Efficient semantic-based IoT service discovery mechanism for dynamic environments

Sameh Ben Fredj, Mathieu Boussard, Daniel Kofman, Ludovic Noirie

Efficient semantic-based IoT service discovery mechanism for dynamic environments

Sameh Ben Fredj
Alcatel-Lucent Bell Labs
route de Villejust
91620 Nozay, France
sameh.ben_fredj
@alcatel-lucent.com

Mathieu Boussard
Alcatel-Lucent Bell Labs
route de Villejust
91620 Nozay, France
mathieu.boussard
@alcatel-lucent.com

Daniel Kofman
Telecom ParisTech
23 avenue d'Italie
75013 Paris, France
daniel.kofman
@telecom-paristech.fr

Ludovic Noirie
Alcatel-Lucent Bell Labs
route de Villejust
91620 Nozay, France
ludovic.noirie
@alcatel-lucent.com

***Abstract*—The adoption of Service Oriented Architecture (SOA) and Semantic Web technologies in the Internet of Things (IoT) enables to enhance the interoperability of devices by abstracting their capabilities as services and enriching their descriptions with machine-interpretable semantics. This facilitates the discovery and composition of IoT services. The increasing number of IoT services, their dynamicity and geographical distribution require to think about mechanisms to enable scalable and effective discovery. We propose in this paper a semantic based IoT service discovery mechanism that supports and adapts to the dynamicity of IoT services. The discovery mechanism is distributed over a hierarchy of semantic gateways. Within a semantic gateway, we implement mechanisms to dynamically organize its content over time, in order to minimize the discovery cost. This cost is measured in terms of numbers of service-request matching operations performed in a gateway to find suitable services. Results show that our approach enables to maintain a scalable and efficient discovery and limits the number of updates sent to a neighboring gateway.**

I. Introduction

According to the Internet of Things (IoT) vision [6] [1], billions of objects will be connected by 2020 and have computation capabilities to interact and cooperate with other objects or users. Moreover, with the large adoption of the Service Oriented Architecture (SOA), real world devices will be able to offer their capabilities as web services [8].

A crucial challenge for the success of this vision is to enable the discovery of IoT services, among thousands of others, matching a set of requirements and expectations. Using models such as OWL-S[1] or WSMO[2], semantic web services can be used to provide rich descriptions about functionalities of IoT services and facilitate their discovery and composition [7].

A common practice in semantic service discovery is to match the semantic input/output signature of a service to a request to select a suitable service among others in a repository. In systems with a large number of IoT services, retrieving services which semantically match a discovery request can become a challenging task [10]. Organizing services into clusters accelerates their discovery [11], as it limits the scope of search. When considering device mobility, service discovery becomes more challenging where service clustering needs to adapt to IoT service dynamicity in order to improve service discovery. In this paper, we propose an approach based on a hierarchy of semantic gateways to accelerate the discovery for IoT semantic web services in a dynamic context. We develop in each semantic gateway mechanisms that adapt to changes induced by service dynamicity. Groups of similar services are created through incremental clustering. The clustering is optimized over time to minimize the discovery cost.

The paper is organized as follows: section II gives an overview of the approach. Section III details the mechanisms to dynamically organize the content of the gateway and accelerate discovery. Section IV provides an experimental evaluation and section V presents the related work.

II. Approach overview

In our approach that we developed in a static context in [2], we model an IoT environment as a tree hierarchy of smart spaces (e.g., country, region, city, streets, buildings, rooms). Each smart space is embodied by a semantic gateway, which is a software component maintaining information about the IoT services in its scope and processing discovery requests. At the lowest level nodes the actual IoT service descriptions are registered in the semantic gateway (the networking-level discovery of these services is out of scope here) while at upper levels only aggregated information is maintained.

A semantic gateway N performs clustering and aggregation of its content. For each of its clusters K, a representative $W_{K,N}$ and radius $r_{K,N}$ are computed based on a quasi-metric Q. The aggregated information (i.e., the representative $W_{K,N}$ and the radius $r_{K,N}$) of each semantic gateway is then sent to its parent gateway where the process is performed recursively. In each gateway, a routing table is built, representing each cluster by its representative, its radius, and the list of its elements.

During discovery request processing, the semantic similarity between a request and a service is measured based on the quasi-metric Q. At a gateway N for a cluster K, an incoming request is matched with the cluster representatives. For each cluster K, we use a specific decision threshold $Td_{K,N}$ to decide to further compare the request with each element of

[1]http://www.w3.org/Submission/OWL-S/
[2]http://www.w3.org/Submission/WSMO/

the cluster or not. For higher level nodes, a fine-grained comparison is performed with elements of a cluster, using the associated $Td_{K',N'}$ decision threshold of the originating node and cluster for these elements. This enables to select the originating node to forward the request to. For lowest level nodes, a matching threshold is used to return final service descriptions matching the request.

An initial prototype of this architecture and the related discovery mechanism were detailed in [2], in a static context. In this work, we consider a dynamic environment and we develop new mechanisms to adapt the discovery mechanism to service mobility.

III. Dynamic management of gateway content

In this part, we detail the dynamic management mechanisms for a semantic gateway to support service dynamicity and reduce search cost. We define first an incremental clustering to create and update clusters of similar services based on service arrivals and departures. Second, we optimize the defined clustering in terms of number of clusters and number of services per cluster in order to minimize the search cost.

A. Dynamic clusters creation and updating

During clusters creation, we aim to make new services quickly discoverable while limiting the number of semantic matching operations required to insert them in the gateway' clusters. In the following, we detail the different service management actions and considered conditions to insert a new service within a node or to delete it. For a node N, $\mathcal{K} = \{K_i, i \in \{1, \cdots, M\}\}$ is a set of clusters in the node N. For a cluster $K \in \mathcal{K}$, $W_{K,N}$ and $r_{K,N}$ are respectively its representative and radius.

1) Service arrival cases: We consider a new service S_1 that will be inserted into the node N. We set $K_{min} = \arg\min_{K \in \mathcal{K}} [Q(S_1, W_{K,N})]$. K_{min} is the cluster in N whose representative is the closest to S_1 and we note d_{min} this distance: $d_{min} = Q(S_1, W_{K_{min},N})$. The arrival of S_1 can represent different scenarios (See Figure 1):

a) If $d_{min} \leq r_{K_{min},N}$, then S_1 is inserted into K_{min} and no change is made within the cluster.
b) If not (a) and $\exists K \in \mathcal{K} \setminus \{K_{min}\}$ where $r_{K,N} \geq Q(S_1, W_{K,N})$, then S_1 is inserted into $K = \arg\min_{K \in \mathcal{K} \setminus \{K_{min}\}} [Q(S_1, W_{K,N})]$, no change is made within the cluster.
c) If not (a) and not (b) and $d_{min} < Q(W_{K,N}, W_{K_{min},N})$, then we enlarge the radius of K_{min} to include the service S_1. We have: $K_{min} = \{W_{k_{min},N}, r_{k_{min},N} = d_{min}\}$.
d) If not (a), not (b) and not (c), then a new cluster $K' \in \mathcal{K}$ is created with a representative $W_{K',N} = S_1$ and a radius $r_{K',N} = 0$.

2) Service departure cases: In the following, we consider a case of service departure where S_1 is leaving the cluster K. We consider $d = Q(S_1, W_{K,N})$. The departure of S_1 can represent different scenarios (See Figure 2):

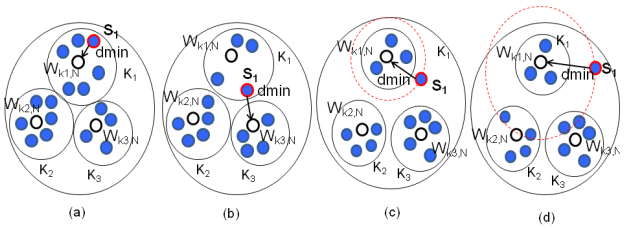

Figure 1. 2D representation of service arrival cases

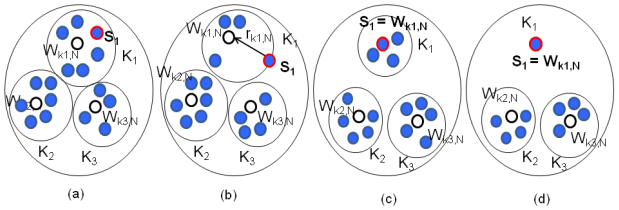

Figure 2. 2D representation of service departure cases

a) $S_1 \neq W_{K,N}$ and $d < r_{K,N}$ then no change is made within the cluster.
b) $S_1 \neq W_{K,N}$ and S_1 is the only service where $d = r_{K,N}$, then we recalculate the radius of the cluster upon S_1 departure to optimize it.
c) $S_1 = W_{K,N}$ and $|K| > 1$ then we consider a virtual representation of S_1 that can be with a service request during service search but it can not be part of the final set of found services that will be returned to a user. No change is made within the cluster.
d) $S_1 = W_{K,N}$ and $|K| = 1$ then we delete the cluster K of node N.

B. Clustering optimization within a node

With the mechanism described in the previous section, we incrementally modify the set of clusters. However, the number of clusters and the number of services per cluster vary during service mobility and can affect the search cost (i.e., the number of matching operations). We define in the following a mechanism that can optimize the distribution of services when it is needed.

1) Minimal Search cost: To evaluate our clustering, we define a referential model of an *ideal clustering* that we intend to approach during service mobility in order to minimize the service search cost. In an *ideal clustering*, a request will not visit any cluster if there is no service matching it and will visit only one cluster if there is a service matching it (See definition 1).

Definition 1: Ideal Clustering.
We consider N_s services distributed over a set of clusters $\mathcal{K} = \{K_i, i \in \{1, \cdots, n\}\}$. R a request sent to $\mathcal{K}$ to find a matching with a service S. β is the number of clusters visited by the request R to search for a matching service S. $\mathcal{K}$ is an ideal clustering if and only if, for any request R:

- *$\beta = 1$ when there is a service S that matches R,*
- *$\beta = 0$ when there is no service S that matches R.*

During service mobility, the number of formed clusters may increase greatly and some clusters may become very populated in terms of number of services. Thus, service distribution within a node impacts the search cost. We should mention that we consider the assumption that for a uniform distribution of requests, the more populated a cluster is, the more likely it will be visited by a request. However, for the same number of services, two clusters have equal probability to be visited by a request. In the following, we define an optimal distribution of services within a node (see proposition 1), where we consider an optimal number of services and an optimal average number of services per cluster to minimize the search cost.

Proposition 1: Optimal service distribution.
We consider a node N with N_s services distributed over the group of clusters $\mathcal{K} = \{K_i, i \in \{1, \cdots, M\}\}$ formed following the process described in section III-A. We consider m_i the number of services per cluster K_i, thus $\sum_{i=1}^{M} m_i = N_s$. A set of requests R are sent to the node N. We define $\langle M_r \rangle$ the average matching cost per request R, n_c the number of clusters, $\langle m_s \rangle$ the average number of services per cluster and $\langle \beta \rangle$ the average number of visited cluster per request. We also define $\langle M_{r_{min}} \rangle$ the minimal average matching cost per request, $n_{c_{op}}$ the optimal number of clusters per node and $\langle m_{s_{op}} \rangle$ the optimal average number of services per cluster. Based on definition 1, we can write:

$$\langle M_r \rangle = n_c + \langle \beta \rangle \times \langle m_s \rangle \text{ and } N_s = n_c \times \langle m_s \rangle .$$

$\langle M_r \rangle$ is minimal for:

$$n_{c_{op}} \approx \sqrt{\langle \beta \rangle \times N_s}, \langle m_{s_{op}} \rangle \approx \sqrt{\frac{N_s}{\langle \beta \rangle}},$$

and we have: $\langle M_{r_{min}} \rangle = 2 \times \sqrt{\langle \beta \rangle \times N_s}$.

Proof: We have: $\langle m_s \rangle = \frac{N_s}{n_c}$ and we consider the function $f : \mathbb{R}_+^* \mapsto \mathbb{R}$ where $f(n) = n + \langle \beta \rangle \times \frac{N_s}{n}$. By studying the variation of the function f, we can determine the value of $n_{c_{op}}$ for which the function f is minimal. ■

Based on definition 1 of the *ideal clustering*, we have $\langle \beta \rangle \in]0, 1[$. Since the *ideal clustering* is a limit that we intend to approximate, we fix $\langle \beta \rangle = 1$ in the following, which is the worst case for $\langle M_{r_{min}} \rangle$. Moreover, we will consider the integer part of $n_{c_{op}}$ and $\langle m_{s_{op}} \rangle$ which are very close to their real values and we will keep the same notation.

2) Reclustering mechanisms: During service mobility, node content changes dynamically, which impacts the number of clusters n_c and the number of services per cluster m_s. In our approach to optimize service distribution within a node, we define mechanisms of re-clustering where a cluster splitting is performed to limit the maximum number of services per cluster (that we note m_{Max}) and a cluster merging is performed to be close to the optimal number of clusters $n_{c_{op}}$. The purpose is to approach the minimal average matching cost per request $\langle M_{r_{min}} \rangle$ defined in proposition 1. Continuous merging and splitting involve additional matching operations and updates that impact the system scalability. To reduce the matching cost due to merge and split operations, we define limits on the number of clusters per node $n_{c_{Limit}}$ and the maximum number of services per cluster $m_{Max_{Limit}}$ (See definition 2). The goal is to always have $n_c \leq n_{c_{Limit}}$ and $m_{Max} \leq m_{Max_{Limit}}$. We are aware that by doing so, we will have: $\langle M_r \rangle > \langle M_{r_{min}} \rangle$. By adding margins to the optimal values $n_{c_{op}}$ and $m_{s_{op}}$ in proposition 1, we expect $\langle M_r \rangle$ to still be advantageous with comparison to a centralized approach where the request is matched with all services within a gateway.

Definition 2: Reclustering limits.
We consider a node N with N_s services distributed over n_c clusters. We consider $\langle m_s \rangle$ the average number of services per cluster and m_{Max} the maximum number of services in a cluster. $n_{c_{op}}$ is the optimal number of clusters and $\langle m_{s_{op}} \rangle$ the optimal number of services per cluster. We set limits on the number clusters per node ($n_{c_{Limit}}$) and on the number of services per cluster ($m_{Max_{Limit}}$) by defining margins ϕ and γ respectively on $n_{c_{op}}$ and $\langle m_{s_{op}} \rangle$ as follows:

- $n_{c_{Limit}} = (1 + \phi) \times n_{c_{op}}$,
- $m_{Max_{Limit}} = (1 + \gamma) \times \langle m_{s_{op}} \rangle$.

For margins, we fix $\gamma = 1$. This allows after a split (based on the splitting approach described below) to have 2 clusters with approximately $\langle m_{s_{op}} \rangle$ services in each. For ϕ, we consider values $\in [0.5, 1]$ in order to limit the number of merging operations.

We consider a node N and N_s services distributed over the group of clusters $\mathcal{K} = \{K_i, i \in \{1, \cdots, M\}\}$. We consider m_i the number of services per cluster K_i. Based on the defined margins and re-clustering limits, cluster merging and splitting are performed as follows:

- If it exists a cluster k_i where $m_i > m_{Max_{Limit}}$, then we split k_i into two new clusters K_{new_1} and K_{new_2} where $m_{new_1} \approx m_{new_2}$. To do so, we consider first the 2 most dissimilar services S_1 and S_2 within the cluster K_i based on a distance Q' that we obtain through symmetrization of the quasi-metric Q. We start with $K_{new_1} = \{S_1\}$ an $K_{new_2} = \{S_2\}$. Services in $K_i \setminus \{S_1, S_2\}$ will be matched against S_1 and S_2 based on the quasi-metric Q and associated with the most similar one until we have: $m_{new_1} \approx m_{new_2} \approx \langle m_{s_{op}} \rangle$. Finally, we calculate the representative and radius of K_{new_1} and K_{new_2}.
- If $n_c > n_{c_{Limit}}$ then we merge K_i and K_j defined by $(K_i, K_j) = \underset{(K_a, K_b) \in \mathcal{E}}{\arg\min} \left[Q\left(W_{K_a,N}, W_{K_b,N}\right) + r_{K_b,N} \right]$ where $\mathcal{E} = \left\{ (K_a, K_b) \in \mathcal{K}^2 \setminus m_a + m_b \leq m_{Max_{Limit}} \right\}$. After cluster merging, we calculate the representative and radius of K_{merg}. Because $\gamma = 1$, we always have $\mathcal{E} \neq \emptyset$.
- In other cases nothing is done.

IV. Experimental Evaluation

In the following, we describe the prototype and the used data set, we define the considered performance metrics and we analyze the results.

A. Prototype and data set

We deployed a prototype of a lowest level semantic gateway as a Java Web Application hosted in a Tomcat server. We

implemented the incremental clustering mechanism described in section III-A and the clustering optimization mechanism described in section III-B. A process of service arrivals and departures was simulated in the gateway. The service arrival process is a Poisson process with an arrival rate λ. The time spent by a service in a node has an exponential distribution with parameter μ (with $\lambda > \mu$). The average number of services within a node is then : $\langle N_s \rangle = \lambda/\mu$. The requests arrival process is a Poisson process with arrival rate η. In real system, we expect that for a given period of time T the request arrival process is faster than the service arrival process (i.e., $\eta >> \lambda$).

Since real and rich data sets about IoT semantic service descriptions are not very common nowadays, we used for our experimental evaluation the known OWL-S service retrieval Test Collection (OWL-S TC v3.0[3]). The data set is composed of 7 different service categories. We chose among them 3 service categories to be represented in the node to ensure service diversity.

B. Metrics

To evaluate our approach we measured the total matching cost M_T which is the sum of the discovery cost (i.e., the number of service-request matching operations involved during service discovery M_{rT}), the mobility matching cost (i.e., the number of service matching operations involved during service arrivals and departures) and the clustering optimization matching cost (i.e., the number of service matching operations involved during cluster merging and splitting). We compare it to the discovery cost M_{rT} to evaluate the cost engendered by the dynamic management mechanisms. We also measured the relative rate of generated updates sent by a node to its parent, which is the ratio between the number of events sent by a child node to its parent node and the total number of events received by a child node over a given period (i.e., total number of service arrivals and departures events).

C. Tests and results

For the measurements, we considered $\langle N_s \rangle = 400$ semantic service descriptions within the gateway. We simulated service mobility (i.e., service arrivals and departures) during $100 \times T_0$, where $T_0 = 1/\mu$. We consider n_r and n_m respectively the number of requests and the total number of service arrivals and departures that happen during T_0. We choose $n_r = 5 \times n_m$and we have $n_m = 2 * \langle N_s \rangle$. During simulation, we measured the different matching costs (i.e., M_{rT} and M_T) and the number of generated updates. To facilitate simulations and to estimate the impact of node content changes on the request matching cost, we did not perform a continuous request arrival process as for services, instead, we measured the average matching cost over 100 requests, every $10 \times T_0$, then we calculated the total request matching cost M_{rT} over requests. We start with an empty node that is filled with service arrivals representing singletons. The routing table of the node is constructed and updated in parallel.

[3]http://projects.semwebcentral.org/projects/owls-tc/

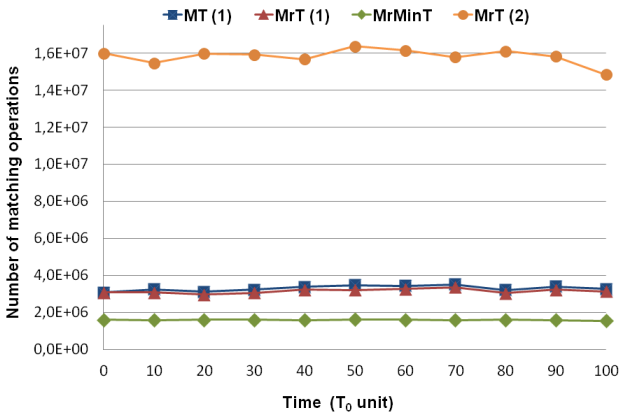

Figure 3. Number of matching operations in approach (1) and approach (2).

Merging clusters starts when there is about 10 singletons in the gateway (at this stage, the optimized clustering become more advantageous in terms of discovery cost than considering singletons). When the node content reaches 400 services, we start measurements. Each new service is chosen randomly among 3 different service categories.

For service arrivals and departures process, we fixed $\mu = 0.0025$ and $\lambda = 1$. For reclustering mechanisms, we fixed $\langle \beta \rangle = 1$ as explained in III-B1, to calculate the optimal values of $n_{c_{op}}$ and $\langle m_{s_{op}} \rangle$. We take a margin $\gamma = 1$ for the maximum number of services per clusters $m_{Max_{Limit}}$, as explained in III-B2. For the maximum number of clusters per node $n_{c_{Limit}}$, we choose a margin $\phi = 0.6$ following test results.

We measured the evolution of the overall matching cost M_T and we compared it to the evolution of the total discovery cost $M_{rT} = nr \times \langle M_r \rangle$, during $100 \times T_0$. Results are shown in figure 3. We compared the discovery cost of our approach (1) to the discovery cost of approach (2) representing a non clustered node. We have $M_{rT}(2) \geq 4 \times M_{rT}(1)$ over time. We also compared the discovery cost M_{rT} of both approaches to the minimal value $M_{r_{minT}} = n_r \times \langle M_{r_{min}} \rangle$ defined in proposition 1. We can see in our approach that $M_T(1)$ and $M_{rT}(1)$ are quasi-constant over the time thanks to the mechanisms that we defined. We have $M_T(1) \approx M_{rT}(1)$ which means that the matching costs generated by the management mechanisms are negligible compared to the discovery cost. Moreover, we have $M_{rT}(1) \approx 2 \times M_{r_{minT}}$. This gap between the two values is due to the margins that we added to the optimal values of $n_{c_{op}}$ and $\langle m_{s_{op}} \rangle$ to reduce the number of reclustering operations. Also, cluster merging and splitting degrade the quality of clusters which increases the average number of visited clusters per request.

We measured the relative rate of generated updates sent by a node to its parent. These updates are due to changes in node content that impact its routing table and need to be notified to a parent node to update its content and ensure an efficient service discovery. Results are shown in figure 4. We can see with our approach, a node generates a low relative rate of updates to its parent ($< 1\%$). Thus, our mechanisms allow for a higher level node to receive mobility events at a rate 100

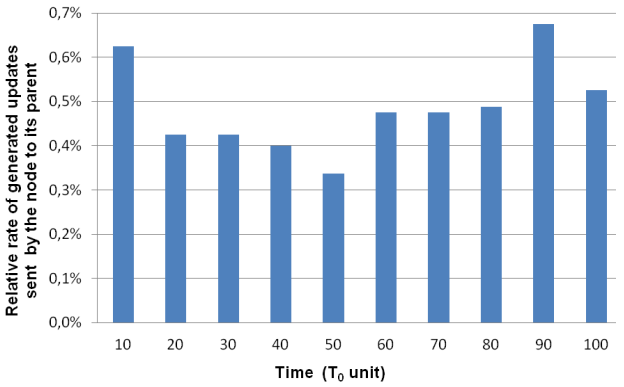

Figure 4. Relative rate of generated updates sent by a node to its parent.

times slower than the one received at a lower level node.

V. Related work

This section presents some related works about techniques that have been investigated in order to improve semantic based service discovery.

METEOR-S [12] is based on a classification system of ontology concepts where peers sharing similar concepts are grouped together and process requests related to these concepts. Used ontologies need to be known in advance to create peers categories, which is problematic in dynamic context.

Clustering approaches [3] try to find information in unlabeled data. These approaches have been used to improve semantic service discovery performances, specially to reduce the scope of search.

Statistical clustering (e.g., K-means, BIRCH, Hierarchical clustering) has been investigated in clustering web services based on similarity metrics. This clustering presents high complexity when dealing with high dimensional data structures such as semantic web services, formed by multiple concepts. Moreover, they do not adapt well to dynamic contexts.

Probabilistic methods for service clustering have been used in [9] and [4]. The dimensionality of the service description is reduced as each service can be described in terms of latent factors. These latent factors are used to group services into clusters. Any newly added service can be clustered with a direct calculation and without requiring to re-calculate the latent variables. Although probabilistic clustering methods have a reasonable search complexity and adapt well to dynamic context, they are significantly less expressive than our approach since they do not take advantage of the logic based description of services. Moreover, the clustering is not optimized over time.

Developed Incremental clustering techniques [5] [13] have been used to cluster dynamic data sets in a metric space. Data joins a cluster if some predefined criteria are verified. Otherwise, a new cluster is created to represent the object. Incremental clustering performs clusters merging under specific conditions in order to preserve the consistency of clusters. They lack split mechanisms which can be useful in certain applications. Moreover, most approaches are considering clustering text documents and not exploiting semantic services.

VI. Conclusion

In this paper, we presented an approach to enable efficient IoT semantic service discovery in a dynamic context and over a hierarchy of semantic gateways. Within a gateway, we defined an incremental clustering mechanism to create groups of similar services. This clustering is optimized over time in terms of number of clusters and number of services per cluster. Results show that the discovery cost is quasi-constant over time and is well optimized with comparison to a non-clustered approach. The matching cost generated by the management mechanisms is negligible compared to the discovery cost. Moreover, the relative rate of generated updates sent by the gateway to its parent is very low. This means that, in the hierarchy of gateways, higher level gateways will be less affected by service dynamicity and changes that happen in lower level gateways. As a next step, we will extend our testing to the whole architecture. We will generalize the mobility support mechanisms to all nodes of a hierarchy and evaluate the whole system.

Acknowledgment

Part of the work presented in this paper has been carried out at Laboratory of Information, Network and Communication Sciences (LINCS, www.lincs.fr).

References

[1] L. Atzori, A. Iera, and G. Morabito, *The Internet of Things: A survey*, Computer Networks (2010), 2787–2805.

[2] S. Ben Fredj, M. Boussard, D. Kofman, and L. Noirie, *A Scalable IoT Service Search Based on Clustering and Aggregation*, Green Computing and Communications (GreenCom), 2013 IEEE and Internet of Things (iThings/CPSCom), IEEE International Conference on and IEEE Cyber, Physical and Social Computing, 2013, pp. 403–410.

[3] P. Berkhin, *A Survey of Clustering Data Mining Techniques*, Grouping Multidimensional Data, 2006, pp. 25–71.

[4] G. Cassar, P. Barnaghi, and K. Moessner, *Probabilistic Methods for Service Clustering*, Proceedings of 4th International Workshop on Service Matchmaking and Resource Retrieval in the Semantic Web, 2010.

[5] M. Charikar, C. Chekuri, T. Feder, and R. Motwani, *Incremental Clustering and Dynamic Information Retrieval*, SIAM J. Comput. (2004), 1417–1440.

[6] E. Dave, *The internet of things: How the next evolution of the internet is changing everything*, CISCO white paper (2011).

[7] S. De, P. Barnaghi, M. Bauer, and S. Meissner, *Service Modelling for Internet of Things*, FedCSIS, 2011, pp. 949–955.

[8] D. Guinard, V. Trifa, S. Karnouskos, P. Spiess, and D. Savio, *Interacting with the SOA-Based Internet of Things: Discovery, Query, Selection, and On-Demand Provisioning of Web Services*, IEEE T. Services Computing (2010), 223–235.

[9] J. Ma, Y. Zhang, and J. He, *Efficiently finding web services using a clustering semantic approach*, CSSSIA, 2008, pp. 5.

[10] S. Ben Mokhtar, D. Preuveneers, N. Georgantas, V. Issarny, and Y. Berbers, *EASY: Efficient semAntic Service discoverY in pervasive computing environments with QoS and context support*, Journal of Systems and Software (2008), 785–808.

[11] Christian Platzer, Florian Rosenberg, and Schahram Dustdar, *Web service clustering using multidimensional angles as proximity measures*, ACM Transactions on Internet Technology (2009), 11.

[12] K. Verma, K. Sivashanmugam, A. P. Sheth, A. A. Patil, S. A. Oundhakar, and J. A. Miller, *METEOR-S WSDI: A Scalable P2P Infrastructure of Registries for Semantic Publication and Discovery of Web Services*, Information Technology and Management (2005), 17–39.

[13] S. Young, I. Arel, T. P. Karnowski, and D. Rose, *A Fast and Stable Incremental Clustering Algorithm*, ITNG, 2010, pp. 204–209.