Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2014, Article ID 760248, 11 pages
http://dx.doi.org/10.1155/2014/760248

Research Article

Fractal Cross-Layer Service with Integration and Interaction in Internet of Things

Bing Jia,[1] Shuai Liu,[1,2] and Yongjian Yang[3]

[1] *College of Computer Science, Inner Mongolia University, Hohhot 010021, China*
[2] *School of Physical Science and Technology, Inner Mongolia University, Hohhot 010021, China*
[3] *College of Computer Science and Technology, Jilin University, Jilin 010021, China*

Correspondence should be addressed to Shuai Liu; cs_liushuai@imu.edu.cn

Received 4 December 2013; Revised 10 January 2014; Accepted 20 January 2014; Published 3 March 2014

Academic Editor: Gelan Yang

Under the Internet of Things (IoT), in order to achieve the adaptive and personalized delivery of services, there was an urgent need to build an intelligent cross-layer service platform to support the services with effective integration and interaction. First, this paper focuses on the huge amounts, dynamic, and heterogeneous nature of services in IoT and presents the architecture of cross-layer services platform based on semantics to extend the service source from the common services in registries to the self-organization services in ubiquitous environment, which mainly contains the service management, demand computing, service discovery, and service selection. Then, some of the key foundational models and algorithms are presented, such as IoT service ontology model, semantic-based IoT service description language OWL-S^{iot}, and service management strategy both in registry mode and self-organization by subcluster. Many large-scale, ubiquitous, and heterogeneous services could be integrated in this platform. It could provide multilevel semantic supports for services to meet the common understanding and interoperability.

1. Introduction

Ubiquitous connection, mature service platform, and the scientific service model based on the service logic which can take intelligent action are the most important aspects of services in Internet of Things (IoT) in the future [1]. IoT can achieve ubiquitous connection; so how to use the technology to build a cross-layer platform about the service in Internet of Things (IoT service) which can support the service with effective integration and interoperability is a very important task.

As a sensing-based technology, IoT is a service-oriented network system blending various applications [2]. And its core value is "smart services" [3]. IoT services not only include common services in traditional internet but also include the ubiquitous services in the new network environment (mobile networks, wireless sensor networks, etc.) [4]. IoT services are massive and heterogeneous. They are mostly facing morphological changes, outward expansion, environmental change, business restructuring, and other situational dynamic adaptabilities. IoT services also face different levels of sharing and interoperability issues. These problems also bring challenges to IoT service platform. In order to solve these problems, we are attempting to use Web Services, SOA, SOC, semantics, and so on to achieve the services with effective integration, sharing, discovery, and interoperability. Then we can achieve an efficient service discovery and personalized delivery [5]. We present a semantic-based and cross-layer service platform for IoT service in this paper, including its architecture and key algorithm descriptions, to form a more complete service system.

2. Related Work

Service platform is logic components for supporting services; IoT services need to be provided to appropriate object in the right situations (right time, right place, right environment, etc.) by the right way. Therefore, IoT service platform is bound to be adaptive, using depth and personalized approaches to enhance the experience to deliver services [1].

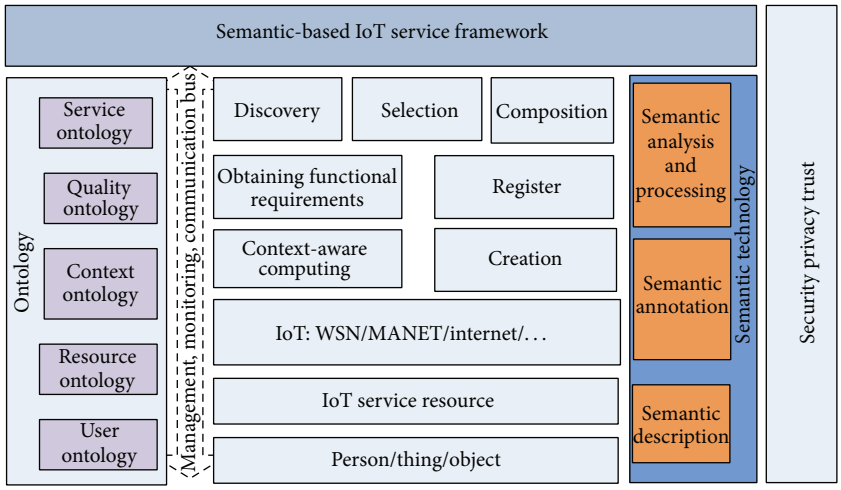

Figure 1: Framework of IoT service based on semantics.

Now the researches about IoT service platform are still in the theory exploratory stage, because the researches of IoT are still in the primary stage. For example, [6] proposed a functional framework of IoT service agent based on enhancing semantics. It uses the adapter agent to handle different types of smart objects' interrelated issues. Li gives the service framework structure based on IoT middleware model, which is future-oriented network, to achieve the transfer of IoT data transfer and the access of services [7]. EU's Seventh Framework Programmer (EU FP7) has organized IoT.est project team dedicated to do researches about the creating and testing of IoT service environment. In its report "Reference Architecture IoT Service Creation and Provision", showed the common reference architecture 5 about IoT service's creation, testing, analysis, and design [8]. They have extended the existing IoT-A architecture by adding a semantic service layer and proposed use-cases that can give an abstract treatment method of heterogeneous objects in IoT [9]. Reference [10] gives semantic-based sensor network services supporting framework, to meet service discovery, composition, compensation, and adaptive functions. Reference [11] proposed a solution about interoperability between devices.

To analyze the service platform from an empirical view, we studied some service platform in mobile communication environment. We mainly selected some more representative platforms such as IMS (IP Multimedia Subsystem), CAMEL (Customized Application for Mobile Enhanced Logic), OSA (Open Service Architecture), WAP (Wireless Application Protocol), and i-mode service platform which represent interaction, information, and integration for Internet [1]. These mobile service platforms are trying to break the traditional mode from end to end, and the service models which are vertical closed two parties (providers/users) are trying to allow the third parties to deploy services and develop application on its platform. Service platform should not be implicitly included within the system anymore. Instead, it should be evolved into a separate part of the display of existing components [12]. The framework and architectures of these platforms have good references for the design of IoT service platform [13].

3. Architecture of IoT Service Platform

3.1. Semantic-Based IoT Service Framework. In this paper, after making a comprehensive analysis on Web service framework and semantic sensor network framework [10] proposed by the EU FP7 [14], we present a semantic-based IoT service framework, as shown in Figure 1.

This framework consists of two parts. One is the service-related concept, both IoT ontology and semantic description, and the other is the corresponding operation of the service. It includes person, thing, and object in the bottom of the framework. And the user ontology is used to map them. IoT service resource needs to be described based on resource ontology. Obtaining functional requirements combines service ontology and quality ontology to achieve semantic description of the user's functional requirements. We can use context ontology to make semantic annotation on context characteristics obtained by context-aware computing. Semantic-based characteristics of the user can help to meet user's adaptive and individual needs of enhanced service. Service ontology is used for service creation and publishing, and it is throughout the whole platform.

This framework includes service discovery, selection, combination and so on, of which the most important are selection and combination. Service composition process can be achieved by the decomposition of requirements. Service discovery is searching and matching process on original advertising services set based on the service requester's service functional requirement. Thereby, it could return the set of service that can meet the functional requirements. The service resource may be organized and managed by registry, or may be decentralized. Service selection is a process to select the personalized service based on the service requester's context.

In particular, it needs to choose service discovery and selection strategy to adapt in different case. Whatever strategy, service ontology, quality ontology, context ontology, and others have played a semantic supporting role. Throughout the outermost of the framework are security, privacy, and trust, which is the basic guarantee for the implementation of the process.

3.2. Architecture of Cross-Layer IoT Service Platform. The Chinese Sensor Network National Standard Working Group has released a report named "Advances in the technology architecture and standards system about Internet of Things." It gives a reference standard of IoT architecture which is mainly divided into three layers, namely, the sensor layer, network layer, and application layer [15]. The sensor layer is the basis for the development and application of IoT. The network layer links the sensor layer and the application layer, including the bearer network and intelligent computing technologies. The application layer can show the business to users. One is a middleware of the business, and the other is a specific application.

It can be seen from IoT's reference architecture that we need a layer of coordinating and handling between the network layer and application layer to achieve the operation model on the data after mixing and cleansing, in order to provide users with more fluid and intelligent services. Then it can better support the industry and the public's application. In addition, since the typical characteristic of IoT is extended to ubiquitous environment like wireless network and ubiquitous sensor network, the service platform should extend the source of IoT service from the traditional common service to the ubiquitous service. In order to meet the extensibility of IoT service, we present the architecture of semantic-based cross-layer IoT service platform according to the standard of IoT architecture and the semantic-based IoT service framework presented in upper section, as shown in Figure 2. It has a special design for open issues-related integrated service produced by the complexity of network.

The platform adopts four-layer structure, adding a data integration and service support layer on the basis of the original IoT architecture. And services are throughout the whole platform. It can meet the requirements of the Times, such as sensor data as a service, network infrastructure as a service, computing platform as a service, and integrated application as a service. Different layers of service have a certain correlation. In general, upper layer service does the job by calling lower layer service. Lower service is the basis of top service. The main functional components of the platform are the various functional modules and associations with the operation between them in the data integration and service support layer. Eventually they can provide support for applications of the "application layer" in specific areas. Platform can also access directly the P2P network formed by the service node in the ubiquitous environment to carry out service discovery in ubiquitous service set.

Sensing layer is mainly used to achieve the real world sensing. The sensing data is also a service. In this layer, we complete collecting a variety of basic data, including user data and environmental data which could be perceived.

The network layer mainly obtains information of network characteristics related to services. Network communication resource performs as a service. This layer also achieves an adaptive mechanism that should transmit the data stably to upper layer and support the transmission of the device control data to the lower layer.

Data integration and service support layer are the core parts of IoT service platform, including service management, demand computing, service discovery, and service selection. Various functional modules and interoperability between them can provide the necessary support for the integration and consolidation in specific different application in application layer. The following describes each module separately.

(i) Service management module: it is the most basic core functionality module in service platform architecture. It is responsible for the standardization of service description and the management of service, respectively, according to the service source's different organizational forms. For service which can be concentrated and registered easily and suitably, we take the management of service registration center model to manage the service's register, update, delete, and so on. But ubiquitous services which has been distributed in the resource pools could not be concentrated and registered easily (e.g., environment-based service, mobile location-based service and highly time-sensitive service, etc.). We can use dynamic autonomous strategies to achieve decentralized management according to the loosely coupled P2P network between them. In particular, for multiple service registration centers which cannot be integrated in a catalogue, each of them could be regarded as a service node. So we can form a loosely coupled P2P service network between them and manage them by a decentralized strategy. In general, decentralized mode is a supplement to registration center model.

(ii) Demand computing module: it achieves interaction with the user and obtains the user's exact needs. One is functional computing, and the other is context-aware computing. Functional requirements are the core primary characteristic of requirements, mainly to know what the specific functions, types, and qualities of service the user wants. Context-aware computing is used to understand the current scene which is the proper way. These are user's enhanced experience needs for services. Context-aware computing is primarily responsible for obtaining the relevant contextual information, making inference and integration to obtain interpretable high-level context (e.g., whether in a meeting, in the hallway, or in the coffee shop, etc.). It could make semantic annotation accordance with the relevant ontology. The demand explanation will be filled into a template to automatically generate a service request description document.

(iii) Service discovery module: it completes the searching and matching of the service. It uses different search and matching algorithms according to the service's management style. It uses a centralized service discovery algorithm for registration center model, focusing on the optimization of semantic matching service. And it uses the P2P service discovery strategy for dispersion mode, because the ubiquitous

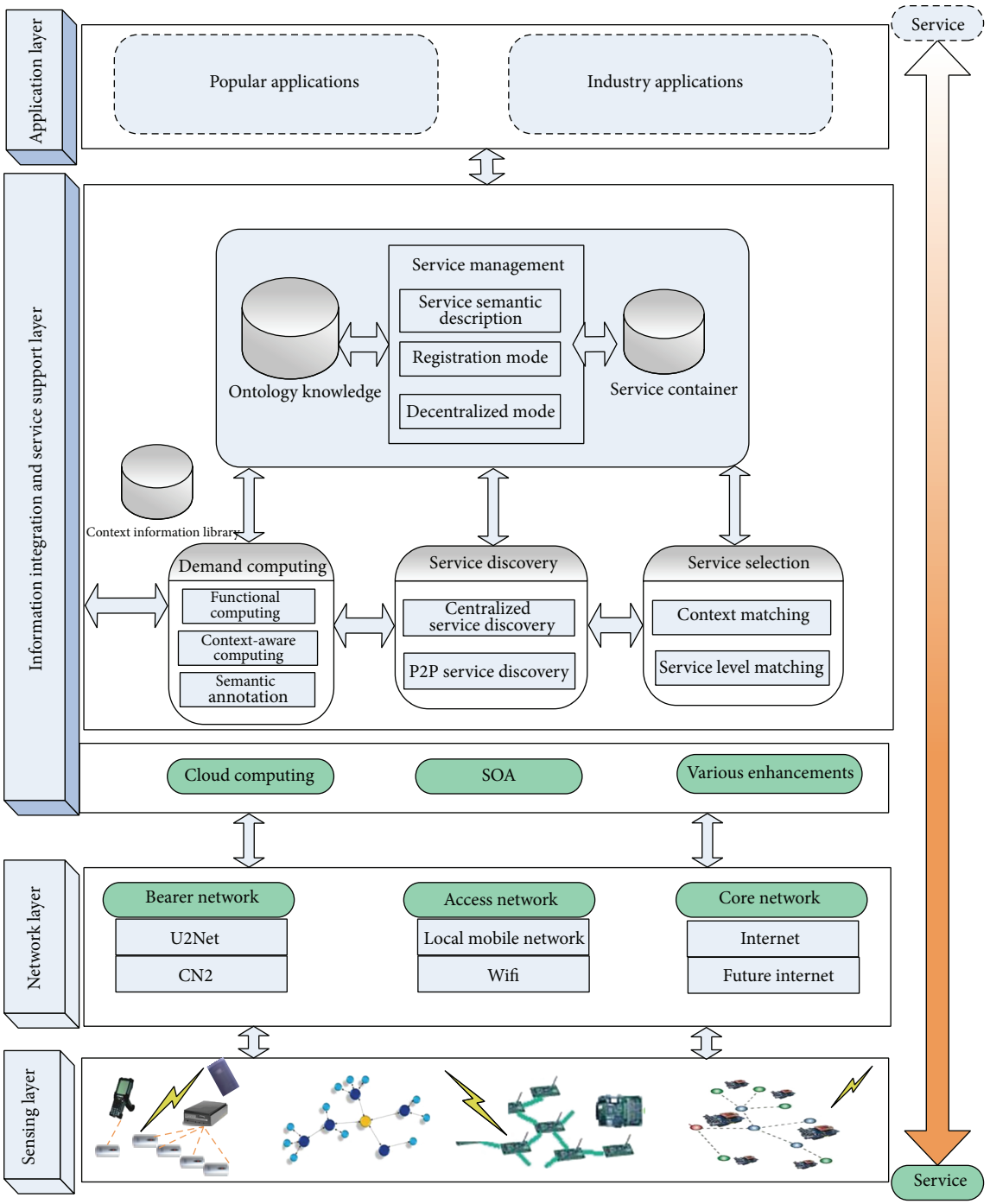

Figure 2: Architecture of IoT service platform.

services could often be formed in a loose P2P network, focusing on the optimization of service search strategy.

(iv) Service selection modules: they select the most adaptive and personalized services from the advertising services, which are obtained from service discovery meeting the functional needs to achieve the personalized pushing on demand. They contain context matching and service level matching.

(v) Service container: it can load services. It is a semantic-based UDDI repository mainly for the registration center model that contains all registered service documentation. The platform can just divide and set the user's region of interest because it cannot predefine the set of services to be discovered for the decentralized mode.

(vi) Ontology knowledge: it is used to store ontology and concepts related to IoT service platform and the relationship

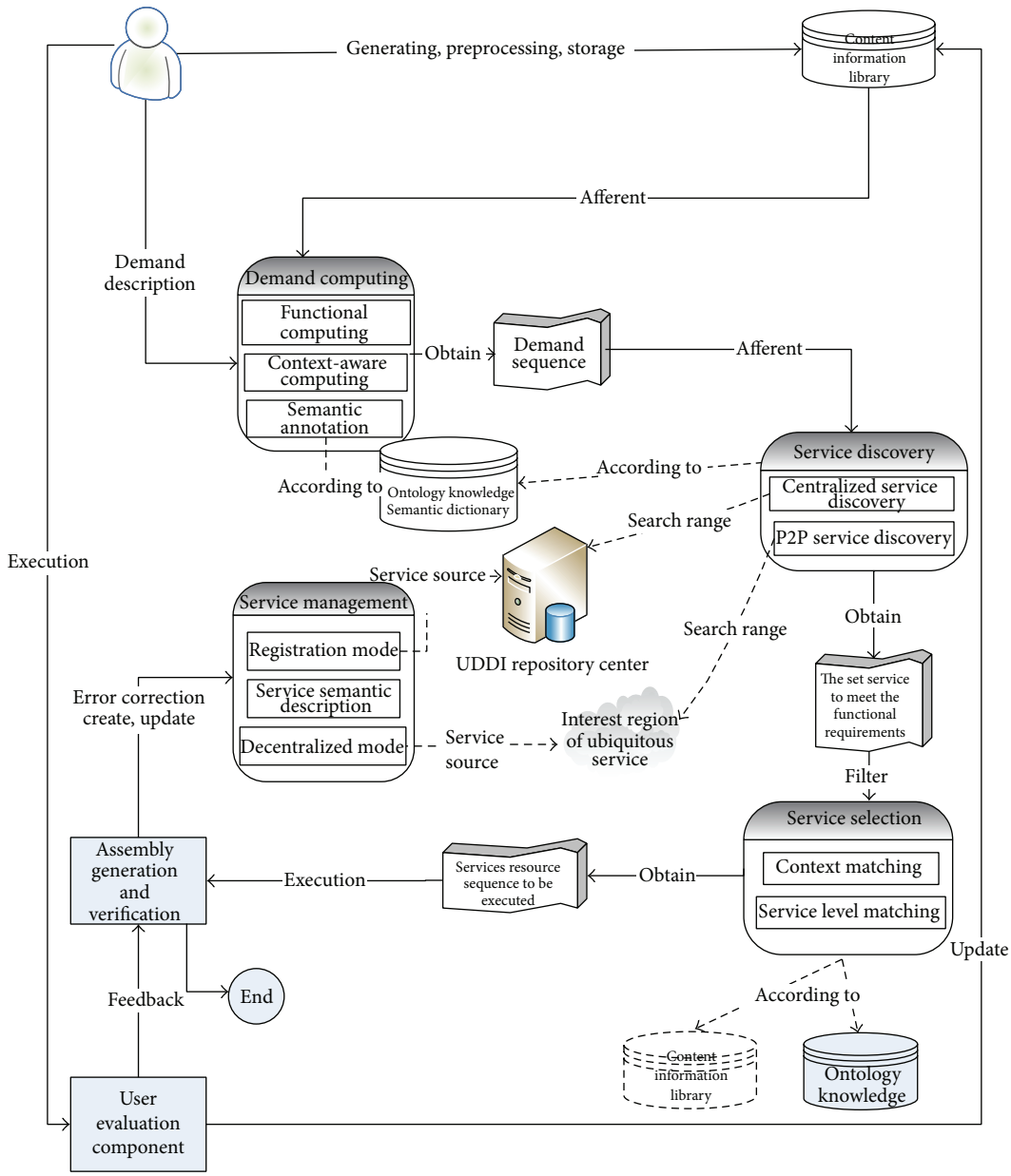

Figure 3: Main flow of service support platform.

between the concepts. It is the core foundation of platform to support semantics.

(vii) Context information library: it is used to store both the original context information and the high-level context after fusion.

Application layer is to show the specific business service in particular areas, including two aspects of the application, both popular applications and industry applications.

3.3. The Main Flow of Each Module of Service Support Layer's Coordination. Figure 3 shows the process of platform processing user's requests. Here the user is the "object" concept instead of the traditional person, which can be any entity in real world.

The key steps in the process are described below.

(i) We can get the user's functional and contextual needs through the interactive between user and demand computing module. Alternatively, we can carry out data mining in the original context information obtained directly through the context-aware computing module to get the implicit functional requirements and context.

(ii) Select a service request description document in turn. First the system should predict in which services source collection could find the suitable service. If it is in a UDDI Services library, it will search and match through a centralized service discovery algorithm. Then it will find a valid service that can meet the functional requirements and constraints. If the searching is in an interest region of

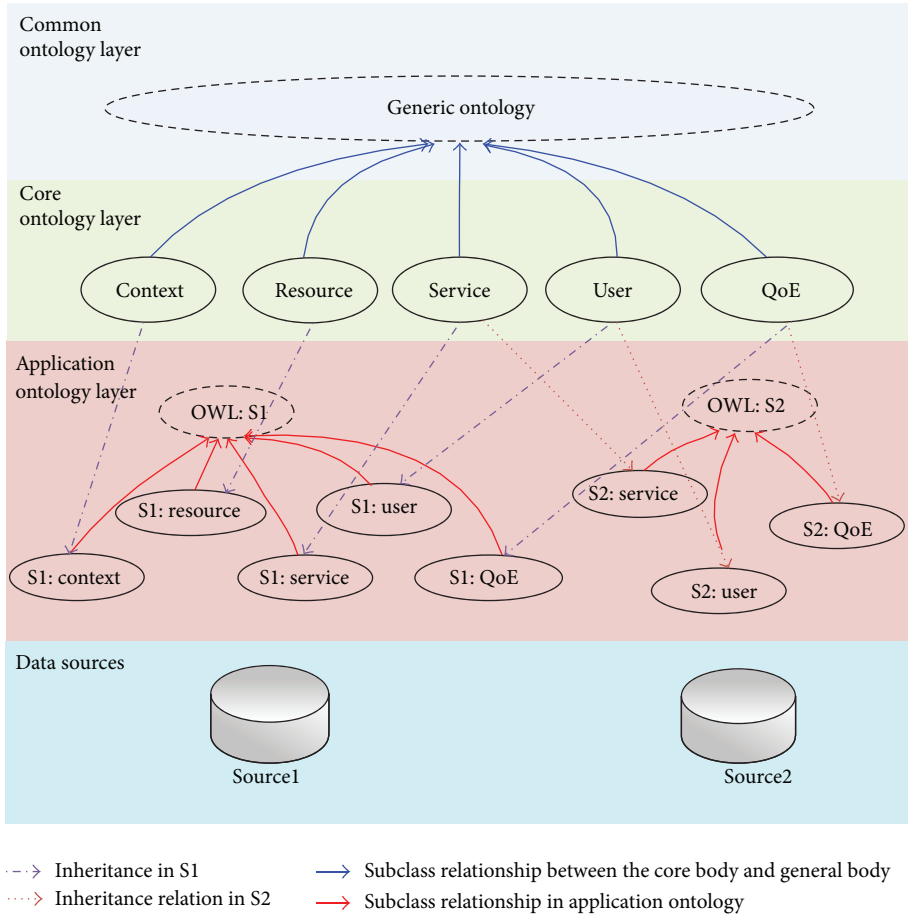

Figure 4: Three-layer ontology model.

ubiquitous service, it needs to call the P2P service discovery algorithms. Platform lets user request client be a Peer by adaptive configuration operation and adds it to a P2P service network formed in the service area which is an interest to the user. The peer can look up and match using P2P service discovery algorithm in the network. In particular, when the user required services cannot be found in the registry, it can also open the P2P service discovery module to continue the search in the ubiquitous service interest areas. Either way, finally we will return a service set sequences that meet the functional requirements and start the next phase of service selection.

(iii) Screen the service sets obtained from the service discovery. It uses more fine-grained personalization selecting based on the user's various context features and service levels. Eventually it returns the most suitable service required sequences adapted to user personalization.

(iv) Take the selected service to users for execution after the assembly verification. The user can choose whether to evaluate after performing the service. Results of the evaluation are returned to service management by the user evaluation component, updating the user's relevant context information.

4. Key Models and Algorithms

4.1. IoT Services Ontology Model Was Constructed. Because of the dynamic of IoT service, it is difficult to construct a single ontology. In order to meet the requirements of the unified sharing and flexible expansion of the related ontologies for IoT service, we propose a three-layer IoT ontology model, as Figure 4 shows.

IoT service ontology includes the generic ontology layer, the core ontology layer, and the application ontology layer from the top to the bottom. In this structure, the lower ontology puts forward the expressing need to the upper ontology. And the upper ontology provides the lower ontology with the design principles. We identified each specific ontology in core ontology layer of the model, namely, the user ontology, service ontology, QoE (quality of experience) ontology, context ontology, and resource ontology. The modeling of each specific ontology in core ontology layer could guide the modeling of service application ontology in the field of various industries. These core ontologies provide the necessary infrastructure to form a moderate coverage for the application ontology for the different information sources. The application ontology layer contains the specific

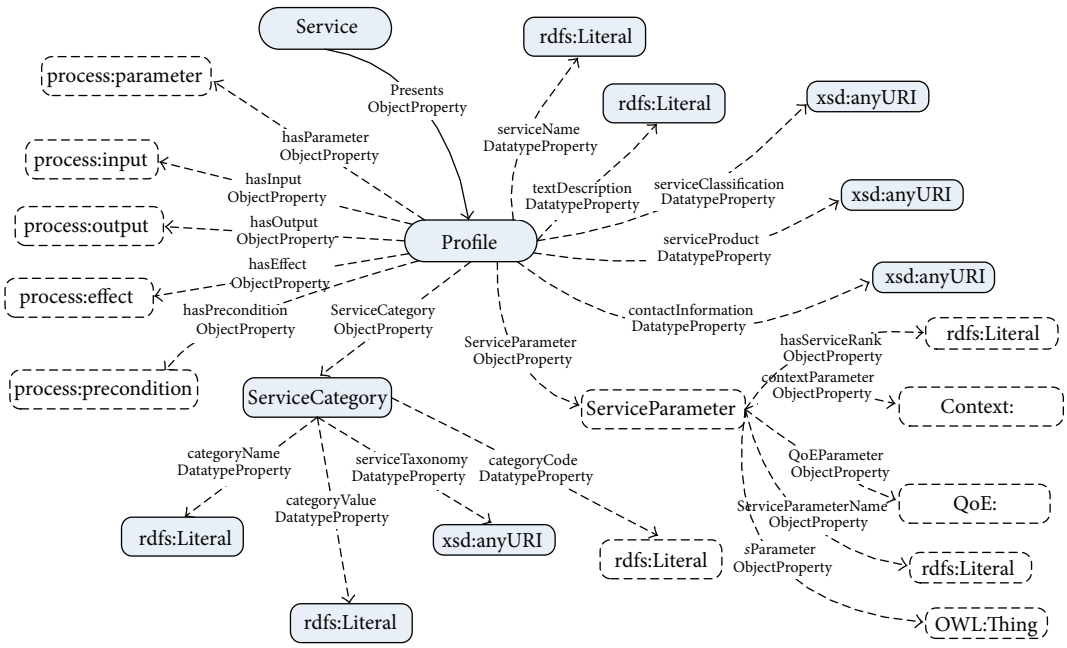

Figure 5: Service profile in OWL-S^{iot}.

application ontology based on the scene-specific information sources, according to the guiding of the core ontology to form a low coverage.

4.2. Semantic-Based IoT Service Description Language OWL-S^{iot}. By contrasting the existing typical Web service semantic description language, OWL-S [16] was selected as a foundation for the evolution of language for its better adaptability and wide application. In order to meet the dynamic flexibility requirements of IoT service description, by some professional processing for OWL-S, a suitable description language for IoT service called OWL-S^{iot} was formed. By Service Parameter in OWL-S, the context ontology and QoE ontology are introduced to form the IoT Service Profile in OWL-S^{iot}, as shown in Figure 5. At the same time, in order to meet the demand of simple cutting, we mark the unnecessary properties of service profile in the resource-constrained environment. These attributes are shown as dashed box in Figure 5.

4.3. Service Management Strategy Both in Registry Mode and Self-Organization by Subcluster. IoT service contained both the traditional common services and the new ubiquitous service. So it needs to be managed, respectively. For the former, we give the service registry architecture based on semantics, as shown in Figure 6.

For the latter, in order to save energy consumption of nodes and to not significantly affect the connectivity of the network, we put forward a self-organization management strategy using sleep scheduling based on "game of life." The main idea and flow of this strategy are described as follows.

(i) Complete the election of service cluster head the election of cluster head takes into account not only random, also the energy consumption of the cluster head [17]. First, the cluster head needs to judge whether they are requested in the last round. If they are, they continue to serve as the cluster head. Otherwise, calculate the node's residual energy threshold in the current round according to (1). Each cluster head node judges whether its current residual energy is greater than the threshold value. And if it is, it continues to be a cluster head. Otherwise, it becomes a common node in the next round. Consider

$$E_{\text{residual}} = E_{\text{initial}} \times \left(1 - \frac{r}{n}\right), \tag{1}$$

where "E_{residual}" is the threshold of residual energy in this round, "E_{initial}" is the average of the initial energy of each node, "r" is the current round, and "n" is the preestimated total rounds.

Then we need to compute a random threshold value in current round to satisfy the randomness of cluster head. The judgment threshold value can be calculated by (2). Then each node generates a random number and judges whether this number is less than the judgment threshold value. If less, then it was elected cluster head, otherwise it becomes a common node. Consider

$$T_{(n)} = \frac{P}{1 - P \times [r \bmod (1/P)]}, \tag{2}$$

where "P" is the percentage of cluster head in all cluster nodes and "r" is the current round.

(ii) The formation of service clusters: cluster head informs all nodes within its communication area that it had become the cluster head by broadcasting. Cluster node will independently judge whether to join the cluster. Because, for a cluster node, it may receive many notification messages from different cluster heads, then it needs to select the nearest cluster head to join to reduce the number of distance-hop

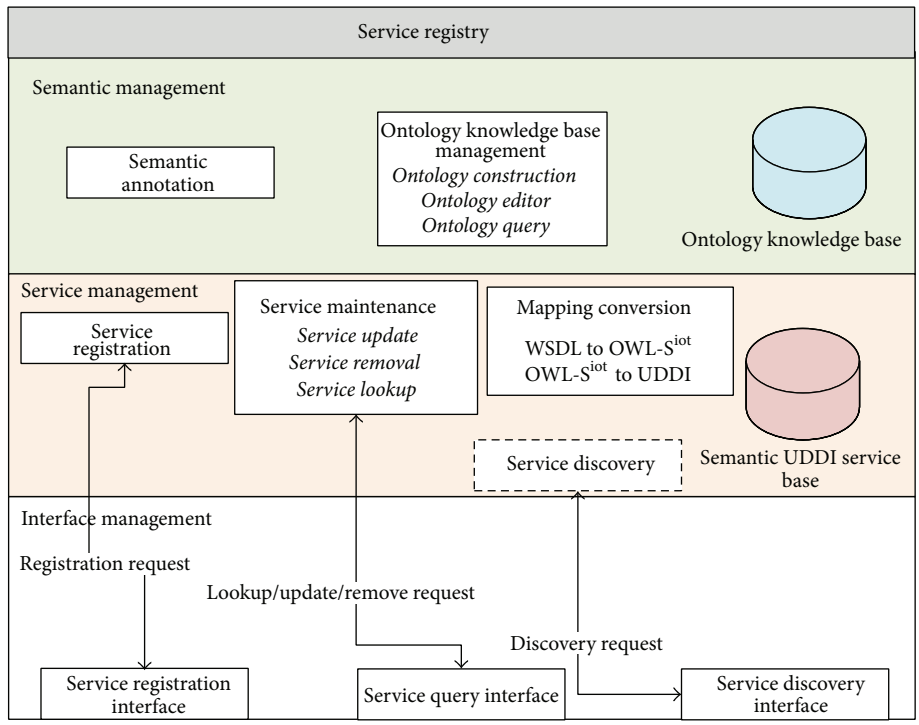

Figure 6: Architecture of service registry.

communication to save energy. Cluster head determines whether the request message is sent to itself after receiving the request message to join its cluster. If so, it allows the node to join the cluster and sends a confirmation message to the requesting node. After a period of interaction, the cluster head could establish a list of the information in this cluster.

(iii) Establishment of a service catalogue: after the formation of the cluster, the cluster node needs to send its service description information to the cluster head. Cluster head finishes collecting service information to create a catalogue of the cluster. Cluster head is responsible for interacting with the client's service request, as well as interacting with other cluster heads.

(iv) Sleep scheduling model based on the "game of life": Initially, the communications functions of all nodes are in the open state. In this period, the cluster node needs to communicate with the cluster head to confirm its identity [18]. If it is the node that needs to execute the customer's service requests, whether the status of the node is working or not in the former round, its state in the next round will be working. And it must immediately communicate and interact with clients to meet the customer's service request. Other cluster nodes without being requested for service by customers will determine their status the next time period based on the "game of life" sleep scheduling rules. The steps are as follows:

(1) determine the survival and birth rule set; they are a set of positive integers;

(2) determine their neighbor nodes "m."

(3) If its last status was working, and $m \leq$ Birth, the status of the node in the next round is working; if $m >$ Birth, then the status of the node in the next round is dormant. If its last status was dormant, and $m \leq$ survival, the status of the node in the next round is working; that is to say, this node will be resurrecting; if $m >$ survival, then the status of the node in the next round is dormant.

After many tests, only when survival value is 1 or 2, birth value is 3; the evolution of the "game of life" sleep scheduling model is relatively stable. And in other situations, the evolution is in shock mode or decay mode. These two modes for this paper do not have research value [17]. So we select the two relatively stable rules "S1B3" and "S2B3."

(v) The interaction process for service request [5]: Figure 7 describes the whole interaction process when a client wants to request the service cluster to service it [19]. We assume that cluster heads have been selected, and the nodes in the cluster have been carried out in accordance with the rules of the sleep schedule.

When a node is in a cluster, it must register its service in the cluster head first. Then it will determine its status in the next period "t" based on sleep scheduling rules. If its status is working, it will listen to the client's request forwarded by cluster head. If its status is dormant, it will send its dormancy request to the cluster head to see if it could go to sleep at time "t". If the current cluster head node does not receive any client requests of the node's service, then allow the node to go to sleep by sending an acknowledgment response. After receiving the dormancy response from the cluster head, the cluster node will be dormant in the following "t" period.

If a client wants to search for a service, it needs to send a request among the cluster heads in the network. The request

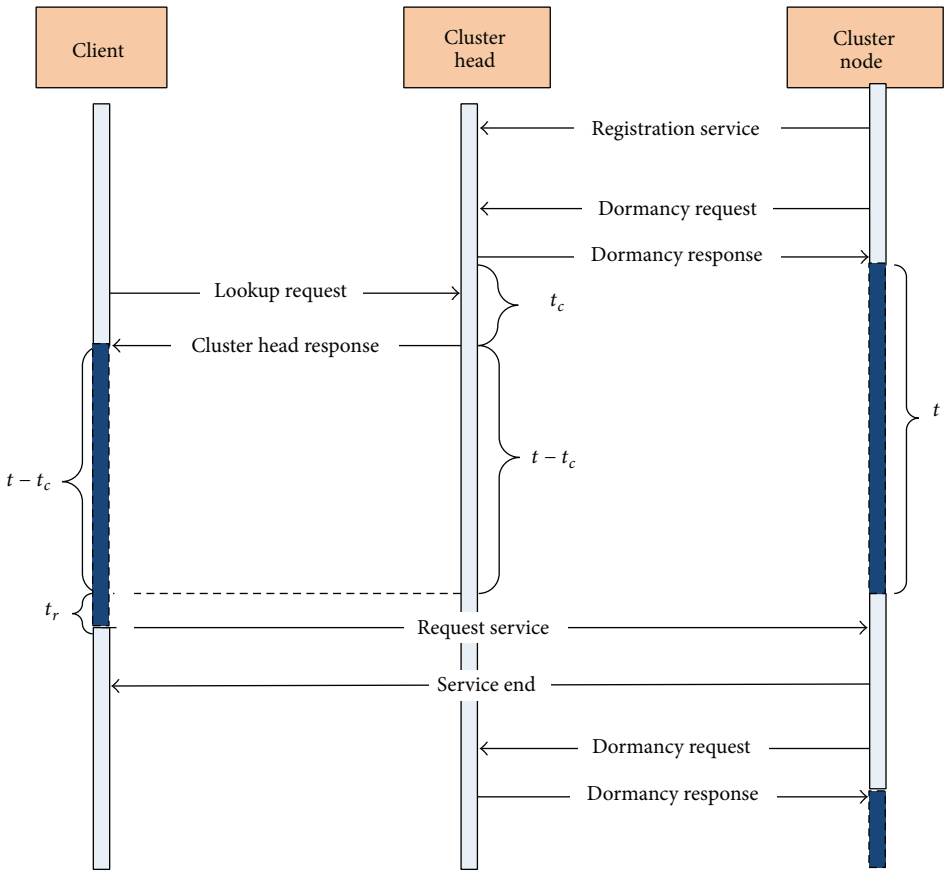

Figure 7: Interaction process for service request.

packet may be transmitted by a hop or multihop between the cluster heads. According to the received service request, each cluster head will find its service catalogue. If the service exists in its service catalogue, then select all the services which meet the requirements to establish a list with the service description. Every service description includes the delayed time ($t - t_c$) that the service could be contacted. The list is sent back to the client. Through this list, the client can choose the most suitable service based on its target. In selecting a service, the client waits for a specified time to contact the service. In this time period, the client can sleep in order to save energy. Before contacting the client service can randomly generate a delay t_r to avoid conflicts with other clients.

(vi) Analysis of the impact of the service interaction delay under this strategy: we use "java" to do a simulation experiment. The amount of service nodes is 100, and we assume that each node provides a service. In the experiment, we ignore the energy management cluster head, periodically and randomly generate cluster heads. We Adopt S1B3 and S2B3 sleep scheduling mechanism for the node to achieve status transition between working and dormant [5]. The time interval of cluster head election is defined as 9 s. The amount number of client requests is a multiple of 10 between 1 and 300. All experiments were repeated 20 times to calculate the average delay time [19]. Each client will wait for a random time interval (selected in [0 s, 9 s]) to broadcast a random service request. If the node that can provide the service is working at this time, it will directly interact without delay. Otherwise, wait for the service node to be awake. As soon as the node is awake, it will form a single request/response interaction with the client. We have designed a service discovery mechanism with no sleep as a comparison method, called nondormancy methods. In order to facilitate the calculation, we ignore the cluster head election delay and state transition delay. Figure 8 shows the S1B3, S2B3, and nondormancy service request interactive delays.

(vii) Nondormancy algorithm has no additional delay besides the base communicating delay, so that it appears as a constant delay. Interaction delay in S1B3 and S2B3 mechanisms is generated when the client waits for the providing service node to wake. For less client requests, the average delay is high. With the increase of the requests by the client, the delay of the interaction is shortened. And the trend is infinitely close to nondormancy algorithm. That is because with the increasing of client's requests, the awake time of the node is longer. In extreme cases, the nodes can be all awake. The connecting delay would be shortened when the service is requested by the client.

5. Conclusion

We propose a semantic-based IoT service framework based on semantic, IoT service features and design principles of IoT

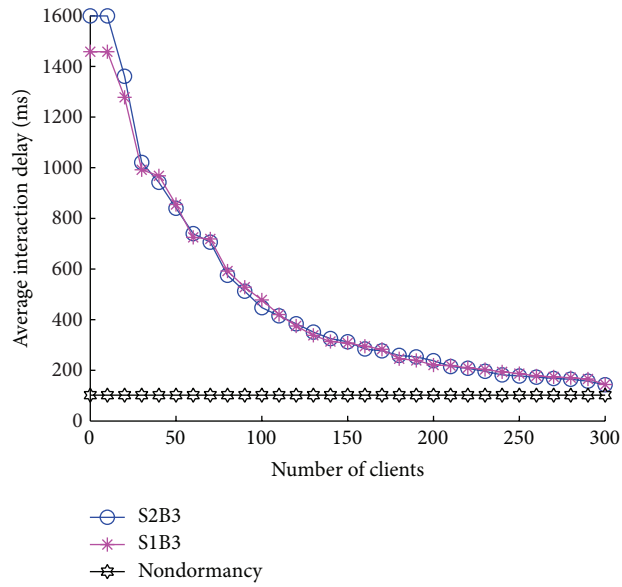

FIGURE 8: The service interaction delay under this strategy.

services framework. The framework is conducted a detailed decomposition referring to the reference standard of IoT architecture. Then we present a semantic-based cross-layer IoT service platform to meet service integration, sharing, and interoperability requirements. And we give the platform's architecture and workflow. Especially, we give the relative fundamental work about semantics, including the model of IoT service ontology and the semantic description method of IoT service. Also we give, respectively, the service registry architecture based on semantics for the traditional common services and self-organization management strategy using sleep scheduling based on "game of life" to save energy consumption of nodes for the new ubiquitous service. The platform extends the range of service source. It could provide the service not only by service providers, but also by normal users.

Conflict of Interests

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

Acknowledgments

This work is supported by Grants from programs of higher-level talents of Inner Mongolia University (nos. 125126, 115117, and 135138), National Natural Science Foundation of China (nos. 61261019, 61262082, and 61272412), scientific projects of High School of Inner Mongolia (no. NJZY13004), Ph.D. Programs Foundation of Ministry of Education of China (no. 20120061110044), Ministry of Education Humanities and Social Sciences Planning Project (no. 13YJAZH130), and Natural Science Foundation of Jilin Province (no. 20130101074JC). The authors wish to thank the anonymous reviewers for their helpful comments in reviewing this paper.

References

[1] H. berndt, *Towards 4G Technologies: Services with Initiative*, John Wiley & Sons, New York, NY, USA, 2008.

[2] Y. H. Liu, *Introduction to Internet of Things*, Science Press, Beijing, China, 2010.

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

[4] S. Reddy, V. Samanta, J. Burke, D. Estrin, M. Hansen, and M. Srivastava, "Mobisense—mobile network services for coordinated participatory sensing," in *Proceedings of the International Symposium on Autonomous Decentralized Systems (ISADS '09)*, pp. 231–236, Athens, Ga, USA, March 2009.

[5] B. Jia, *Research on Semantic-Based Service Architecture and Key Algorithms for the Internet of Things*, Jilin University, 2013, (Chinese).

[6] S. Alam and J. Noll, "A semantic enhanced service proxy framework for internet of things," in *Proceedings of the IEEE/ACM International Conference on Green Computing and Communications & IEEE/ACM International Conference on Cyber, Physical and Social Computing (CPSCom '10)*, pp. 488–495, Hangzhou, China, December 2010.

[7] J. Li, Y. Shvartzshnaider, J. A. Francisco et al., "Enabling internet-of-things services in the mobility first future internet architecture," in *Proceedings of the 13th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM '12)*, pp. 1–6, 2012.

[8] W. Wang and J. V. Sesmero, "Report on reference architecture IoT service creation and provision," EC FP7 Project IoT.est, 2012.

[9] J. V. Sesmero and L. L. Muñiz, "Report on use-case scenarios and requirements for the internet of things," EC FP7 Project IoT.est, 2012.

[10] W. Wang, P. Barnaghi, G. Cassar et al., "Semantic sensor service networks," in *Proceedings of the IEEE Sensors Conference*, 2012.

[11] Z. Song, A. A. Cárdenas, and R. Masuoka, "Semantic middleware for the internet of things," in *Proceedings of the 2nd International Internet of Things Conference (IoT '10)*, pp. 1–8, December 2010.

[12] N. Kong, X.-D. Li, W.-M. Luo, and B.-P. Yan, "Model of the resource addressing in the internet of things," *Journal of Software*, vol. 21, no. 7, pp. 1657–1666, 2010 (Chinese).

[13] H. M. Chen, L. Cui, and K. B. Xie, "A comparative study on architectures and implementation methodologies of Internet of things," *Chinese Journal of Computer*, vol. 36, no. 1, pp. 168–188, 2013 (Chinese).

[14] W. Wang, S. De, and A. Lehmann, "Semantic description framework for IoT services," EC FP7 Project IoT.est, 2012.

[15] Chinese Sensor Network National Standard Working Group, Advances in the technology architecture and standards system about Internet of Things [EB/OL], (Chinese), http://wenku.baidu.com/view/4ea7e769a45177232f60a288.html.

[16] OWL-S: Semantic Markup for Web Services[EB/OL], 2004, http://www.w3.org/Submission/OWL-S/.

[17] J. Wang, *Research on Topology Control Algorithm in Delay Tolerant Wireless Sensor Networks*, Jilin University, 2012, (Chinese).

[18] Y. J. Yang, B. Jia, and J. Wang, "An improved algorithm for LEACH protocol in wireless sensor network," *Journal of Beijing University of Posts and Telecommunications*, vol. 36, no. 1, pp. 100–104, 2013 (Chinese).

[19] G. Schiele, C. Becker, and K. Rothermel, "Energy-efficient cluster-based service discovery for ubiquitous computing," in *Proceedings of the 11th Workshop on ACM SIGOPS European Workshop (EW '11)*, Leuven, Belgium, September 2004.