to work, the proposed model exhibits flexibility and robustness, two features that are desirable in any artificial system [17].

C. Mapping the Response Threshold Model to the Decentralised Service Discovery and Selection Problem

For defining entities as required by the RTM model, an agent-based approach is adopted. Each SPN is represented by an autonomous decision-making entity, termed service requester agent, which is able to transmit user requests for services towards its neighbours based on local information. One of the most important benefits of agent-based systems is their ability to capture emergent phenomena resulting from the interactions of individual entities [18]. For this reason, in the proposed model SPNs are modelled as agents which act locally based on the bio-inspired response threshold model, and the satisfaction of a user request emerges as the result of individual agent actions. In the proposed model, service requester agents can be either active or passive entities depending on whether they act as service requesters or service providers for a particular interaction.

Considering a service requester agent $A(v_i)$ with a request $Req(v_i)$, each service requester agent $A(v_j)$ in the neighbourhood of $A(v_i)$, that is $A(v_j) \in N(v_i)$, is considered as a passive entity characterized by a stimulus st_{ij}, ($i,j \in \mathbb{N}, v_i, v_j \in V^P$). st_{ij} denotes the stimulus generated from service requester agent $A(v_j)$ with respect to a user request $Req(v_i)$ and can be calculated by (2) as follows:

$$st_{ij} = \left|S(v_j) \cap Req(v_i)\right| \tag{2}$$

where, $|...|$ denotes the cardinality of the respective set and $st_{ij} \in [0, L]$, where L is the total number of 2-tuples requested. Therefore, the stimulus from $A(v_j)$ towards $A(v_i)$ is equal to the number of services that $A(v_j)$ can provide as a response to $Req(v_i)$.

Service requester agent $A(v_i)$ with a request $Req(v_i)$, on the other hand, is considered as an active entity associated with a response threshold θ_i that reflects the tendency of $A(v_i)$ to send $Req(v_i)$ towards the other service requester agents in its neighbourhood. The response threshold θ_i with respect to $Req(v_i)$ is calculates according to (3) as follows:

$$\theta_i = \frac{|Req(v_i)|}{N} \tag{3}$$

where, $|...|$ denotes the cardinality of the respective set, $N \in \mathbb{N}$ and $\theta_i \in (0, L]$, where L is the total number of 2-tuples requested. The higher the value of N, the lower the value of θ_i.

Consequently, the probability T_{ij} of service requester agent $A(v_i)$ to send user request $Req(v_i)$ towards $A(v_j) \in N(v_i)$ can be calculated by (4):

$$T_{ij} = \frac{st_{ij}}{st_{ij} + \theta_i} \tag{4}$$

where, st_{ij} is the stimulus of service requester $A(v_j) \in N(v_i)$ and θ_i is the response threshold for $A(v_i)$ with respect to $Req(v_i)$. $T_{ij} \in [0, 1)$ with values close to 0 representing low probability and values close to 1 standing for high probability for sending $Req(v_i)$ towards $A(v_j)$. Algorithm 1 describes the procedure followed for all user requests received by each service requester agent.

Algorithm 1: Service Discovery and Selection

For each $Req(v_i)$ **do**
  Calculate $Sel(v_i) = S(v_i) \cap Req(v_i)$
  Store $\langle ID, PreNode, Sel(v_i)\rangle$ in the request table
  Calculate $Req'(v_i) = Req(v_i) - Sel(v_i)$
  If $Req'(v_i) = \emptyset$ **do**
    Set $Col(v_i) = Sel(v_i)$
    Send $\langle ID, Hops, Col(v_i)\rangle$ to $PreNode$
  End if
  If $Req'(v_i) \neq \emptyset$ **do**
    For each $A(v_j)$ in $N(v_i)$ **do**
      Calculate T_{ij}
      Send $Req'(v_i)$ to $A(v_j)$ with probability T_{ij}
    End for
  End if
End for
For each request ID **do**
  Select $Res(v)$ with lowest number of $Hops$
  Set $Col(v_i) = Col(v_i) + Sel(v_i)$
  Send $\langle ID, Hops, Col(v_i)\rangle$ to $PreNode$
End for

III. Evaluation

To evaluate the proposed decentralized service discovery and selection model for the pervasive environment of IoT networks, a simulation tool was developed in Java using Mason multi-agent simulation toolkit [51]. The routing performance and scalability of the proposed model was evaluated by measuring the average number of hops required for a user request to be satisfied in different network settings. 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.

Following the experimental design introduced in [1], each IoT 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 service 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.

To evaluate the routing performance of the proposed approach, the average number of hops needed for a user request to be satisfied is correlated to the number of services included in the request. The number of services included in a request was set to 5, 10 and 15 services respectively, respectively according to [12]. In each simulation the user request was provided randomly to an SPN of the network. From the results obtained, as shown in Fig. 6, it can be concluded that routing performance is highly dependable on the number of services included on 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, 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.

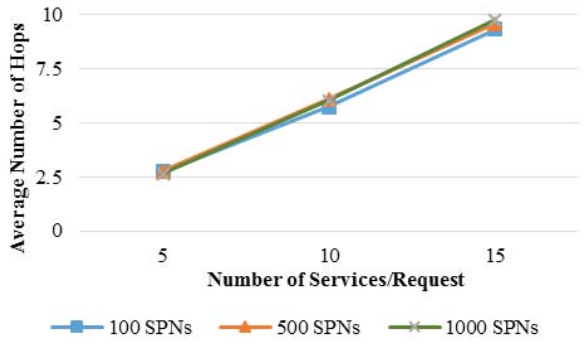

Figure 2. Average number of hops required for the discovery of 5, 10 and 15 services included in a request with respect to the network size

IV. Conclusions

In this paper an approach for decentralized service discovery and selection in the pervasive environment of IoT networks based on the Response Threshold Model is presented. The proposed approach achieves service discovery and selection based only on the local view of each SPN in the network. Further research efforts will include the extension of the proposed work so as to incorporate dynamic adaptation to address changes in network topology. In addition, the integration of semantic matching of services would be highly beneficial, as the mobile environment of IoT networks is characterized by the 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 will be compared with both centralized and decentralized approaches.

References

[1] S. Cirani, L. Davoli, G. Ferrari, R. Léone, P. Medagliani, M. Picone, *et al.*, "A scalable and self-configuring architecture for service discovery in the internet of things," *Internet of Things Journal, IEEE,* vol. 1, pp. 508-521, 2014.

[2] H.-D. Ma, "Internet of things: Objectives and scientific challenges," *Journal of Computer science and Technology,* vol. 26, pp. 919-924, 2011.

[3] T. Ahmed, A. Tripathi, and A. Srivastava, "Rain4Service: An approach towards decentralized web service composition," in *Services Computing (SCC), 2014 IEEE International Conference on*, 2014, pp. 267-274.

[4] P. Spiess, S. Karnouskos, D. Guinard, D. Savio, O. Baecker, L. Souza, *et al.*, "SOA-based integration of the internet of things in enterprise services," in *Web Services, 2009. ICWS 2009. IEEE International Conference on*, 2009, pp. 968-975.

[5] T. Teixeira, S. Hachem, V. Issarny, and N. Georgantas, "Service oriented middleware for the internet of things: a perspective," in *Towards a Service-Based Internet*, ed: Springer, 2011, pp. 220-229.

[6] F. Zambonelli and M. Viroli, "A survey on nature-inspired metaphors for pervasive service ecosystems," *International Journal of Pervasive Computing and Communications,* vol. 7, pp. 186-204, 2011.

[7] M. J. Csorba, H. Meling, and P. E. Heegaard, "A Bio-inspired Method for Distributed Deployment of Services," *New generation computing,* vol. 29, pp. 185-222, 2011.

[8] P. Papadopoulos, H. Tianfield, D. Moffat, and P. Barrie, "Decentralized multi-agent service composition," *Multiagent and Grid Systems,* vol. 9, pp. 45-100, 2013.

[9] 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," *Services Computing, IEEE Transactions on,* vol. 3, pp. 223-235, 2010.

[10] Y.-K. Chen, "Challenges and opportunities of internet of things," in *Design Automation Conference (ASP-DAC), 2012 17th Asia and South Pacific*, 2012, pp. 383-388.

[11] Q. He, J. Yan, Y. Yang, R. Kowalczyk, and H. Jin, "A decentralized service discovery approach on peer-to-peer networks," *Services Computing, IEEE Transactions on,* vol. 6, pp. 64-75, 2013.

[12] Q. He, J. Yan, Y. Yang, R. Kowalczyk, and H. Jin, "Chord4s: A p2p-based decentralised service discovery approach," in *Services Computing, 2008. SCC'08. IEEE International Conference on*, 2008, pp. 221-228.

[13] A. Bicchi, A. Marigo, G. Pappas, M. Pardini, G. Parlangeli, C. Tomlin, *et al.*, "Decentralized air traffic management systems: performance and fault tolerance," in *the Proceedings of the IFAC International Workshop on Motion Control*, 1998.

[14] E. Bonabeau, G. Theraulaz, and J.-L. Deneubourg, "Fixed response thresholds and the regulation of division of labor in insect societies," *Bulletin of Mathematical Biology,* vol. 60, pp. 753-807, 1998.

[15] A. Y. Quiñonez Carrillo, "Response threshold models, stochastic learning automata and ant colony optimization-based decentralized self-coordination algorithms for heterogeneous multi-tasks distribution in multi-robot systems," Informatica, 2012.

[16] T. Yasuda, K. Kage, and K. Ohkura, "Response threshold-based task allocation in a reinforcement learning robotic swarm," in *Computational Intelligence and Applications (IWCIA), 2014 IEEE 7th International Workshop on*, 2014, pp. 189-194.

[17] C. R. Kube and E. Bonabeau, "Cooperative transport by ants and robots," *Robotics and autonomous systems,* vol. 30, pp. 85-101, 2000.

[18] E. Bonabeau, "Agent-based modeling: Methods and techniques for simulating human systems," *Proceedings of the National Academy of Sciences,* vol. 99, pp. 7280-7287, 2002.