Semantic-based middleware for IoT service search

Sameh Ben Fredj

2014-ENST-0063

EDITE - ED 130

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

TELECOM ParisTech

Spécialité Informatique et Réseaux

présentée et soutenue publiquement par

Sameh BEN FREDJ

le 27 octobre 2014

Intergiciel Sémantique pour la recherche des services de l'Internet des Objets

Directeur de thèse : **Daniel KOFMAN**
Co-encadrement de la thèse : **Mathieu BOUSSARD**

T H È S E

TELECOM ParisTech
école de l'Institut Mines-Télécom - membre de ParisTech
46 rue Barrault 75013 Paris - (+33) 1 45 81 77 77 - www.telecom-paristech.fr

Telecom ParisTech
Alcatel Lucent-Bell Labs France

Semantic-based middleware for IoT service search

Sameh BEN FREDJ

A thesis submitted and defended for the degree of
Doctor of Philosophy from Telecom ParisTech

Academic supervisor : Prof. Daniel KOFMAN
Industrial supervisor : Mr. Mathieu BOUSSARD
Industrial Co-supervisor : Mr. Ludovic NOIRIE

27 October 2014

A mes chers parents, mon frère et ma soeur,
A mon cher époux.

Acknowledgement

During my PhD, I had the opportunity work on a passionate subject which is the Internet of Things and with amazing people from whom I learned a lot. Without these people, this PhD experience could not have the same taste. For that, I would like first to thank my supervisors and managers from Telecom ParisTech and Alcatel-Lucent Bell Labs that gave me the chance to do this PhD. They have always supported me, guided me, advised me and challenged me repeatedly. Thank you for your patience and your precious help.

I also would like to thank my friends and colleagues from Telecom ParisTech, Alcatel-Lucent Bell Labs and outside. Thank you for your support, advices and for the great moments that we shared together.

This thesis is also dedicated to my family : my mum, my dad, my sister and my brother. You have always supported me, believed in me and encouraged me to reach my dreams. I could not find the right words to thank you enough for all what you did for me and to express how much I love you and how blessed I am to have you in my life.

I would like also to thank my amazing family-in-law that has always supported me during my PhD.

Finally, I could not be here today without the great support of my dear husband. You have been amazingly patient and supportive during this entire journey. I cannot thank you enough for all the sacrifices you made for me, for your strong believe in me and you strong support even in times when I no longer believed in myself and wanted to give up. You have always been the light that guided me to the final point. I love you.

Abstract

With the advent of the Internet of Things (IoT), we are facing a proliferation of connected devices distributed over physical locations, so called smart spaces. A major part of these devices are mobile and can roam from one smart space to another. The adoption of Service-Oriented Architecture (SOA) enables to abstract device capabilities as services that we call IoT services. These services can be composed with others and offered to users (i.e., humans or other devices). Enabling an easy and seamless discovery of these IoT services is crucial for the success of the Internet of Things. The characteristics of IoT services, such as their sheer number, their heterogeneity and their dynamicity induced by the mobility of the related devices, make discovering them a challenge. To enhance the interoperability of IoT services, some solutions rely on the use of Semantic Web technologies to enrich their descriptions with machine-interpretable semantics. This enables to reason on semantic service descriptions and to support accurate and flexible service search. However, in large scale and dynamic systems, retrieving services which semantically match a search request can become an arduous task due to induced computational costs that impact systems' scalability.

In this thesis, we propose a system architecture and the associated mechanisms to enable efficient and scalable semantic-based IoT service discovery supporting dynamic contexts. Our approach relies on distributed semantic gateways that represent different smart spaces and host semantic descriptions of the services offered by devices in these smart spaces. The discovery system embeds clustering and information aggregation mechanisms based on a quasi-metric that we defined. Based on a semantic routing, the proposed system allows searching and retrieving all suitable services matching an incoming request, and reduces the service search cost by limiting useless service-request semantic matching operations. Moreover, we consider the dynamic aspect of IoT environments where devices are roaming from one smart space to another. We propose mechanisms to monitor the system's evolution to ensure that the properties of the discovery system are still verified. We prove theoretically that our method ensures a flexible and exhaustive search of all matching services within the system. Experimental results show that our method enables to reduce greatly the semantic matching cost (i.e., the number of semantic matching operations) during service search. Moreover, the defined mobility management mechanisms limit the number of system updates and keep the semantic matching cost at a low level over time.

Keywords : Internet of Things, middleware, IoT services, Semantic Web, service search, scalability, service mobility.

Résumé

Avec l'avènement de l'Internet des objets, nous observons une prolifération des appareils connectés répartis dans des emplacements physiques, appelés des espaces intelligents. La majorité de ces appareils connectés sont mobiles et peuvent se déplacer d'un espace intelligents vers un autre. La large adoption de l'Architecture Orientée Services (SOA) permet l'abstraction des capacités des appareils connectés en services qu'on appelle services de l'Internet des objets. Permettre une découverte facile et homogène de ces services est crucial pour le succès de l'Internet des objets. Les caractéristiques des services de l'Internet des objets, tels que leur nombre, leur hétérogénéité et leur dynamicité induite par la mobilité des appareils connectés, rendent leur découverte difficile. Pour améliorer l'interopérabilité des services de l'Internet des Objets, certaines solutions s'appuient sur l'utilisation des technologies du Web Sémantique pour enrichir leurs descriptions de services et les rendre interprétables par une machine ce qui rend leur recherche précise et flexible. Cependant, à grande échelle et dans un contexte dynamique, extraire des services qui répondent à une requête peut devenir ardu en raison des coûts de calcul induits.

Dans cette thèse, nous proposons une architecture de système et ses mécanismes associés pour permettre une découverte efficace et scalable des services de l'Internet des Objets, en se basant sur le Web Sémantique et en supportant des contextes dynamiques. Notre approche se base sur des passerelles sémantiques distribuées qui représentent différents espaces intelligents et enregistrent leurs services associés. Le système de découverte intègre des mécanismes de groupement, d'agrégation de l'information et de routage sémantique. Le système proposé permet de trouver tous les services qui répondent à une requête donnée en limitant les opérations inutiles d'appariement sémantiques. En outre, nous considérons l'aspect dynamique des environnements de l'Internet des Objets où les appareils sont mobiles. Dans ce contexte, nous proposons des mécanismes de suivi de l'évolution du système afin de s'assurer que ses propriétés de découverte sont toujours vérifiées. Nous montrons théoriquement que notre méthode assure une recherche flexible et exhaustive de tous les services existants dans le système et répondants à une requête donnée. Les résultats expérimentaux montrent que notre méthode permet de réduire considérablement le coût d'appariement sémantique. De plus, les mécanismes définis pour la gestion de la mobilité limitent le nombre de mises à jour du système et maintiennent un coût bas d'appariement sémantique au cours du temps.

Mots-clefs : Internet des Objets, intergiciel, services de l'Internet des Objets, Web Sémantique, la recherche des services, extensibilité, mobilité des services.

Synthèse en français

1-Introduction

1.1-Contexte et Motivation

Depuis son introduction par Mark Weiser en 1988, l'informatique ubiquitaire, aussi appelée informatique omniprésente, vise à intégrer les ressources informatiques dans les environnements physiques et joindre ainsi le monde physique au monde virtuel avec une vision de base qui est celle de créer des espaces intelligents dans lesquels les utilisateurs interagissent de façon transparente avec les services informatiques par le biais de dispositifs mobiles (par exemple smartphones, tablettes, PDA, etc.) [1].

Les progrès réalisés dans l'informatique embarquée ont beaucoup aidé au développement de cette vision. Le coût de l'électronique à incorporer dans les objets de tous les jours a fortement baissé ces derniers temps, ce qui a permis la connexion de périphériques de tous les jours (par exemple, les téléviseurs, alarmes, Horloges, lampes, réfrigérateurs, etc.) et aussi de petits objets (par exemple des capteurs, étiquettes RFID, etc.) créant ainsi l'Internet des Objets, où les objets deviennent accessibles via le réseau Internet.

Ceci a permis les multiplications des espaces intelligents et la génération de plusieurs nouveaux services qu'on appellera dans la suite, les services de l'Internet des Objets (on notera : services de l'IoT). L'utilisateur sera donc entouré par un grand nombre de services offerts par ses objets connectés et aura besoin de les trouver pour interagir avec eux. Cependant, les environnements de l'Internet des Objets sont caractérisés par le nombre volumineux des objets connectés, leurs mobilités et leurs hétérogénéités. Ces caractéristiques rendent la recherche de leurs services adjacents difficile.

Considérant ce contexte, le mécanisme de recherche doit également répondre à certaines exigences pour être efficace. En effet, avec un large volume de services IoT, il serait difficile pour l'utilisateur de connaitre les caractéristiques de tous les services autour de lui qui peuvent répondre à son besoin. Ainsi, le mécanisme de recherche doit être flexible dans les sens où il serait possible à l'utilisateur de donner une description vague du service recherché et que sur la base de cette description, le système de recherche serait capable de retourner des services qui correspondent exactement ou partiellement à la requête exprimée. Dans certains cas, il serait utile aussi que le système de recherche soit exhaustif, c'est à dire qu'il est en mesure de trouver tous les services répondant à la requête dans un périmètre de recherche donné. Enfin, le système de recherche doit pouvoir être extensible par rapport à un nombre volumineux de services considérés.

Sur la base de ce qui précède, la question que nous nous posons est : Comment construire un mécanisme de recherche efficace qui répond à ces exigences tout en prenant en compte les contraintes de l'enivrement IoT ?

Pour surmonter les problèmes d'hétérogénéité syntaxique des services de l'IoT, des approches reposent sur la sémantique des services pour les chercher. Cependant, l'utilisation de la sémantique dans un système à large échelle et avec des contraintes de mobilité peut causer des problèmes d'extensibilité qu'il faudra considérer.

1.2-Contribution

Pour répondre à ces problématiques, nous introduisons dans cette thèse une solution de découverte des services IoT basée sur la sémantique et répondant aux exigences de la flexibilité, exhaustivité et extensibilité. Les différentes contributions sont les suivantes :

- **Le Chapitre 2** donne un aperçu des efforts visant à régler le problème de l'hétérogénéité dans l'Internet des Objets. Il passe également en revue les solutions existantes dans l'IoT pour chercher des services en se basant sur leur sémantique.
- **Le chapitre 3** présente l'architecture de notre solution de découverte des services IoT sur la base de leur sémantique. Nous donnons d'abord un aperçu de l'architecture globale de la couche sémantique qui gère la recherche, la réception des requêtes de l'utilisateur et l'enregistrement des descriptions de services sémantiques. Nous détaillons ensuite les blocs fonctionnels de la partie de découverte des services. Enfin, nous introduisons le concept de passerelles sémantiques qui sont une instanciation de la couche sémantique.
- **Le chapitre 4** considère un contexte statique et détaille le mécanisme de découverte de service basé sur la sémantique. Le mécanisme de découverte est supporté par les passerelles sémantiques. Sur la base de groupement et d'agrégation des descriptions de services dans chaque passerelle sémantique, le système est capable de retourner tous les services qui répondent à une requête de l'utilisateur en limitant le nombre d'appariements sémantiques.
- **Le chapitre 5** considère un contexte dynamique caractérisé par la mobilité des services. Il détaille les mécanismes qui permettent une gestion dynamique du contenu des passerelles sémantiques qui change au cours du temps. Pour cela, des mécanismes de groupement incrémental et des mécanismes d'optimisation de ces groupements sont introduits pour mettre à jour le contenu des passerelles sémantiques et minimiser le coup de recherche des services.

2-La recherche des services de l'Internet des Objets : Etat de l'art

Dans ce chapitre, nous présentons un état de l'art des efforts développés pour la recherche des services de l'Internet des Objets. Ce chapitre se compose de 4 sous parties.

En premier lieu, nous présentons les efforts pour connecter les objets physiques au réseau Internet et exposer leurs services IoT à travers des architectures d'interlogiciel orientées services (SOA). Ces approches ont permis de résoudre des problèmes liés à l'hétérogénéité physique des objets.

En second lieu, nous présentons les efforts des descriptions de services par des technologies du Web Sémantique afin de surmonter les problématiques d'hétérogénéité syntaxique au niveau des descriptions des services de l'Internet des Objets. Un exemple des technologies du Web Sémantique considérées sont les ontologies et les modèles de descriptions des services Web Sémantique.

En troisième lieu, nous nous focalisons sur les méthodes de recherche sémantique en générale. Nous présentons des techniques basées sur l'appariement des descriptions des services sémantiques à l'aide des raisonneurs sémantiques, des méthodes basées sur l'appariement des sous parties du graph à l'aide des technologies de type SPARQL et enfin des méthode basées sur l'indexation des descriptions de services sémantiques comme celles utilisées dans les moteurs de recherche. Nous expliquons les principes de chaque technique de recherche.

En quatrième lieu, nous nous focalisons sur les solutions existantes de recherche de services de l'IoT en se basant sur la sémantique. Certaines approches implémentent les techniques de recherche décrites précédemment. Nous décrivons ces approches et nous analysons leur capacité à remplir les exigences définies dans le chapitre précédent, à savoir la flexibilité de la recherche, l'exhaustivité, l'extensibilité et leurs capacités à s'adapter à des environnements dynamiques. Notre analyse montre qu'aucune de ces solutions ne répond à toutes les exigences demandées du système de recherche.

Nous présentons dans la suite notre solution pour la recherche sémantique des services de l'IoT en répondant aux exigences de la flexibilité, l'exhaustivité, l'extensibilité et la capacité à s'adapter à la mobilité des objets.

3-Architecture d'Intergiciel pour la recherche des services de l'Internet des Objets

3.1-Présentation globale de l'architecture

L'Intergiciel proposé permet la découverte et facilite l'interaction entre les applications de l'Internet des Objets et les objets connectés. Il cache l'hétérogénéité matérielle et logicielle des dispositifs de l'Internet des objets et offre une vue uniforme au développeur des applications IoT. Le middleware permet la découverte des services des objets connectés. Nous nous focalisons sur cette fonctionnalité dans la suite de notre travail. L'Intergiciel permet de recevoir les requêtes de l'utilisateur envoyées à travers des applications de l'Internet des objets (par exemple de son Smartphone) pour trouver un service particulier. Pour cela, l'utilisateur peut saisir des mots-clés décrivant le service désiré (par exemple, les concepts à mesurer ou à agir dessus). L'utilisateur peut aussi spécifier un emplacement s'il veut effectuer une recherche de service basée sur la localisation. La Figure 1 est une esquisse de la vue globale de l'Intergiciel.

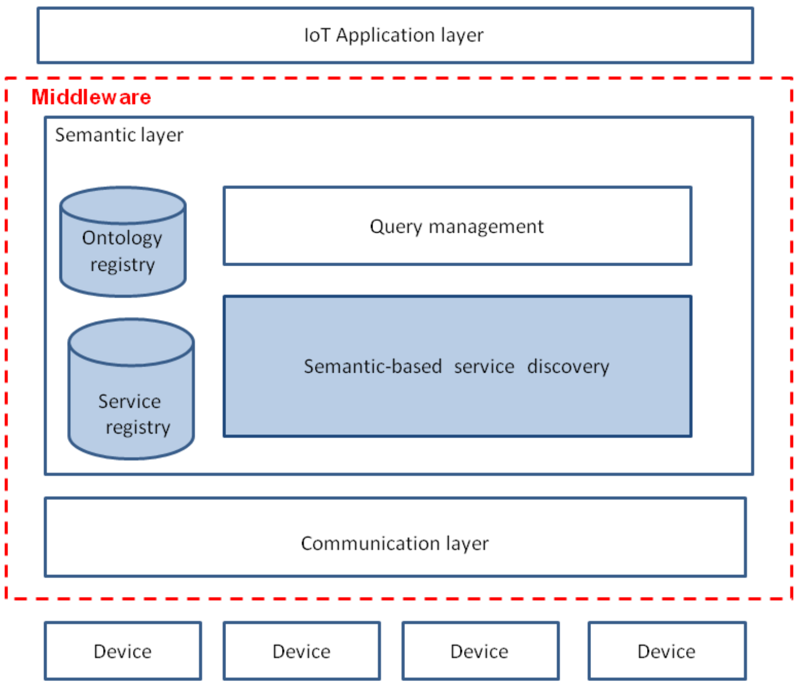


Figure 1 : Architecture globale de l'intergiciel pour la découverte des services en se basant sur leurs sémantiques.

L'Intergiciel est composé essentiellement de deux couches principales :

- **La couche de communication :** Elle offre deux fonctionnalités essentielles. Tout d'abord, elle fédère des protocoles de communication hétérogènes et permet la découverte des objets au niveau de la couche réseau. Typiquement, elle permet la détection des objets mobiles à l'aide des protocoles de découvert ([2], [3], [4]), à extraire des informations pertinentes sur les services offerts, à générer leurs descriptions sémantiques et à les enregistrer au niveau de la couche sémantique. En second lieu, cette couche de communication permet à l'utilisateur d'accéder au service demandé et à interagir avec les objets connectés sans se soucier des différents protocoles d'accès.
- **La couche sémantique :** La couche sémantique permet la découverte des services de l'IoT en se basant sur leurs descriptions sémantiques. Elle contient différents composants : le composant de gestion des requêtes de l'utilisateur, le registre des ontologies, le registre des descriptions des services sémantiques et le registre de découverte des services. Ces différents composants seront détaillés dans la partie suivante.

3.2-Présentation de la couche sémantique

Nous détaillons dans cette section la couche sémantique. Nous présentons d'abord le modèle d'information utilisé. Ensuite, nous décrivons les fonctionnalités et les interactions entre les différents composants de la couche sémantique.

Présentation du modèle d'information

Nous décrivons dans cette partie le modèle d'ontologie, des services et des requêtes utilisés au niveau de la couche sémantique. Dans notre approche, nous ne cherchons pas à fournir de nouveaux modèles sémantiques pour décrire les services de l'Internet des Objets. Cependant, nous voulons que notre solution puisse utiliser les modèles existants d'ontologies et de descriptions de services sémantiques.

Ontologie et concepts :

Dans notre approches nous considérons une ontologie $\mathcal{O}$ formée par un ensemble de concepts $\mathcal{C} = \{C_i, i \in \{1, \cdots, M\}\}$ liés entre eux par une relation binaire $\mathcal{R}$. Cette relation binaire décrit les relations de subsumption entre ces concepts. Ainsi, pour un couple de concepts $(C_i, C_j) \in \mathcal{C}^2$, on a :

- C_i est un super-concept de C_j si et seulement s'il existe un chemin entre $(\mathcal{C}, \mathcal{R})$ qui commence à C_i et se termine à C_j. On note $C_i \sqsupset C_j$.
- C_i est un sous-concept de C_j si et seulement si C_j est un super-concept de C_i. Nous notons $C_i \sqsubset C_j$.

Pour inclure les cas d'égalité dans l'espace de concepts équivalents, nous notons : $(C_i \sqsupseteq C_j) \Leftrightarrow (C_i \sqsupset C_j) \vee (C_i = C_j)$.

Services et requêtes :

Nous considérons que la description de service est formée par un ensemble de paramètres (fonctionnels et non fonctionnels) et que chaque paramètre est annoté avec un concept C_i de l'ontologie $\mathcal{O}$.

Nous considérons que la requête R est une description d'un service désiré et qu'elle est également représentée par un sous-ensemble de l'ensemble des concepts tels que défini pour un service

Les principaux composants de la couche sémantique

La Figure 2 donne une vue détaillée des principaux composants de la couche sémantiques. Nous décrivons dans la suite ces composants :

le composant de la découverte sémantique des services :

Ce composant est formé essentiellement par deux blocs fonctionnels :

- **l'enregistrement des services :** Ce bloc fonctionnel se charge de l'insertion des nouveaux services dans le registre des services ou de leur suppression suite à la détection de l'arrivée ou du départ d'un objet connecté dans l'espace physique considéré. Les descriptions de services sémantiques sont générées automatiquement après la détection d'un nouvel objet. L'enregistrement des services dans le registre ou leur suppression se fait d'une façon incrémentale. Les services dans le registre sont groupés en groupe de services similaires et ce groupement est optimisé au cours du temps en fonction de la mobilité des services.
- **La recherche des services :** Ce bloc fonctionnel est chargé d'identifier les descriptions de services qui répondent à une requête envoyée au système et renvoie des pointeurs à l'utilisateur pour lui permettre d'accéder aux services demandés. Une opération d'appariement

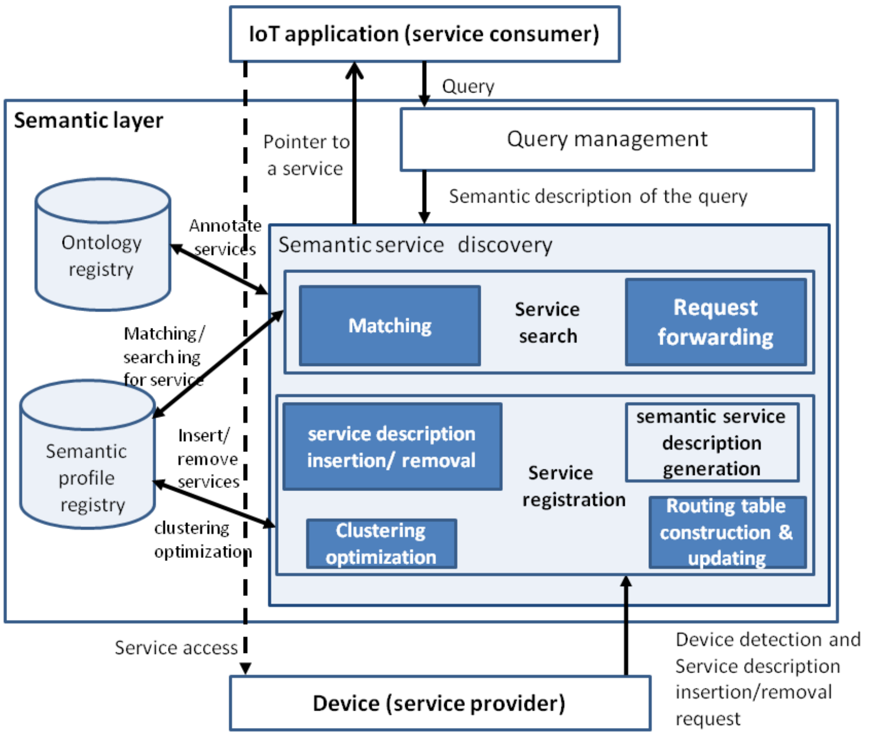

IoT application (service consumer)
Query
Semantic layer
Pointer to a service
Query management
Semantic description of the query
Ontology registry
Annotate services
Semantic service discovery
Matching
Service search
Request forwarding
Matching/ searching for service
Semantic profile registry
Insert/ remove services
service description insertion/ removal
Service registration
semantic service description generation
Routing table construction & updating
Clustering optimization
clustering optimization
Service access
Device (service provider)
Device detection and Service description insertion/removal request

Figure 2 : Les composants de la couche sémantique

sémantique et d'expédition de la requête d'une passerelle sémantique vers une autre (voir les passerelles dans la partie suivante) sont effectués pour assurer la recherche des bons services.

Le composant de gestion des requêtes : Ce composant reçoit la requête de l'utilisateur d'une application de l'Internet des Objets. La requête de l'utilisateur contient des informations sur le service désiré, où l'utilisateur précise des mots clés décrivant les paramètres fonctionnels du service (par exemple les concepts physiques à mesurer) et aussi les paramètres non-fonctionnel (par exemple, l'emplacement physique). Le composant de gestion des requêtes reçoit la demande de l'utilisateur, extrait les mots clés et génère une description de service sémantique liée à cette requête qui sera utilisée lors de la recherche des services.

Les registres : Nous considérons deux types de registres : le registre des ontologies et le registre des services. Le premier registre détient les ontologies utilisées décrivant la base de connaissance. Le dernier registre contient les descriptions sémantiques des services de l'IoT qui implémente des concepts des ontologies du registre précédent.

3.3-Déploiement des passerelles sémantiques

Nous donnons dans cette section une vue d'ensemble du déploiement de l'architecture de découverte des services dans un environnent IoT composé par un ensemble d'espaces intelligents interconnectés. Bien que notre solution soit destinée à être déployée dans un environnement à grande échelle (par exemple une ville), nous considérons dans cette section et à titre d'exemple illustratif, le cas d'un bâtiment intelligent.

Le bâtiment est composé de k étages et de l chambres dans chaque étage. Chaque chambre contient un nombre m d'objets connectés (par exemple des capteurs de température, des capteurs d'humidité, des capteurs de détection de présence) et des actionneurs (par exemple un projecteur, une lampe, des appareils de chauffage, etc.).

La couche de communication et la couche sémantique sont respectivement instanciées en passerelles physiques et sémantiques que nous décrivons ci-après.

Passerelles physiques

Les passerelles physiques assurent la communication avec les objets connectés autour d'elles. Elles agrègent différents protocoles de communications et permettent de communiquer avec les objets physiques malgré leurs hétérogénéités protocolaires. Chaque passerelle physique est associée à un lieu géographique spécifique (par exemple la salle de réunion, le bureau, le couloir, etc.) en s'appuyant sur leurs portées radios. Elles ont différentes interfaces (par exemple Bluetooth, Zigbee, Wifi, etc.) afin de détecter et se connecter avec les nouveaux appareils entrant dans l'emplacement physique et couvert par les passerelles physique.

Passerelles sémantiques

Les passerelles sémantiques sont des composants logiciels qui hébergent des descriptions de services sémantiques de l'IoT. Ces services sont liés à un emplacement spécifique. Les passerelles sémantiques se chargent d'effectuer la recherche des services sémantiques suite à la réception d'une requête. Ces passerelles sémantiques peuvent être soit centralisées en un seul serveur ou entièrement réparties entre plusieurs serveurs dans un scénario à grande échelle. Dans un environnement IoT donné, les passerelles sémantiques sont interconnectées et communiquent entre elles sur la base d'une topologie en arbre représentant les espaces physiques avec différents niveaux de granularité. Les passerelles sémantiques du plus bas niveau sont associées à des espaces contenant des objets physiques, tandis que les passerelles sémantiques de niveaux supérieurs sont associées à des espaces qui comprennent ceux du niveau inférieur.

Chaque nœud N dans le graph en arbre (Voir Figure 3) représente une passerelle sémantique qui représente un espace géographique. Nous caractérisons le graphe de nœuds par le nombre n de niveaux et le nombre de nœuds enfants d_i pour chaque nœud parent du niveau $i+1$ avec $i \in [1, n-1]$.

Association entre la passerelle physique et sémantique

Chaque passerelle sémantique du niveau le plus bas est associée à une passerelle physique. Dans notre approche, passerelles physiques et sémantiques sont immobiles et leur association est statique.

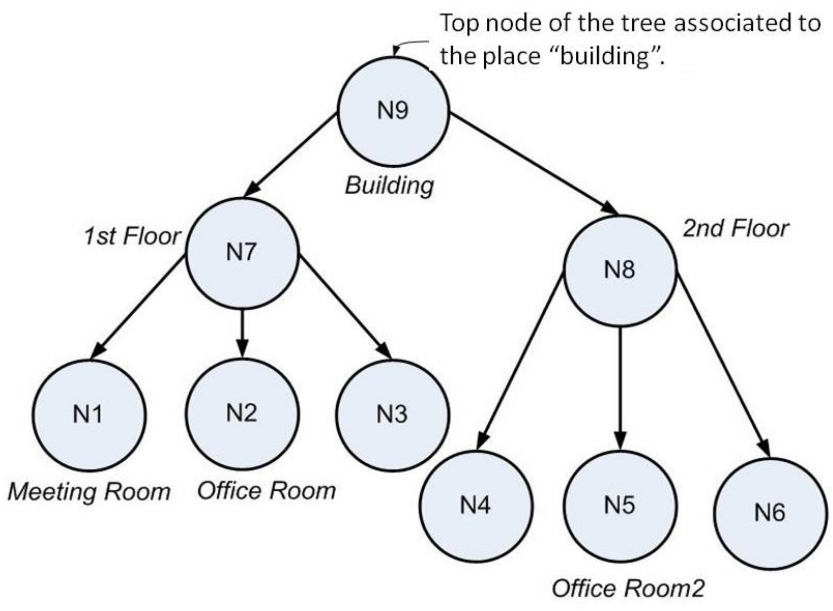


Figure 3 : Déploiement logique des passerelles sémantiques dans le cas d'un bâtiment.

4-La recherche des descriptions de services sémantiques dans un contexte statique

Dans ce chapitre, nous détaillons les différents mécanismes intégrés dans les passerelles sémantiques pour l'enregistrement et la recherche des descriptions de services. Nous considérons dans ce chapitre un contexte statique où les objets connectés et leurs services associés sont constamment liés au même espace intelligent.

Dans la suite, nous spécifions d'abord le problème traité afin d'assurer une recherche de services efficace répondant aux exigences de la flexibilité, exhaustivité et extensibilité tels qu'elles ont été exprimées dans le premier chapitre.

Ensuite, nous présentons notre solution et détaillons ses caractéristiques. Enfin nous présentons quelques évaluations effectuées sur un prototype d'immeuble intelligent pour valider notre approche.

4.1-Enoncé du problème

Avec la multiplication des objets connectés et l'augmentation de leur hétérogénéité, il deviendrait difficile pour un utilisateur d'avoir une connaissance exacte de tous les services de l'IoT autour de lui (Par exemple la description, l'ID, l'adresse, etc.). Aussi, l'hétérogénéité syntaxique des services rend leur découverte inefficace. En considérant ces contraintes, il serait plus pertinent de baser la recherche des services de l'IoT sur leur sémantique.

Les technologies du Web sémantique permettent la modélisation sémantique des informations et des fonctionnalités offertes par les services IoT. L'information est ainsi enrichie et devient interprétable par les machines. En effet, grâce à la logique implémentée dans la description sémantique des services et avec l'aide de raisonneurs, il devient possible aux machines de comprendre le sens des services et permettre leur découverte en se basant sur leur sens.

Toutefois, ces approches sont coûteuses en termes de ressources de calcul et peuvent impacter l'extensibilité du système surtout pour des systèmes à large échelle. Ainsi, il est important de considérer des techniques d'optimisation lors de l'utilisation des technologies du Web Sémantique afin de limiter leur coût en termes de nombre d'opérations d'appariement nécessaires pour trouver services pertinents.

Nous présentons dans la partie suivante notre solution pour permettre une recherche extensible, exhaustive et flexible des descriptions des services sémantiques dans des environnements de l'Internet des Objets.

4.2-Présentation de la solution

Nous définissons dans cette section un mécanisme extensible de recherche des descriptions sémantiques des services IoT. Il est basé sur un appariement sémantique flexible et permet de trouver tous les services qui répondent à une requête donnée, tout en limitant le nombre d'opération d'appariement nécessaires pour trouver ces services. Chaque passerelle sémantique implémente des mécanismes pour gérer son contenu et effectuer une recherche des services, que nous détaillons dans la suite. Nous définissons d'abord les métriques utilisées pour estimer une "distance sémantique" entre la requête et une description sémantique d'un service donné. Ensuite, nous présentons la notion de "table de routage" construite dans chaque passerelle sémantique et qui résulte de l'agrégation de l'information dans chaque passerelle. Enfin nous détaillons le mécanisme de routage de la requête entre les passerelles sémantiques à l'aide de seuils d'appariements que nous définissons.

Les métriques d'appariement sémantique

Pour assurer la flexibilité de l'appariement sémantique de la requête avec un service, nous avons considéré l'approche du "plug in matching" [5] où un service S est pertinent pour une requête R les concepts de S sont équivalents ou inclus dans ceux de R. Cette démarche assure que le service renvoyé contient au moins les fonctionnalités demandées dans la requête tout en assurant un certain degré de flexibilité.

Avec la définition de la méthode d'appariement précédente, un super-concept d'un concept dans le service ou la requête n'affecte pas le résultat final. Donc, nous définissons dans la suite *la représentation étendue* d'un service où nous ajoutons à la requête ou au service des super concepts de leur représentation. Nous notons la Représentation étendue d'un service S, $\tilde{S}$.

On a aussi définie la fonction *cordonnés d'un service* dans l'espace des concepts qui permet d'évaluer si un concept donné existe dans l'espace des concepts d'un service S (on associe à la fonction coordonnée 1 dans ce cas) ou non.

Sur la base de ces deux fonctions, nous avons définie une métrique Q qui permet dévaluer la distance entre une requête R et un service S. Nous la définissons de la façon suivante :

Definition 1: Métrique d'appariement.
Nous définissons la fonction d'appariement Q comme suit :

$$Q \ : \ \mathcal{S} \times \mathcal{S} \to \mathbb{R} \tag{1}$$

$$(R, S) \mapsto Q(R, S) = \sum_{i=1}^{M} \alpha_i \times q\left(\chi_i\left(\tilde{R}\right), \chi_i\left(\tilde{S}\right)\right), \tag{2}$$

où $(\alpha_i)_{i\in\{1,\cdots,M\}}$ est un M-uplet de nombres réels strictement positifs, et la fonction $q : \{0,1\} \times \{0,1\} \mapsto [0,1]$ est définie come suit :

– $q(0,0) = 0,$
– $q(0,1) = \epsilon_1 \in]0,1[,$
– $q(1,0) = 1 - \epsilon_2,\ avec\ \epsilon_2 \in [0, 1 - \epsilon_1[,$
– $q(1,1) = 0.$

On peut vérifier que cette métrique Q vérifie les propriétés d'une quasi-métrique en terme de positivité, séparabilité et inégalité triangulaire. Enfin, en utilisant cette métrique Q, nous définissons aussi une fonction d'appariement M_f, qui permet d'estimer s'il ya un appariement entre un service S et une requête R à l'aide d'un seuil T.

les tables de routage

Dans une passerelle sémantique, nous créons des groupes de services similaires basés sur un mécanisme de groupement hiérarchique et sur la base d'un seuil de similarité TC que nous fixons d'une façon heuristique. Ensuite nous agrégeons le contenu de ces groupes en sélectionnant un représentant et en définissant un rayon à l'aide des métriques définies précédemment. Ces informations agrégées sont stockées dans une table de routage. Ainsi, la table de routage contient un résumé du contenu de nœud sémantique. Ces tables de routages sont construites dans chaque nœud sémantique du graph. Le processus est basé sur un groupement et une agrégation de l'information récursivement tout au long de la hiérarchie des nœuds. En commençant par les nœuds inférieurs, l'information est agrégée et envoyée aux nœuds parents où un groupement et une agrégation du contenu de ces nœuds seront effectués aussi. La Figure 4 donne une vue globale du processus de groupement et d'agrégation tout au long des nœuds de la hiérarchie. L'information agrégée de chaque nœud est stockée dans une table de routage associée au nœud en question.

Les seuils de décisions et transfert de la requête

Nous avons définit deux seuils qui sont utilisé lors de la recherche des services : le seuil d'appariement, qu'on note θ, qui permet de sélectionner les descriptions de services finaux qui correspondent à une requête (c'est à dire, les services finaux qui seront retournés à l'utilisateur) sur la base de la condition suivante : $Q\,(R, S) \leq \theta$ et qui signifiera que $M_f\,(R, S) = 1$.

Pour sélectionner le bon cluster K à visiter dans une passerelle sémantique N ou pour transférer la requête d'une passerelle sémantiques à une autre, on a défini un seuil de décision $Td_{K,N}$.

Un cluster K contient un service S qui répond à une requête R implique que $Q\,(R, W_{K,N}) \leq Td_{K,N}$, i.e, $M_f\,(R, W_{K,N}) = 1$, où $W_{K,N}$ correspond au représentant du cluster K dans le nœud N.

Sur la base de ses seuils, le transfert de la requête d'un nœud sémantique vers un autre s'effectue selon de schéma sur la Figure 5.

4.3-Tests et Résultats

Nous validons dans cette partie que le mécanisme de recherche répond aux exigences de flexibilté, exhaustivité et extensibilité.

La flexibilité du système est vérifiée à travers la définition du la méthode d'appariement qui permet de retourner non seulement des services exactement similaires à celui qui est demandé mais aussi des services partiellement similaires.

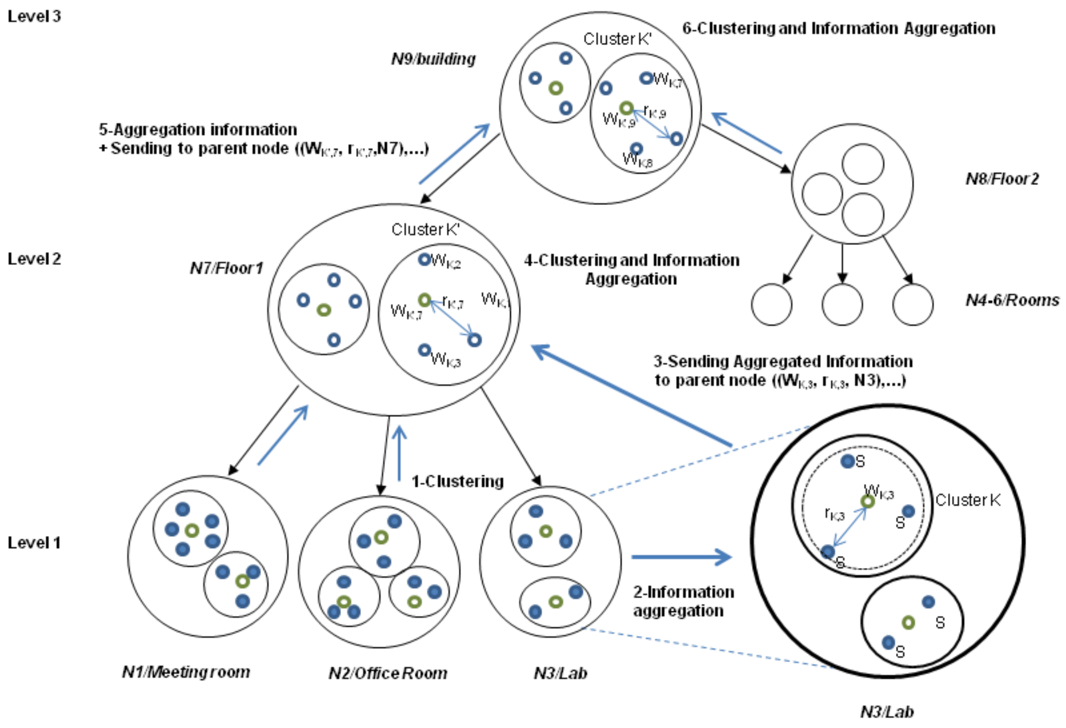

Level 3
N9/building
Cluster K'
6-Clustering and Information Aggregation
5-Aggregation information
+ Sending to parent node ((W_{K',7}, r_{K',7}, N7),...)
N8/Floor2
N4-6/Rooms
Level 2
N7/Floor1
Cluster K'
4-Clustering and Information Aggregation
3-Sending Aggregated Information to parent node ((W_{K,3}, r_{K,3}, N3),...)
1-Clustering
2-Information aggregation
Cluster K
Level 1
N1/Meeting room
N2/Office Room
N3/Lab
N3/Lab

Figure 4 : Groupement et agrégation récursifs du contenue des passerelles sémantiques tout au long de la hiérarchie des nœuds.

Nous pouvons vérifier formellement que la recherche dans le système est exhautif à l'aide de l'inégalité triangulaire et les différents seuils d'appariement definis.

A travers une analyse numérique, nous évaluons l'extensibilité du système. Pour cela nous effectuons différents testes en variant différents paramètres. Dans la suite, nous considérons un prototype de building intelligent et nous effectuons différents testes en variant certains paramètres. Nous évaluons d'abord l'extensibilité du système en fonction du nombre de services dans le système. Ensuite, nous étudions l'impacte des distributions de services considérées, sur les performances du système de recherche et enfin nous étudions l'impacte de la topologie des graphs des nœuds.

Nous nous limitons dans la suite à détailler l'expérience de l'étude de l'impacte du nombre de services sur les performances du système.

Dans cette expérience, nous considérons un prototype d'un système simulant un bâtiment intelligent composé de 3 étages et de 5 chambres dans chaque étage. Nous varions le nombre de services dans les nouds du plus bas niveau de 50 services jusqu'à 3000 services en considérant une distribution des services donnée, ce qui correspond à un nombre total de services dans le système variant de 750 à 45000. Nous envoyons 100 requêtes au nœud du plus haut niveau et qui vont parcourir toute la hiérarchie des nœuds jusqu'à attendre les services utiles dans les nœuds du plus bas niveau. Nous considérons comme métriques le coup moyen d'appariement par requête et la longueur moyenne du parcours d'une requête dans la hiérarchie des nœuds. Nous comparons notre approche à une approche purement centralisée et non optimisée (nous ne considérons pas de l'agrégation de l'information). La Figure 6 montre les résultats obtenus.

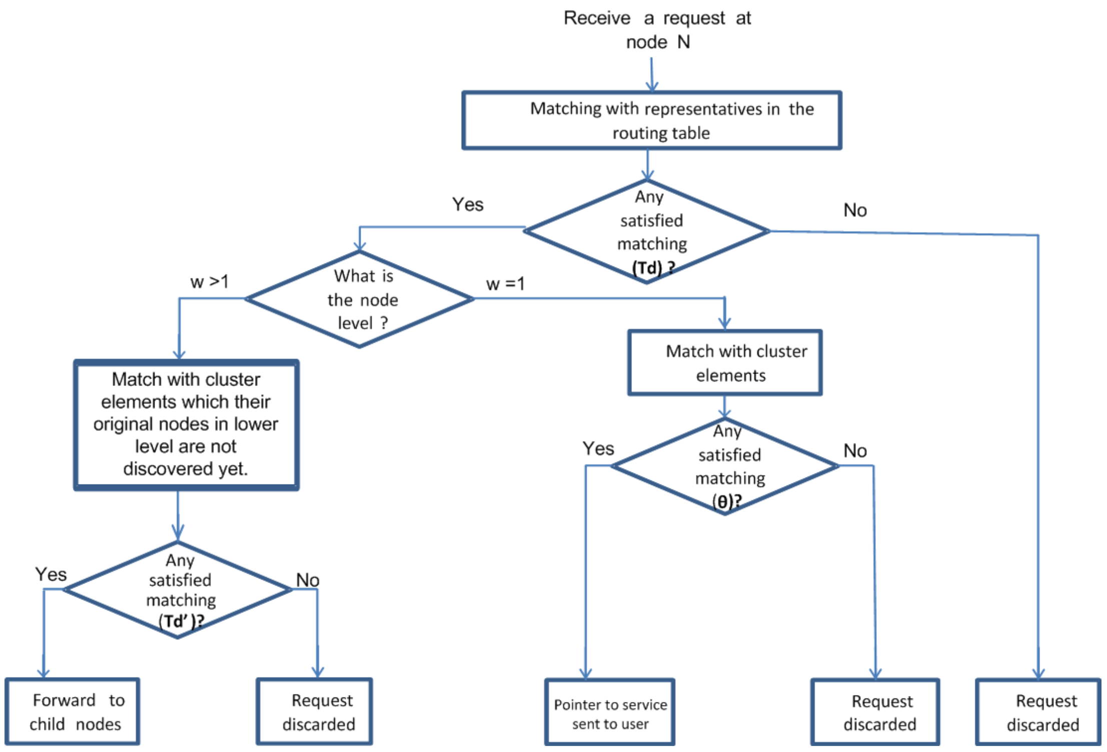

Receive a request at node N
Matching with representatives in the routing table
Any satisfied matching (Td) ?
Yes
No
What is the node level ?
w >1
w =1
Match with cluster elements which their original nodes in lower level are not discovered yet.
Match with cluster elements
Any satisfied matching (Td')?
Yes
No
Any satisfied matching (θ)?
Yes
No
Forward to child nodes
Request discarded
Pointer to service sent to user
Request discarded
Request discarded

Figure 5 : Le transfert de la requête à travers la hiérarchie des nœuds.

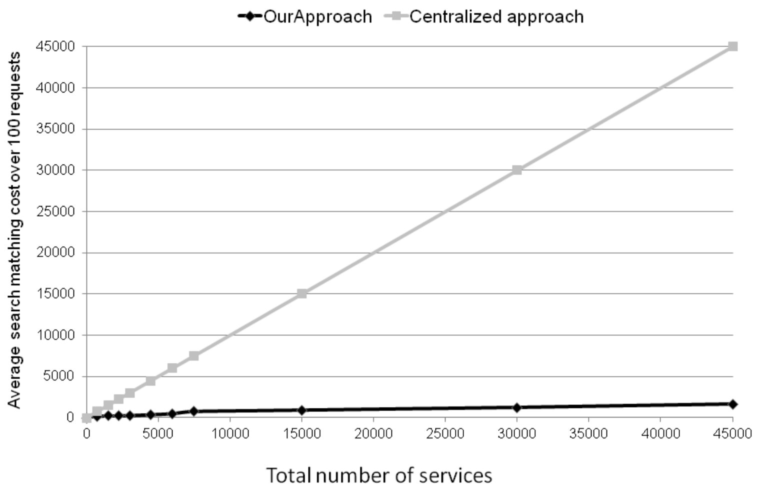


Figure 6 : L'évolution du coup d'appariement par requête.

Nous pouvons voir que dans l'approche centralisée, comme prévu, le coût moyen d'appariement par requête évolue linéairement avec le nombre de services dans le système. Cependant, dans notre approche, l'évolution du coût est sous-linéaire. Les valeurs de coût d'appariement sont faibles par rapport à l'approche centralisée, par exemple, pour 45000 services, on a un gain d'environ 25. La combinaison des mécanismes de groupement et d'agrégation de l'information permet de réduire ces coups de recherche.

Dans un deuxième temps, nous mesurons la longueur moyenne réelle du chemin de recherche par requête et nous le comparons à la longueur moyenne parfaite du chemin de recherche d'une requête dans le système.

Nous comparons l'évolution des deux approches lorsque le nombre global de services Augmente dans le système. Les résultats de cette expérience sont présentés sur la Figure 7.

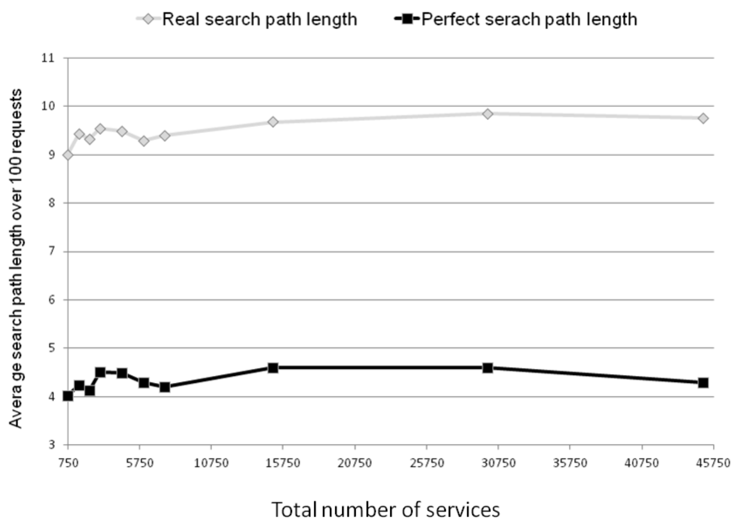


Figure 7 : L'évolution de la longueur du chemin de recherche en fonction du nombre de services.

La longueur de la trajectoire idéale est de l'ordre de 4 nœuds, ce qui correspond à la moitié de la longueur de trajectoire réelle, cela est dû à l'appariement faux positif où la requête est transmise à un nœud qui est considéré à tort comme contenant les services appropriés. Cependant, la longueur réelle du chemin reste acceptable si on la compare à l'approche de l'inondation où on visite tous les nœuds du système (19 nœuds dans ce cas).

Les autres tests qui étudient l'impacte de la distribution des services et le type de graphe considéré sur les performances du système montrent que ces paramètres affectent le coup d'appariement dans le système et le chemin parcouru par la requête. Cependant cet impacte est modéré et les résultats restent très avantageux par rapport à une approche purement centralisée surtout quand on considère un grand nombre de services dans le système.

5-La recherche des descriptions de services sémantiques dans un contexte dynamique

Dans ce chapitre, nous considérons un contexte dynamique où les services IoT sont mobiles et bougent d'un espace intelligent vers un autre (Par exemple, les capteurs embarqués dans les téléphones intelligents). Par conséquence, le contenu des passerelles sémantiques, qui enregistrent des descriptions sémantiques des services de l'IoT évoluent au cours du temps.

Cette mobilité des services induit de nouveaux défis pour le système de découverte défini dans la partie précédente. Ces défis sont principalement en termes de gestion de données pour permettre une recherche efficace et extensible. Dans ce qui suit, nous définissons d'abord le problème traité et les exigences pour une découverte efficace des services IoT sémantiques dans un contexte dynamique. Ensuite, nous présentons notre solution où nous définissons des mécanismes au sein des passerelles sémantiques afin de s'adapter à la mobilité des services et modifier le contenu de chaque passerelle. Ces mécanismes permettent de créer dynamiquement des groupes de services semblables et optimiser le regroupement au cours du temps afin de réduire au minimum le coût de la recherche.

5.1-Enoncé du problème

La dynamicité des services apporte de nouveaux défis pour le système de découverte que nous énumérons ci-après :

Assurer une recherche efficace dans un nœud et dans la hiérarchie des nœuds (c'est-à-dire trouver tous les services qui répondent à une requête donnée). Avec la mobilité des services, une nouvelle description de service sémantique peut être ajoutée à la passerelle sémantique en raison de l'arrivée d'un nouvel objet dans son champ d'application géographique. De la même façon, une description de service peut être supprimée de la passerelle sémantique en raison du départ de son objet correspondant. Au cours de cette mobilité des services, nous avons besoin de garantir, dans un nœud, la cohérence de regroupement des services et l'actualisation permanente de sa table de routage. Les nœuds enfants doivent aussi informer les nœuds parents des changements effectués à leurs niveaux. Ainsi, une gestion dynamique et efficace du contenu des nœuds est nécessaire.

Assurer un mécanisme de gestion et de recherche de services extensible dans un nœud. Les coûts à l'intérieur d'un nœud sont liées au nombre d'opérations d'appariements sémantiques produites dues à la réception d'une requête de recherche d'un service ou à l'insertion d'un nouveau service dans la passerelle et à la mise à jour de son contenue. Ces coûts Ils peuvent être divisés en :

- Le coût M_{rT} lié à la recherche de services : Le nombre total d'opérations d'appariement effectuées pour trouver les services qui répondent à une requête donnée.
- Le coût M_{mT} lié à la mobilité des services : Le nombre total d'opérations d'appariement effectuées entre les services lors de l'insertion ou la suppression d'un service une passerelle sémantiques. .
- Le coût M_{RCT} lié à l'optimisation du groupement : Le nombre total d'opérations d'appariement entre services lors de la mise à jour des groupements (par exemple suite aux opérations de fusion ou de fractionnement des groupements).

Le coût des mis à jour envoyés à un nœud parent sont liés aux changements dans les tables de routage d'un nœud enfant et qui seront notifiées à son nœud parent. Une fréquence élevée de mises à jour impacte l'extensibilité du système. Ainsi, une minimisation des coûts d'appariement et de mise à jour est nécessaire.

5.2-Présentation de la solution

Nous présentons dans la suite de notre approche pour adapter le système à la mobilité des services. Nous considérons d'abord un nœud du plus bas niveau et nous détaillons les mécanismes de soutien à la mobilité au sein de ce nœud. Pour cela, nous définissons un groupement incrémental qui prend en compte l'arrivée et le départ des nouveaux services dans le nœud. Ce regroupement incrémental est optimisé en termes de distribution de services entre les clusters afin de minimiser le coût de recherche M_{rt}. Nous proposons ensuite une approche pour optimiser le coût de la recherche dans l'ensemble du système (c'est à dire la hiérarchie des passerelles sémantiques).

Groupement incrémental dans un noeud du bas niveau

Dans cette partie, nous définissons les mécanismes pour créer et mettre à jour des groupes de services similaires au niveau d'un nœud. Lors de la création et la mise à jour d'un cluster, nous visons à rendre les nouveaux services insérés dans la passerelle rapidement découvrables tout en limitant le nombre d'opérations d'appariement nécessaires pour les insérer dans la passerelle. Dans la suite, nous détaillons les différentes actions pour insérer ou supprimer un nouveau service d'une passerelle sémantique.

Nous considérons dans la suite un nœud de niveau le plus bas dans la hiérarchie des nœuds et nous analysons l'impact de sa mobilité sur un nœud de niveau supérieur.

Soit N un nœud qui représente une passerelle sémantique et $\mathcal{K} = \{K_i, i \in \{1, \cdots, n\}\}$ un ensemble de clusters dans le nœud N. Pour un cluster $K \in \mathcal{K}$, nous considérons $W_{K,N}$ et $r_{K,N}$, respectivement son représentant et son rayon. Nous considérons la quasi-métrique Q définie dans le chapitre précédent.

Cas d'arrivée d'un nouveau service :

Nous considérons S_1 un nouveau service qui sera inséré dans le nœud N. Nous considérons $K_{min} = \underset{K \in \mathcal{K}}{\arg\min} \left[Q\left(S_1, W_{K,N}\right)\right]$. K_{min} est le cluster en N dont le représentant est le plus proche de S_1 en terme de quasi-métrique Q. On note d_{min} cette distance. $d_{min} = Q\left(S_1, W_{K_{min},N}\right)$. L'insertion de S_1 dans N peut présenter différents scénarios que nous illustrons par la suite (Voir la Figure 8) :

a) Si $d_{min} \leq r_{K_{min},N}$, alors S_1 est inséré dans K_{min} et aucune modification n'est apportée au cluster.

b) Si le cas (a) n'est pas vérifié et $\exists K \in \mathcal{K} \setminus \{K_{min}\}$ où $r_{K,N} \geq Q\left(S_1, W_{K,N}\right)$, alors S_1 est inséré dans $K = \underset{K \in \mathcal{K}\setminus\{K_{min}\}}{\arg\min} \left[Q\left(S_1, W_{K,N}\right)\right]$, aucune modification n'est apportée dans le cluster.

c) Si ce n'est pas (a) et pas (b) et $d_{min} < Q\left(W_{K,N}, W_{K_{min},N}\right)$, alors nous agrandissons le rayon de K_{min} pour inclure le service S_1. Nous avons : $K_{min} = \{W_{k_{min},N}, r_{k_{min},N} = d_{min}\}$.

d) Si ce n'est pas (a), pas (b) et pas (c), alors un nouveau cluster $K' \in \mathcal{K}$ est créé avec un représentant $W_{K',N} = S_1$ et un rayon $r_{K',N} = 0$.

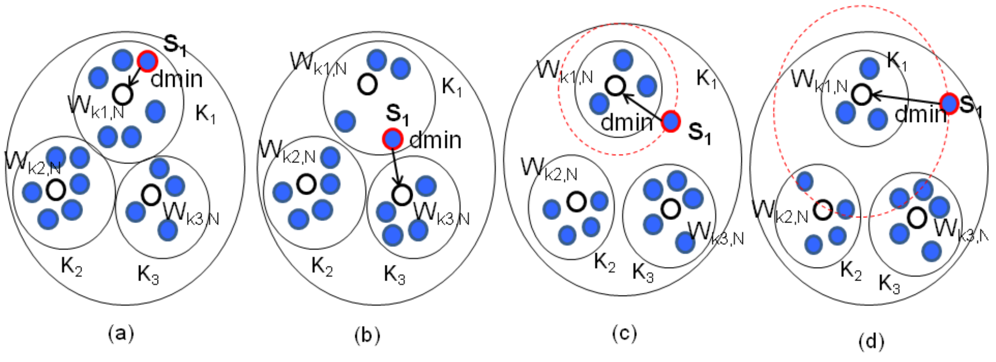


Figure 8 : Représentation 2D des cas d'arrivée des services.

Cas de départ d'un service :

Dans ce qui suit, nous considérons le cas d'un départ d'un service S_1 qui quitte le groupement K. Nous considérons $d = Q(S_1, W_{K,N})$. Le départ de $S1$ peut représenter différents scénarios que nous détaillons dans la suite (Voir Figure 9)

a) $S_1 \neq W_{K,N}$ et $d < r_{K,N}$, alors aucune modification n'est apportée au cluster.
b) $S_1 \neq W_{K,N}$ et S_1 est le seul service où $d = r_{K,N}$, on recalcule le rayon du groupe après le départ de S_1 pour l'optimiser.
c) $S_1 = W_{K,N}$ et $|K| > 1$, alors nous considérons une représentation virtuelle de S_1. Aucun changement n'est apporté au sein du cluster.
d) $S_1 = W_{K,N}$ et $|K| = 1$, alors nous supprimons le cluster K du noeud N.

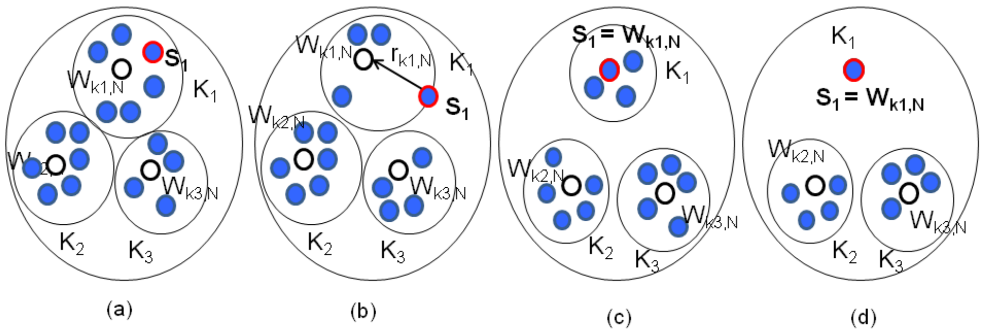


Figure 9 : Représentation 2D des cas de départ des services

Optimisation du groupement dans un nœud de niveau bas

Avec le mécanisme décrit dans la section précédente, le nombre de groupement de services formés et le nombre de services par groupement peuvent varier au cours de la mobilité des services, ce qui peut affecter le coût de la recherche M_{rT}. Puisque nous cherchons à minimiser les coûts pendant la mobilité des services, il est important d'optimiser la distribution de services dans les groupements afin de minimiser le coût de recherche correspondante.

Groupement Idéal :

Pour évaluer la qualité du groupement, nous définissons un modèle de référence de groupement idéal qui n'existe pas. Cependant, il peut servir de référence pour définir une limite basse pour le coût de la recherche que nous essayons d'approcher pendant la mobilité des services.

Dans un groupement idéal, une requête ne visitera aucun groupe s'il n'y a pas un service qui répond à la requête dans le groupement. Sinon, la requête visitera un seul groupe s'il y a un service qui répond à la requête dans le groupement.

Groupement optimisé :

Le regroupement défini ci-dessus est idéal. Cependant, il n'est pas optimal. Nous définissons, ci-dessous, une approche pour l'optimiser. En fait, au cours de la mobilité des services, certains groupes peuvent se développer pour inclure un grand nombre de services. Ceux-ci ont une incidence négative sur le coût de recherche.

Dans ce qui suit, nous définissons une répartition optimale des services au sein d'un nœud (voir la proposition ci-dessous), où on considère un nombre optimal de groupes de services et un nombre moyen optimal de services par groupe afin de minimiser le coût de la recherche.

Proposition 1: Distribution optimale des services.
On considère un nœud N avec N_s services distribués sur un groupement $\mathcal{K} = \{K_i, i \in \{1, \cdots, n\}\}$. m_i est le nombre de services par cluster K_i. On a alors $\sum_{i=1}^{n} m_i = N_s$. Un nombre de requêtes est envoyé à N.

$\langle M_r \rangle$ est le coup moyen d'appariement par requête R, n_c est le nombre de clusters, $\langle m_s \rangle$ est le nombre moyen de service par groupe de services et $\langle \beta \rangle$ est le nombre moyen de groupes de services visités par requête. $\langle M_{r_{min}} \rangle$ est le coup minimal moyen d'appariement par requête, $n_{c_{op}}$ est le nombre optimal de clusters par nœud et $\left\langle m_{s_{op}} \right\rangle$ est le nombre moyen optimal de services par groupe de services. En se basant sur la définition du groupement idéal, on peut écrire :

$$\langle M_r \rangle = n_c + \langle \beta \rangle \times \langle m_s \rangle \, et \tag{3}$$
$$N_s = n_c \times \langle m_s \rangle \,. \tag{4}$$

$\langle M_r \rangle$ est minimal pour :

$$n_{c_{op}} \approx \sqrt{\langle \beta \rangle \times N_s}, \left\langle m_{s_{op}} \right\rangle \approx \sqrt{\frac{N_s}{\langle \beta \rangle}}, \tag{5}$$

et on a :

$$\langle M_{r_{min}} \rangle = 2 \times \sqrt{\langle \beta \rangle \times N_s}. \tag{6}$$

En se basant sur la définition du groupement idéal, nous avons$\langle \beta \rangle \in]0,1[$. Dans la suite, nous considérons le pire cas et nous fixons $\langle \beta \rangle$ à 1.

Opérations de fusion et séparation du groupement :

Au cours de la mobilité des services, les changements effectués sur le groupement de services dans une passerelle sémantique affecte le nombre de groupes et le nombre moyen de services par groupe.

Dans notre approche pour optimiser la distribution des services au sein d'un nœud, nous vérifions après chaque événement de mobilité (par exemple, une arrivée ou départ de service) le nombre de groupes N_c et le nombre maximal de services par groupe m_{Max}. Nous définissons ainsi les mécanismes de regroupement où un fractionnement d'un groupe est effectué quand la limite m_{Max} est dépassée et une fusion de groupes est effectuée quand la limite n_{c_op} est dépassée. Le but est d'approcher le coût de recherche au coup minimal $\langle M_{r_{min}} \rangle$.

Cependant, le fractionnement et la fusion continue des groupes de services peut impacter négativement l'extensibilité du système. C'est la raison pour laquelle nous définissons des valeurs limites en termes de nombre de groupes $n_{c_{Limit}}$ et nombre moyen maximal de services par groupe $m_{Max_{Limit}}$ en ajoutant des marges ϕ et γ comme suit :

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

Ainsi, à chaque fois que la limite $n_{c_{Limit}}$ est dépassée, on effectue une fusion de deux groupes de services les plus similaires. Et à chaque fois que la limite $m_{Max_{Limit}}$ est dépassée, on effectue une fraction du plus gros groupe en deux groupes de tailles similaires. Les simulations on montré que les valeurs idéales des marges sont $\gamma = 1$ et $\phi \in [0.5, 1]$.

Optimisation du coût de la recherche dans toute la hiérarchie des nœuds

Pour une optimisation globale d'une hiérarchie des nœuds, nous appliquons le même principe d'optimisation pour un nœud de bas niveau aux autres nouds de la hiérarchie. Ceci permet d'optimiser le coup de la recherche dans tout le système en limitant les contraintes entre les nœuds.

5.3-Quelques tests et résultats

Nous avons effectué différentes expériences pour évaluer la performance des mécanismes de support à la mobilité que nous avons présentées dans cette partie. Nous avons évalué les performances au niveau d'un nœud en fonction de différents paramètres (par exemple le nombre de services et les marges considérées) et au niveau d'une hiérarchie de nœuds. Nous nous limitons dans cette partie à présenter l'évaluation d'un nœud de bas niveau. Pour cela, nous considérons un nœud représentant un espace intelligent et contenant 400 services. Nous simulons des arrivées et des départs de services aléatoires. Pour évaluer la performance de notre approche nous définissons deux métriques :

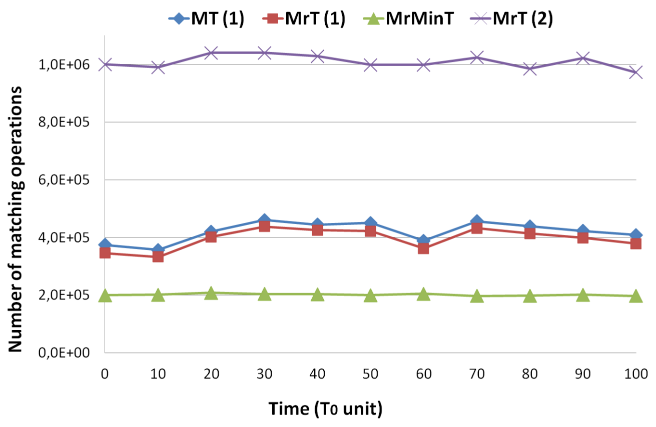
- Le coût total d'appariement M_T : C'est la somme du coût de recherche, le coût de la mobilité et le coût d'optimisation du groupement.
- Le taux relatif des mises à jour générées et envoyées par un nœud enfant à son parent : Il s'agit du rapport entre le nombre d'événements envoyés par un nœud enfant à son nœud parent et le nombre total d'événements reçus par un nœud enfant sur une période de temps donnée.

Nous évaluons, dans cette partie, l'impact du nombre de services sur le coût total d'appariement et sur le taux relatif des mises à jour générées et envoyées par un nœud enfant à son parent. Pour cela, nous considérons 3 nœuds avec les mêmes paramètres en termes de marges ($\gamma = 1$ et $\phi = 0,5$). Le premier nœud contient un nombre moyen de 100 services au cours du temps (nous fixons $\mu = 0,01$ et $\lambda = 1$), le second nœud contient un nombre moyen de 400 services (nous fixons $\mu = 0,0025$ et $\lambda = 1$)) et le troisième nœud contient un nombre moyen de 800 services (nous fixons $\mu = 0,00125$ et $\lambda = 1$). Nous comparons également notre approche (1) avec une approche centralisée (2) représentant un nœud sans groupement de services.

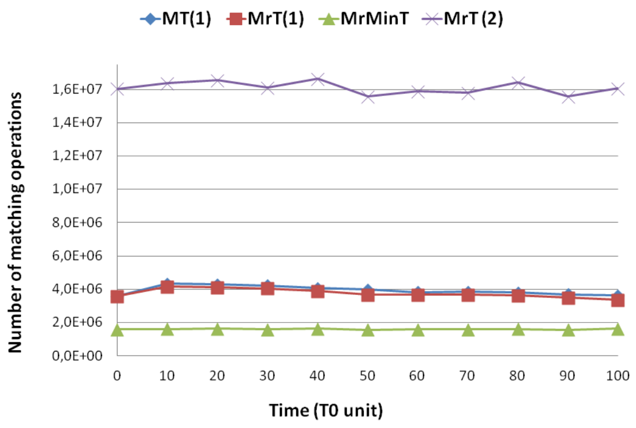
Nous mesurons l'évolution du coût total d'appariement M_T et nous le comparons avec l'évolution du coût total de recherche $M_{rT} = N_R \times \langle M_r \rangle$, au cours de $100 \times T_0$. Les résultats sont présentés dans la Figure 10 où nous montrons respectivement l'évolution d'un nœud avec 100 services, 400 services et des services 800.

Nous comparons le coût de recherche de notre approche (1) avec le coût de recherche de l'approche (2) représentant un nœud sans groupements. Nous avons $M_{rT}(2) \geq 6 \times M_{rT}(1)$ pour les nœuds de 800 services. Dans les deux cas, notre approche (1) est plus performante qu'une approche sans groupements (2) en termes de coût de recherche. Nous pouvons aussi voir dans notre approche que $M_T(1)$ et $M_{rT}(1)$ sont quasi-constants au cours du temps. Nous avons $M_T(1) \approx M_{rT}(1)$ pour les 3 nœuds, ce qui confirme que les coûts générés par les mécanismes de gestion de la mobilité sont négligeables par rapport au coût de la recherche. Nous avons aussi $M_{rT}(1) \approx 2 \times M_{r_{menthe}}$. Cet écart entre les deux valeurs est due aux marges que nous avons ajoutées à $n_{c_{op}}$ et $\langle m_{s_{op}} \rangle$ pour réduire le nombre d'opérations de regroupements.

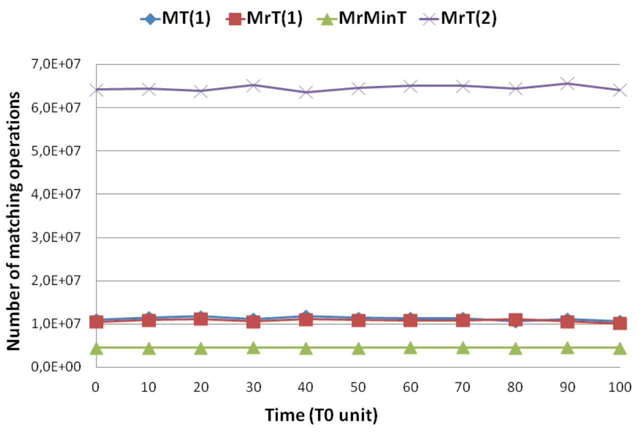
Nous avons mesuré le coût relatif des mises à jour générées et envoyées par un nœud à son parent. Ces mises à jour sont dues à des changements dans le contenu du nœud qui doivent être notifiées au nœud parent pour assurer une recherche efficace des services. Les résultats sont présentés dans la Figure 11.



(a) 100 services

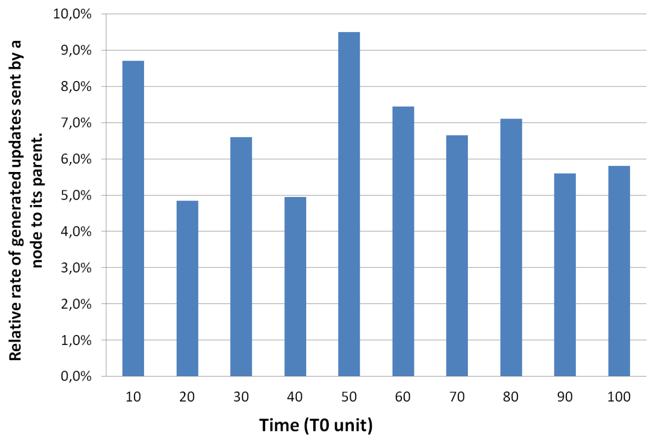


(b) 400 services

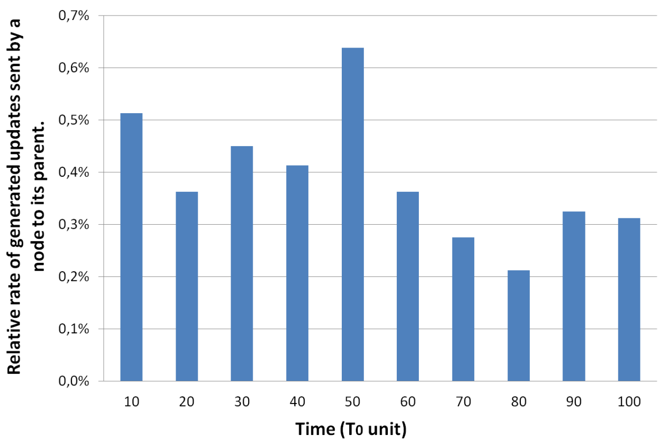


(c) 800 services

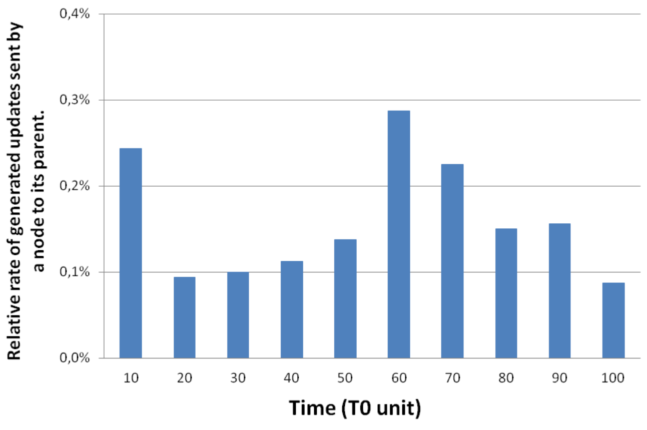
Figure 10 : Impacte du nombre de services par noeud sur le coup d'appariement pour une marge of $\phi = 0.5$



(a) 100 services



(b) 400 services



(c) 800 services

Figure 11 : Impact du nombre de services sur le taux relatif de mises à jour générées et envoyées par un nœud à son parent pour $\phi = 0,5$

On peut voir que dans les 3 cas de 100 services, 400 services 800 services par nœud que le taux relatif de mises à jour du nœud parent est faible. En effet, il est à moins de 1% dans le cas des 400 services et 800 services. Le système semble être plus stable lorsque le de services par nœud augmente. Pour les 3 nœuds considérés, nos mécanismes permettent à un nœud de niveau supérieur de recevoir des événements de mobilité à un rythme beaucoup plus lent que celui reçu par un nœud enfant. Ceci peut refléter que nos mécanismes de mobilité ont un impacte faible sur la hiérarchie des nœuds.

6-Conclusion

Dans cette thèse, nous avons détaillé l'architecture d'un mécanisme de découverte des services de l'Internet des Objets en se basant sur leur sémantique. Nous avons présenté la notion de passerelles sémantiques liées à des espaces intelligents. Elles se chargent d'enregistrer les descriptions de services sémantiques liées à ce lieu et traiter les requêtes de découverte reçues à son niveau. Les passerelles sémantiques sont interconnectées entre elles sous forme de graph en arbre.

Ces passerelle supportent un mécanisme de découverte basé sur du groupement et de l'agrégation de l'information sur la base d'une quasi-métrique et des seuils d'appariement que nous avons définis. En considérant un cas statique, nous avons montré que notre mécanisme de recherche est capable de répondre aux exigences de la recherche flexible, exhaustive et extensible.

Nous avons étendu notre mécanisme de recherche à un cas dynamique caractérisé par la mobilité des services IoT et nous avons définies des mécanismes de mise à jour des groupements. L'évaluation de la solution a montré que les exigences des mécanismes de recherche démontrées dans un cas statique sont conservées dans le cas dynamique et que les mécanismes de mobilité qu'on a définit impactent très faiblement l'extensibilité du système.

Outre les contributions de cette thèse, certains points peuvent encore être étudiés et améliorés dans le futur. En effet, on peut envisager d'enrichir notre mécanisme de découverte avec la fonction de composition/décomposition des services où un utilisateur peut envoyer une requête complexe qui se décompose en requêtes de recherche simples. Chaque requête sera recherchée et le résultat final sera une composition de services trouvés. Aussi, dans notre travail nous avons considéré des ontologies simples basées sur une hiérarchie de concepts. L'appariement est basé sur l'évaluation des relations de subsumption entre concepts dans les descriptions de services sémantiques. Dans une prochaine étape, nous aimerions envisager des ontologies basées sur des logiques de descriptions plus riches afin de décrire des services complexes de l'IoT. Ceci nécessitera la révision de nos métriques pour les adapter à ce type de descriptions sémantiques. Enfin, dans l'architecture actuelle, nous considérons des passerelles physiques et sémantiques fixes. Dans une prochaine étape, nous envisageons de considérer des passerelles mobiles qui peuvent être hébergées par exemple dans des smartphones. Ceci nécessitera l'adaptation de notre solution à des topologies de passerelles inconnues.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

1.1 IoT vision, scenarios and challenges

Since its introduction by Mark Weiser in 1988, ubiquitous computing, also called pervasive computing, aims at integrating computing and networking resources into physical environments, with the core vision of creating spaces in which users seamlessly interact with computing services typically through mobile devices (i.e., smartphones, tablets, PDAs, etc.) [1]. Progress made in embedded computing helped significantly the development of this vision. The cost of embedding electronics in everyday objects has decreased greatly lately, which enabled connecting everyday devices (e.g., TVs, alarm clocks, lamps, refrigerators,etc.) and also smaller objects (e.g., sensors, RFID tags,etc.). As a result, nowadays, interconnected networks of physical objects are slowly emerging, and creating the *Internet of Things* (IoT).

The term Internet of Things was popularized by the work of the Auto-ID center [1] at MIT in 2000. This work has mainly focused on associating RFID tags to physical objects to get information about them through connected databases. Later on, the term has become more general to encompass connecting all physical objects implementing different communication protocols and becoming reachable via the Internet. The Internet of Things has faced a growing interest from the research community through these ten past years [6]. Consequently, several definitions have been proposed to characterize the Internet of Things [2]. For instance, CASAGRAS project [7] defines the Internet of Things as a network infrastructure for connecting objects :

"A global network infrastructure, linking physical and virtual objects through the exploitation of data capture and communication capabilities. This infrastructure includes existing and evolving Internet and network developments. It will offer specific object-identification, sensor and connection capability as the basis for the development of independent cooperative services and applications. These will be characterised by a high degree of autonomous data capture, event transfer, network connectivity and interoperability."

The Center of European Research Projects on the Internet of Things (CERP-IoT), defined the Internet of Things in [8] as an enabler of new services that take part in different applications:

"In the IoT, 'things' are expected to become active participants in business, information and social processes where they are enabled to interact and communicate among themselves and

1. Auto-IDlabs:http://www.autoidlabs.org/
2. http://postscapes.com/internet-of-things-definition

with the environment by exchanging data and information 'sensed' about the environment, while reacting autonomously to the 'real/physical world' events and influencing it by running processes that trigger actions and create services with or without direct human intervention. Interfaces in the form of services facilitate interactions with these 'smart things' over the Internet, query and change their state and any information associated with them."

Connecting things has enabled the development of new applications in diverse domains. For instance, connected home appliances such as TVs, alarm clocks and Hi-Fi music systems can communicate together to offer cross-devices multimedia services. Smart HVAC (Heating, Ventilating, and Air Conditioning) systems make homes and buildings more energy efficient by monitoring and controlling their energy consumption. RFID-tagged objects along supply-chains facilitate logistics, surveillance and tracking of transported products ([9], [10]). However all these applications are built in silos. They are domain, technology and purpose-specific. Based on the definitions given above, we believe that the IoT vision and the future IoT applications are about breaking these silos, bringing more values and offering more services to users. In the following we propose a simple scenario to better illustrate our some aspects of our IoT vision:

Mary's daughter wants to go playing in the nearby park. Mary wants to know whether the park is crowded. Through an IoT application installed on her smartphone, she sends a request to find a service that can give her an instant picture of the park. The application returns a list of services pointing to cameras around the park that can instantly take pictures. In the proposed list, Mary can choose among a number of fixed cameras that have been installed by the local community to oversee the park, or cameras from smartphone users who are at the park at that time and are willing to let her use their phone cameras. Mary can select a service, connect to a camera and have an instant image of the park.

This simple scenario shows an example of loosely coupled interactions between a user and connected objects through an IoT application. The user is able to express a query, search and access functionalities offered by connected objects. He has neither a priori knowledge about possible services that he can access, nor a priori knowledge about the underlying technology to connect to objects and interact with. In our IoT vision, we also consider that a user can give an approximate description of desired functionalities and that the search mechanism is able to find services similar to the given request. We call this a flexible search. This is useful, for instance, if the user does not have an exact knowledge of a service or if the requested service is not available, then a similar one is returned. Another aspect to consider in our vision, is the ability for the user to find all the services that respond to a query. For instance, this is useful if the user wants to find all the services that measure the temperature in a location to estimate an average value. In this case, the search mechanism is able to return all the services that match a request within a scope of search. We call this an exhaustive search.

1.2 Toward a service-oriented middleware for searching IoT things

In the following, we call the services offered by the connected objects as *IoT services*. Devices and their related IoT services are associated to a physical space that we call a smart space. Besides, we define service search, the action of looking for and retrieving service descriptions that satisfy, at a certain degree, a given request and without having any priori knowledge about them. Finally, we call service discovery the action of both service registration in a registry and service search.

The realization of the scenario and the IoT vision mentioned above requires searching for IoT services while dealing with a number of issues related to the characteristics of an IoT environment. These IoT environment characteristics are: its scale, its heterogeneity and its dynamicity [11].

Regrading the scale and according to [12], there will be around 50 billions of connected objects by 2020. Moreover, the number of connected devices per person increased from 0.08 in 2003 to 1,84 in 2010 and the number of smartphone users will surpass 1.75 billions in 2014 [13]. When considering the number of possible sensors and actuators hosted in each smartphone and when knowing that the number of Internet users is growing fast, we can expect a rapid increase in the number of IoT devices. With scale, several constraints can be considered. For instance, the system needs to be able to identify effectively and access appropriate things, among billions, to provide needed functionalities [14].

Another challenge related to IoT environment characteristics is its dynamicity. This dynamicity is due to the mobility of devices. It is also due to the fact that devices can be broken and replaced by new ones. Moreover, some devices can have limited availability due to their constrained resources to partake in a sensing, actuation or processing task. As a consequence, IoT services availability will be changing over time and sometime their location will be changing due to mobility. The system needs to be able to manage such dynamicity and identify quickly available IoT services within a given smart space [15].

Finally, interoperability is essential to have a global infrastructure of things able to communicate together and to compose their services despite their heterogeneity. However, things in the IoT, manufactured by different entities, will likely have different communication interfaces, different ways to be addressed and different capabilities [16]. This heterogeneity will make finding suitable services to interact with, challenging. Thus, it is crucial to think about solutions to enable homogeneity among things.

To overcome these challenges, middleware helps realize and maintain complex distributed systems, by providing the necessary abstractions, interfaces and system support mechanisms. For IoT systems, such an approach typically allows providing an abstract interface not only to the heterogeneous and possibly constrained underlying hardware but also to the defined support software components of the system. It facilitates software development and execution by relying on software orchestration mechanisms. As a result, a great deal of work has been put into advocating and designing middleware frameworks for IoT to facilitate and reuse such developments. Service-Oriented Computing (SOC)/ Service-Oriented Architecture (SOA) [17] are dominating approaches when considering IoT middleware design ([18], [19], [20], [21], [22]) where devices and their functionalities are abstracted as a services.

Nevertheless, even after interoperability has been established at the hardware and application level, searching and interacting with IoT services requires service providers and requesters to agree on the semantics of services. One could assume that the same syntax, for describing service semantics, is shared between the provider and the requester. Although such assumption can be achieved at a small scale when considering a particular system, in the context of the Internet of Things, when considering billions of things, such assumption becomes hard to achieve. In this context, *Semantic Web Technology*, introduced by Berners-Lee in 2001 [23], is a promising approach to address syntactic heterogeneity of service descriptions. It enables semantic modeling of information and functionalities offered by a service and a global common understanding of service semantics. Semantically enriched information is machine interpretable and refers to ontologies representing a specific area of knowledge [24]. Ontology languages support formal descriptions and machine reasoning. Multiple research efforts have been conducted to use ontologies to model sensors and actuators in the context of IoT ([25], [26], [27], [28]). Nevertheless, finding services whose semantic description is in conformance with a given request often implies semantic reasoning which is costly in terms of time and computational resources [29] specially when considering large scale, mobile and resource-constrained devices [30].

1.3 Contributions and Outlines

To address the above challenges, we introduce in this thesis a service-oriented, semantic-based middleware with a main focus on the service search functionality. The middleware is able to manage heterogeneous, large scale and mobile devices while performing flexible, exhaustive and large scope search of IoT services. The most important contributions are structured along this document as follows:

- **Chapter 2** gives an overview about efforts to address the heterogeneity issue in the IoT. It also surveys existing solutions in the IoT to search for semantic-based IoT services and analyzes their ability to support large scale and mobile services and to perform flexible, exhaustive and scalable search.
- **Chapter 3** presents our semantic-based middleware architecture for IoT service discovery in heterogeneous, large scale and dynamic context. We give an overview of the middleware architecture and we focus on its semantic layer that handles semantic-based IoT service search. We detail its main components and their interactions. The semantic layer is instantiated into semantic gateways, each one mapped to a smart space and hosting its IoT semantic service descriptions. Semantic gateways handle search requests. We published initial ideas about the semantic gateways in the *IOT-A* project deliverable D4.1 [31].
- **Chapter 4** considers a static context and details the semantic-based service discovery mechanism to register and search large scale semantic-based service descriptions within semantic gateways. Based on clustering and aggregation mechanisms of IoT semantic service descriptions in each semantic gateway, the system is able to return all services that respond to a request while limiting the number of needed service-request matching operations. We described parts of this work in our publication at *iThings 2013* [32] and the *IOT-A* project deliverable D4.3 [33]. A patent on some of these ideas was filed [34].
- **Chapter 5** considers a dynamic context due to IoT service mobility and details mechanisms to enable dynamic management of semantic gateways content. For that, we propose an incremental clustering mechanism to update the service clustering in each semantic gateway. Thanks to these mechanisms, the search mechanism is flexible, exhaustive and scalable time. We described most of this work in a paper that will be published at *PIMRC 2014* [35].
- **Chapter 6** concludes the thesis. It provides a summary of our contributions along with the remaining research issues and perspectives to be further investigated.

Appendix A provides some details on the prototype implementation used in simulations.

Chapter 2

Searching IoT services: State of the Art

We survey in this chapter efforts for searching services in the Internet of Things. Before introducing the different approaches, we first give an overview about efforts to integrate physical objects to the Internet and tackle their heterogeneity issue. Section 2.1 gives an overview about techniques to connect things and expose their functionalities as services within the network. While section 2.2 gives an overview about different efforts to tackle the syntactic heterogeneity within IoT service descriptions based on Semantic Web technologies. Then, we describe in section 2.3 the main existing techniques in the state of the art to search for semantic based services. Finally, we focus in section 2.4 on IoT specific existing solutions to search for semantic-based IoT services. These solutions implement search techniques described in the previous section. We analyze their ability to fulfill the search requirements defined in the previous chapter for our system: scalability, flexibility, exhaustivity, large scope and mobile services support.

2.1 From Physical Things to Virtual Services

Many efforts have been deployed to integrate physical objects into the digital world. We survey in this part efforts to connect things and make them available on the network to interact with them despite their communication heterogeneity. To facilitate the integration of things into existing structures, one widely used approach is to abstract them as services following Service-Oriented Approaches (SOA).

2.1.1 Connecting and interacting with things over the Internet

The Internet of Things is characterized by the diversity and heterogeneity of connected devices and their supporting technologies. These devices are ranging from resource-constrained (e.g., wireless sensors and actuator networks (WS&AN)) to rich-resources (e.g., printers, lamps, home appliances). They are implementing different communication protocols (e.g., bluethooth, Zigbee, Wifi) to interact together. To bridge things to the Internet, specially for low power devices, approaches relying on the concept of gateways have been adopted ([36], [37], [38]) to federate different network protocols and to ensure bi-directional translation between different protocol stacks (for instance between IP and non-IP protocols). Bridging the physical world to the virtual one, has been enhanced by the development of uniform protocols such as CoAP (Constrained Application Protocol) [39] that enables lightweight communication with constrained devices. CoAP enables end to end HTTP communications between applications and resource constrained devices by compacting the form of HTTP request at the device level. At The networking layer, 6LoWPAN [40] enables to carry IP packets on constrained IP networks.

2.1.2 Abstracting things as services: SOA meets IoT

After taming the networking level heterogeneity, several efforts have been undertaken to decouple the hardware specifications of devices from their functionalities by abstracting them as services. The needed abstraction can be provided by the Service-Oriented Computing (SOC)[17] that considers services as fundamental elements for developing applications and solutions. SOC is typically embodied through SOA (Service-Oriented Architecture) that provides an architectural design pattern to achieve loose coupling among interacting services. SOA was first aimed to be used in the business computing entities in order to enable computers connected over a network to cooperate and provide large software applications. SOA defines a relationship between 3 participants: a service provider, a service consumer and a registry. The service provider defines a service description of the service it hosts and publishes it to a service registry that makes it discoverable. The service consumer searches for a service description in the service registry, and uses this service description to contact the service provider (See figure 2.1).

Service descriptions typically contain information about the service capabilities (purpose of the service and expected results), interface (publishes service signature), behavior (behavior of the service during execution) and quality (functional and non-functional service quality attributes, such as service performance, security, reliability, etc.). This information enable service discovery, selection, binding and composition.

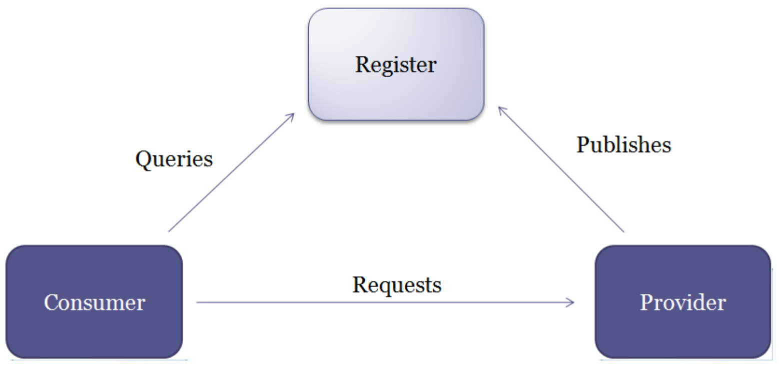


Figure 2.1: SOA main components.

2.1.2.1 Web services

Web services [41] are the most widespread implementation of SOA. The W3C distinguishes 2 classes of Web services:

- REST-compliant Web services, they are Web services " in which the primary purpose of the service is to manipulate XML representations of Web resources using a uniform set of *stateless* operations". Based on the definition in [42], they are "simple Web services" implemented based on the principle of REST and using HTTP [1] methods (e.g., POST, GET,

1. HTTP:http://www.w3.org/Protocols/

PUT or DELETE). They have no standard to define service interfaces that can be specified in various way (e.g., XML, WSDL, etc.). We call them in the following: *RESTful Web services.*
- Arbitrary Web services, in which "the service may expose an arbitrary set of operations". To detail more, they are designed to support interoperable machine to machine interaction over a network. They have an interface described in a machine-processable format, specifically Web Service Description Language (WSDL). We call them in the following *(WS-*) Web services.*

URIs [2] are used by both classes of Web services to identify resources. They use Web protocols such as (HTTP and SOAP [3]) for messaging.

2.1.2.2 SOA, middleware and IoT

Middleware solutions have been proposed in the Internet of Things as a software layer that can be decomposed into sub-layers interposed between the technological and the applications levels [6]. It hides the technological details and facilitates the development of IoT applications. Middleware architectures proposed in the last years for the IoT (e.g., HYDRA [43], SENSEI [18], SOCRADES [19]) often follow the Service-Oriented Architecture (SOA) as the approach for abstracting things as services [22].

Both types of Web services have been used in the IoT. (WS-*) Web services offer great granularity in describing concepts such as addressing, security, discovery and service composition. WSDL is well structured and beneficial to overcome heterogeneity issues. However it is a heavyweight XML document that requires expensive parsing which makes it unsuitable for constrained things. Efforts have been done to adapt WSDL to the needs of resource-constrained devices such as sensors [44]. Furthermore, lighter forms of (WS-*) such as DPWS (Device Profile for Web Services) [4] were proposed, defining a minimal set of implementation constraints to enable secure Web service messaging, discovery, description, and eventing on resource-constrained devices ([45], [19]).

Several recent research projects implement RESTful services for smart things [46] facilitating their integration to the actual Web and thus enabling what is called the *Web of Things* ([47], [48]). Although they have no standard to define service interfaces as for (WS-*) Web services, they are more suitable for devices with limited resources since they require less computational efforts than (WS-*) Web services [49]. Recently, WADL (Web Application Description Language) [5] has been proposed for RESTful services to provide a machine processable description of such HTTP-based Web applications.

2.1.3 Local service discovery protocols

The previous approaches enable to connect physical objects to a network (eventually to the Internet) and abstract them as services. To interact with these services, it is important to provide mechanisms to enable devices to declare and register them within the network, to make them discoverable by other devices. For that, a number of service discovery protocols have been proposed, such as JINI [6], SLP [7] and UPNP [8]. These protocols have been developed to enable applications to discover services in the local network. For that, when a device connects

2. URI:http://www.w3.org/Addressing/
3. SOAP:http://www.w3.org/TR/soap/
4. DPWS:http://docs.oasis-open.org/ws-dd/ns/dpws/2009/01
5. WADL:http://www.w3.org/Submission/wadl/
6. JINI:https://river.apache.org/doc/specs/html/discovery-spec.html
7. SLP:http://www.ietf.org/rfc/rfc2608.txt
8. UPNP:http://www.upnp.org/

to the network, it advertises its service and registers it to a service register that keeps track of networked services. These discovery protocols rely on broadcasting or multi-casting of the discovery queries to discover the registry in order to find or register a service. While these protocols are limited to local networks, mechanisms are required to search IoT services at a larger scope. However, when considering large scope of search, service descriptions are more likely to be heterogeneous.

2.2 Toward homogeneous service descriptions using Semantic Web technologies

Even after tackling the networking-level heterogeneity and abstracting things as services, the issue of syntactic heterogeneity in service descriptions remains. Semantic Web technologies have been proposed to tackle this issue. In this section, we first introduce the Semantic Web and its main technologies, then we provide an overview about efforts to semantically describe IoT services.

2.2.1 The Semantic Web

Based on Tim Berners-Lee vision [23], the Semantic Web constitutes an extension to the current Web, where information has associated semantics that can be processed and understood by machines to overcome data heterogeneity. Thus, using Semantic Web technologies enable autonomous interaction between computers and improve the cooperation between people and computers.

2.2.1.1 Main Semantic Web technologies

The Semantic Web stack is composed of many technologies as shown in figure 2.2. Several technologies like the Resource Description Framework (RDF) [9], Resource Description Framework Schema (RDFS) [10] or Web Ontology Language (OWL) [11] have been standardized and provide today the basic functionalities of the Semantic Web. We give a brief description of each technology in the following.

A core data representation format for Semantic Web is Resource Description Framework (RDF). RDF is a framework for representing information about resources. They are based on triples subject-predicate-object that form a graph of data. All data in the Semantic Web use RDF as the primary representation language. RDF Schema (RDFS) was created to describe taxonomies of classes (e.g, sub-class, super-class) and use them to create lightweight ontologies.

In [50], an ontology is defined as “a formal, explicit specification of a shared conceptualization” and it represents knowledge within a domain through a set of concepts (called classes) which are related to each other. More detailed ontologies can be created with the Web Ontology Language OWL. OWL is a language derived from description logic, and offers additional constructs over RDFS. OWL comes in three species: OWL Lite for taxonomies and simple constrains, OWL DL for full description logic support, and OWL Full for maximum expressiveness. A class is a type of resource on the web identified with a URI. In the following we refer to classes of an ontology as “concepts”.

9. RDF:http://www.w3.org/RDF/
10. RDFS:http://www.w3.org/TR/rdf-schema/
11. OWL:http://www.w3.org/2001/sw/wiki/OWL

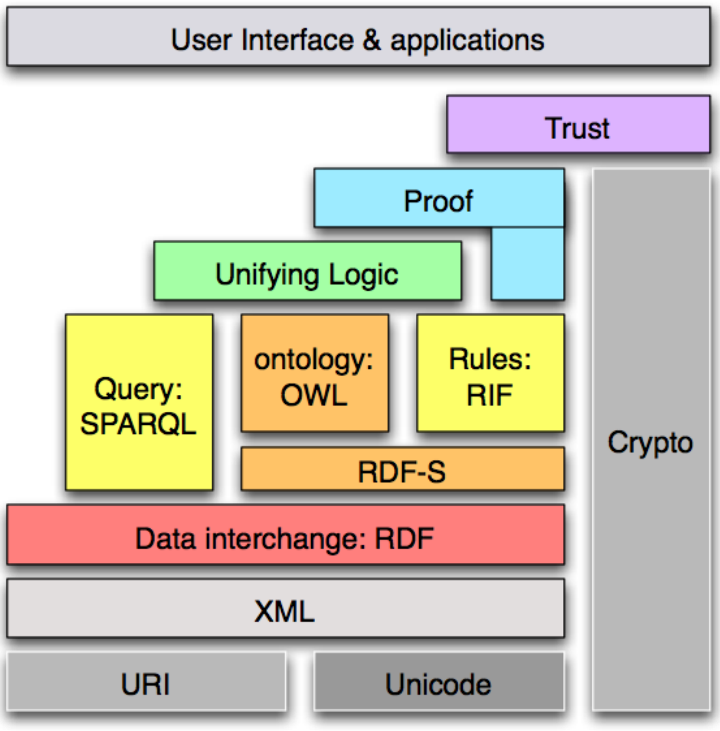


Figure 2.2: The Semantic Web stack from W3C

A semantic reasoner is a piece of software able to infer logical consequences from a set of asserted facts or axioms. The inference rules are commonly specified by means of an ontology language. Examples of existing semantic reasoners include: Racer [12], FaCT++c and Pellet [13].

Linked Data [51] describes a method of publishing data encoded in RDF and spanning different knowledge domains (e.g., nature, geography, science, culture, etc.) so that it can be interlinked.

Semantic Web technologies cited above, provide an environment where applications can query data and draw inferences. They build upon standard Web technologies to share information in a way that can be read automatically by computers. This enables data from different sources to be connected and queried and to create a Web of Data (as opposed to a sheer collection of data sets).

2.2.1.2 Semantic Web service descriptions

In order to overcome the shortcomings of syntax-based service descriptions, several researchers have proposed the usage of Semantic Web technologies in order to annotate service descriptions to make them machine-interpretable and facilitate their discovery, execution and composition [52]. Several initiatives have contributed to (Semantic Web Services) SWS standardization. We list in the following some examples of SWS:

OWL-S [14](Web Ontology Language for Web Services), is a service description ontology based

12. Racer:http://racer.sts.tuhh.de/index.html
13. Pellet:http://clarkparsia.com/pellet/
14. OWL-S:http://www.w3.org/Submission/2004/07/

on the W3C standard Web Ontology Language (OWL). It comprises three main parts (see figure 2.3): Service Profile, which is for advertising and searching service capabilities (e.g., inputs, outputs); Process Model, which gives a detailed description of a service's operation; and Service Grounding, which provides details on how to inter-operate with a service, via messages. Generally speaking, the OWL-S service profile provides both the functional and non-functional information needed to search a service. The OWL-S service model and the OWL-S service grounding, provide both enough information to tell how to make use of a service, once found.

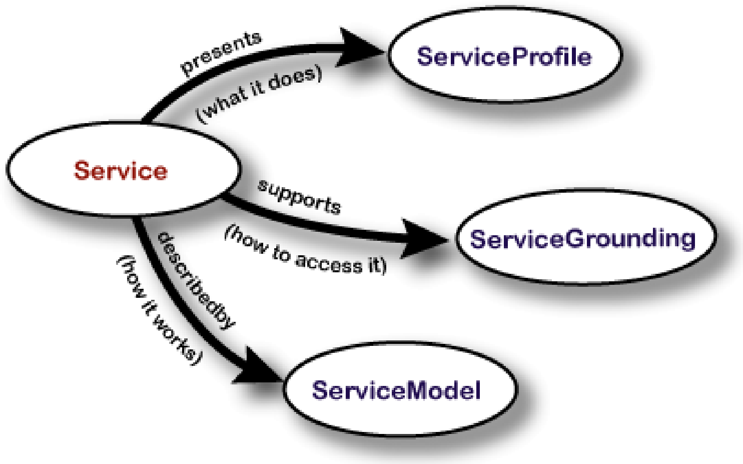


Figure 2.3: Top level of the OWL-S service model

WSMO [15] (Web Service Modeling Ontology) provides an in-depth conceptual model for describing service semantics in order to facilitate the automation of service discovery, composition, and invocation. WSMO comprises four subsystems: (1) ontologies that formalize domain knowledge, (2) goals that describe users' objectives, (3) Web Services and their semantic descriptions; and (4) mediators for enabling interoperability and resolving heterogeneity. The description of WSMO elements is represented using the Web Service Modeling Language (WSML). WSMO also defines the conceptual model for WSMX, a semantic Web services execution environment. Thus, WSMO, WSML, and WSMX form a comprehensive framework for modeling, describing, and executing semantic Web services. Due to its complexity, light forms of WSMO were proposed such as MicroWsmo [16] and WSMO-Lite [17].

SAWSDL [18] (Semantic Annotations for WSDL and XML Schema) and **WSDL-S** [19] (Web service semantics) enable semantic annotations for Web Services by using and building on the existing extensible framework of WSDL.

We should note that lightweight semantic annotations such as RDFa [20] and Microformats [21] have been also proposed to semantically annotate (X)HTML documents.

15. WSMO:http://www.w3.org/Submission/WSMO/
16. MicroWsmo:http://www.wsmo.org/TR/d38/v0.1/
17. WsmoLittle:http://www.w3.org/Submission/2010/SUBM-WSMO-Lite-20100823/
18. SAWSDL:http://www.w3.org/2002/ws/sawsdl/
19. WSDL-S:http://www.w3.org/Submission/WSDL-S/
20. RDFa:http://rdfa.info/
21. Microformats:http://microformats.org/

2.2.2 Semantic Web technologies and IoT

Many efforts have been made to semantically model IoT resources. For instance, the Sensor Web Enablement(SWE) initiative of The Open Geospatial Consortium (OGC) [53] proposes taxonomies to model sensors:

- Sensor Model Language (SensorML): Provides standard models to describe geometric and observational characteristics of sensors. It can be used to characterize a wide range of sensors from thermometers to satellites.
- Observations & Measurements (O&M): It is a conceptual schema encoding for observation and measurements. It is used in the context of geographic information systems and defines a core set of properties for an observation such as features of interest, observation result and observation time.

The taxonomies above are provided in XML, they are useful for data presentation. However, they do not enable to reason over the meaning beyond the syntactic level. As a consequence, ontologies were built on the SWE taxonomy such as SNN [54] provided by the W3C Semantic Sensor Network Incubator Group.

Recently, efforts aimed at extending the ontologies with IoT-specific semantics to describe things and functionalities they provide. In this context, we can list Sense2Web [55] and also the reference model in IoT-A European project [56] that provide ontologies to model the following thing-related concepts as described in [11]: the Entity (equivalent to a feature on interest); the Device, which is the hardware component (equivalent to a Thing); the Resource, which is a software component representing the entity; and the Service through which a resource is accessed. Other efforts to extend ontologies with IoT-specific semantics, including things and the functionalities they provide have been proposed (e.g., [57], [58], [59]).

These ontologies are used to annotate IoT resource descriptions and facilitate their search.

2.3 Techniques for searching semantic-based service descriptions

We investigate in the following main techniques from the state of the art to search for semantic based service descriptions and describe their characteristics.

2.3.1 Semantic service description matchmaking

Semantic service matchmaking [60] is the process of finding suitable services to fulfill a given service request [61]. A reasoner is implemented within a matchmaker to enable such a process. We can distinguish different categories of semantic service matchmaking:

Logic-based approaches in service search are based on reasoning over the semantic descriptions of Web services to match queries with service parameters. A number of efforts have been conducted in the area of matching semantic Web services. Service matchmakers, such as OWLS-UDDI [62], Inter-OWLS [63] and SAM [64], match service advertisements with queries by inferring the relations between their underlying concept classes [65]. A widely-used classification of matching degrees in logic-based software component matching is defined by Zaremski and Wing in [66]. Based on this classification, they present in [5] four levels of matching between a query presented by a set of concepts $C_R = \{C_1, C_2, ..., C_n\}$ and a service presented by a set of concepts $C_S = \{C'_1, C'_2, ..., C'_n\}$:

- **Exact**, which represents perfect semantic identity of the query parameters (e.g., input) with the service advertisement properties, denoted as $C_R \equiv C_S$,
- **Plugin**, in which parameters of the service advertisement are fully contained in the concepts of the query, so that the query has at least one concept which is more general than a corresponding property of the service ($C_S \subseteq C_R$),
- **Subsume**, in which concepts of the query are fully contained in the parameters of the service advertisement, so that some of the service parameters may not fully answer the query ($C_S \supseteq C_R$), and
- **Disjoint**, in which none of the parameters of the service can be related to any concept of the query ($C_S \cap C_R = \emptyset$).

Logic-based approaches are accurate, however, in some cases and due to their rigidity, pur logic-based matching may lead to some false negative matching. Moreover, logic-based matching is complex and lacks scalability when dealing with a large number of services [67].

Hybrid approaches have been proposed to augment logic-based matching with other non-logic matching methods, especially from the information retrieval (IR) field. Works such as LARKS [68] and OWLS-MX [69] introduce methods that combine text-based similarity and logic based matching of concept classes. The purpose is to reduce the complexity of semantic matchmaking and return more possible results.

2.3.2 Graph-based query: SPARQL based retrieval

Graph-based query languages allow to fetch RDF [70] triples based on matching triple patterns with RDF graphs to find the sub-graph that matches the query triple patterns and retrieve information from the Web of Data (e.g., from Linked Data). There are several graph-based query languages with different features to query the web of data [71]. SPARQL (Simple Protocol and RDF Query Language) is the only language that is a W3C Recommendation [72] and is the most widely used query language for the Semantic Web. SPARQL is a SQL-like language for querying RDF data. It uses RDF triples and resources for both matching parts of the query and for returning results of the query. Since both RDFS and OWL are built on RDF, SPARQL can be used for querying ontologies as well. SPARQL extracts resources that match exactly the query.

2.3.3 Indexing-based search: Semantic search engine

With the development of the semantic web, classic search engines such as Google and Yahoo extended their search to support semantics. For instance, Google uses expansion queries [73] to augment a query with additional terms that convey the same meaning. This enlarges the scope of search and provides better results. Google also integrated the Knowledge Graph [22] to enhance its search results with semantic-search information gathered from a wide variety of sources. Some semantic search engines were also proposed such as Swoogle [74] or Semserach [75] to perform semantic-based search. They are based on indexing techniques similar to classical search engines. However, they do not only index keywords in web pages, they also index semantic data which is used during search.

Three main steps are performed to perform search: first, crawling the web; where a bot visits web pages and navigates the web based on hyperlinks and downloads the visited pages. The crawler actually sends HTTP requests to fetch pages and send them to the indexer. The frequency of crawling a page is proportional to the frequency of change in the page. The indexer

22. KnowledgeGraph:http://www.google.com/insidesearch/features/search/knowledge.html

stores the document in a structured data base. It extracts keywords from documents and also semantic data and index them. The most used indexing method is the inverted index [76] where each keyword in the document is extracted and associated to a list of documents that contain these keywords. During search, the keywords of the user are matched with indexed keyword to extract documents.

2.4 Searching semantically-described IoT services

Some of the techniques surveyed in the previous section have been implemented in IoT middleware to support searching for IoT services. We provide in this part a state of the art about main existing approaches. We focus on semantic-based IoT service search techniques as they tackle heterogeneity issues and we analyze their ability to support service mobility and provide flexible, exhaustive, large scope and scalable search.

The adoption of REST approaches facilitates the integration of smart things into the Web [47]. Thus, one may think about indexing and searching Web pages describing IoT services using existing search engines to enable large scope search. However, searching for services describing things is significantly more complicated than searching for documents [77] as things are tightly bound to contextual information such as location, they can move from one context to another and have no easily indexable properties such as human readable text in the case of a *classical* Web document. Semantic annotation formats such as RDFa[23] and Microformats[24] can be embedded into IoT service descriptions to facilitate their search by search engines. However, these search engines do not support mobility and rapid contextual changes. Search engines are still largely based on scheduled indexing and might not reflect the latest context of registered smart things. Snoogle [78] has been proposed for searching real world and mobile devices. Also, Dyser, a Real-time search engine has been proposed in [79] to deal with service mobility and frequent changes of generated data. However, these approaches were not integrated into large scale systems.

In [80] they consider local search engines related to smart spaces. Each search engine is hosted in a smart gateway where IoT services are registered. Service descriptions are enhanced using Semantic annotations and indexed using the inverted index technique [81]. Smart gateways are interconnected hierarchically (e.g, building, floor, room). The benefit from this approach is to perform localized search queries based on context information. Different types of queries can be supported, including the exhaustive search query that spans a whole sub-tree from the hierarchy. The proposed approach is limited to exact matching of the query and lacks scalability since all nodes within a sub-tree are visited in case of an exhaustive search, even when the node does not contain the required resource.

In SOCRADES project [19], they demonstrated how to use a Web service standard (DPWS) and Web oriented patterns (REST) to easily integrate physical devices into existing enterprise information systems. To search for devices, user queries are augmented via other third parties web sites (e.g., yahoo!, Google, Wikipedia) to find synonyms. This enables to avoid having ontologies and minimizes the amount of data that devices need to provide upon discovery and service registration.

In [82] they propose a hybrid matchmaker to search for semantic based IoT services. The purpose is to provide an efficient and lightweight approach to search semantic web services.

23. RDFa:http://rdfa.info/
24. Microformats:http://microformats.org/

Although, their approach presents better results than a pure-logic based approach, they did not evaluate its scalability when considering large scale IoT services.

In [83], they present a search process relying on semantic profiles for connected objects. They establish similarities between semantic profiles to regroup them into clusters in order to accelerate the search and selection of the suitable services. The prototype scalability was not evaluated and the considered use cases were limited to a small scope.

Similar to [80], in [57], they propose a hierarchy of nodes representing smart places to manage massive data of the Internet of Things. Nodes are federated together and are able to detect new service arrivals and departures and to share the information with nearby nodes in order to enable scalable search and management mechanisms for the IoT services. The search is based on SPARQL query to retrieve resources. Another approach in [84] proposed an architecture aiming to open the access to sensors, allowing enhancing integration of data and services of heterogeneous sensors and facilitating novel applications. They provided vocabularies allowing to integrate descriptions of sensors with Linked Open Data (LOD) [85] and proposed search mechanisms taking into account states of sensors (e.g. availability). Their approach to answer a user request was to query a triple store by using the SPARQL protocol to retrieve data. The use of federated nodes in [57] allows more scalable search than the approach in [84]. However, by using SPARQL, the search is only limited to resources that provide an exact match to the original query and thus does not allow flexible and exhaustive search as we required.

A summary of the approaches presented in this section and their properties are presented in figure 2.4.

	Flexibility	Exhaustivity	Support for large scope search	Support for service mobility	Scalability
[78] [79]		✓		✓	
[80]		✓	✓	✓	
[19]	✓	✓			✓
[82]	✓	✓			
[83]	✓	✓			
[57]			✓	✓	✓
[84]		✓			✓

Figure 2.4: Comparaison of existing approaches for semantic-based IoT service search

2.5 Discussion

We presented in this chapter approaches for searching IoT services while considering the heterogeneity, mobility and scalability issues. For the heterogeneity issue, many efforts have been provided to connect IoT devices to Internet and abstract their functionalities as services following Service-Oriented Approaches (see section 2.1). Moreover, Semantic Web technologies have been investigated to model IoT services with machine-interpretable semantics (see section 2.2) and thus, enable to tackle the syntactic-level heterogeneity issue.

In our approach, we are interested in solutions that are able to support large scale and mobile services while providing flexible and exhaustive search. Semantic-based solutions have the ability to provide flexible search, however they lack scalability when considering large scale systems, due to the cost of semantic reasoning.

We surveyed existing semantic based search solutions in the IoT (see section 2.4) and analyzed their ability to fulfill the requirements mentioned above. From figure 2.4, we can see that existent solutions respond only to a sub-set of the mentioned requirements. Issue related to searching large scale and mobile IoT services while providing flexible, exhaustive and scalable search have not been fully resolved. We proceed in the following chapter to present our approach towards addressing these requirements.

Chapter 3

Middleware architecture for searching semantic-based IoT service

As stated in the previous chapter, Service-Oriented Architecture (SOA) is a design pattern that has been investigated to develop middleware for IoT. Within these middleware, the functionalities of a device are abstracted as a service that can be used by IoT applications. This enables to decouple the heterogeneous hardware and software specifications of a device from its functionalities. Besides, Semantic Web technologies enhance the interoperability of service descriptions. They enable semantic modeling of functionalities offered by a service in a machine interpretable way. Although SOA defines a set of core functionalities to implement [17], we mainly focus in our work on service discovery and we optimize the service search [1] to adapt it to IoT constraints in terms of heterogeneity, scale and mobility.

In the following, we give first an overview of the middleware architecture in section 3.1. Then, we focus in section 3.2 on the semantic layer that handles semantic-based service discovery and which constitutes our core contribution. We detail its main components and describe their interactions. Finally, in section 3.3, we describe possible cases of middleware deployment in smart spaces. For all figures in this chapter, the middleware components that represent our core contributions are colored in blue.

3.1 Architecture overview

The proposed middleware enables easy discovery and interaction between IoT applications and IoT devices. It hides the hardware and software heterogeneity among IoT devices and provides a uniform view to the application developer. It also provides a smart management of a large scale and mobile things in order to answer efficiently a user requests. It should be able to manage a large number of mobile things (sensors, actuators, etc.) which can communicate over different network technologies (from low-rate wireless personal area networks to Wifi and 3G/4G networks [86]). Semantic service descriptions are stored within a registry in the middleware. The system supports users sending queries through IoT applications (for instance from his

1. In SOA, service discovery is defined as service registration and service look-up. In our approach, we defin in section 1.2 and use the term service search instead of service look-up.

smartphone) to find a suitable service offered by a device, to acquire data from or to perform an action on it. Using a dedicated IoT application, the user enter keywords about desired service parameters (i.e., inputs/outputs). These keywords describe concepts to measure or act on (e.g., temperature). The user may optionally specify a location if he wants to perform a location-based service search.

In the following, the overall architecture of the Middleware is sketched in figure 3.1. We represent components that allow end-to-end discovery and communication between IoT applications and devices. In the literature, some existing IoT middleware implement also a composition component to create new services from the existing ones based on the user request ([18], [45], [87]). For our tests and evaluation we implemented the service discovery component that represents our main contribution (for more details about the implementation see Appendix A)

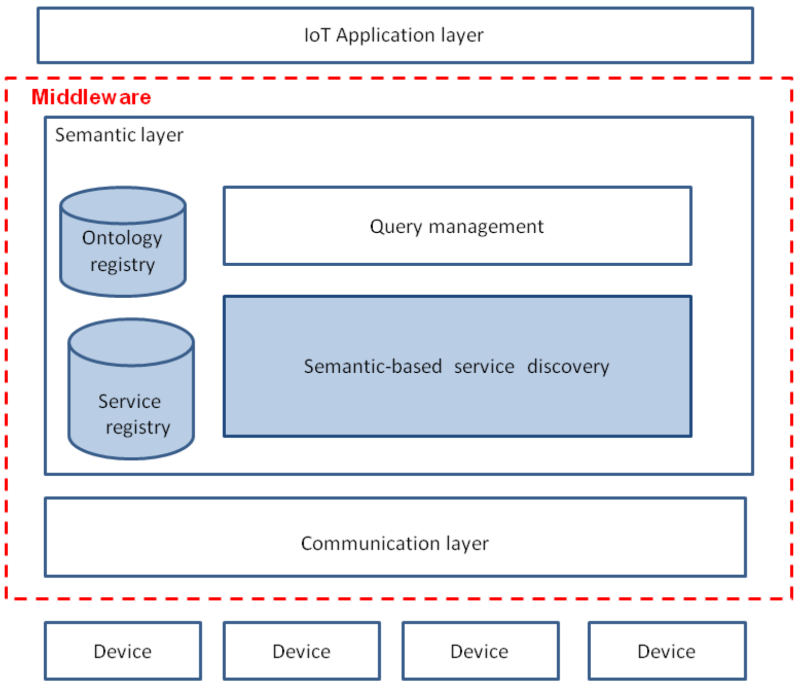


Figure 3.1: Semantic Service-Oriented Middleware for service discovery in the context of the Internet of Things

The middleware is composed mainly of 2 layers that we describe in the following:

- **The communication layer**: The communication layer provides two mains functionalities. First, it federates heterogeneous communication protocols and enables network-level discovery of devices. Typically, it detects mobile devices based on service discovery protocols ([2], [3], [4]) and help gather relevant information [28] to compute the semantic service descriptions to be registered in the semantic layer. Second, when a pointer to a service is returned to the user (in response to a search request), the communication layer allows interacting with the device and accessing its service without worrying about access protocols.
- **The semantic layer**: The semantic layer enables semantic-based service discovery. It implements a set of components: the query management component that receives the user queries, extracts their main concepts and generates a semantic description that will be used to perform service search. The semantic-based service discovery component registers semantic service descriptions in the service registry and performs service search upon the reception

of a user request. The Ontology registry contains ontologies used to annotate the semantic service descriptions in the service registry. The semantic layer is the core of our contribution. Its components will be further detailed in section 3.2. The mechanisms implemented in the semantic layer are detailed in chapters 4 and 5.

Figure 3.2 details the different interactions between components of the semantic layer and the other components represented in figure 3.1, upon the reception of a request from an IoT application to search and access an IoT service offered by a device.

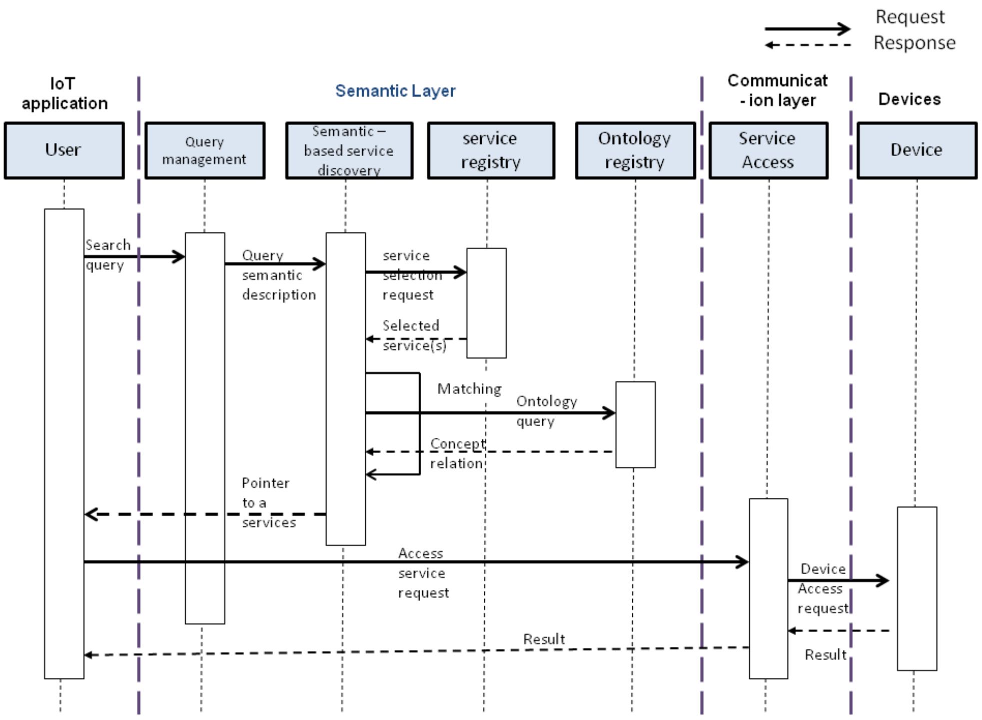


Figure 3.2: Sequence diagram representing the interactions between the different components of the semantic layer and the others middleware components to search and access an IoT service offered by a device, upon the reception of a user query.

3.2 The semantic layer

We detail in this section the semantic layer. We first present its information model and then detail the main functions of its components. In the following, we describe the information model (i.e., ontologies, semantic service descriptions, requests) used in the semantic layer. We also describe the functionalities and interactions among the different components of the semantic layer.

3.2.1 Information model

We describe in this section the ontology model, service and request model used by the semantic layer. As we mentioned in chapter 2, many approaches using IoT ontologies and semantic service descriptions have already been developed (see section 2.2.2)to describe IoT entities (i.e., services and devices). We consider that semantic modeling of IoT entities has reached a mature stage and can be exploited to provide interoperability among things. We do not aim in this work to provide new semantic models to describe IoT entities. However, we want existing models (i.e., ontologies and semantic service descriptions) to be used within our middleware. For that, we give in the following a high level description about the considered models for ontology, service and request descriptions.

3.2.1.1 Ontology and concepts:

We consider an ontology $\mathcal{O}$ that includes a set of concepts $\mathcal{C} = \{C_i, i \in \{1, \cdots, M\}\}$, and the binary relation $\mathcal{R}$ that describes subsumption relations between the concepts where for any couple of concepts $(C_i, C_j) \in \mathcal{C}^2$, we say that:

- C_i is a super-concept of C_j if and only if it exists a path in $(\mathcal{C}, \mathcal{R})$ that starts with C_i and ends with C_j. We note $C_i \sqsupset C_j$.
- C_i is a sub-concept of C_j if and only if C_j is a super-concept of C_i. We note $C_i \sqsubset C_j$. A sub-concept inherits everything from the super-concept.

To include cases of equality in the space of equivalent concepts, we note: $(C_i \sqsupseteq C_j) \Leftrightarrow (C_i \sqsupset C_j) \vee (C_i = C_j)$.

3.2.1.2 Services and requests:

Models for semantic service descriptions have been proposed in the literature (see section 2.2.1.2) where service functional parameters (e.g., inputs, outputs) and non-functional parameters(e.g., locations) are annotated with a set of concepts from an ontology $\mathcal{O}$. In the following, we consider a service description formed by a set of parameters (functional and non-functional) where each parameter is annotated with a concept from the ontology $\mathcal{O}$ as follows:

Definition 2: Service.
We consider an ontology $\mathcal{O}$ that includes a set of concepts $\mathcal{C} = \{C_i, i \in \{1, \cdots, M\}\}$. A service S is represented by a subset of the whole set of concepts: $S = \{C_i, i \in I \subset \{1, \cdots, M\}\} \subset \mathcal{C}$. The whole set of potential services is $\mathcal{S} = \mathcal{P}(\mathcal{C})$. The whole set of existing services is a subset of $\mathcal{S}$.

We consider that a request R is a description of a desired service and it is also represented by a subset of the whole set of concepts as defined for a service in definition 2.

3.2.2 Main functions in the semantic layer

In the following, figure 3.3 gives a detailed view of the main components in the semantic layer.

In the following we describe each component of the semantic layer and its main functions:

3.2.2.1 The semantic service discovery component

The semantic service discovery component is composed of two main functional blocks:

- **Service registration:** This component contains several functions. The **service description insertion/removal function** is responsible for inserting new services in the service registry upon detection of a new device by the communication layer (or the removal of a service from the registry in the case of a device departure). The services are inserted in or

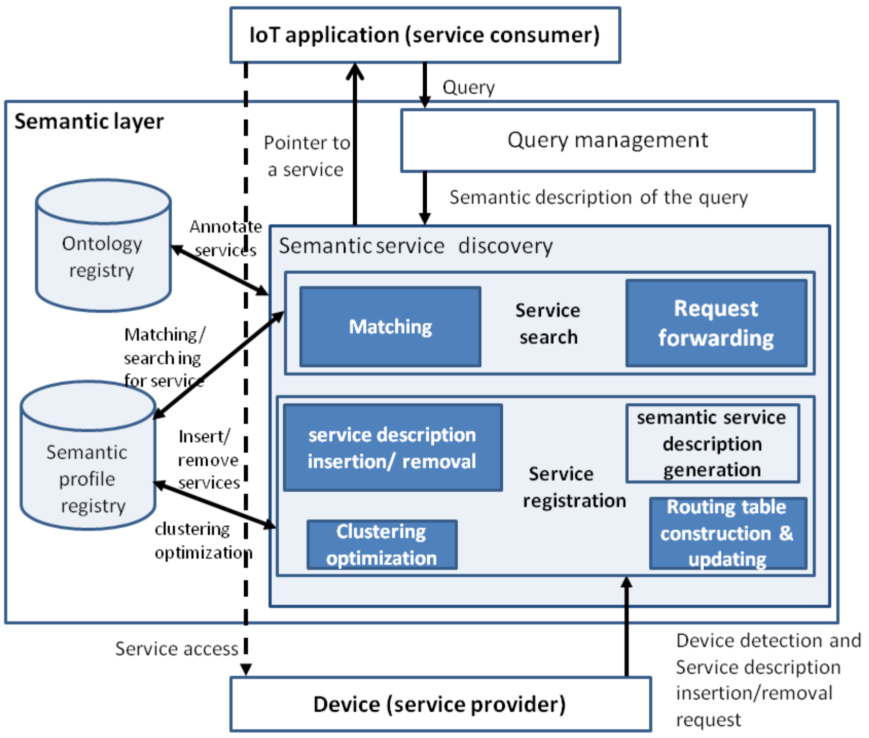

IoT application (service consumer)
Query
Semantic layer
Pointer to a service
Query management
Semantic description of the query
Ontology registry
Annotate services
Semantic service discovery
Matching
Service search
Request forwarding
Matching/ searching for service
Semantic profile registry
Insert/ remove services
service description insertion/ removal
semantic service description generation
Service registration
clustering optimization
Clustering optimization
Routing table construction & updating
Service access
Device detection and Service description insertion/removal request
Device (service provider)

Figure 3.3: The semantic layer components

removed from the service registry following an incremental clustering that creates groups of similar services. Before registering a service in the service registry, the **semantic service description generation function** generates the semantic service description of an IoT service based on semantic concepts from the ontology registry. After service insertion or removal following an incremental clustering as described above, some management operations are handled in the service registry. **Clustering optimization function** is responsible for optimizing the the repartition of services among clusters. **Routing table construction and updating function** aggregates the content of each cluster in the service registry into a routing table that is used during the service search. This function also updates the routing table when changes happen in the clustering.

- **Service search:** This component is responsible for identifying the service descriptions that respond to a request and returns pointers to these service descriptions to the user. It is composed of two functions. **Matching function** is responsible for matching the request description with service descriptions based on metrics and select the suitable services based on thresholds that we defined. **Request forwarding function** is responsible for sending the request to the suitable clusters within the service registry, based on the content of the routing table. It is also responsible for forwarding the request to suitable semantic gateways which realize the semantic layer functions in a distributed fashion (Semantic gateways are described in section 3.3).

All the mechanisms handled by the different functions are detailed in chapters 4 and 5.

3.2.2.2 The query management component

The query management component receives the user request from an IoT application. The query should contain some information about the desired service, where the user should specify some keywords to describe the desired functional and non-functional parameters (e.g., a physical concept to measure/act on in the input or output of the service, location). The query management component receives the user request, extracts its keywords and generates semantic descriptions to be matched with services. Many works have been investigated in the domain of automatic generation of semantic descriptions that can be used in this case ([88], [89], [28]).

3.2.2.3 The registries

We consider two types of registries: Ontology registry and Service registry. The former holds the knowledge base used to describe IoT services. The latter contains semantic service descriptions of IoT services. These descriptions are based on semantic concepts from the ontologies hosted in the ontology registry.

3.3 Deploying the solution through gateways

We give in this section a comprehensive view of the middleware deployment within an IoT environment realized as a collection of interconnected smart spaces. Although the middleware is intended to be deployed in a large scale environment (e.g., a city), we consider in this section the case of a smart building as an illustrative example

We consider a building with k floors and l rooms in each floor. We consider that each room contains a number m of connected objects. Examples of connected objects include sensors (e.g., temperature sensors, humidity sensors, presence detection sensors, etc.) and actuators (e.g., a projector, lamps, heaters, etc.). A user, using an IoT application on his smartphone, searches for IoT services around him. Figure 3.4 gives an example of connected object distribution over rooms of a floor within a building.

The semantic and communication layers of the middleware (see figure 3.1) are instantiated into respectively, semantic and physical gateways that we detail in the following.

3.3.1 Physical gateways

The physical gateways deal with device communication, supporting networks and protocols heterogeneity. Since embedded devices have limited resources which may prevent them from implementing full IP protocols, and as multiple networks and discovery protocols are used in IoT environments to cope with different requirements, *physical gateways* have been proposed in the literature to support multi-protocols and network federation ([48], [90], [91]). In our case, these physical gateways are typically mapped to a specific geographical location (e.g., a meeting room, office room, corridor, etc.) relying on their radio range. They have different interfaces (e.g., Bluetooth, Zigbee, Wifi, etc.) to detect and connect with new devices entering the location they cover. Upon new device detection, the physical gateway sends its IoT service description to the semantic gateway where it is annotated and registered. Physical gateways detect also device departures from their covered geographical zone and notify the semantic layer about these departures.

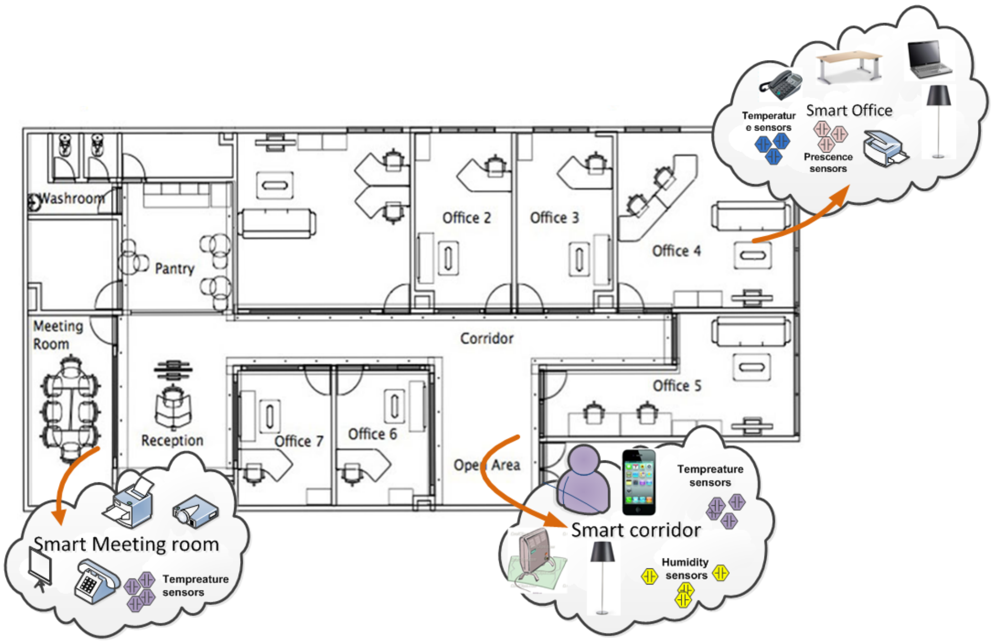


Figure 3.4: Deployment of connected devices around rooms of a floor within a building.

3.3.2 Semantic gateways

Semantic gateways are software components that host semantic service descriptions of IoT services related to a specific location. Semantic gateways perform semantic-based search of requested services. They can be either centralized into one server or fully distributed across many servers when considering a large scope scenario.

Semantic gateways are interconnected and communicate together based on a tree topology (see figure 3.5) representing the physical spaces with different levels of granularity (e.g., building, floor, rooms). Lowest level semantic gateways are mapped to spaces containing physical connected objects served by one physical gateway (e.g., rooms) while upper-level semantic gateways are mapped to spaces that include lower-level ones (e.g., floors, building).

In the tree graph topology of semantic gateways, each node N models a semantic gateway that represents a geographical space. In this way, the tree graph allows to have loosely-coupled and independent semantic gateways since they do not need to communicate directly together but only to their parents or children. We characterize the graph of nodes by the number n of levels and the number of children nodes d_i for each parent node from level $i+1$ with $i \in [1, n-1]$. For instance, in the graph example given by figure 3.5, we have a graph with 3 levels ($n = 3$) and 3 children nodes per parent node at the first level ($d_1 = 3$) and 2 children nodes per parent node at the second level of the graph ($d_2 = 2$).

3.3.3 Gateways mapping

Each Lowest level semantic gateway is mapped to a physical gateway (see figure 3.6). In our approach, we consider that the physical and semantic gateways are immobile and their mapping is static. This approach requires no efforts to maintain the tree structure of semantic gateways or to establish a dynamic mapping between a lowest level semantic gateway and a physical gateway, which come with communication overhead.

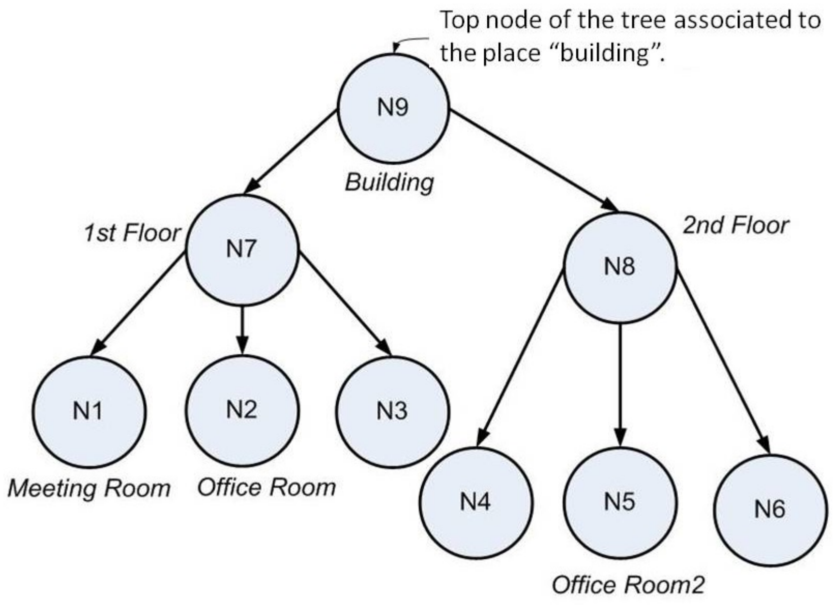


Figure 3.5: Logical deployment of semantic gateways in the case of a building.

Moreover, in a dynamic context where IoT services are mobile, it becomes easier to locate them based on the location of the semantic gateway where they are registered. The dynamic aspect of physical and semantic gateways is part of the future work.

3.3.4 Semantic gateway selection and user request forwarding

The user may optionally perform a location-based service search by giving a location as part of his request description. If a location constraint is expressed in the request, the IoT application, which has an updated list of available semantic gateways and their related locations, is able to associate the expressed location with a suitable semantic gateway and forward the user query to it. In case no location constraint is specified in the request, the IoT application can associate the user position to the nearest semantic gateway in terms of location proximity to the user and forward the query to it. The position of the user can be determined for example via GPS (Global Positioning System) if the user is outdoor or via WPS (Wifi-based Positioning System) [92] if the user is indoor.

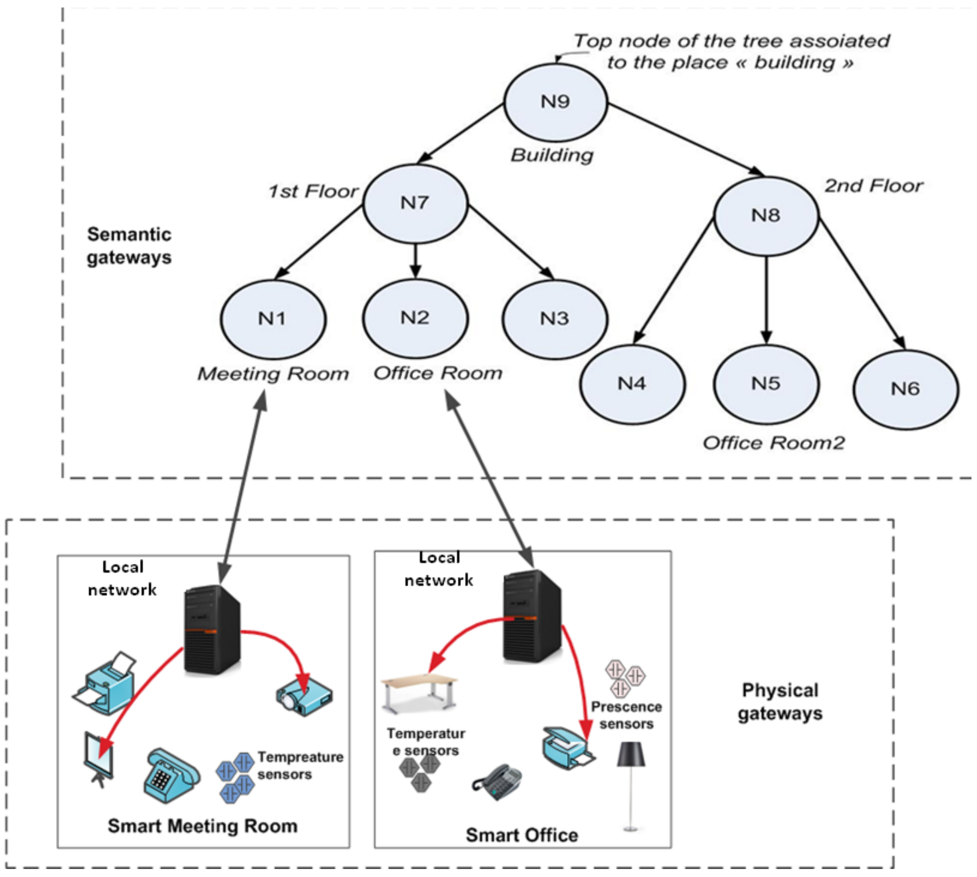


Figure 3.6: Mapping between the physical gateways and the semantic gateways

3.4 Concluding remark

In this chapter, we presented a high-level overview of the architecture of a semantic-based middleware built on a SOA approach for IoT environments. This middleware is designed to address large scale, heterogeneous and dynamic IoT services. First, we presented the different layers that constitute the middleware. Then, we focused on the semantic layer. This layer handles semantic based service discovery based on mechanisms of clustering and aggregation that reduce the scope of search when considering a large number of services. The semantic layer allows accurate search and implements mechanisms to support service mobility. In the following chapters, we first consider a static context and detail the mechanisms involved during the semantic-based service discovery. Then, we extend the system to a dynamic context and detail the mechanisms developed in order to support service mobility.

Chapter 4

The service search mechanism in a static context

In the previous chapter, we gave an overview of the middleware architecture to support semantic-based IoT service discovery. We detailed the main functions of its semantic layer and introduced the hierarchy of semantic gateways which are instantiations of the semantic layer in smart spaces. A semantic gateway, mapped to a smart space, register its semantic IoT service descriptions and process search requests for IoT services. In this chapter, we detail the different mechanisms embedded in semantic gateways to register and search IoT services. We consider in this chapter a static context where devices and their related IoT services are constantly associated with the same smart space. We extend our work to a dynamic context in the next chapter.

In the following, section 4.1 defines the problem and requirements for searching large scale semantic-based IoT services. Section 4.3 details our solution for an efficient and scalable discovery mechanism. Semantic matching operations performed during service search induce semantic reasoning which is time and resources consuming. The idea is to be able to retrieve all suitable services matching an incoming request and to reduce the search cost by limiting useless service-request matching operations. The discovery system embeds clustering and information aggregation mechanisms based on a quasi-metric that we defined. Based on routing tables and thresholds, the request is forwarded over the hierarchy of semantic gateways. Results in sections 4.4 and 4.5 show that the proposed mechanism performs exhaustive search while limiting its cost in term of numbers of matching operations.

4.1 Problem statement and requirements

With the expecting fast growing number of connected objects [12] and their heterogeneity increase, it would become hard for a user to have an exact knowledge about IoT services that he or she can interact with (i.e., description, ID, address, etc.). As we mentioned in our IoT vision in section 1.1, to search for a service, the user would rather give an approximate description. Moreover, in some cases where an explicitly requested service is not available (i.e., in case the related device is broken or being in use), a similar service can be returned instead. Thus, supporting a flexible search which allows to return services semantically close to the one requested by the user is a useful feature. Besides, in some use cases an exhaustive search is needed. By exhaustive search, we mean the ability to find all services that respond to a given request. For instance, this is useful if the average temperature of a building is requested and it implies to find all services that measure the temperature within the building to estimate an average value.

Semantic Web technologies allow semantic modeling of information and functionalities offered by IoT services. Information is enriched with machine interpretable semantics that refer to ontologies representing specific areas of knowledge. Logic-based approaches (see section 2.3.1) using the reasoning over the semantic descriptions of Web services to match queries with service parameters, enable accurate and flexible search. However, they are costly in terms of computational resources and present scalability issues ([67], [93]) specially when dealing with a large scale of services. Thus, it is important to consider the use of Semantic Web technologies in conjunction with distributed processing and mechanisms to optimize the search and limit its cost in terms of number of matching operations needed to return all services responding to a request. To summarize, we aim in the following to provide a semantic -based service search mechanism that responds to these requirements: flexibility, exhaustivity and scalability.

4.2 Background

We present in this part a background about techniques to optimize the semantic-based service search by reducing the number of matching operations needed to resolve a service request. Two approaches can be investigated:

- In the case of having distributed registries, optimizing the selection of registries to forward the request to, in order to avoid searching in registries that will not contain services matching a request.
- Organize semantic service descriptions in service registries, so that the number of matching performed for resolving a request is reduced.

To optimize the selection of registries to forward the request, Distributed Hash Tables (DHT) [94] have been investigated in pure P2P approaches. In DHT-based systems, peers and data are structurally organized so that the location of a specific piece of data (i.e., peer that contains this particular piece of data) can be determined by a hash function. Chord [95] is an example of such approach that employs a DHT-based scheme. While this approach is scalable and exhaustive (i.e., it is possible to find all peers that contain the requested data), it can only support exact match queries and comes with a cost to maintain consistency between nodes. Other approaches to optimize the query forwarding are based on caching mechanisms within nodes, where knowledge from previous search queries is used by peers that can not satisfy a query to forward the request to peers that can respond to the query [96]. However, this approach comes with a high storage cost within registries and does not enable efficient search in the case of receiving new requests that have not been treated before by the system. Approaches in hybrid P2P consider super peers that act as heads of sub structure within P2P networks systems. In [97] and [98] classification approaches are used, where peers sharing the same service category are grouped together under the same super node. However, classification approaches require that categories used in the classification are known in advance. Moreover, grouping nodes based only on service categories does not enable a fine-grained selection of nodes to which the request will be forwarded. Other approaches use Bloom filers [99] to summarize node content. Computation of the register's Bloom filter is based on hashing service descriptions within registries. However, Bloom filter presents some limitation to perform flexible and exhaustive search. For instance, It is not able to detect the case where a service query contains only a subset of a given service description in the register. Thus, the register would not be selected.

To optimize the search within a registry, approaches based on clustering ([100], [101]) and service classification [102] have been investigated to organize the registry content into groups of semantically similar services with respect to their descriptions. We want to avoid classification approaches that require a pre-knowledge of concepts and service categories used within the system. While this is feasible in small scale systems with a limited number of service categories, when dealing with large systems that involve a great diversity of service categories it become challenging to have such a priori knowledge of the whole system content. Clustering allows to create groups of similar services based on their description using defined metrics. An example of a clustering method is the hierarchical agglomerative clustering [103] which is often used in information retrieval for grouping similar documents [104]. This method uses a bottom-up strategy that starts by placing each service description in its own cluster, and then successively merges clusters together until a stopping criterion is satisfied. This process results in arbitrarily shaped clusters where similar services are grouped together. While these clustering approaches allow to reduce the search space by selecting groups of services within a registry to match with, the method to select the suitable cluster within a registry is generally based on indexing its content [105] which may not fully characterize its content to enable flexible and exhaustive search.

In our approach, we build a solution where we both select the registries where to forward the request and the set of services with which the request is matched in a registry, while ensuring flexible and exhaustive search.

4.3 Proposed solution

We define in this section an efficient and scalable semantic-based IoT service search mechanism based on flexible matching which is able to retrieve all services that respond to a request within the area of search while limiting the number of service-request matching operations. The discovery mechanism is distributed over the semantic gateways presented in the previous chapter (see section 3.3). We consider in this part the same notation defined in the previous chapter where we model the hierarchy of semantic gateways as a tree graph G with multiple levels ω that contain nodes N.

In the following, we detail the mechanisms in each semantic gateway that enable searching for IoT services. First, we define metrics used during request-service matching to estimate a "semantic distance" between the request and the matched service (see section 4.3.1). A request can be forwarded among semantic gateways in order to find matching services. In order to limit the number of service-request matching operations while ensuring flexible and exhaustive search, routing tables (see section 4.3.2) are created in each semantic gateway to "summarize" its content based on clustering and data aggregation mechanisms. Using defined thresholds in each semantic gateway (see section 4.3.3), it is possible to forward the request to semantic gateways that are more likely to host matching services and within a semantic gateway to limit matching operations in order to return suitable services (see section 4.3.4).

4.3.1 Matching metrics

The search mechanism is based on matching operations between the concepts of a request description and a service description, to determine whether a service responds to a request. Thus, it is important to define first what we mean by a "matching service" (i.e., a service that matches a request). Then, we characterize the matching through metrics that assess the degree of matching. These metrics are used during the service search. We defined in the previous chapter the semantic description of a service and a request (see section 3.2.1). We consider these definitions in the following.

4.3.1.1 Matching definition

For the IoT context, we stated in section 4.1 the importance of flexible matching in the search process in order to satisfy a user need. As we mentioned in 2.3.1, a number of efforts have been conducted in the area of matching semantic Web services (i.e., the identification of subsumption relationships between concepts describing parameters of a service and a request). In our approach, we consider the definition of Plugin matching given in [5], where a service S is relevant to a request R if the concepts of S are equivalent or subsumed by the ones of R. This ensures the requester that the provided service offers at least the requested functionality while providing more flexibility than an exact match. In the following, we define matching a request by a service as follows:

Definition 3: Matching a request by a service.
A service $S \in \mathcal{S}$ matches a request $R \in \mathcal{S} \Leftrightarrow \forall C \in R,\ \exists C' \in S | C' \sqsubseteq C$.

With definition 3, any super-concept of a concept in the service or the request does not affect the matching result. So, we can add to the request or the service any super-concepts of their representation, this will not change the status about the matching of the request by a given service. Therefore, we define in the following, the extensive representation of a service:

Definition 4: Extensive representation of a service.
The extensive representation $\tilde{S}$ of a service $S \in \mathcal{S}$ is:

$$\tilde{S} = \{C \in \mathcal{C} \mid (\exists C' \in S | C \sqsupseteq C')\}\,. \tag{4.1}$$

$\tilde{S}$ represents the set of all super concepts of its representation S.

The whole set of extended services is $\tilde{\mathcal{S}} = \left\{\tilde{S} \mid S \in \mathcal{S}\right\}$.

Based on this definition, we can now have a simpler characterization of the matching of a request by a service, where a service $S \in \mathcal{S}$ matches a request $R \in \mathcal{S}$ in the sense of definition 3 $\Leftrightarrow \tilde{R} \subset \tilde{S}$.

Proof. If there is a service $S \in \mathcal{S}$ that matches a request $R \in \mathcal{S}$ in the sense of definition 3, then $\forall C \in R,\ \exists C' \in S | C' \sqsubseteq C$. By applying the extensive representation of the service on R and S, we have $\tilde{R} \subset \tilde{S}$. Reversely, if $\tilde{R} \subset \tilde{S}$ then, $\forall C \in \tilde{R},\ \exists C' \in \tilde{S} | C' \sqsubseteq C$ and based on the definition of an extensive representation of a service, we have: $\forall C_1 \in R,\ \exists C'_1 \in S | C'_1 \sqsubseteq C_1$. Then, the service S matches the request R based on definition 3. □

The extensive representation will be used in the following to define matching metrics.

4.3.1.2 Matching metrics

Based on the matching definition given in the previous section, we formulate metrics that will be used during the matching process to estimate the "distance" between a service and a request. Then, we define a matching function to evaluate if there is a matching between a service and a request. To do so, we need first to define the coordinates of a service in the "space" of concepts which will be helpful in the next notations and definitions.

Definition 5: Coordinate functions.
For any service $S \in \mathcal{S}$, for each concept $C_j \in \mathcal{C}$, $j \in \{1, \cdots, M\}$, we define the following coordinate function:

$$\chi_j \ : \ \mathcal{S} \rightarrow \{0,1\} \tag{4.2}$$
$$S \mapsto \chi_j\left(S\right) = 1 \Leftrightarrow C_j \in S. \tag{4.3}$$

The coordinate function χ_j is equal to 1 if the concept C_j exists in S.

When matching a request with a service, we define the following function Q that evaluates how far a service is from a request, using the extensive representation:

Definition 6: Metric measurement.
We define a function Q as follows:

$$Q \ : \ \mathcal{S} \times \mathcal{S} \rightarrow \mathbb{R} \tag{4.4}$$
$$\left(R,S\right) \mapsto Q\left(R,S\right) = \sum_{i=1}^{M} \alpha_i \times q\left(\chi_i\left(\tilde{R}\right), \chi_i\left(\tilde{S}\right)\right), \tag{4.5}$$

where $(\alpha_i)_{i\in\{1,\cdots,M\}}$ is a M-uplet of strictly positive real numbers (some given weights), and the function $q : \{0,1\} \times \{0,1\} \mapsto [0,1]$ is defined as follows in order to clearly distinguish the cases where S is lacking some concepts of R.
– $q(0,0) = 0$,
– $q(0,1) = \epsilon_1 \in]0,1[$,
– $q(1,0) = 1 - \epsilon_2$, *with* $\epsilon_2 \in [0, 1 - \epsilon_1[$,
– $q(1,1) = 0$.

If $S \in \mathcal{S}$ is a service and $R \in \mathcal{S}$ is a request, $Q\left(R,S\right)$ evaluates the "distance" that separates the service from the request. It can serve to test the matching of a request by a service. We choose an asymmetric function q (i.e., $q(0,1) \neq (1,0)$) to clearly separate the cases where R has semantic concepts that encompass the ones of S and where S is lacking some concepts of R. This will be useful later to determine whether R matches S based on a fixed threshold (see theorem 2).

We can verify below that the function Q verifies a quasi-metric [106] on the set $\tilde{\mathcal{S}}$ of extensive representations of services (see definition 4). The properties of the quasi-metric (see theorem 1) will be useful in the following to perform efficient search of IoT services.

Theorem 1: Quasi-metric function.
We consider the function Q. $\forall \left(R,S,T\right) \in \mathcal{S} \times \mathcal{S} \times \mathcal{S}$, it verifies the properties of a quasi-metric:

a) The positivity: $Q\left(R,S\right) = Q\left(\tilde{R},\tilde{S}\right) \geq 0$,

b) The separability: $Q\left(R,S\right) = Q\left(\tilde{R},\tilde{S}\right) = 0 \Leftrightarrow \tilde{R} = \tilde{S}$,

c) The triangular inequality:
$Q\left(R,S\right) = Q\left(\tilde{R},\tilde{S}\right) \leq Q\left(\tilde{R},\tilde{T}\right) + Q\left(\tilde{T},\tilde{S}\right) = Q\left(R,T\right) + Q\left(T,S\right)$.

Proof. Because the weights α_i are strictly positive, we can derive by summation that Q is a quasi-metric from the fact that q is a quasi-metric on $\{0,1\}$. □

Q is an asymmetric function where $\forall \left(R,S\right) \in \mathcal{S} \times \mathcal{S}$, when $R \neq S$, in general $Q\left(R,S\right) \neq Q\left(S,R\right)$. Q can be a distance on the set $\tilde{\mathcal{S}}$ of extensive representations of services if the function q in definition 6 is symmetrized (i.e., $\Leftrightarrow q(0,1) = q(1,0) = d$, with d $\in]0,1[$). We note this distance Q'.

Using these metrics, we need to estimate if there is a matching between a service R and a service S. For that, we define the matching function M_f:

Definition 7: Matching function.
We consider a threshold value $T \in \mathbb{R}$ and $D \in \{Q, Q'\}$. We consider that a service S matches a request R if $D(R, S)$ is less or equal to a threshold T.

$$M_f \ : \ \mathcal{S} \times \mathcal{S} \rightarrow \{0, 1\} \tag{4.6}$$
$$(R, S) \mapsto M_f\,(R, S) = 1 \Leftrightarrow D\,(R, S) \leq T \tag{4.7}$$

We should note that at this point, there is no equivalence between definition 3 and 7. We will prove later in this chapter that they are equivalent for a given T.

4.3.2 Routing table

Within a semantic node, we create groups of similar services based on a clustering mechanism and we aggregate these clusters content by selecting a representative and defining a radius based on the metrics defined in the previous section (see 4.3.1). This aggregated information is stored in a routing table. Thus, the routing table contains an overview of the semantic node content. We detail in the following the different mechanisms to create the routing table.

4.3.2.1 Service clustering

The purpose of service clustering is to create groups of similar services based on some metrics. As we mentioned in section 4.2, the hierarchical agglomerative clustering [103] is often used in information retrieval for grouping similar documents. Moreover, it responds better to our requirements among other approaches such as K-means clustering, since it does not require to fix the number of formed clusters in advance. Instead, it accepts a similarity threshold between elements of a cluster to stop the clustering when it is reached. The used clustering method is depicted in algorithm 1. We characterize the dissimilarity between services at each tree level ω by a *clustering threshold* Tc_ω. Two services are clustered together if they match based on the distance Q' (see section 4.3.1.2) and the clustering threshold Tc_ω (i.e., $Q'\,(S_1, S_2) \leq Tc_\omega \Leftrightarrow M_f\,(S_1, S_2) = 1$), following the matching function defined in definition 7. Since services in higher levels come from different semantic gateways representing different smart spaces, we consider that their service are more likely to be more dissimilar than the ones in lower levels, we consider: $Tc_1 \leq Tc_2 \leq Tc_3 ... \leq Tc_n$. In this chapter and during experiments, these values are fixed (e.g., by the system administrator) based on heuristic measures (see section 4.5.2).

4.3.2.2 Cluster aggregation

After service clustering, we perform information aggregation on each cluster. Information aggregation in a cluster K enables to characterize its content based on semantic description of its services. Below, we define a representative and a radius based on the quasi metric Q of a cluster K in a semantic node N at a level ω of the graph G :

Algorithm 1 Hierarchical agglomerative clustering in [103] applied to our context.

```
Input: .
  $\Phi = \{S_1, S_2, S_3, ..., S_n\}$ : initial $n$ services to cluster in node N of level $\omega$.
  $Tc_\omega$: clustering threshold for level $\omega$.
  $SimMatrix$ : contains the matching distance of each pair of
  services $Q'(S_z, S_y)_{(z,y)\in[1..n]}$
Output: .
  $\mathcal{K} = \{K_1, K_2, ..., K_p\}$: dendrogram of the partition of services in $\Phi$.

  $\mathcal{K} \leftarrow \{\{S_1\}, \{S_2\}, \{S_3\}, ..., \{S_n\}\}$   {Considering clusters with single element (singleton) at the beginning.}
  $Tc_{min} \leftarrow 0$
  while $Tc_{min} < Tc_\omega$ do
    $Tc_{min} \leftarrow Tc_\omega$
    $Cluster_1 \leftarrow NULL$
    $Cluster_2 \leftarrow NULL$
    $d \leftarrow 0$
    for $K_i$ in $\mathcal{K}$ do
      for $K_j$ in $\mathcal{K}$ do
        $d \leftarrow \max_{S_z\in K_i} \max_{S_y\in K_j} Q'(S_z, S_y)$
        if $d \leq Tc_{min}$ then
          $Tc_{min} \leftarrow d$
          $Cluster_1 \leftarrow K_i$
          $Cluster_2 \leftarrow K_j$
        end if
      end for
    end for
    if ($Cluster_1 \neq NULL$) AND ($Cluster_2 \neq NULL$) then
      $K_{new} \leftarrow Cluster_1 \cup Cluster_2$   {Merge the two clusters into a new one.}
      $\mathcal{K} \leftarrow (\mathcal{K} \setminus \{Cluster_1, Cluster_2\}) \cup K_{new}$   {Delete the merged clusters from $\mathcal{K}$ and add the new one.}
    end if
  end while
```

Definition 8: Representative and radius of a cluster
We consider the quasi-metric Q. We consider K a cluster from node N that includes a set of similar services $K = \{S_i, i \in \{1, \cdots, M\}\}$, in level ω of the tree nodes. For $\omega > 1$, we consider the representative $W_{K,N}$ and the radius $r_{K,N}$ of the cluster K. We also consider the representative $W_{K',N'}$ and radius $r_{K',N'}$ of a cluster K' of node N' in level $\omega - 1$. We have the following relations:

- $W_{K,N} = \arg\min_{Y\in K} \max_{S\in K} [Q(S, Y) + r_{S,K',N'}]$.
- $r_{K,N} = \max_{S\in k} [Q(S, W_{K,N}) + r_{K',N'}]$.

We should notice that for the use case of $\omega = 1$, we have $r_{K',N'} = 0$ since the services contained in the cluster are not representatives of other clusters in lower level nodes. Thus, the representative $W_{K,N}$ and the radius $r_{K,N}$ of the cluster K are respectively defined as:

- $W_{K,N} = \underset{Y \in K}{\arg\min} \max_{S \in K} Q(S, Y)$.
- $r_{K,N} = \max_{S \in K} Q(S, W_{K,N})$.

For a cluster K of a node N of level ω the aggregated information is then represented by the couple $(W_{K,N}, r_{K,N})$ corresponding to the center of the cluster and its radius.

4.3.2.3 Routing table construction process

Routing tables are constructed in each semantic node of the graph. The process is based on a recursive clustering and information aggregation of semantic gateways content over the tree hierarchy of nodes. Starting from the lowest-level of the tree, semantic gateways perform clustering to create groups of similar services, then aggregate their content by selecting a representative and radius for each cluster based on a quasi-metric. Local clusters representatives of each semantic gateway are then sent to its parent gateway where the process is performed recursively (see figure 4.1). Following, is an overview of the different steps performed by a semantic gateway at level ω of the tree:

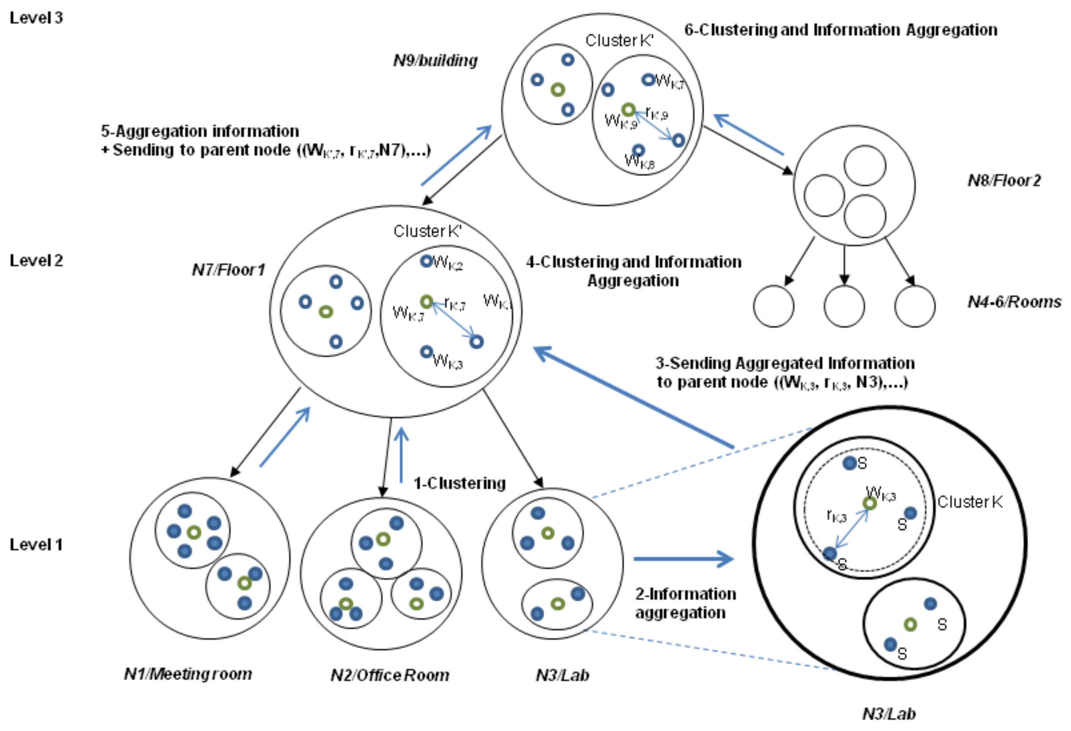
a) Receiving aggregated information from all children gateways (level $(\omega-1)$, none if $\omega = 1$).
b) Clustering services.
c) Information aggregation.
d) Routing table construction.
e) Sending aggregated information to parents gateway (level $(\omega + 1)$).

A routing table related to a semantic gateway represents each cluster by its representative, its radius, the list of its elements and their originating node. The purpose is to allow upper level nodes of the tree hierarchy to have a "summarized" view of the content of their children nodes, and to facilitate request forwarding. We should note that the elements of the clusters for the lowest level semantic gateways ($\omega = 1$) are the semantic descriptions of the IoT services to be eventually returned to user. For the other gateways ($\omega > 1$) the elements of their clusters are representatives and radius of clusters from child semantic gateways with an indication of their origin nodes.

In the following we provide a formal representation of a routing table Rt_N of a node N at a level ω of a the graph hierarchy. $\{K_i, i \in \{1..p\}\}$ is the set of clusters of node N. $(W_{K_i,N})_{i\in\{1..p\}}$ and $(r_{K_i,N})_{i\in\{1..p\}}$ are respectively the representative and radius of clusters $(K_i)_{i\in[1..p]}$ of node N at level ω and $\left(W_{K'_j,N'_k}\right)_{j\in\left\{1..p'_k\right\}}$ and $\left(r_{K'_j,N'_k}\right)_{j\in\left\{1..p'_k\right\}}$ respectively the representative and radius for each cluster K'_j of nodes N'_k from lower level $\omega - 1$ that are included in cluster K_i. We define the routing table Rt_N of node N at a level ω as:

$$Rt_N = \bigcup_{i\in\{1..p\}} \left\{ \left(W_{K_i,N}, r_{K_i,N}, \left(W_{K'_j,N'_k}, r_{K'_j,N'_k} \right)_{j,k | W_{K'_j,N'_k} \in K_i} \right)_i \right\} \tag{4.8}$$

The routing table construction process should typically be a dynamic process that can be triggered by new service arrivals and departures due to objects roaming across smart spaces. The dynamic process will be presented in the next chapter (see chapter 5).



Level 3
Level 2
Level 1
N9/building
Cluster K'
6-Clustering and Information Aggregation
$W_{K,7}$
$r_{K,9}$
$W_{K,9}$
$W_{K,8}$
5-Aggregation information
+ Sending to parent node (($W_{K,7}$, $r_{K,7}$,N7),...)
N8/Floor2
N7/Floor1
Cluster K'
$W_{K,2}$
$W_{K,7}$
$r_{K,7}$
$W_{K,}$
$W_{K,3}$
4-Clustering and Information Aggregation
N4-6/Rooms
3-Sending Aggregated Information to parent node (($W_{K,3}$, $r_{K,3}$, N3),...)
1-Clustering
S
$W_{K,3}$
$r_{K,3}$
S
S
Cluster K
S
S
2-Information aggregation
N1/Meeting room
N2/Office Room
N3/Lab
N3/Lab

Figure 4.1: Recursive clustering and information aggregation of semantic gateways content over the tree hierarchy of nodes

4.3.3 Thresholds for service search

In the previous section, we presented routing tables that aggregate the content of a semantic gateway. In this section, we define thresholds that are considered when matching a request with services to decide whether a given service matches a request during service search. We consider two types of thresholds: the decision threshold and the matching threshold. Decision thresholds are used when the request is matched with representatives of clusters in the routing table using the matching function in definition 7. By comparing the result of the matching to the value of the decision threshold, it is possible to state whether it is pertinent to perform further matching with its elements or not. Thus, decision thresholds limit the number of visited clusters per semantic gateway. Matching thresholds are used at lowest level nodes ($\omega = 1$) and select the final services (i.e., services that will be returned to user) that match the request based on definition 3. Thresholds can have an impact on the number of retrieved services when a request is issued. In our approach, as we stated in the requirements in section 4.1, these thresholds are set in order to perform an exhaustive search within the scope of search and to find all services that match a given request.

4.3.3.1 Matching threshold

We note the matching threshold θ to select the final service descriptions that match a request (i.e., services that will be returned to user). For that, we can set different system parameters (see theorem 2) to define a matching threshold. This is used to select service descriptions matching a request following the matching definition 3.

Theorem 2: Matching of a request by a service
We consider the different parameters in definition 6 and the matching function M_f of definition 7 in section 4.3.1.2, where $D = Q$ and $T = \theta$. We set:

$$1)\ \alpha_T = \sum_{i=1}^{M} \alpha_i, \tag{4.9}$$

$$2)\ \alpha_m = \min_{1 \leq i \leq M} \{\alpha_i\} \quad (\leq \alpha_T), \tag{4.10}$$

$$3)\ \epsilon_1 \in]0, \alpha_m/\alpha_T[\quad (\subset]0,1[), \tag{4.11}$$

$$4)\ \epsilon_2 \in [0, 1 - (\alpha_T/\alpha_m) \times \epsilon_1[\quad (\subset [0, 1-\epsilon_1[), \tag{4.12}$$

$$5)\ \theta \in [\alpha_T \times \epsilon_1, \alpha_m \times (1-\epsilon_2)[\quad (\subset [0, \alpha_m[). \tag{4.13}$$

A service $S \in \mathcal{S}$ matches a request $R \Leftrightarrow Q(R,S) \leq \theta \Leftrightarrow M_f(R,S) = 1$.

Proof. The bounds of the intervals for ϵ_1 and ϵ_2 have been chosen in such a way that the intervals are not empty and that they are included in the subsets as indicated above. According to the matching of the request by the service:

- If S matches R, we have $\tilde{R} \subset \tilde{S}$, thus for all $i \in \{1, \cdots, M\}$, $q\left(\chi_i\left(\tilde{R}\right), \chi_i\left(\tilde{S}\right)\right) \leq \epsilon_1$, $Q(R,S) \leq \alpha_T \times \epsilon_1 \leq \theta$, and $M_f(R,S) = 1$.
- If S does not match R, there is at least a concept C_i of the request for which there is no sub-concept in the service. Thus $q\left(\chi_i\left(\tilde{R}\right), \chi_i\left(\tilde{S}\right)\right) \geq 1 - \epsilon_2$. This implies that $Q(R,S) \geq \alpha_m \times (1-\epsilon_2) > \theta$ and $M_f(R,S) = 0$.

$\square$

Conditions 1 to 5 in theorem 2 prove that definitions 3 and 7 in section 4.3.1 are equivalent for a given value of threshold T.

4.3.3.2 Decision threshold

For a cluster K in a semantic gateway represented by a node N at level ω, we define a decision threshold $Td_{K,N}$. Each decision threshold enables to test whether a cluster is *likely* to contain a service that matches the request or not, and thus to decide whether it is pertinent to perform further matching with the elements of the cluster.

Theorem 3: Decision threshold
We consider K a cluster in node N of level ω with $W_{K,N}$ its representative and $r_{K,N}$ its radius. We define the decision threshold as following:

$$Td_{K,N} = r_{K,N} + \theta. \tag{4.14}$$

A cluster K contains a service $S \in \mathcal{S}$ that matches a request $R \in \mathcal{S}$ implies that $Q(R, W_{K,N}) \leq Td_{K,N}$, i.e, $M_f(R, W_{K,N}) = 1$.

Proof. Considering the triangular inequality property of the matching function Q in theorem 1, the matching condition in theorem 2 and the definition 8, we have: $Q(R, W_{K,N}) \leq Q(R,S) + Q(S, W_{K',N'}) + Q(W_{K',N'}, W_{K,N}) \leq$
$\theta + \max_{i \in \{k',N'\}} \{r_{k',N'}\}_i + \max_{W \in k'} Q(W_{K',N'}, W_{K,N}) \leq$
$\theta + \max_{W \in k'} [Q(W_{K',N'}, W_{K,N}) + r_{K',N'}]$. $\square$

We will illustrate, below, the usage of these thresholds during request matching and forwarding through the hierarchy of nodes to find suitable services that respond to a user request.

4.3.4 Request forwarding

In this part, we describe in details the process of request matching and forwarding through the hierarchy of nodes. Request matching and forwarding among semantic gateways is performed based on the routing tables and thresholds defined above (see sections 4.3.2 and 4.3.3).

The search system can either handle location-based search if a location constraint is expressed in the user request, or a location-free search if no location constraint is expressed in the request. Figure 4.2 illustrates some examples of the search scope.

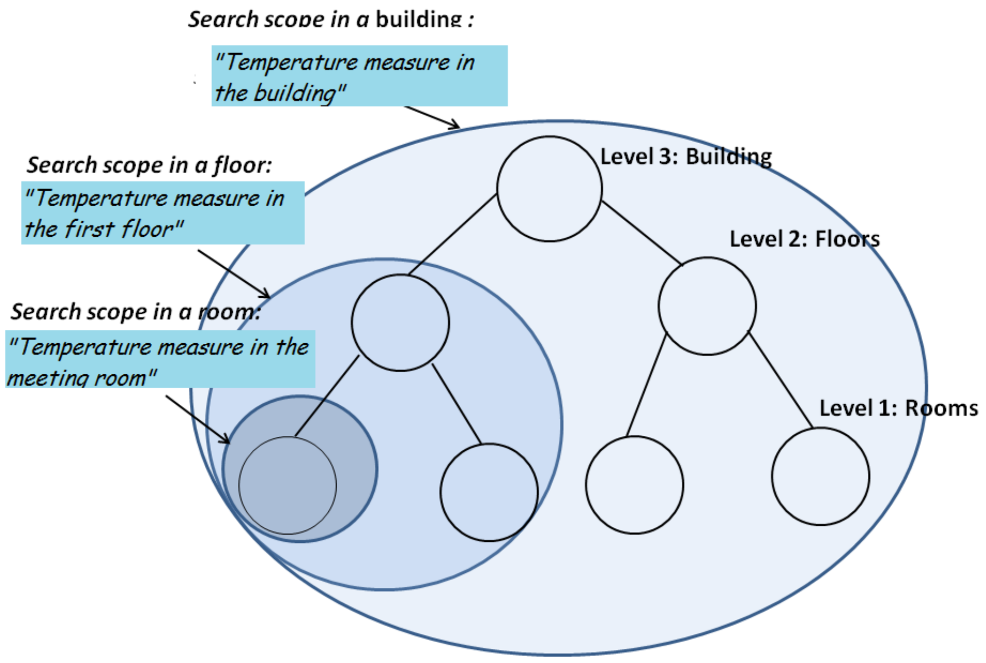

Search scope in a building :
"Temperature measure in the building"
Search scope in a floor:
"Temperature measure in the first floor"
Search scope in a room:
"Temperature measure in the meeting room"
Level 3: Building
Level 2: Floors
Level 1: Rooms

Figure 4.2: Requests involving different search scopes within the node hierarchy based on their location constraints

If the request contains a location constraint, it is then forwarded to the suitable node (i.e., semantic gateway) based on that location constraint. For example, for a request expressing directly a search at a lowest level semantic gateway (e.g. “temperature measure in the Meeting room”, with “Meeting room” being mapped to a specific node of level $\omega = 1$), then the used IoT application forwards the query to the corresponding semantic gateway (see section 3.3.4) to search for all services that “ measure temperature” in the meeting room. If the request expresses constraints on a broader location (e.g. “temperature measure in this building”, with “this building” being mapped to a node of level $\omega = 3$ for example), the query is directed to the corresponding semantic gateway. If the request does not express any any location-wise constraints (e.g. “temperature measure”), the closest lowest level semantic gateway to the user in terms of geographical proximity will receive the request and process search within that gateway. If no service corresponding to the user query is found, the IoT application may notify the user about the result and propose him to enlarge his scope of search. For example, the user be prompted for a location. Alternately, the IoT application could request the search facility to increase the scope of search, resulting e.g. in the request being sent to the parent node of the last requested gateway.

To forward the request from an upper-level semantic gateway to lower level ones, it is matched with the representatives of clusters in the routing table of the upper level gateway. The result of such matching, based on a *Decision threshold* proper to each cluster, allows to select the suitable clusters and perform a refined search to find destination gateways. Figure 4.3 provides a view of the search process that we describe its main steps in the following (the node level in the figure 4.3 is noted w):

- When the request reaches a semantic gateway represented by a node of level $\omega > 1$ (corresponding to level $w > 1$ in the figure 4.3), it is matched with representatives $W_{K,N}$, in the routing table, of local clusters based on their related Decision thresholds (Td in figure 4.3). If there is a match with a representative of a cluster K, the request will then be matched with its elements (which are representatives of clusters from its children nodes). During this matching process within the cluster, for a given element, we check that its origin node has not been selected yet to receive the considered request. If so, then the request is matched with the element based on its Decision threshold (Td') from its lower level node. If there is a matching with the request, then the request is forwarded to the original node at level $\omega - 1$.
- When the request reaches a semantic gateway represented by a node of level $\omega = 1$ (corresponding to level $w > 1$ in the figure 4.3), it is matched with representatives of clusters in the routing table based on the related Decision thresholds. If there is a match with a representative of a cluster K, the request is then matched with all services of that cluster based on the Matching threshold (θ in figure 4.3). All matching service descriptions are appended to the search result that will be sent back to the user.

In a semantic gateway, if there is no matching with representatives in its routing table, or if no final service or node destination is found after matching with its clusters' elements, then the request is discarded as shown in figure 4.3.

4.4 Exhaustive search: Formal validation

We presented in previous sections the routing table (see section 4.3.2) and the different thresholds (see section 4.3.3) that are used in the search mechanism (see section 4.3.4) to find services that respond to a given request. Based on these thresholds, the aggregated information and the defined metrics, we prove formally in this section that, using our search mechanism, that it is possible to find all services that respond to a given request.

For that, we consider in the following a graph G of hierarchical nodes that represent distributed semantic gateways. We consider a request R sent to a node N_α of level $\alpha > 1$ (with $\alpha \in \mathbb{N}$) to find services that respond to this request. For the seek of simplicity, we index here the nodes by the level to which they belong. This is possible here as we consider only the chain of nodes from the one of the lowest level to the node of higher level to which the request is sent. $N_{\alpha+1}$ is the parent node of N_α. The service S matching the request R is in the cluster K_1 with representative W_{K_1,N_1} in the lowest level node N_1, and recursively the representative W_{K_α,N_α} of the cluster K_α in the node N_α is in the cluster $K_{\alpha+1}$ with representative $W_{K_{\alpha+1},N_{\alpha+1}}$ in the node $N_{\alpha+1}$. Td_α is the decision threshold of the cluster K_α in the node N_α of the tree hierarchy. In the following, we demonstrate that:

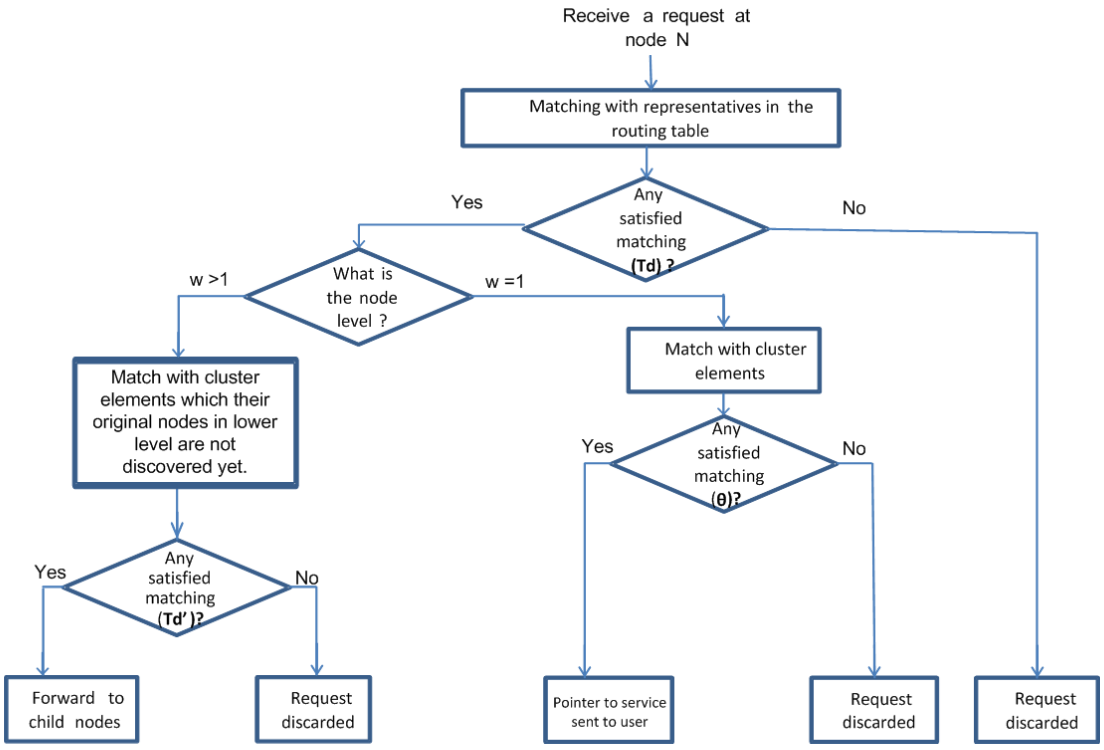

Receive a request at node N
Matching with representatives in the routing table
Any satisfied matching (Td) ?
Yes
No
What is the node level ?
w >1
w =1
Match with cluster elements which their original nodes in lower level are not discovered yet.
Match with cluster elements
Any satisfied matching (θ)?
Yes
No
Any satisfied matching (Td')?
Yes
No
Forward to child nodes
Request discarded
Pointer to service sent to user
Request discarded
Request discarded

Figure 4.3: Request forwarding process among a hierarchy of nodes.

$$\forall \alpha \geq 1, \exists W_{K_\alpha,N_\alpha} \text{ where } Q(R, W_{K_\alpha,N_\alpha}) \leq Td_\alpha. \tag{4.15}$$

By recursion:

1) We verify that is true for $\alpha = 1$:

$$Q(R, W_{K_1,N_1}) \leq Td_1. \quad \text{(Based on theorem 3)}$$

2) We suppose that the statement is true for α and we verify that is true for $\alpha + 1$:

$$\begin{aligned}
Q\left(R, W_{K_{\alpha+1},N_{\alpha+1}}\right) &\leq Q\left(R, W_{K_\alpha,N_\alpha}\right) + Q\left(W_{K_\alpha,N_\alpha}, W_{K_{\alpha+1},N_{\alpha+1}}\right) \quad \text{(Triang. inequa.)} \\
&\leq Td_\alpha + Q\left(W_{K_\alpha,N_\alpha}, W_{K_{\alpha+1},N_{\alpha+1}}\right) \quad \text{(The statement is true for } \alpha\text{)} \\
&\leq r_{K_\alpha,N_\alpha} + \theta + Q\left(W_{K_\alpha,N_\alpha}, W_{K_{\alpha+1},N_{\alpha+1}}\right) \quad \text{(Definition of } Td_\alpha\text{)} \\
&\leq \theta + \max_{W_{K,N_\alpha} \in K_{\alpha+1}} \left[Q\left(W_{K,N_\alpha}, W_{K_{\alpha+1},N_{\alpha+1}}\right) + r_{K,N_\alpha}\right] \\
&\leq \theta + r_{K_{\alpha+1},N_{\alpha+1}} \quad \text{(Definition of } r_{K_{\alpha+1},N_{\alpha+1}}\text{)} \\
&\leq Td_{\alpha+1}. \quad \text{(Definition of } Td_{\alpha+1}\text{)}
\end{aligned}$$

Thus the request R will match with elements of the cluster K_α and will be forwarded to the suitable nodes along the hierarchy. We demonstrate here that in a given node, we are able to find all clusters that are likely to contain matching services. Thus, the search mechanism is able to find all possible services that respond to a given request. This means that the search mechanism is able to perform an exhaustive search within a given scope.

4.5 Scalable search: Numerical analysis

We consider in this section different experiments to evaluate the performance of the semantic-based search mechanism in terms of average matching cost and search path length per request. We developed a prototype of distributed semantic gateways representing a building as an illustrative use case. We also developed the associated discovery mechanism (i.e., service registry, service clustering and aggregation and service seraph mechanisms). Details about the prototype implementation are in appendix A. In the following, we first present the used metrics and we fix the different experiment parameters. Then, we consider three different experiments where we respectively study the impact of the number of services, the service categories distribution and the type of graph on the search performance.

4.5.1 Metrics

We use the following metrics to measure the performance of our system:

- Matching cost: The total number of request-service description matching operations during the search process.
- Real search path length: The total number of visited nodes by a request from source until final destinations (i.e., nodes that may host the suitable services).
- Perfect search path length: The shortest path length that the query should take in order to find all suitable services, if the system had the full knowledge about the location of each service.

4.5.2 Experiment parameters and settings

We present in the following some experiments parameters and explain their associated value. We fixed the clustering thresholds (Tc_1, Tc_2 and Tc_3) defined in 4.3.2.1 to 0.5 based on heuristic measures. We also fixed the weights in the matching function in definition 6 to 0.2. Then, $\alpha_1 = \alpha_2 = ... = \alpha_M = \alpha_{min} = 0.2$, where M represents the total number of concepts in the set of ontologies in the used data set and is equal in our case to 3500. Finally, we set the matching parameters as follows: $\epsilon_1 = \epsilon_2 = \alpha_{min}/4\alpha_T$ and $\theta = \alpha_{min}/2$ in order to fulfill the conditions of theorem 2.

We consider hierarchies of nodes in each experiment to model buildings (see section 3.3.2) and we use the OWL-S service retrieval Test Collection (OWL-S TC v3.0 [1]) as data set as we explain in section A.2.

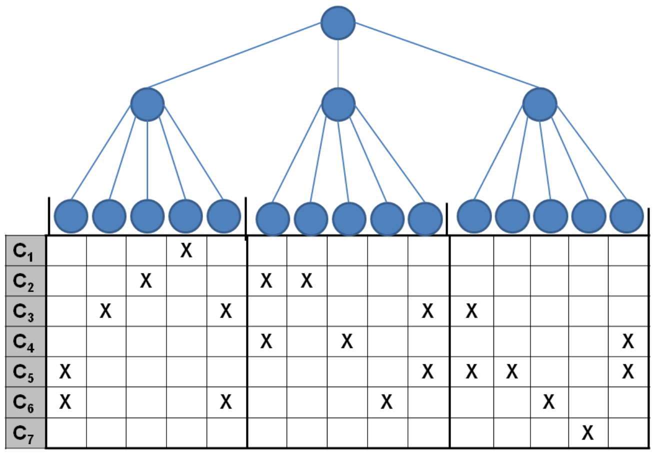
In all the following experiments, we fill lowest level nodes with services from 7 different service categories from C_1 to C_7 (see table A.1). We consider different service categories distribution over lowest level nodes in each experiment that can reflect real world cases. For instance, within a building, the type of objects depends on the purpose of the room or the floor. In the case of a working building, in each floor, we can have different rooms based on their functionalities (e.g., meeting room, office room, lab, etc.). Moreover, in the case of a mall, each floor can be specialized in a certain type of shops (e.g., clothes, home furniture, restaurants, etc.). Rooms within different floors can also have shared service categories among floors (e.g., food services). Moreover, some objects are common utilities and their services are likely to be found everywhere (e.g., lighting). Through our experiments, we tried to build images of these realities.

1. http://projects.semwebcentral.org/projects/owls-tc/

During service search, we span the whole building to find all the services that respond to a request. Thus, we are considering the worst case of search since it is the most expensive in terms of search cost. The request is then received at the highest node of the tree and goes down until reaching the lowest level nodes that may contain matching services. We calculate the average matching cost and the real and perfect average search path length over a set of 100 considered queries. The queries are chosen in a way so as to have an equal probability to belong to an existing service category. we added queries that do not have any corresponding services in order to simulate a real use case where a user may express a service description that does not exist in the building. These queries represent 25% of the total used queries.

4.5.3 Impact of the number of services on search performance

In this experiment, we evaluate the impact of the number of services per node on the search performance in terms of matching cost and search path length while considering a given graph of nodes and a given distribution of service categories over lowest level nodes. We deployed a prototype of a system simulating a smart building composed of 3 floors and 5 rooms in each floor (i.e., n=3, d_1=3 and d_2= 5 as we defined the graph parameters in chapter 3). Based on a defined service categories distribution, we fill each lowest level node with 500 randomly selected services. The considered graph of nodes and the service categories distribution over lowest level nodes are showed in figure 4.4.



C_1				X											
C_2			X			X	X								
C_3		X			X					X	X				
C_4						X		X							X
C_5	X									X	X	X			X
C_6	X				X				X				X		
C_7														X	

Figure 4.4: Considered distribution of services over 15 lowest level nodes

In a first step, the system is initialized. Thus, clustering and information aggregation (i.e., calculation of the representative and the radius of each cluster, and construction of the routing tables) are performed at each level of the tree, starting from the lowest level nodes until reaching the highest node. The system is then ready to receive service requests and perform service search. We compared the prototype performance to a nonoptimized centralized approach, simulated by a single node where all contained services are matched with a received request. We vary the number of services per lowest level node from 50 to 3000, thus the total number of services in the system has increased from 750 to 45000 services. Results about the average matching cost over 100 requests are shown in figure 4.5.

We can see that in the centralized approach, as expected, the matching cost evolves linearly with the number of services in the system. However, in our approach, the matching cost evolves sub-linearly. Matching cost values are low compared to the centralized approach, e.g. for 45000 services we have a gain of about 25. The combination of information aggregation and cluster

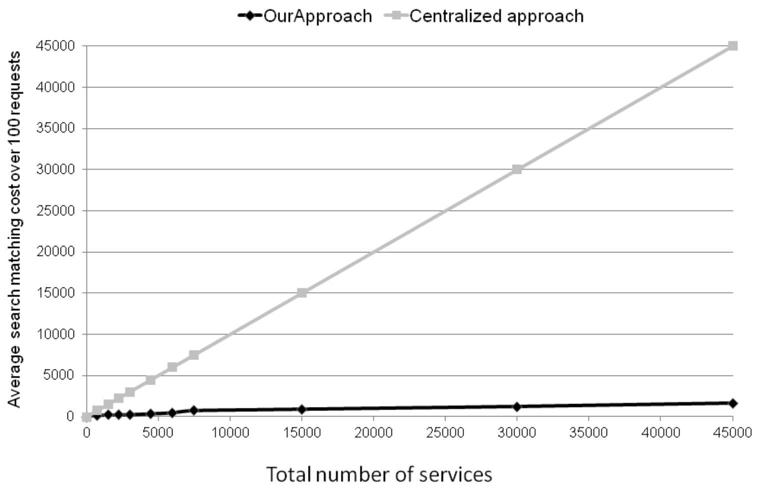


Figure 4.5: Matching cost evolution per number of services in the system.

selection based on Decision thresholds reduces the matching cost. Indeed, based on information aggregation, the request is forwarded to the suitable node that may contain suitable services, avoiding matching with services in irrelevant nodes. Within a node, Decision thresholds allow to select the relevant clusters that may contain the suitable services and reduce the number of services to match with.

In a second step, we measure the average real search path length over 100 requests which is the average number of nodes visited by requests from the highest node until reaching final destination nodes. We also measure the average perfect search path length over 100 requests, which is the minimum number of nodes that each request should visit in order to find all the suitable services. We compare the evolution of the two paths when the global number of services increases. Results of this experiment are shown in Figure 4.6.

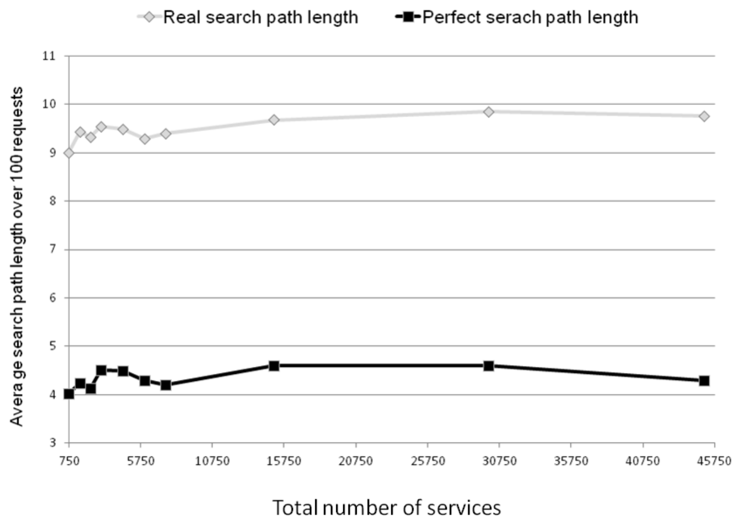
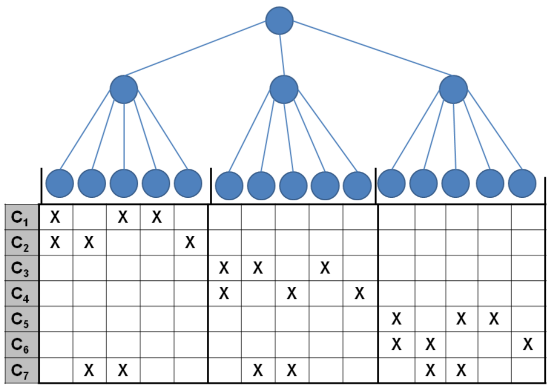


Figure 4.6: Search path length evolution per number of services.

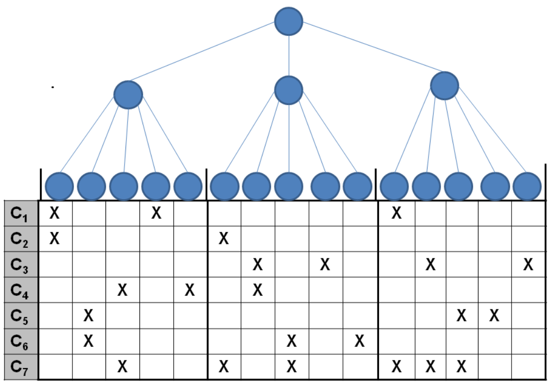
The perfect path length is around 4 nodes which corresponds to the average perfect path length with regard to the considered distribution. Both measures of average real path length and perfect path length have similar variation. However, the real path length value is about twice that of the perfect search path length: this is due to false positive matching where the request is forwarded to a node that it is wrongly thought to contain the suitable services. But, the real path length is acceptable if we compare it with the flooding approach where a request visits all nodes (path length=19).

4.5.4 Impact of service categories distribution on search performance

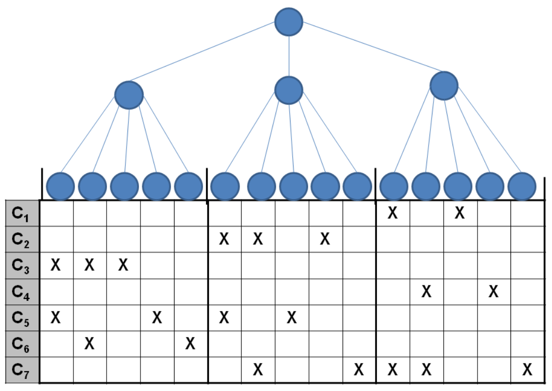
We evaluate in this experiment the impact of service category distribution over the lowest level nodes on the search performances. To do so, we consider the same graph of nodes as in the previous experience (i.e., n=3, d_2=3, d_1=5) and 7500 services (i.e., 500 services per lowest level node). However, we consider different cases of service category distributions (based on the 7 service categories presented in section A.2) over lowest level nodes (see figure 4.7).



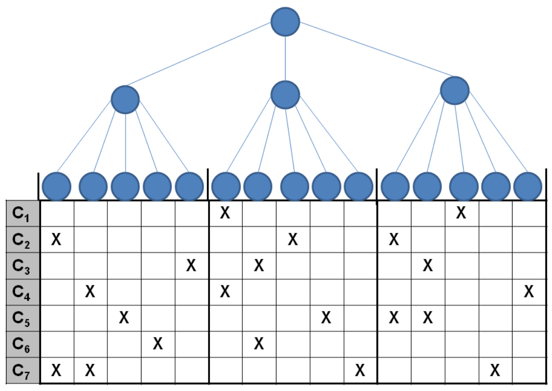
(a) distribution 1: service set 1 and category distribution type 1



(b) distribution 2: service set 1 and category distribution type 2



(c) distribution 3: service set 2 and category distribution type 1



(d) distribution 4: service set 2 and category distribution type 2

Figure 4.7: Different distributions for service categories among 15 lowest level nodes

In more details, we consider two cases with different sets of services in each. For each case we apply two different distributions of service categories. Thus, for each set of same services we consider two different types of service distribution. In the first type of service category distribution, each floor of the building has at most one common service category with the other floors. In the second type, the different floors share at least four common service categories. Thus, we chose two distinct distributions which are quite dissimilar to see the impact on the matching cost and on the search path length.

We send a set of 100 service search requests to the highest node of each graph and perform service search starting from the highest node of the graph. For each request, we measure the matching cost and the real and perfect search path lengths. Then, we calculate the average values over 100 request. Results are shown respectively in figures 4.8 and 4.9.

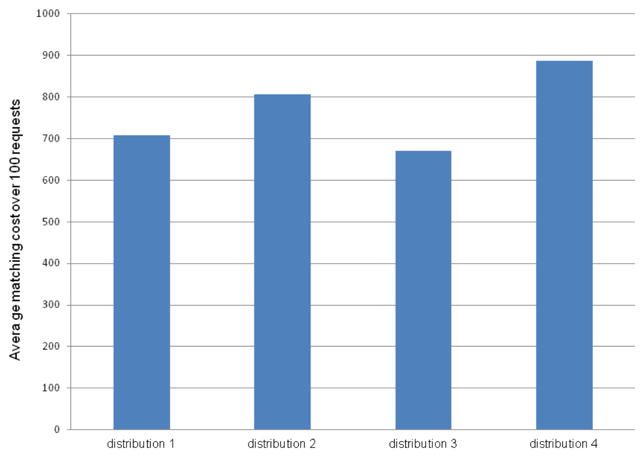


Figure 4.8: The average matching cost over 100 requests for the 4 cases of service category distributions.

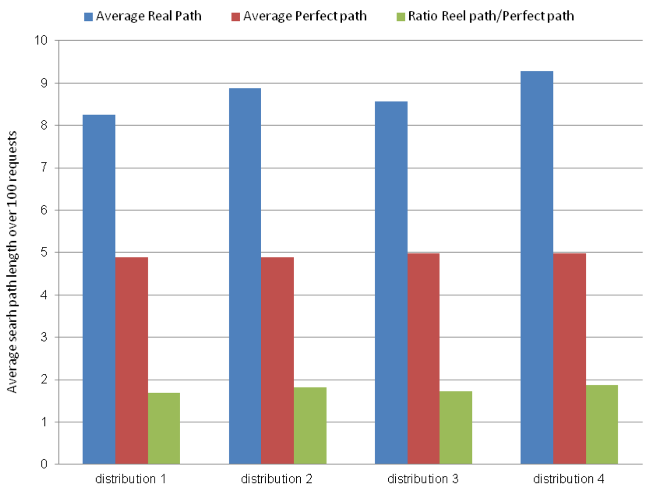
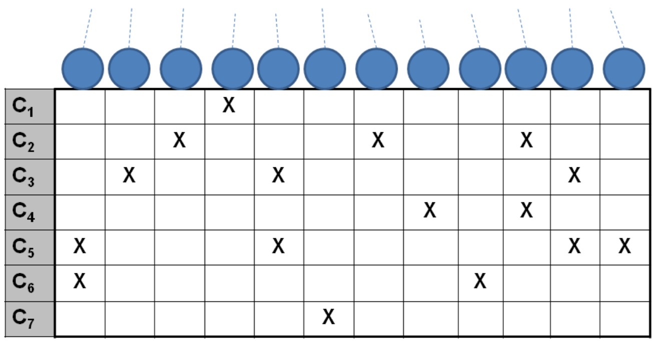


Figure 4.9: The average real search path length and perfect search path length over 100 requests for the 4 cases of service category distribution.

Results in figure 4.8 show that the matching cost for distributions 1 and 3 are lower than for distributions 2 and 4. Moreover, search path lengths (figure 4.9) in distributions 1 and 3 are lower than in distribution 2 and 4. We can see that there is a correlation between the results given by distributions 1 and 3 and those given by the distributions 2 and 4. Service categories distributions 1 and 3 have few common service categories (at most 1) at the floor level (i.e., n=2). However, service categories distribution 2 and 4 have many common service categories (at least 4) at the floor level. Having distinct service categories distribution in distribution 1 and 3 allows to better select the nodes to visit for service search than in the case of distributions 2 and 4, and reduces false positive results. Thus, few nodes are visited and few service matching operations are performed. Although more exhaustive experiments are required to better analyze the results, we can see, based on this experiment, that the gap in terms of average matching cost per request between the two types of service category distributions is quite moderate (maximum gap of 200 services per request between distribution 3 et 4 over 7500 initial services in each distribution).

4.5.5 Impact of nodes graph topology on search performances

In this experiment, we study the impact of the graph topology on the search performances in terms of matching cost and search path length. To do so, we consider 12 lowest level nodes with 500 services in each. We consider a service category distribution among these lowest level nodes (see figure 4.10).



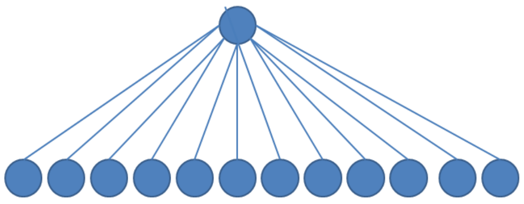
C_1				X								
C_2			X				X			X		
C_3		X			X						X	
C_4								X		X		
C_5	X				X						X	X
C_6	X								X			
C_7						X						

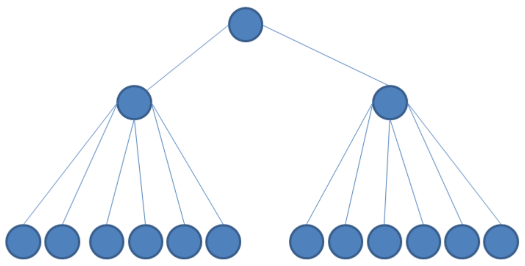
Figure 4.10: Considered service categories distribution of services over 12 lowest level nodes

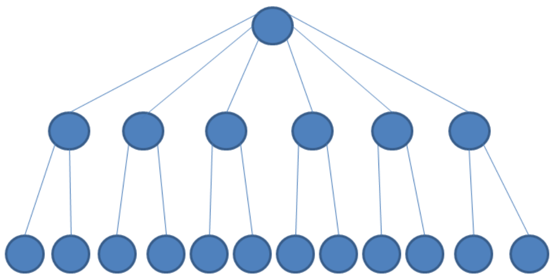
We consider in the following different configurations of graphs in terms of number of levels (i.e., n) and number of children nodes (i.e., d_i with $i \in [1, n-1]$). Figure 4.11 shows the different cases of considered graphs.

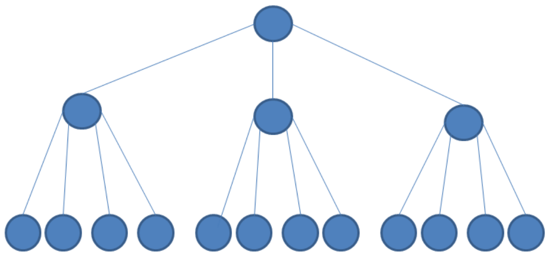
We perform service search in each graph by sending service search requests to their top-most node and compare their average matching cost and average search path length over 100 requests. As in the previous experiments, the request is sent to the highest node of the graph and spans the whole graph. Results of the service search are shown respectively in figures 4.12 and 4.13.

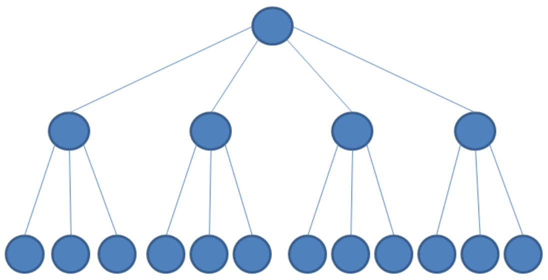
The results show that the average matching cost for a given request depends on the graph topology. The same observation is made for the average real search path length and the average perfect path length. We can see that the graph 3 has the lowest average matching cost per request compared with the other graphs, whereas, graph 1 has the worst case. Since we have the same services in lowest level nodes for all graphs, we have the same formed clusters (the average number of formed clusters is about 10 per node). The representatives of these clusters will be sent to parent nodes. The number of services in parent nodes depends on the number of its children nodes, the more children nodes a parent node has, the more service representatives it receives and the more matching a request will perform. Since graph 3 has the lowest number of services at level $n = 2$, the matching cost is the lowest. For the search path, we can see that, the more parent nodes we have at intermediate levels (e.g., $n = 2$), the longer the search path


(a) graph 1: $n = 2$, $d1 = 12$


(b) graph 2: $n = 3$, $d2 = 2$, $d1 = 6$


(c) graph 3: $n = 3$, $d2 = 6$, $d1 = 2$


(d) graph 4: $n = 3$, $d2 = 3$, $d1 = 4$


(e) graph 5: $n = 3$, $d2 = 4$, $d1 = 3$

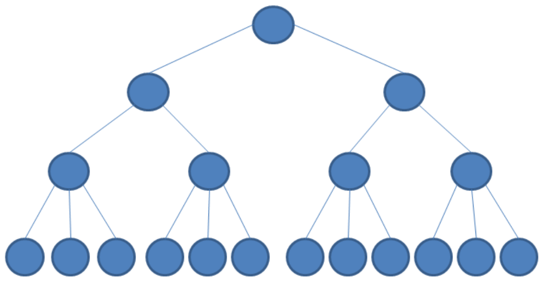

(f) graph 6: $n = 4$, $d3 = 2$, $d2 = 4$, $d1 = 3$

Figure 4.11: Different graph configurations based on 12 lowest-level nodes

is. In fact, request forwarding at higher-level nodes (e.g., from level 3 to level 2) is not very accurate because parent nodes are sharing common services categories resulting from service representatives sent by their children nodes.

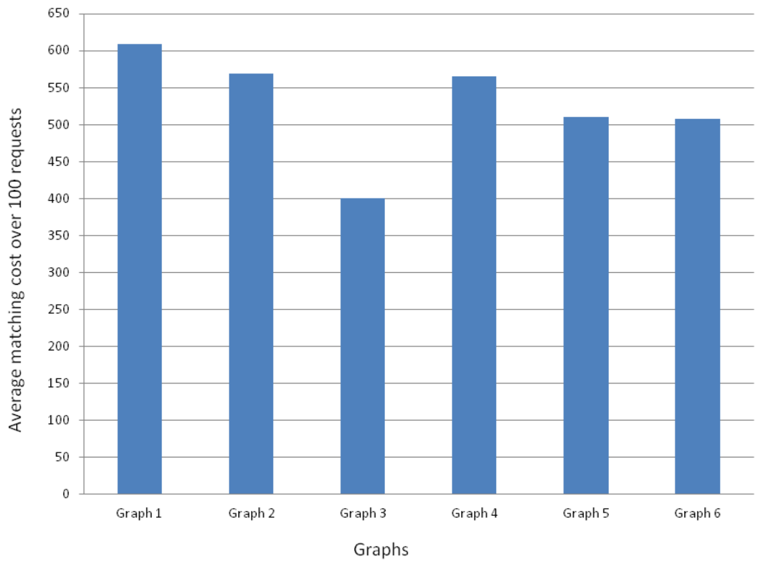


Figure 4.12: Average matching cost over 100 requests for the different graphs.

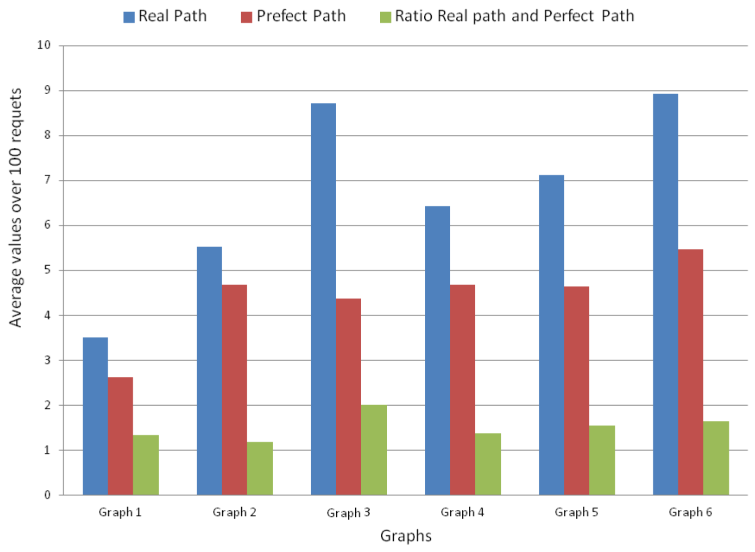


Figure 4.13: The average real search path and perfect search path length over 100 requests for the different graphs.

4.6 Concluding remarks

This chapter defines a semantic-based IoT service discovery mechanism over a network of hierarchical semantic nodes that we call semantic gateways. These semantic gateways represent smart spaces where IoT devices expose their semantic-based service descriptions. Semantic gateways can be deployed over a small scope such as a building or a larger one such as a city. Semantic service descriptions in a semantic gateway are clustered into groups of similar services and aggregated into a routing table based on a quasi-metric where a cluster representative and a radius are defined. Based on thresholds, the request is matched with content of a semantic gateway and forwarded from one node to another to find suitable services. Supporting a flexible search, we formally proved that our mechanism is also able to perform an exhaustive search (i.e., find all the suitable services that respond to a request within the scope of search). We also performed experimental evaluations to study the impact of the different parameters. These experiments show that our method is scalable and enables to greatly reduce search cost with comparison to a centralized approach. Moreover, the search path length over the hierarchy of nodes is acceptable. Besides, experiments show that the performances of the search mechanism can be impacted by the type of considered graph of nodes and by the service category distribution over the lowest level nodes. However, on the light of these results, one may consider that these impacts are moderate (i.e., there is not a large gap between experiment results when considering different parameters). We estimate that more experiments are needed to confirm this statement.

Chapter 5

Service Mobility Support in a dynamic context

In the previous chapter, we considered a static context where a resource is immobile and mapped to the same smart space over time, thus the content of their related semantic gateway is static. This can be true when considering devices such as computers, printers, faxes or home appliances which mobility is very limited. In this chapter, we consider a dynamic context where IoT services are mobile and roaming from one smart space to another. For instance, this can be true when considering sensors embedded into smart phones. Thus, the content of semantic gateways registering IoT service descriptions are evolving over time.

Service mobility induces new challenges to the discovery system defined in the previous chapter. We consider challenges in terms of data management to enable efficient and scalable search. In the following, we define in section 5.1 the problem and the requirements for efficient semantic-based IoT service discovery in a dynamic context. In section 5.3, we present our solution where we define mechanisms within semantic gateways in order to adapt to service mobility and content change in each gateway. These mechanisms enable to dynamically create clusters of similar services and optimize the clustering over time in order to minimize the search cost. Results in section 5.4 show that the overhead in terms of matching operations due to the defined mechanisms and the number of updates sent to a neighboring gateway are limited.

5.1 Problem statement and requirements

Service dynamicity brings new challenges to the discovery system that we list in the following:

Ensuring efficient search within a node and within the hierarchy of nodes (i.e., finding all services that respond to a given request within the scope of search). A new semantic service description can be added to the semantic gateway due to new device arrival within its geographical scope. In the same way, a service description can be deleted from the semantic gateway due to the departure of its corresponding device. During this service mobility, we need to guarantee within a node, the consistency of clustering where similar services are clustered together and the continuous updating of its routing table. Children nodes need also to notify parent nodes about their changes. Thus, an efficient dynamic management of node content is required.

Ensuring a scalable service management and service search within a node. Matching costs within a node are related to the number of service-request or service-service matching operations performed within a node. They can be further divided into:

- Search matching cost M_{rT}: The total number of service-request matching operations performed during service search.
- Mobility matching cost M_{mT}: The total number of service-service matching operations induced by service arrivals and departures.
- Clustering optimization matching cost M_{rcT}: The total number of service-service matching operations during re-clustering operations (e.g., cluster merging and splitting).

Updating costs are related to changes on the routing tables of a node that need to be notified to its parent node. A high frequency of updates impacts the system scalability. Thus, minimizing updating and matching costs is required.

5.2 Background

We present in the following a quick and short background about Incremental clustering approaches. These approaches have been developed in order to manage dynamically changes within clusters, where a new data item can be assigned to a cluster without affecting the existing clusters significantly [107]. These changes can be discrete [108] where new data can be inserted or deleted from clusters from time to time or changes can be continuous such as in the case of data stream [109]. Incremental clustering allow to avoid re-clustering the whole data-set and to perform instead incremental changes which is advantageous when dealing with a large amounts of data [110].

In incremental clustering, data joins a cluster if some predefined criteria are verified. Otherwise, a new cluster is created to represent it. Cluster merging and splitting are also performed under specific conditions to preserve the consistency of clusters. We give in the following some examples of incremental clustering and their criteria. In [111], the authors define an incremental clustering to cluster groups of documents while efficiently maintaining clusters of small diameter for clusters. In [112] they propose an incremental clustering that is able to reach stability (i.e., inserting data in existing clusters instead of creating new ones) quickly. COBWEB [113] which is a Hierarchical incremental clustering utilizes incremental learning and instead of following divisive or agglomerative approaches to update clusters, it dynamically builds a dendrogram by processing one data point at a time. BIRCH [114] is another hierarchical clustering suitable for very large databases allowing to produce the best cluster quality within the constraints of available resources (i.e., available memory and time constraints). In [115] and [116] they propose incremental clustering approaches to create dynamically groups of similar semantic-based resources to facilitate their discovery. However, in their approaches they mainly focus on providing efficient clustering and do not consider the generated matching and computing cost.

Many examples of incremental clustering algorithms exists in the literature, however, while a major part of these approaches focuses on creating efficient clusters with coherent groups of similar data, they do not fully respond to our requirements mentioned in section 5.1. In fact, we do not only consider efficient clustering where similar services are clustered together, we also consider minimizing updating and matching operations costs during clustering and during the service search cost, in order to ensure system scalability.

5.3 Proposed solution

We present in the following our approach to support service mobility. We consider a lowest level node to detail the mobility support mechanisms within a node. In section 5.3.1 we define an incremental construction and update of clusters and routing tables. This incremental clustering is optimized in 5.3.2 in terms of service distribution among clusters to minimize the search cost M_{rt}. We evaluate the total cost optimization within a node in 5.3.3. An approach to optimize the search cost within the whole system is proposed in 5.3.4.

5.3.1 Dynamic clusters creation and updating within a node

In this part, we define mechanisms to create and update clusters of similar services. During cluster creation and updating, we aim at making new services quickly discoverable while limiting the number of matching operations required to insert them in the clusters of the gateway. In the following, we detail the different service management actions and consider conditions to insert a new service within a node or to delete it. In the following, we consider a lowest level node and we analyze the impact of its mobility on a higher level node.

We consider N a node and $\mathcal{K} = \{K_i, i \in \{1, \cdots, n\}\}$ a set of clusters in the node N. For a cluster $K \in \mathcal{K}$, we consider $W_{K,N}$ and $r_{K,N}$ respectively its representative and radius. We consider the quasi-metric Q of definition 6.

5.3.1.1 Service arrival cases

We consider S_1 a new service that will be inserted into the node N. We consider $K_{min} = \underset{K \in \mathcal{K}}{\arg\min} \left[Q\left(S_1, W_{K,N}\right)\right]$. K_{min} is the cluster in N whose representative is the closest to S_1 in terms of quasi-metric Q and we note d_{min} this distance; $d_{min} = Q\left(S_1, W_{K_{min},N}\right)$. The insertion of S_1 in N can present different scenarios that we illustrate in the following (See Figure 5.1):

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 $r_{K,N} \geq Q\left(S_1, W_{K,N}\right)$, then S_1 is inserted into $K = \underset{K \in \mathcal{K} \setminus \{K_{min}\}}{\arg\min} \left[Q\left(S_1, W_{K,N}\right)\right]$, no change is made within the cluster.

c) If not (a) and not (b) and $d_{min} < Q\left(W_{K,N}, W_{K_{min},N}\right)$, 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 $r_{K',N} = 0$.

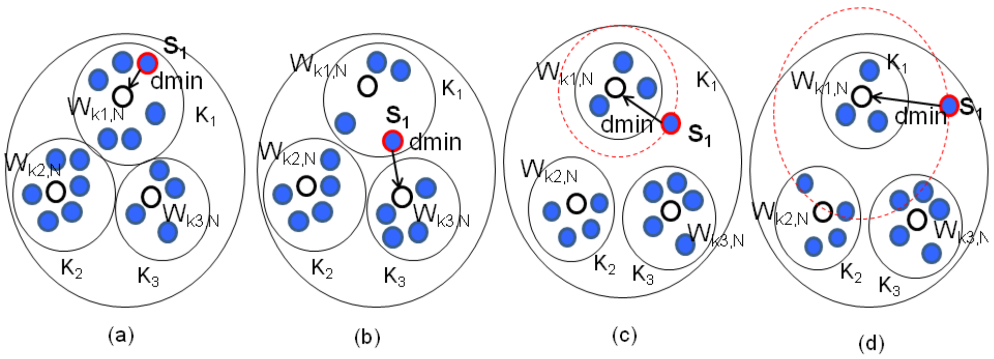


Figure 5.1: 2D representation of service arrival and departure cases

5.3.1.2 Service departure cases

In the following, we consider a case of a service departure where $S_1 \in K$ is leaving the cluster. We consider $d = Q(S_1, W_{K,N})$. The departure of S_1 can represent different scenarios that we detail in the following (See Figure 5.2):

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 matched 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.

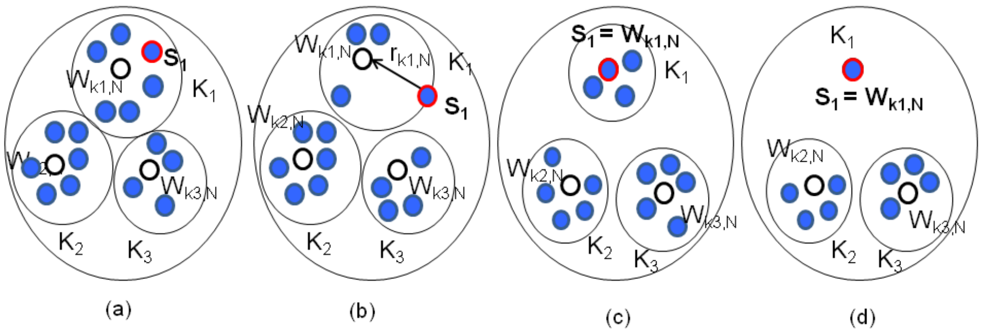


Figure 5.2: 2D representation of service departure cases

5.3.2 Clustering optimization within a node

With the mechanism described in the previous section, the number of clusters and the number of services per cluster may vary during service mobility, which can affect the search matching cost M_{rT}. Since we aim at minimizing the costs during service mobility, it is important to optimize the distribution of services within the clusters in order to minimize the search matching cost. In the following, we will first define an optimal distribution of services within clusters. Then, we define mechanisms of re-clustering (i.e., cluster merging and splitting) to conserve this optimal distribution during service mobility.

5.3.2.1 Optimal service distribution

To evaluate the clustering quality, we define a referential model of an *ideal clustering* that does not exist. However it can serve as a reference to define a lower bound to the search cost that we try 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 9).

Definition 9: Ideal Clustering.
We consider N_s services S distributed over a set of clusters $\mathcal{K} = \{K_i, i \in \{1, \cdots, n\}\}$. We consider R a request sent to $\mathcal{K}$ to search matching services S. We consider β the number of clusters visited by the request R to search a matching service S. $\mathcal{K}$ is an ideal clustering if and only if, for any request R:

- $\beta = 1$ *when there are services S that match R,*
- $\beta = 0$ *when there are no services S that match R.*

The above defined clustering is ideal, however, it is not optimal. We define, below, an approach to optimize it. Actually, during service mobility, some clusters may grow to include a sheer number of services. These impact negatively the search cost as requests will have to be compared with each of them.

We consider the hypothesis that for a uniform distribution of requests, the more populated a cluster is, the more likely it will be visited by a request. Whereas, for the same number of services, two clusters have equal probability to be visited by a request. Thus, for a node N with N_s services distributed over the group of clusters $\mathcal{K} = \{K_i, i \in \{1, \cdots, n\}\}$, we consider m_i the number of services per cluster K_i, with $\sum_{i=1}^{n} m_i = N_s$. A set of n' requests R are sent to the node N. We consider p_i the probability for a request R to visit a cluster K while knowing that the request will visit a cluster in the node N. We have $\sum_{i=1}^{n'} p_i = 1$. We consider 2 clusters K_i and K_j. If $m_i > m_j$ then $p_i > p_j$, else if $m_i = m_j$ then $p_i = p_j$.

In the following, we define an optimal distribution of services within a node (see proposition 2), where we consider an optimal number of services and an optimal average number of services per cluster to minimize the search cost.

Proposition 2: Optimal service distribution.
We consider a node N with N_s services distributed over the group of clusters $\mathcal{K} = \{K_i, i \in \{1, \cdots, n\}\}$ formed following the process described in section 5.3.1. m_i is the number of services per cluster K_i, thus $\sum_{i=1}^{n} m_i = N_s$. A set of requests are sent to the node N. $\langle M_r \rangle$ is 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 clusters per request. $\langle M_{r_{min}} \rangle$ is 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 9 of the Ideal clustering, we can write:

$$\langle M_r \rangle = n_c + \langle \beta \rangle \times \langle m_s \rangle \text{ and} \tag{5.1}$$

$$N_s = n_c \times \langle m_s \rangle . \tag{5.2}$$

$\langle M_r \rangle$ is minimal for:

$$n_{c_{op}} \approx \sqrt{\langle \beta \rangle \times N_s}, \langle m_{s_{op}} \rangle \approx \sqrt{\frac{N_s}{\langle \beta \rangle}}, \tag{5.3}$$

and we have:

$$\langle M_{r_{min}} \rangle = 2 \times \sqrt{\langle \beta \rangle \times N_s}. \tag{5.4}$$

Proof. We have: $\langle m_s \rangle = \frac{N_s}{n_c}$ and we consider the function $f : \mathbb{R}_+^* \mapsto \mathbb{R}$ where $f(n) = n + \langle \beta \rangle \times \frac{N_s}{n}$. We consider f' the derivative of the function f, then $f' = \frac{df}{dn} = 1 - \langle \beta \rangle \times \frac{N_s}{n^2}$, f is minimal when $f' = 0$ which means that $n = n_{c_{op}} = \sqrt{\langle \beta \rangle \times N_s}$. □

Based on definition 9 of the *ideal clustering*, we have $\langle \beta \rangle \in]0, 1[$. In practice, we estimate that in most cases the request will be matched with the services in the system. Thus $\langle \beta \rangle$ will be approaching 1 in the case of an *ideal clustering*. We are aware that the *real clustering* is not ideal, That $\langle \beta \rangle$ will probably be higher than 1 but this will not have a big impact on the final result as we will see in the following.

We conduct some studies to understand the impact of $\langle\beta\rangle$ on $\langle M_r\rangle$. First of all, we studied the impact of N_s on $\langle M_r\rangle$ for a given value of $\langle\beta\rangle$. The figure 5.3 shows the impact of N_S on $\langle M_r\rangle = n_c + \langle\beta\rangle \times \frac{N_s}{n_c}$ with $\langle\beta\rangle = 1$. We can see that $\langle M_r\rangle$ has the same variation for different values of N_s. Then, we studied the impact of $\langle\beta\rangle$ on $\langle M_r\rangle$ for a given value of N_s. Figure 5.4 shows the variation of $\langle M_r\rangle$ for different values of $\langle\beta\rangle$ and for $N_s = 100$. We can see that for the same number of services N_s, the variation of $\langle\beta\rangle$ values impacts the minimum value of $\langle M_r\rangle$ which is $\langle M_{r_{min}}\rangle$.

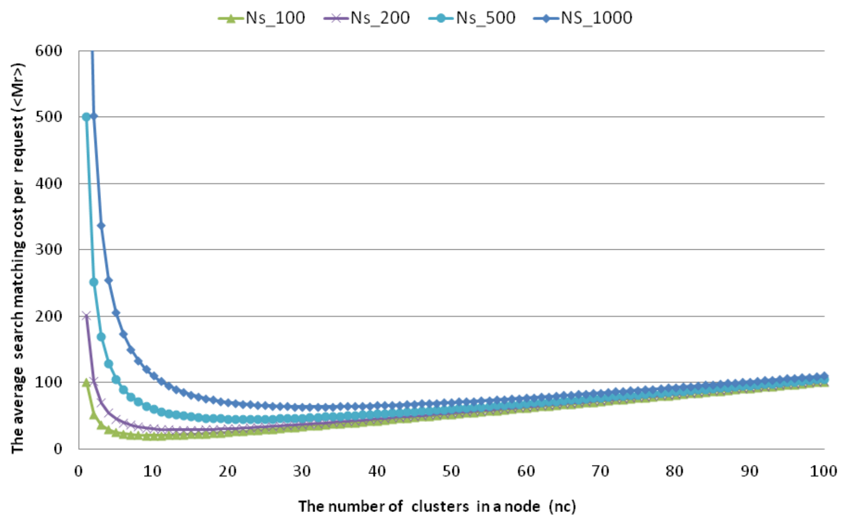


Figure 5.3: The impact of N_s on $\langle M_r\rangle$ for $\langle\beta\rangle = 1$.

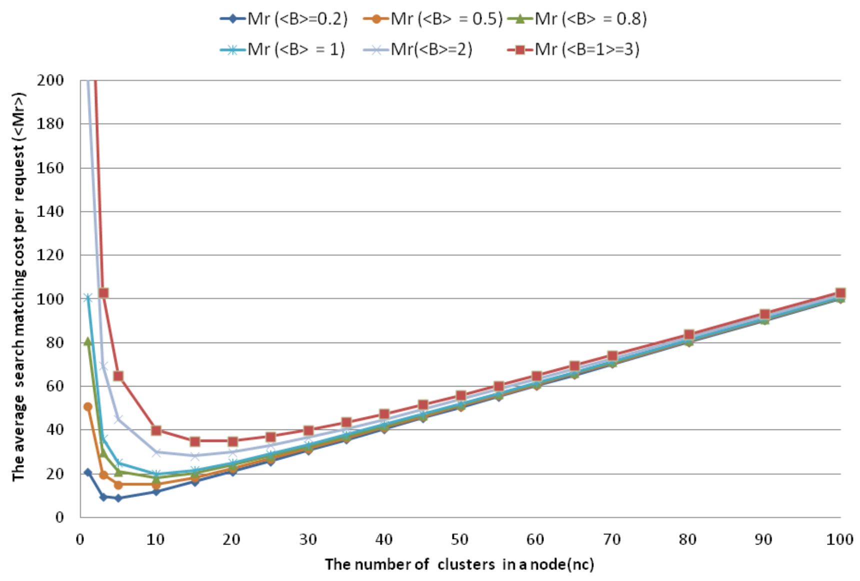


Figure 5.4: The impact of $\langle\beta\rangle$ on $\langle M_r\rangle$ for $N_s = 100$.

In the following, we study the matching overhead in $\langle M_r\rangle$ due to a bad approximation of the value of $\langle\beta\rangle$. For that, we consider $\langle\beta_1 = 1\rangle$ an estimated value for $\langle\beta\rangle$, and we consider $\langle\beta_2\rangle$ the real value for $\langle\beta\rangle$ measured during simulation. We also consider $\langle M_{r_{min2}}\rangle = \sqrt{\langle\beta_2\rangle \times N_s}$ the minimal value of $\langle M_r \rangle$ for a given value of $\langle \beta_2 \rangle$ and $\langle M_{r_{min2error}} \rangle = \sqrt{\langle \beta_1 \rangle \times N_s} \times \left(1 + \frac{\langle \beta_2 \rangle}{\langle \beta_1 \rangle}\right)$ the wrong estimated minimal value of $\langle M_r \rangle$ by considering $\langle \beta_1 = 1 \rangle$ during simulation. The matching overhead $\langle M_{r_{minOverhead}} \rangle$ in the minimal value of $\langle M_r \rangle$ is the following:

$$\langle M_{r_{minOverhead}} \rangle = \langle M_{r_{min2error}} \rangle - \langle M_{r_{min2}} \rangle = \sqrt{\langle \beta_1 \rangle \times N_s} \times \left(1 - \frac{\sqrt{\langle \beta_2 \rangle}}{\sqrt{\langle \beta_1 \rangle}}\right)^2 . \qquad (5.5)$$

In the following, we consider $N_s = 100$, we fix $\langle \beta_1 = 1 \rangle$ and consider different values of $\langle \beta_2 \rangle$. We measure the evolution of the percentage overhead of $\langle M_{r_{min}} \rangle$ when the estimated value $\langle \beta_1 \rangle$ and the real value $\langle \beta_2 \rangle$ are not equal. Figure 5.5 shows the results.

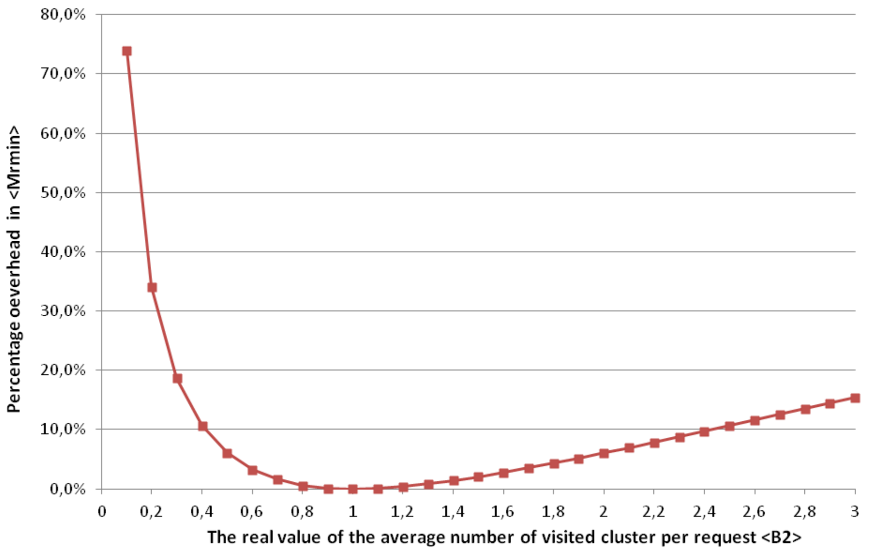


Figure 5.5: The percentage overhead of $\langle M_{r_{min}} \rangle$ for different values of $\langle \beta_2 \rangle$ when $\langle \beta_1 \rangle = 1$ and $N_s = 100$.

We can see that in the case of $\langle \beta_1 \rangle = \langle \beta_2 \rangle$, the percentage overhead in the value of $\langle M_{r_{min}} \rangle$ is null. In the case of $\langle \beta_1 \rangle \neq \langle \beta_2 \rangle$, the percentage overhead in the value of $\langle M_{r_{min}} \rangle$ is not null, however, its value is low for $\langle \beta_2 \rangle \geq 0.5$ (the most realistic cases). We can see that for $\langle \beta_2 \rangle = 3$ (i.e., the real value of $\langle \beta \rangle$ is 3 times its estimated value, which is higher than what we will find in experiments in section 5.4) the percentage overhead on $\langle M_{r_{min}} \rangle$ is only about 15% of the real value $\langle M_{r_{min2}} \rangle$. This means that an error on the estimation of $\langle \beta \rangle$ would not greatly affect the value of $\langle M_{r_{min}} \rangle$. As a result, in the following, we can consider $\langle \beta \rangle = 1$.

Moreover, we 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 in the following.

5.3.2.2 Re-clustering mechanism

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 check after each mobility event (i.e., a service arrival or service departure) number of clusters n_c and the maximum number of services per cluster m_{Max}. We define mechanisms of re-clustering where a cluster splitting is performed to limit 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 2.

Continuous merging and splitting involves additional matching operations and updates that impact the system scalability. To limit the clustering matching cost M_{rcT} (i.e., the total number of service-service matching operations during re-clustering operations) and the number of generated updates, we define limits on the number of clusters per node $n_{c_{Limit}}$ and the maximum number of services per cluster $m_{Max_{Limit}}$ by adding margins to the optimal values $n_{c_{op}}$ and $m_{s_{op}}$ (See definition 10). 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: $\langle M_r \rangle > \langle M_{r_{min}} \rangle$. However, we expect $\langle M_r \rangle$ to still be advantageous in comparison to a centralized approach where the request is matched with all the services within a repository as described in the chapter 4.

Definition 10: Re-clustering 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 of 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:

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

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. We consider $\phi \in [0, 1]$. However, to limit the number of merging operations we estimate that it is better to take values of $\phi \in [0.5, 1]$.

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

- If there is 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 the distance Q' defined in section 4.3.1. We start with $K_{new_1} = \{S_1\}$ and $K_{new_2} = \{S_2\}$. Services in $K_i \setminus \{S_1, S_2\}$ will be matched with S_1 and S_2 based on the quasi-metric Q (see definition 6) 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}. (See algorithm 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 $\mathcal{E} \neq \emptyset$. (See algorithm 3)
- In other cases nothing is done.

This section detailed the different mechanisms performed in a lowest level node of the hierarchy to optimize service distribution and minimize the average matching cost per request $\langle M_r \rangle$.

Algorithm 2 Splitting Mechanism

Input: .

$K = \{S_1, S_2, ..., S_{Max_{Limit}}\}$: a set of Max_{Limit} services in cluster K. d, d_1, d_2, d_{max}: Reals that represent distances.

Output: .

K_{new1}, K_{new2}: The 2 newly formed clusters after the split of cluster K.

```
1: $d$, $d_1$, $d_2$, $d_{max} \leftarrow 0$
2: for $S_i$ in $K$ do
3:     for $S_j$ in $K$ do
4:         $d \leftarrow Q'(S_i, S_j)$
5:         if $d > d_{max}$ then
6:             $d_{max} \leftarrow d$
7:             $K_{new1} \leftarrow \{S_i\}$
8:             $K_{new2} \leftarrow \{S_j\}$
9:         end if
10:    end for
11: end for
12: while $(|K_{new1}| \leq Max_{Limit}/2)$ do
13:     $S_{min} = \underset{S_p \in K \setminus \{S_1, S_2\}}{\arg\min} [Q(S_p, S_1)]$
14:     $K_{new1} \leftarrow S_{min}$
15:     $K \leftarrow K \setminus \{S_{min}\}$
16: end while
17: $K_{new2} \leftarrow K$
```

5.3.3 Total Costs optimization within a node

In this section, we provide an overview of the global costs (i.e., updates and matching costs) and their optimization within a node. Within a node, to minimize the number of updates, we defined conditions on service insertion and deletion. We also defined margins on re-clustering operations in order to reduce the number of cluster merging and splitting and thus that of routing tables updating. For the matching operations optimization, we consider $\langle M_r \rangle$, $\langle M_m \rangle$ and $\langle M_{rc} \rangle$ respectively the average matching cost per request, the average matching cost per service mobility event (i.e., service arrival or service departure) and the average matching cost per re-clustering operation as defined in section 5.1. We also consider n_r, n_m and n_{rc} respectively the number of requests, the total number of service arrivals and departures and the number of re-clustering operations (i.e., number of cluster merging and splitting) that happen during a considered period of time T. We should mention that in our approach, we consider a system where there is more incoming requests than mobile events, thus for a given period of time T, we have $n_r >> n_m$. Based on our re-clustering mechanisms, we have $n_m \geq n_{rc}$. Then, the total matching cost M_T within a node is:

$$\begin{aligned} M_T &= M_{rT} + M_{mT} + M_{rcT} && (5.6)\\ &= n_r \times \langle M_r \rangle + n_m \times \langle M_m \rangle + n_{rc} \times \langle M_{rc} \rangle && (5.7)\\ &\approx n_r \times \langle M_r \rangle && (5.8) \end{aligned}$$

Thus, minimizing M_T consists in minimizing $nr \times \langle M_r \rangle$ and we have $M_T \approx n_r \times \langle M_r \rangle >$

Algorithm 3 Merging Mechanism

```
Input: .
    $\mathcal{K} = \{K_i, i \in \{1, \cdots, n\}\}$: A set of $n$ clusters of size $m_i$ each in a node N.
    $n_{c_{Limit}}$: The maximum number of clusters in a node.$m_{Max_{Limit}}$: The maximum
    number of services per cluster.$d$, $d_{min}$: Reals that represent distances.$K_1$, $K_2$:

    Intermediate clusters
Output: .
    $K_{merge}$: The newly formed cluster after merging.
 1: $d \leftarrow 0$
 2: $d_{min} \leftarrow 10^6$    {Corresponding to infinity in a practical case.}
 3: for $K_i$ in $\mathcal{K}$ do
 4:   for $K_j$ in $\mathcal{K}$ do
 5:     if $m_i + m_j < m_{Max_{Limit}}$ then
 6:       $d \leftarrow [Q(W_{j,N}, (W_{i,N}) + r_{i,N}]$
 7:       if $d < d_{min}$ then
 8:         $d_{min} \leftarrow d$
 9:         $(K_1, K_2) \leftarrow (K_i, K_j)$
10:       end if
11:     end if
12:   end for
13: end for
14: $K_{merge} \leftarrow K_1 \cup K_2$
```

$n_r \times \langle M_{r_{min}} \rangle$.

Proof. We fixed $\langle \beta \rangle = 1$, then we have $\langle M_r \rangle = n_c + \langle m_s \rangle$. The new service to insert will be compared with representatives of each cluster in a node and in some cases of service departure, cluster radius is calculated, so $\langle M_m \rangle \leq n_c + \langle m_s \rangle$. $\langle M_r \rangle \geq \langle M_m \rangle$ and since $n_r >> n_m$, then, $n_r \times \langle M_r \rangle >> n_m \times \langle M_m \rangle$. Based on the merging and splitting methods described in section 5.3.2, the average merging cost is less than $2 \times {m_{Max_{Limit}}}^2$ and the average splitting cost is less than $4 \times {m_{Max_{Limit}}}^2$. So, we have $\langle M_{rc} \rangle \leq 4 \times {m_{Max_{Limit}}}^2$ and $\frac{n_m \times \langle M_m \rangle}{n_{rc} \times \langle M_{rc} \rangle} \geq \frac{n_m \times n_c}{n_{rc} \times 8 \times m_{sop}^2}$. We have $\frac{n_m \times \langle M_m \rangle}{n_{rc} \times \langle M_{rc} \rangle} \geq 1$ if $\frac{n_m}{n_{rc}} \geq \frac{8 \times \sqrt{N_s}}{1+\phi}$, which will be verified during the tests. □

5.3.4 Search cost optimization within the whole system

In the previous sections, we considered solely a lowest level node from the tree hierarchy. We proposed mechanisms to adapt the discovery approach defined in chapter 4 to a dynamic context. We also proposed mechanisms to minimize updating costs as well as search cost M_{rT} which constitutes the major matching cost within a node (see section 5.3.3).

In the following, we try to optimize the service discovery in the whole system by minimizing the global search cost in the tree hierarchy. To do so, we depict two approaches that we expose and compare in section 5.3.4.2. However, before that, we detail in section 5.3.4.1 the service mobility model for a higher level node of the tree hierarchy.

5.3.4.1 Mobility model for a higher-level node

We detailed in the previous sections 5.3.1 and 5.3.2 the mobility model for a lowest level node (i.e., the different cases of service arrivals and departures, their impact on the node organization and the clustering optimization process through splitting and merging operations). Higher level nodes implement the same cases of service management as the ones defined for lowest level nodes (see section 5.3.1). However, while the service arrivals and departures in lowest level nodes are induced by device mobility, in higher-level nodes, they are induced by changes happening at lowest level nodes. In the following, we list the different cases of changes happening at a lowest level node and their impact on the parent node:

- **When considering cases of service arrivals in a lowest level node** (see section 5.3.1.1)
  • If cases (a) and (b) happen where a service is inserted into an existing cluster without need for further cluster modification, then the parent node will not be impacted.
  • If case (c) happens where the radius of a cluster in lowest level node is enlarged to insert a new service, then the value of this radius which is associated to a cluster representative in the parent node will be updated. After this update, the representative and the radius of the cluster in the parent node hosting the representative from the child node, will also be re-calculated.
  • If case (d) happens where a new cluster is created, then the representative of the new cluster will be inserted in the parent node.
- **When considering cases of service departures in a lowest level node** (see section 5.3.1.2)
  • If case (a) happens where a service leave a cluster from lowest level node without need for further modification, then the parent node is not impacted.
  • If case (b) happens where the radius of a cluster in lowest level node is updated due to the departure of a service, then the new value of this radius which is associated to a cluster representative in the parent node will be updated. After this update, the representative and the radius of the cluster in the parent node hosting the representative from the child node, will also be re-calculated.
  • If case (c) happens where the center of a cluster becomes a Virtual Element, then the parent node is not impacted since there is no impact on the search process.
  • If case (d) happens where a cluster from lowest level node is deleted, then the representative of this cluster in the parent node will departure.
- **When considering cases of cluster merging and splitting in lowest level nodes** (see section 5.3.2.2)
  • If two clusters K_1 and K_2 are merged to create a new cluster K_{new} in lowest level node, then the cluster representatives of K_1 and K_2 in the parent node will departure and the representative of the cluster K_{new} will be inserted.
  • If a cluster K in lowest level node is splitted into two clusters K_{new_1} and K_{new_2} , then the cluster representative of K will departure from the parent node and the clusters' representatives of K_{new_1} and K_{new_2} will be inserted.

5.3.4.2 System optimization approaches

To optimize (i.e., minimize) the cost of service search in the whole system we illustrate two approaches that we compare in the following:

Individual node-based optimization approach: In this approach, the system is considered as a composition of sub-systems which are the different nodes in the tree hierarchy. During service mobility, the optimization process starts from lowest level nodes to higher level nodes.

By minimizing the search cost individually in each node, we reduce the global search cost in the whole system.

Global system optimization approach: In this approach, we consider the system as a global entity and we minimize its global search cost. In the following, we first depict the expression of the global search cost within the system. Then, we propose an approach to minimize it.

We consider a tree graph G of h levels. At each level K where $k \in [1, h]$, we consider N_k nodes with N_{s_k} services, n_{c_k} clusters and $\langle m_{s_k} \rangle$ the average number of services per cluster. To simplify the problem, we consider an average value of the number of services that we note N_{s_k}, the number of clusters n_{c_k} and the average number of services per cluster $\langle m_{s_k} \rangle$ in each node of level k. We should note that we kept the same notations for average values. We note d_k the number of children nodes for each parent node at level $k+1$ and b_k the average number of visited nodes per level per request. We should mention that we want to avoid locally overloading the nodes within the node hierarchy. Thus, it is important to keep an equivalent number of services at each level of the hierarchy. To do so, we consider the following condition:

$$\forall k \in [1, h], d_k \leq \frac{N_{s_{k+1}}}{n_{c_k}} \text{ with } d_h = 1 \tag{5.9}$$

In the following, we express $\langle M_r \rangle$, the average matching cost per request in the whole system. For a given request R crossing the graph from level h to level 1 and being forwarded to b_k nodes at each level nodes, we have:

$$\langle M_r \rangle = \sum_{k=1}^{h} [b_k \times (n_{c_k} + \langle \beta_k \rangle \times \langle m_{s_k} \rangle)]. \tag{5.10}$$

Based on the hierarchy of nodes and on the number of children nodes per parent node, we can write:

$$\forall k \in]1, h], n_{c_k} \times \langle m_{s_k} \rangle = n_{c_{k-1}} \times d_{k-1}, \tag{5.11}$$

$$\text{and for } k = 1, n_{c_1} \times \langle m_{s_1} \rangle = N_{s_0}. \tag{5.12}$$

If we set $n_{c_0} = 1$ and $d_0 = N_{s_0}$, then we can write in the following:

$$\forall k \in [1, h], \langle M_r \rangle = \sum_{k=1}^{h} [b_k \times (n_{c_k} + \langle \beta_k \rangle \times \frac{n_{c_{k-1}}}{n_{c_k}} \times d_{k-1})], \tag{5.13}$$

$\langle M_r \rangle$ is thus a function $f : \mathbb{N}^* \longmapsto \mathbb{R}$ with multiple variables $(n_{c_1}, n_{c_2}, ..., n_{c_k}, .., n_{c_h})$:

$$f(n_{c_1}, n_{c_2}, ..., n_{c_k}, .., n_{c_h}) = \sum_{i=1}^{h} [b_k \times (n_{c_k} + \langle \beta_k \rangle \times \frac{n_{c_{k-1}}}{n_{c_k}} \times d_{k-1})]. \tag{5.14}$$

To minimize f, we calculate when its partial derivatives $\frac{\partial f}{\partial n_{c_k}} = 0 \; \forall k \in [1, h]$. After calculation, we have $\forall k \in [1, h-1]$:

$$n_{c_{k+1}} = \frac{b_{k+1} \times \langle \beta_{k+1} \rangle \times d_k}{b_k \times \langle \beta_k \rangle \times d_{k-1} \times \frac{n_{c_{k-1}}}{n_{c_k}^2} - b_k}, \tag{5.15}$$

and for $k = h$:

$$n_{c_k}^2 = \langle \beta_k \rangle \times d_k \times n_{c_{k-1}}. \tag{5.16}$$

This system of non linear-equations can be resolved by expressing variables of the highest level node n_{c_h} with the lowest level one n_{c_1} and then resolving the system iteratively. Some numerical iterative methods for non-linear equations, such as Seidel or Newton's Methods ([117]) can be used to speed up the resolution and find $n_{c_k} \forall k \in [1, h-1]$ then minimize f. However taking into consideration the system, this approach is constraining since the optimization of each level in the hierarchy of nodes requires information from other levels. Moreover, any modification within a node will require to re-compute the variables (i.e., $n_{c_k} \forall k \in [1, h-1]$) which comes with communication and computational overhead.

Comparaison and discussion: We compare in table 5.1 the advantages and limitations of the individual node-based optimization approach and the global system optimization approach.

Individual node-based optimization approach	**Global system optimization approach**
Advantages	
Local optimization of the search cost per node	Global minimal value for the search cost
Low relative rate of generated updates sent by a node to its parent	
Limited communication exchanges between nodes	
Flexible system, low constraint among nodes	
Limitations	
Global minimal value for the search cost	Need for a global knowledge of the system's parameters
	Each modification in a single node to impact the optimization parameters in the whole system
	Need for a high level rate of information exchange among nodes

Table 5.1: Comparison of the advantages and limitations of the Individual node-based optimization approach and the global system optimization approach label.

The global optimizing approach allows to have the minimal value of the search cost. However, it presents a number of drawbacks that may impact the system scalability. This approach requires a global knowledge of the system at each instant of time, which produces high level rate of information exchange among the different nodes. Although the Individual node based optimization does not provide the minimal value of the search cost as the global optimization does, it does not require a global knowledge of the system parameters and the rate of information exchange among nodes is very low and does not impact system scalability. The compromise between the search cost and the system's scalability is more interesting with the latter approach than with the former one. The latter approach will be considered in the following.

5.4 Numerical evaluation

We carried out in this section different experiments to evaluate the performance of the mobility support mechanisms that we presented in this chapter. A prototype of distributed semantic gateways, representing a building, and the associated discovery mechanisms have been developed. Details about the prototype implementation are in appendix A. In the following, we first present the used metrics and we explain the different experimental parameters and settings. Then, we detail three different experiments where we consider a single node in the two first ones and a hierarchy of nodes in the third one.

5.4.1 Metrics

We use the following metrics to measure the system's performances:

- The total matching cost M_T: It is the sum of the search matching cost (i.e., the number of service-request matching operations involved during service search), 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)
- The relative rate of generated updates sent by a node to its parent: It 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).

5.4.2 Experimental parameters and settings

To evaluate the system, we implemented the incremental clustering mechanism as described in 5.3.1 and the clustering optimization mechanism as described in 5.3.2. We explain the different settings below.

5.4.2.1 Experiment parameters

The following experiment parameters are considered for a lowest level node within the hierarchy of nodes. For service mobility, we simulated service arrivals and departures. The service arrival process is a Poisson process [118] with an arrival rate λ. The time spent by a service in a node has an exponential distribution with parameter μ. The average number of services within a node is then : $\langle N_s \rangle = \lambda/\mu$. We simulate service mobility during $100 \times T_0$, where $T_0 = 1/\mu$ is the average time spent by a service within a node. The request arrival process follows a Poisson process with an arrival rate η. We consider a system where there is more incoming requests than mobile events (i.e., $\eta >> \lambda$) for a given period of time T. During T_0, n_r and n_m are respectively the number of sent requests and the total number of service arrivals and departures within a node. For tests we choose $n_r = 5 \times n_m$ and we have $n_m = 2 * \langle N_s \rangle$.

During simulation, we measure the different matching costs (i.e., M_{rT} and M_T) and the number of updates generated by a node and sent to its parent node. To measure the search matching cost M_{rT} within a node, 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.

For re-clustering mechanisms, we fixed $\langle \beta \rangle = 1$ to calculate $n_{c_{op}}$ and $\langle m_{s_{op}} \rangle$. We took a margin $\gamma = 1$, for all experiments, for the maximum number of services per cluster $m_{Max_{Limit}}$ as we explained in 5.3.2.2.

5.4.2.2 Clustering initialization step

Starting with empty lowest level nodes, they are filled with service arrivals representing singletons. The routing table of the node is constructed and updated. Cluster representatives are sent to higher level nodes. In a node, merging clusters starts when there is about 10 singletons, at this stage, the optimized clustering becomes more advantageous in terms of search cost than considering singletons.

5.4.3 Impact of the number of services per node: case of a single lowest node

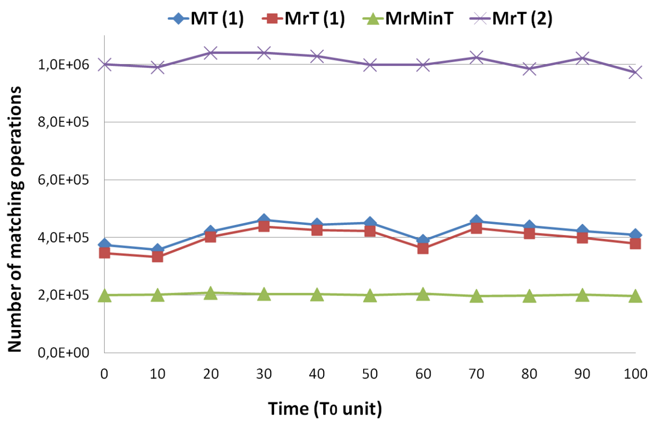
We evaluate, in this part, the impact of the number of services on the matching cost and on the relative rate of generated updates sent by a node to its parent. For that, we consider 3 nodes with the same parameters in terms of margins ($\gamma = 1$ and $\phi = 0.5$). The first node contains an average number of 100 services over time (we fix $\mu = 0.01$ and $\lambda = 1$), the second node contains an average number of 400 services over time (we fix $\mu = 0.0025$ and $\lambda = 1$)) and the third node contains an average number of 800 services over time (we fix $\mu = 0.00125$ and $\lambda = 1$). We also compare our approach (1) with a centralized approach (2) representing a non-clustered node.

We measure the evolution of the overall matching cost M_T and we compare it with the evolution of the total search cost $M_{rT} = n_r \times \langle M_r \rangle$, during $100 \times T_0$. Results are shown in figure 5.6, where we show respectively the evolution of a node with 100 services, 400 services and 800 services

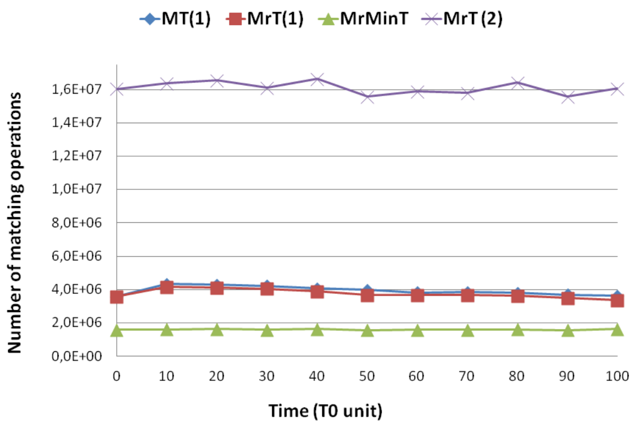
‘ We compare the search cost of our approach (1) with the search cost of approach (2) representing a non-clustered node. We have $M_{rT}(2) \geq 2 \times M_{rT}(1)$ for the 100-service nodes and $M_{rT}(2) \geq 4 \times M_{rT}(1)$ for the 400-service nodes, and $M_{rT}(2) \geq 6 \times M_{rT}(1)$ for the 800-service nodes, over time. In both cases, our approach (1) is performing better than a non-clustering approach (2) in terms of search matching cost. We can see that the gain in terms of search matching cost increases when the number of services per node increases.

We can see in our approach that $M_T(1)$ and $M_{rT}(1)$ are quasi-constant over time thanks to the mechanisms that we defined. We have $M_T(1) \approx M_{rT}(1)$ for the 3 nodes, which confirms that the matching costs generated by the management mechanisms are negligible compared to the search cost. However, there is a small difference between the 3 nodes. We see that in the case of 400 and 800 services $M_T(1)$ and $M_{rT}(1)$ are more stable over time that the ones of 100 services. Moreover, we can see that the difference between $M_T(1)$ and $M_{rT}(1)$ is more attenuated than in the case of 400 and 800 services than in the case of 100 services. The performances of the node in terms of matching cost seem better when the number of services increases.

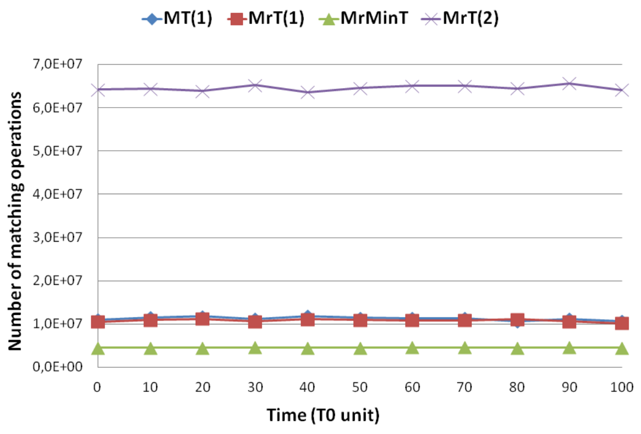
We also compare the search matching cost M_{rT} with the minimal value $M_{r_{minT}} = n_r \times \langle M_{r_{min}} \rangle$ defined in proposition 2. We have $M_{rT}(1) \approx 2 \times M_{r_{minT}}$. This gap between the two values is due to the margins that we added to $n_{c_{op}}$ and $\langle m_{s_{op}} \rangle$ to reduce the number of re-clustering operations. Also, cluster merging and splitting degrade the quality of clusters, which increases the average number of visited clusters per request $\langle \beta \rangle$ which is in $]1, 2]$ for the 3 nodes. However,



(a) 100 services



(b) 400 services



(c) 800 services

Figure 5.6: Impact of service number per node on the matching cost for margin $\phi = 0.5$

we think that having a cost only twice as higher than the minimal theoretical value could be considered as a good result.

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 search. Results are shown in figure 5.7.

We can see that in the 3 cases of 100 services, 400 services per node and 800 services per node, the relative rate of updates to parent node is low. However, it is less than 1% in the case of 400 services and 800 services and higher than 10% in the case of 100 services. As stated above, the system seems to perform better when the number of services per node increases. For the 3 considered nodes, our mechanisms allow for a higher-level node to receive mobility events at a rate much slower than the one received at a lower-level node.

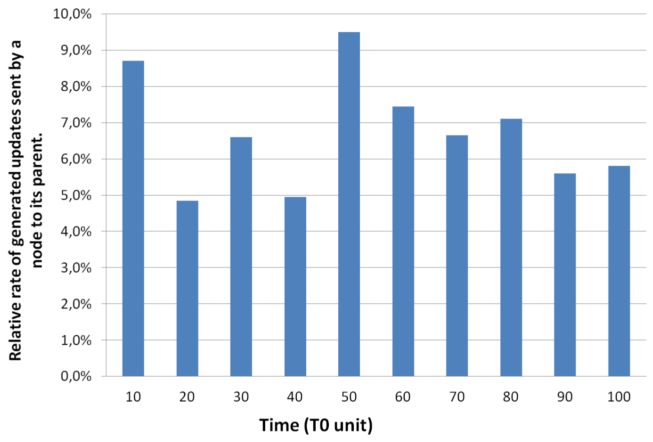
5.4.4 Impact of margin values: case of a single lowest level node

We evaluate in this section, the impact of margins defined in definition 10, on the matching cost and on the relative rate of generated updates sent by a node to its parent. For that, we consider 5 nodes, with different parameters : for the nodes 1, 2 and 3 we consider respectively $\phi = 0.3$, $\phi = 0.5$ and $\phi = 0.8$ as margins on the optimal number of clusters $n_{c_{op}}$ (we fix $\gamma = 1$ as mentioned in section 5.4.2.1). For the two last nodes (nodes 4 and 5), we do not consider our re-clustering mechanisms. However, we perform a total re-clustering of the nodes 4 and 5 respectively every T_0 and $10T_0$, based on the hierarchical clustering used in the previous chapter and presented in section 4.3.2.1. During simulation, all nodes have the same services at a given time (i.e., same service arrivals and departures). We consider, 100-service nodes.

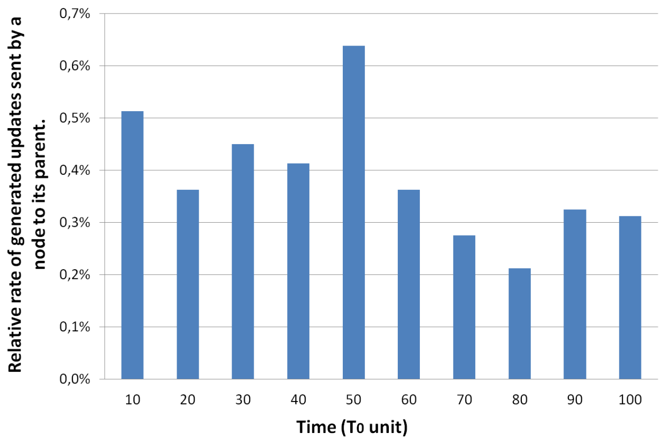
Results related to the evolution of the matching costs are shown in figure 5.9. We can see that the variation in node parameters impact the matching costs (i.e., respectively M_T and M_{rT}). In figures (a), (b) and (c) the margin ϕ related to the number of clusters per node evolves from 0.3 to 0.8. The matching cost is impacted by the variation of ϕ value. We can see that M_rT increases with ϕ. When ϕ increases, the number of clusters per node goes up which augments the number of representatives in the routing table of the node and thus, the number of matching operations. The variation of the matching costs are almost similar in the 3 cases. However, the value of M_T differs from case (a) to (c). We can see that the bigger ϕ is, the smaller the difference between M_T and M_{rT} is. Actually, when the margin ϕ is small, the number of merging operations increases and so the number of matching operations involved during merging. Thus, there is a trade off to find between the value of ϕ and the value of M_T in order to limit the cost induced by mobility mechanisms. We think that it is better to consider the values of ϕ in $[0.5; 1]$ since they limit the number of re-clustering operations while maintaining an acceptable search cost ($M_{rT} \approx 2 \times M_{r_{minT}}$).

Nodes 4 and 5 present cases where we do not perform cluster merging and splitting (i.e., we do not perform clustering optimization). However, we consider a re-clustering of the whole node content based on a hierarchical clustering defined in chapter 4. In case (d) we perform a re-clustering every T_0 and in case (e) every $10T_0$. In case (d), we can see that although M_{rT} is low, M_T is high due to the cost engendered by the global re-clustering. This impacts the system scalability. However, in case (e), we can see that the performances are similar to case (a) and a bit worse than (b) and (c) in terms of matching costs generated by mobility support mechanisms.

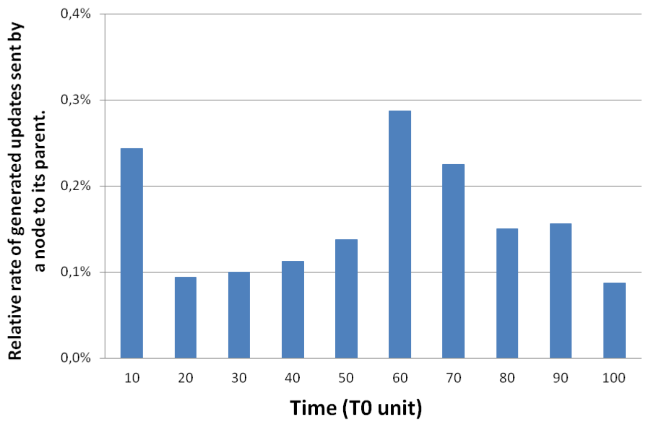
We measured the relative rate of generated updates sent by a node to its parent considering the different parameters. Results are shown in figure 5.8. We see that for nodes (a), (b) and (c), the relative rate of generated updates sent by a node to its parent considering the different parameters is lower than 10%. However, for nodes (d) and (e) it is higher than 10%. We can see in the three first nodes that the smaller ϕ is, the higher is the relative rate of generated



(a) 100 services



(b) 400 services



(c) 800 services

Figure 5.7: Impact of the service number on the relative rate of generated updates sent by a node to its parent for margin $\phi = 0.5$

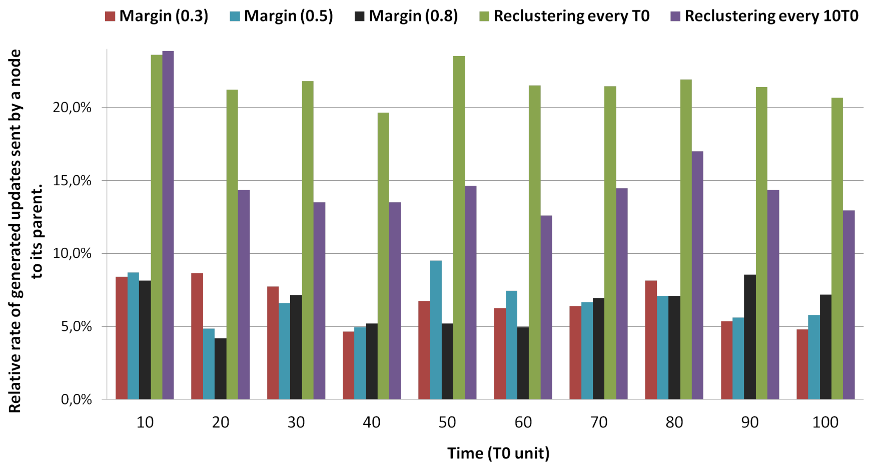


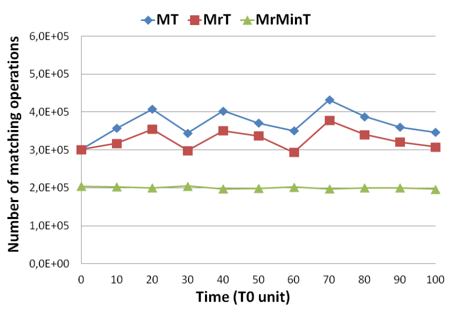
Figure 5.8: Relative rate of generated updates sent by a node to its parent for different values of margin ϕ

updates sent by a node to its parent. This is due to the increase in the number of merging operations. In nodes (d) and (e), re-clustering the whole node engenders a high level of relative rate of generated updates sent by a node to its parent. This makes the system not scalable due to the frequency of updates. Consequently, we think that this approach is not suitable for managing service mobility even if the performances in terms of matching cost of nodes (a) and (e) are similar.

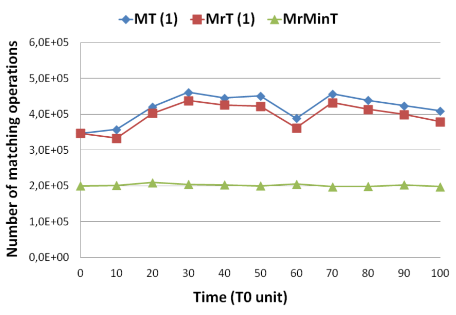
5.4.5 Impact of service mobility on the nodes hierarchy

We evaluate in this part the mobility support mechanisms over a hierarchy of nodes. We want to evaluate the impact of service mobility on higher level nodes in terms of matching cost evolution and generated updates. We consider a 2-level hierarchy of nodes with 3 children nodes N_1, N_2, N_3 and their parent node N_4. We simulated the mobility of 100 services in each child node with $\mu = 0.01$ and $\lambda = 1$ (see figure 5.10). We detailed in section 5.3.4.1, the different cases of changes happening at a lowest level node and their impact on the parent node.

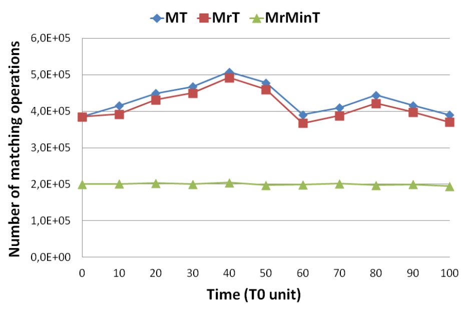
For re-clustering mechanisms, we fixed $\gamma = 1$ and $\phi = 0.5$ in all nodes. Based on these parameters, the average number of services in the parent node is around 48. Although the number of services in the parent node (N_4) is low, our main purpose here is to observe the impact of mobility on the parent node. As we did in the previous experiments, 100 requests are sent to the parent node to perform service search over the hierarchy of nodes. We measure the average matching cost per request and use it to calculate M_{rT}. In this case, for the parent node, we considered $n_r = 5 \times n_{mT}$ with n_{mT} the total number of events received by the 3 children nodes. Then, $n_r = 3 \times 5 \times 10 \times 200 = 30000$ requests sent to the parent node each $10 \times T_0$. The requests are then forwarded to children nodes based on the location of the searched services. We chose a non-uniform distribution of services over the 3 children nodes in order to ensure service diversity within the parent node and to have an average of 1 visited child node per request. Moreover, in this experiment, we choose a distribution of request which is equiprobable over the 3 nodes. Thus, each child node receives about 10000 requests per $10 \times T_0$. We measure the evolution of the overall matching cost $M_T(4)$ and the search cost $M_rT(4)$ of the parent node N_4 and we compare it with the overall matching cost $M_T(1)$ and search cost $M_rT(1)$ of the child node N_1 (We display results for one child node). Figure 5.11 shows the results of matching costs for the parent node (N_4) and the child node (N_1).



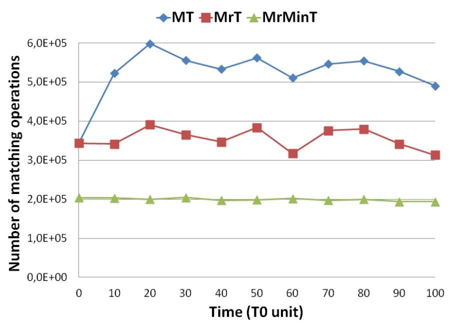
(a) Node 1: Margin $\phi = 0.3$



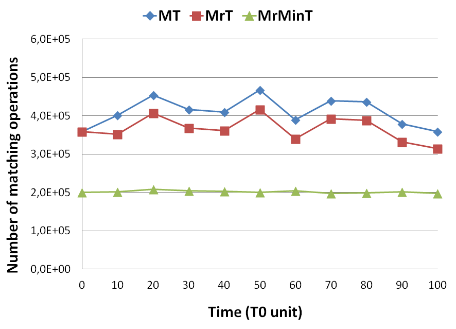
(b) Node 2: Margin $\phi = 0.5$



(c) Node 3: Margin $\phi = 0.8$



(d) Node 4: No margins and re-clustering every T_0



(e) Node 5: No margins and re-clustering every $10T_0$

Figure 5.9: The evolution of the matching cost based on different margins' parameters

Results show that both nodes have similar evolutions in terms of matching costs. In the parent node N_4 as well as in the children node N_1 we have $M_{rT}(i) \approx 2 \times M_{r_{minT}}(i)$ and $M_T(i) \approx M_{rT}(i)$ with $i \in \{1, 4\}$. Moreover, the evolution of the matching costs is constant over time for both nodes. This illustrates that for a given hierarchy of nodes, each node is able to minimize locally its matching costs and keep it constant over time. This enables to limit the computational efforts needed in each semantic gateway to process service search and to manage service mobility.

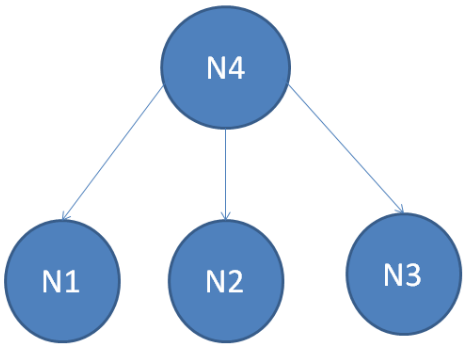
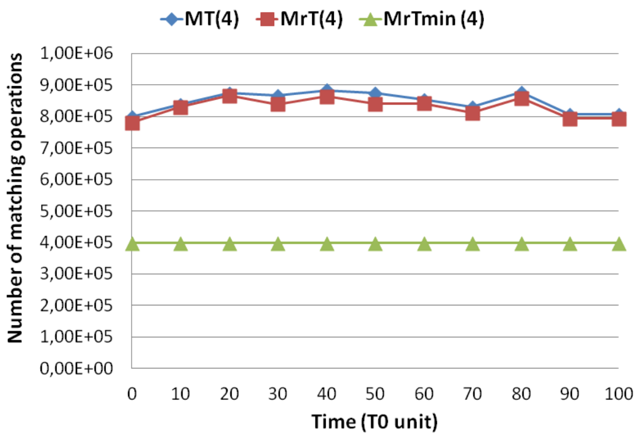
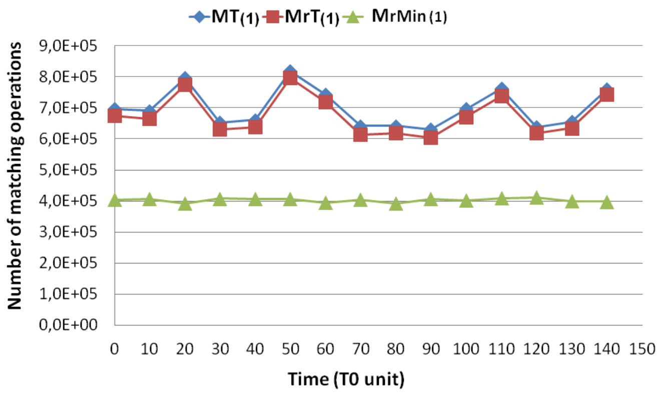


Figure 5.10: 2-level hiearchy of nodes: N1, N2 and N3 100-service children nodes and N4 a parent node



(a) Parent node:N_4



(b) Child node: N_1

Figure 5.11: Matching costs for the parent node N_4 and its child node N_1 for $\phi = 0.5$

We also measured the ratio between the number of events sent by the node N_4 to its parent node and the total number of events received by the children nodes N_1, N_2 and N_3, every

$10 \times T_0$. In this way, we are able to evaluate the impact of service mobility in lowest level nodes on the higher level nodes. Figure 5.12 shows the results. We can see that the ratio is lower than 1% which confirms the fact that higher level nodes are only slightly affected by service mobility in lowest level nodes and that we can expect negligeable induced mobility events in the highest nodes. This shows that our system is stable and able to well manage content changes in each semantic gateway while ensuring efficient and scalable search.

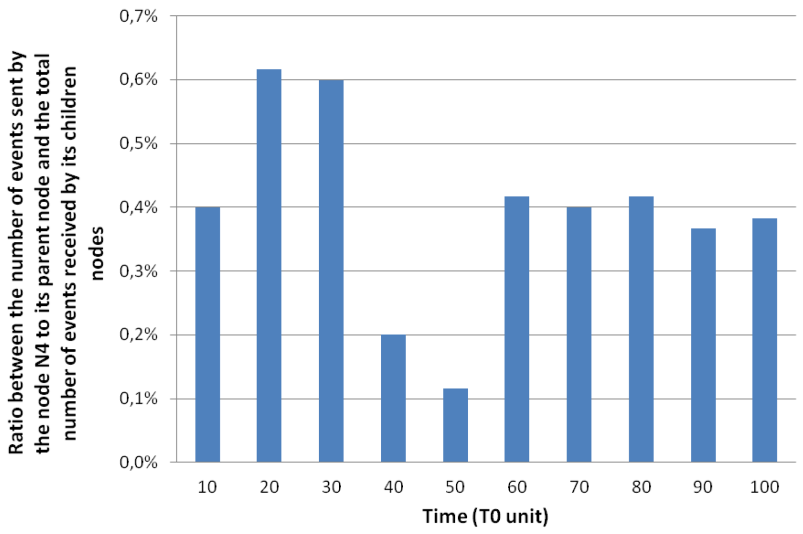


Figure 5.12: Ratio between the number of events sent by the node N_4 to its parent node and the total number of events received by its children nodes N_1,N_2 and N_3.

5.5 Concluding remarks

This chapter details the mobility support mechanisms for IoT service discovery in a dynamic context. Within a semantic node, we proposed a method to create and update dynamically clusters of similar services in order to ensure efficient search within the system despite continuous changes. We also optimized the clustering in terms of number of clusters and number of services per cluster based on defined conditions to minimize continuously service search matching cost. Tests performed on lowest level nodes showed that the search matching cost is quasi-constant over time and that our approach scales well with the increase in the number of services per node. Results also showed that our approach scales compared to a centralized approach where a request is matched against all services within the registry. Besides, the percentage of updates sent by a node to its parent node is low which means that higher level nodes are only slightly affected by service mobility. This is confirmed by experiments on a hierarchy of nodes where we could see that the number of updates sent by a higher node to its parent node is small. Moreover, all nodes within the hierarchy are able to minimize their matching costs over time. As a result, the global matching costs and updating costs in the whole system are optimized. Based on the defined mechanisms for service mobility, the system is stable and able to manage service dynamicity and continuous changes in a scalable way.

Chapter 6

Conclusion

Internet is extending to the physical world, forming the premises of the Internet of Things and creating smart spaces (i.e., geographical locations that contain connected objects and their IoT services). However, while the *Intranet* of Things exists today, where connected objects in a local area can communicate together in silos and offer IoT services (e.g., within a house), the vision of the Internet of Things where a large number of objects are cooperating together still has a way to go.

A key feature to the success of this vision is to enable service search, where a human user or a device can find another device offering specific functionalities among billions of others. However, the huge number of IoT services, their heterogeneity and their mobility (which results in objects roaming from one smart space to another), all contribute to making the search of relevant IoT services for a given situation challenging.

Building upon service-oriented middleware allows having a homogeneous view of the diverse devices as they are abstracted as services and are discovered and communicate using standard protocols. While this is feasible at small scale, it becomes hard to achieve when considering the Internet of Things context where each platform brings its own service descriptions syntax, discovery and access "standards". Thus, interoperability methods have to be developed in order to facilitate service discovery and access.

In the context of the Internet of Things on a large number of mobile devices, it becomes hard for the user to have the exact information about existing services. Searching services based on a given description and allowing flexible search where the returned services can be close to the search request rather than matching it exactly, seems to be a good approach for service search in IoT. Semantic Web technologies allow unambiguous semantic specification of service functionalities. Thus a service can be searched based on its description. Adding semantics to IoT service descriptions makes such information machine interpretable and enables more interoperability and flexibility in the search. However, the use of semantic technologies come with a cost in terms of semantic reasoning and computational resources which makes its use for large number of resources challenging. Hence, efficient methods to handle large scale-semantic-based and mobile service descriptions are needed to enable service discovery in the context of the Internet of Things.

6.1 Contributions

In this thesis, we presented an overview of a semantic-based middleware. We developed its discovery mechanism to perform flexible, exhaustive and scalable search for semantic-based IoT services while supporting their dynamicity. Our main contributions can be summarized in the following:

Definition of an asymmetric matching function for semantic service descriptions: It is a quasi-metric that measures the "distance" between semantic service descriptions or between a request and a semantic service description. This measurement takes into consideration the different parameters of a service and allows an accurate and flexible matching with a request. This measurement is defined is a way to allow a clear distinction between matching services and non-matching services.

Efficient semantic-based service search mechanism supported by semantic gateways: We describe a service discovery infrastructure based on distributed *semantic gateways* that represent smart spaces to better exploit the location of devices and support semantic-based service search. Each semantic gateway registers the semantic descriptions of IoT services in its vicinity. These semantic gateways are modeled as a hierarchical tree graph that represents the hierarchy of smart spaces (e.g., country, region, city, streets, buildings, rooms). Each semantic gateway performs clustering and aggregation of its semantic service descriptions, based on the quasi-metric defined above, and builds a routing table that provides an overview of its content. Based on thresholds, the search mechanism is able to search exhaustively, within the tree of nodes, all services that match a request while selecting the semantic gateways that are more likely to contain suitable services. Within a gateway, the search mechanism is able to limit the number of useless matching operations in order to find services matching a request.

Efficient management of service registry in a dynamic context: We implement mechanisms to support service dynamicity induced by device mobility. Thanks to the infrastructure of physical and semantic gateways, device arrivals and departures are detected by physical gateways (although this part is not treated in this thesis) and the content of semantic gateways is updated automatically. Moreover, with the incremental clustering that we defined, it is possible to update dynamically clusters of similar services with no need to re-cluster the whole content of the registry. We also implement a method to optimize the distribution of services within clusters in order to minimize the search cost over time. This cost is measured in terms of the number of service-request matching operations performed in a gateway to find services matching an incoming request. Besides, the relative rate of generated updates sent by a gateway to its parent is very low. These mechanisms allow to maintain a scalable semantic-based discovery in a dynamic context.

We also provide a prototype implementation of our middleware that serves to evaluate our mechanisms' performances.

6.2 Perspectives

Besides the contributions in this thesis, some issues still needed to be investigated. In fact, throughout the work presented in this thesis, we made a number of assumptions that can be discussed and addressed in future research activities. We briefly list them in the following:

- While in this work, we consider only the discovery of services, a possible extension would be to enhance the functionalities of this middleware by taking into consideration service compositions. In this case, a user can send a complex query that is decomposed into simpler search requests. Each request will be searched and the final result will be a composition of the retrieved services. The use of semantic-based service descriptions will allow dynamic and automatic selection of conformed services to compose.

- In our work, we consider mainly "simple" ontologies based on a hierarchy of concepts. The matching is based on assessing the subsumption relations between concepts in semantic service descriptions. As a next step, we would like to consider ontologies based on rich Description Logic (DL) with complex formal definition of concepts using logical operators, with the purpose to describe complex and composed IoT services. To do so, we need to investigate how to adapt our metrics to such descriptions.
- In this thesis, we assume that semantic service descriptions are automatically generated. It is an assumption that can be considered to propose an approach to automatically and efficiently generate semantic service descriptions from the description of service devices. Ideas from existing approaches can be investigated.
- In the described architecture, we consider fixed gateways while objects are mobile. As a next step, we can consider mobile gateways that can be hosted, for instance, on the users' smartphones. Thus, issues related to the unknown topology of gateways need to be taken into consideration to enable efficient search.
- Security and privacy issues are serious challenges in the Internet of Things. We did not mention them in this thesis because they are not part of the current work. However, in the future we intend to deal with security and privacy issues at the service layer and try to propose some models and approaches to deal with these challenges.

Bibliography

[1] M. Weiser, "The computer for the 21st century," *Scientific american*, vol. 265, no. 3, pp. 94–104, 1991.

[2] F. Zhu, M. W. Mutka, and L. M. Ni, "Service discovery in pervasive computing environments," *IEEE Pervasive computing*, vol. 4, no. 4, pp. 81–90, 2005.

[3] 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, no. 3, pp. 223–235, 2010.

[4] E. Guttman, "Service location protocol: Automatic discovery of ip network services," *Internet Computing, IEEE*, vol. 3, no. 4, pp. 71–80, 1999.

[5] E. Toch, I. Reinhartz-Berger, and D. Dori, "Humans, semantic services and similarity: A user study of semantic web services matching and composition," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 9, no. 1, pp. 16–28, 2011.

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

[7] "Casagras, eu. fp7 project, rfid and the inclusive model for the internet of things. technical report, 2012." www.grifs-project.eu.

[8] "Internet of things: Strategic research roadmap," http://www.grifs-project.eu/, 2009.

[9] S. Haller, S. Karnouskos, and C. Schroth, "The internet of things in an enterprise context," in *Future Internet–FIS 2008*. Springer, 2009, pp. 14–28.

[10] D. Guinard, "A Web of Things Application Architecture – Integrating the Real-World into the Web," Ph.D., ETH Zurich, 2011. [Online]. Available: http://webofthings.org/dom/thesis.pdf

[11] S. Hachem, "Middleware pour l'internet des objets intelligents," Ph.D. dissertation, Université de Versailles-Saint Quentin en Yvelines, 2014.

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

[13] "emarketer article: Smartphone users worldwide will total 1.75 billion in 2014." http://www.emarketer.com/Article/Smartphone-Users-Worldwide-Will-Total-175-Billion-2014/1010536.

[14] I. Toma, E. Simperl, A. Filipowska, G. Hench, and J. Domingue, "Semantics-driven interoperability on the future internet," in *Semantic Computing, 2009. ICSC'09. IEEE International Conference on*. IEEE, 2009, pp. 551–558.

[15] M. Zorzi, A. Gluhak, S. Lange, and A. Bassi, "From today's intranet of things to a future internet of things: a wireless-and mobility-related view," *Wireless Communications, IEEE*, vol. 17, no. 6, pp. 44–51, 2010.

[16] F. Mattern and C. Floerkemeier, "From the internet of computers to the internet of things," in *From active data management to event-based systems and more*. Springer, 2010, pp. 242–259.

[17] M. P. Papazoglou, "Service oriented computing: Concepts, characteristics and directions," in *Web Information Systems Engineering, 2003. WISE 2003. Proceedings of the Fourth International Conference on.* IEEE, 2003, pp. 3–12.

[18] M. Presser, P. M. Barnaghi, M. Eurich, and C. Villalonga, "The sensei project: integrating the physical world with the digital world of the network of the future," *Communications Magazine, IEEE*, vol. 47, no. 4, pp. 1–4, 2009.

[19] 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*, pp. 223–235, 2010.

[20] T. Gu, H. K. Pung, and D. Q. Zhang, "A service-oriented middleware for building context-aware services," *Journal of Network and computer applications*, vol. 28, no. 1, pp. 1–18, 2005.

[21] J. King, R. Bose, H.-I. Yang, S. Pickles, and A. Helal, "Atlas: A service-oriented sensor platform: Hardware and middleware to enable programmable pervasive spaces," in *Local Computer Networks, Proceedings 2006 31st IEEE Conference on.* IEEE, 2006, pp. 630–638.

[22] 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.* Springer, 2011, pp. 220–229.

[23] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web," *Scientific American*, vol. 284, no. 5, pp. 34–43, 2001.

[24] "Ontologies," http://www.w3.org/standards/semanticweb/ontology.

[25] M. Eid, R. Liscano, and A. El Saddik, "A universal ontology for sensor networks data," in *Computational Intelligence for Measurement Systems and Applications, 2007. CIMSA 2007. IEEE International Conference on.* IEEE, 2007, pp. 59–62.

[26] B. Christophe, "Semantic profiles to model the "web of things"," *Semantics, Knowledge and Grid, International Conference on*, vol. 0, pp. 51–58, 2011.

[27] W. Wang, S. De, R. Toenjes, E. Reetz, and K. Moessner, "A comprehensive ontology for knowledge representation in the internet of things," in *Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on.* IEEE, 2012, pp. 1793–1798.

[28] Z. Song, A. A. Cárdenas, and R. Masuoka, "Semantic middleware for the internet of things," in *Internet of Thingsb(IOT), 2010.* IEEE, 2010, pp. 1–8.

[29] V. Haarslev and R. Möller, "On the scalability of description logic instance retrieval," *J. Autom. Reason.*, vol. 41, no. 2, pp. 99–142, Aug. 2008. [Online]. Available: http://dx.doi.org/10.1007/s10817-008-9104-7

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

[31] IOT-A Consortium, "Concepts and Solutions for Identification and Lookup of IoT Resources (D4.1)," IOT-A Project, Tech. Rep., 2011. [Online]. Available: http://www.iot-a.eu/public/public-documents/documents-1

[32] S. Ben Fredj, M. Boussard, D. Kofman, and L. Noirie, "A Scalable IoT Service Search Based on Clustering and Aggregation," in *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.

[33] IOT-A Consortium, "Concepts and Solutions for Entity-based Discovery of IoT Resources and Managing their Dynamic Associations (D4.3)," IOT-A Project, Tech. Rep., 2012. [Online]. Available: http://www.iot-a.eu/public/public-documents/documents-1

[34] S. Ben Fredj, M. Boussard, L. Noirie, and D. Kofman, “Method and system for discovery of services in interconnected smart spaces,” European Patent EP 13 305 641.6, May 17, 2013.

[35] S. Ben Fredj, M. Boussard, D. Kofman, and L. Noirie, “Efficient semantic-based IoT service discovery mechanism for dynamic environments,” accepted, to appear in IEEE 25th Ann. International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), 2014.

[36] D. Guinard, “Towards the web of things: Web mashups for embedded devices,” in *In MEM 2009 in Proceedings of WWW 2009. ACM*, 2009.

[37] R. Kawamoto, T. Emori, S. Sakata, K. Furuhata, K. Yuasa, and S. Hara, “Dlna-zigbee gateway architecture and energy efficient sensor control for home networks,” in *Mobile and Wireless Communications Summit, 2007. 16th IST*. IEEE, 2007, pp. 1–5.

[38] H.-j. He, Z.-q. Yue, and X.-j. Wang, “Design and realization of wireless sensor network gateway based on zigbee and gprs,” in *Information and Computing Science, 2009. ICIC’09. Second International Conference on*, vol. 2. IEEE, 2009, pp. 196–199.

[39] C. Bormann, A. P. Castellani, and Z. Shelby, “Coap: An application protocol for billions of tiny internet nodes,” *IEEE Internet Computing*, vol. 16, no. 2, pp. 62–67, 2012.

[40] Z. Shelby and C. Bormann, *6LoWPAN: The Wireless Embedded Internet*. Wiley Publishing, 2010.

[41] W3C working group, “Web Services Architecture,” W3C, Tech. Rep., 2004. [Online]. Available: http://www.w3.org/TR/ws-arch/

[42] L. Richardson and S. Ruby, *Restful Web Services*, 1st ed. O’Reilly, 2007.

[43] M. Eisenhauer, P. Rosengren, and P. Antolin, “A development platform for integrating wireless devices and sensors into ambient intelligence systems,” in *Sensor, Mesh and Ad Hoc Communications and Networks Workshops, 2009. SECON Workshops’ 09. 6th Annual IEEE Communications Society Conference on*. IEEE, 2009, pp. 1–3.

[44] N. B. Priyantha, A. Kansal, M. Goraczko, and F. Zhao, “Tiny web services: Design and implementation of interoperable and evolvable sensor networks,” in *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, ser. SenSys ’08. New York, NY, USA: ACM, 2008, pp. 253–266. [Online]. Available: http://doi.acm.org/10.1145/1460412.1460438

[45] L. M. S. De Souza, P. Spiess, D. Guinard, M. Köhler, S. Karnouskos, and D. Savio, “Socrades: A web service based shop floor integration infrastructure,” in *The internet of things*. Springer, 2008, pp. 50–67.

[46] W. Drytkiewicz, I. Radusch, S. Arbanowski, and R. Popescu-Zeletin, “prest: a rest-based protocol for pervasive systems,” in *Mobile Ad-hoc and Sensor Systems, 2004 IEEE International Conference on*, Oct 2004, pp. 340–348.

[47] D. Guinard, V. Trifa, and E. Wilde, “A resource oriented architecture for the web of things,” in *Internet of Things (IOT), 2010*, Nov 2010, pp. 1–8.

[48] M. Boussard, B. Christophe, O. Le Berre, and V. Toubiana, “Providing user support in web-of-things enabled smart spaces,” in *Proceedings of the Second International Workshop on Web of Things*, ser. WoT ’11. New York, NY, USA: ACM, 2011, pp. 11:1–11:6. [Online]. Available: http://doi.acm.org/10.1145/1993966.1993981

[49] D. Guinard, I. Ion, and S. Mayer, “In search of an internet of things service architecture: Rest or ws-*? a developers’ perspective,” in *Mobile and Ubiquitous Systems: Computing, Networking, and Services*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, A. Puiatti and T. Gu, Eds. Springer Berlin Heidelberg, 2012, vol. 104, pp. 326–337. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-30973-1_32

[50] N. Guarino, M. Carrara, and P. Giaretta, "An ontology of meta-level categories." *KR*, vol. 94, pp. 270–280, 1994.

[51] "Linked data," http://linkeddata.org/.

[52] S. A. McIlraith, T. C. Son, and H. Zeng, "Semantic web services," *IEEE intelligent systems*, vol. 16, no. 2, pp. 46–53, 2001.

[53] M. Botts, G. Percivall, C. Reed, and J. Davidson, "Ogc® sensor web enablement: Overview and high level architecture," in *GeoSensor networks.* Springer, 2008, pp. 175–190.

[54] M. Compton, P. Barnaghi, L. Bermudez, R. GarcíA-Castro, O. Corcho, S. Cox, J. Graybeal, M. Hauswirth, C. Henson, A. Herzog *et al.*, "The ssn ontology of the w3c semantic sensor network incubator group," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 17, pp. 25–32, 2012.

[55] S. De, T. Elsaleh, P. Barnaghi, and S. Meissner, "An internet of things platform for real-world and digital objects," *Scalable Computing: Practice and Experience*, vol. 13, no. 1, 2012.

[56] IOT-A Consortium, "Converged Architectural Reference Model for the IoT (D1.4)," IOT-A Project, Tech. Rep., 2012. [Online]. Available: http://www.iot-a.eu/public/public-documents/documents-1

[57] B. Christophe, "Managing massive data of the internet of things through cooperative semantic nodes," in *Semantic Computing (ICSC), 2012 IEEE Sixth International Conference on*, 2012, pp. 93–100.

[58] W. Wang, S. De, R. Toenjes, E. Reetz, and K. Moessner, "A comprehensive ontology for knowledge representation in the internet of things," *2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, vol. 0, pp. 1793–1798, 2012.

[59] B. Christophe, V. Verdot, and V. Toubiana, "Searching the 'web of things'," in *Semantic Computing (ICSC), 2011 Fifth IEEE International Conference on*, Sept 2011, pp. 308–315.

[60] H. Dong, F. K. Hussain, and E. Chang, "Semantic web service matchmakers: state of the art and challenges," *Concurrency and Computation: Practice and Experience*, vol. 25, no. 7, pp. 961–988, 2013.

[61] E. Cimpian, H. Meyer, D. Roman, A. Sirbu, N. Steinmetz, S. Staab, and I. Toma, "Ontologies and matchmaking," in *Semantic Service Provisioning.* Springer, 2008, pp. 19–54.

[62] M. Paolucci, T. Kawamura, T. R. Payne, and K. Sycara, "Semantic matching of web services capabilities," in *The Semantic Web—ISWC 2002.* Springer, 2002, pp. 333–347.

[63] E. Sirin, J. Hendler, and B. Parsia, "Semi-automatic composition of web services using semantic descriptions," in *In Web Services: Modeling, Architecture and Infrastructure workshop in ICEIS 2003.* Citeseer, 2002.

[64] A. Brogi, S. Corfini, and R. Popescu, "Semantics-based composition-oriented discovery of web services," *ACM Transactions on Internet Technology (TOIT)*, vol. 8, no. 4, p. 19, 2008.

[65] M. Schumacher, H. Helin, and H. Schuldt, *CASCOM: Intelligent Service Coordination in the Semantic Web.* Springer, 2008.

[66] A. M. Zaremski and J. M. Wing, "Specification matching of software components," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 6, no. 4, pp. 333–369, 1997.

[67] R. Möller, V. Haarslev, and M. Wessel, "On the scalability of description logic instance retrieval," in *KI 2006: Advances in Artificial Intelligence.* Springer, 2007, pp. 188–201.

[68] K. Sycara, S. Widoff, M. Klusch, and J. Lu, "Larks: Dynamic matchmaking among heterogeneous software agents in cyberspace," *Autonomous agents and multi-agent systems*, vol. 5, no. 2, pp. 173–203, 2002.

[69] M. Klusch, B. Fries, and K. Sycara, "Owls-mx: A hybrid semantic web service matchmaker for owl-s services," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 7, no. 2, pp. 121–133, 2009.

[70] "Rdf: Resource description framework," c.

[71] J. Bailey, F. Bry, T. Furche, and S. Schaffert, "Web and semantic web query languages: A survey," in *Proceedings of the First international conference on Reasoning Web*. Springer-Verlag, 2005, pp. 35–133.

[72] "Sparql," http://www.w3.org/TR/sparql11-overview/.

[73] C. Carpineto and G. Romano, "A survey of automatic query expansion in information retrieval," *ACM Computing Surveys (CSUR)*, vol. 44, no. 1, p. 1, 2012.

[74] L. Ding, T. Finin, A. Joshi, R. Pan, R. S. Cost, Y. Peng, P. Reddivari, V. Doshi, and J. Sachs, "Swoogle: a search and metadata engine for the semantic web," in *Proceedings of the thirteenth ACM international conference on Information and knowledge management*. ACM, 2004, pp. 652–659.

[75] Y. Lei, V. Uren, and E. Motta, "Semsearch: A search engine for the semantic web," in *Managing Knowledge in a World of Networks*. Springer, 2006, pp. 238–245.

[76] J. Zobel and A. Moffat, "Inverted files for text search engines," *ACM computing surveys (CSUR)*, vol. 38, no. 2, p. 6, 2006.

[77] D. Guinard, V. Trifa, F. Mattern, and E. Wilde, "From the internet of things to the web of things: Resource oriented architecture and best practices 1."

[78] H. Wang, C. C. Tan, and Q. Li, "Snoogle: A search engine for pervasive environments," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 21, no. 8, pp. 1188–1202, 2010.

[79] B. Ostermaier, K. Romer, F. Mattern, M. Fahrmair, and W. Kellerer, "A real-time search engine for the web of things," in *Internet of Things (IOT), 2010*. IEEE, 2010, pp. 1–8.

[80] S. Mayer and D. Guinard, "An extensible discovery service for smart things," in *Proceedings of the Second International Workshop on Web of Things*, ser. WoT '11. New York, NY, USA: ACM, 2011, pp. 7:1–7:6. [Online]. Available: http://doi.acm.org/10.1145/1993966.1993976

[81] J. Zobel, A. Moffat, and K. Ramamohanarao, "Inverted files versus signature files for text indexing," *ACM Transactions on Database Systems (TODS)*, vol. 23, no. 4, pp. 453–490, 1998.

[82] G. Cassar, P. Barnaghi, W. Wang, and K. Moessner, "A hybrid semantic matchmaker for iot services," in *Green Computing and Communications (GreenCom), 2012 IEEE International Conference on*. IEEE, 2012, pp. 210–216.

[83] B. Christophe, V. Verdot, and V. Toubiana, "Searching the'web of things'," in *Semantic Computing (ICSC), 2011 Fifth IEEE International Conference on*. IEEE, 2011, pp. 308–315.

[84] D. Pfisterer, K. Romer, D. Bimschas, O. Kleine, R. Mietz, C. Truong, H. Hasemann, A. Kroller, M. Pagel, M. Hauswirth *et al.*, "Spitfire: toward a semantic web of things," *Communications Magazine, IEEE*, vol. 49, no. 11, pp. 40–48, 2011.

[85] L. Yu, "Linked open data," in *A Developer's Guide to the Semantic Web*. Springer, 2011, pp. 409–466.

[86] I. Howitt and J. A. Gutierrez, "Ieee 802.15. 4 low rate-wireless personal area network coexistence issues," in *Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE*, vol. 3. IEEE, 2003, pp. 1481–1486.

[87] S. Hachem, A. Pathak, and V. Issarny, "Probabilistic registration for large-scale mobile participatory sensing," *2014 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, vol. 0, pp. 132–140, 2013.

[88] P.-A. Chirita, S. Costache, W. Nejdl, and S. Handschuh, "P-tag: large scale automatic generation of personalized annotation tags for the web," in *Proceedings of the 16th international conference on World Wide Web*. ACM, 2007, pp. 845–854.

[89] H. Zhuge and X. Luo, "Automatic generation of document semantics for the e-science knowledge grid," *Journal of Systems and Software*, vol. 79, no. 7, pp. 969–983, 2006.

[90] A. Helal, B. Winkler, C. Lee, Y. Kaddoura, L. Ran, C. Giraldo, S. Kuchibhotla, and W. C. Mann, "Enabling Location-Aware Pervasive Computing Applications for the Edlerly," in *PerCom*, 2003, p. 531.

[91] L. D. Souza, P. Spiess, D. Guinard, M. Köhler, S. Karnouskos, and D. Savio, "SOCRADES: A Web Service Based Shop Floor Integration Infrastructure," in *IOT*, 2008, pp. 50–67.

[92] N. Brachet, F. Alizadeh-Shabdiz, J. N. Nelson, and R. K. Jones, "Method and system for selecting and providing a relevant subset of wi-fi location information to a mobile client device so the client device may estimate its position with efficient utilization of resources," Feb. 4 2013, uS Patent App. 13/758,154.

[93] S. B. 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*, vol. 81, no. 5, pp. 785–808, 2008.

[94] W. Galuba and S. Girdzijauskas, "Distributed hash table," in *Encyclopedia of Database Systems*. Springer, 2009, pp. 903–904.

[95] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 4, pp. 149–160, 2001.

[96] D. Chakraborty, A. Joshi, Y. Yesha, and T. Finin, "Toward distributed service discovery in pervasive computing environments," *Mobile Computing, IEEE Transactions on*, vol. 5, no. 2, pp. 97–112, 2006.

[97] M. Schlosser, M. Sintek, S. Decker, and W. Nejdl, "A scalable and ontology-based p2p infrastructure for semantic web services," in *Peer-to-Peer Computing, 2002.(P2P 2002). Proceedings. Second International Conference on*. IEEE, 2002, pp. 104–111.

[98] K. Verma, K. Sivashanmugam, A. Sheth, A. Patil, S. Oundhakar, and J. Miller, "Meteor-s wsdi: A scalable p2p infrastructure of registries for semantic publication and discovery of web services," *Information Technology and Management*, vol. 6, no. 1, pp. 17–39, 2005.

[99] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[100] K. Elgazzar, A. E. Hassan, and P. Martin, "Clustering wsdl documents to bootstrap the discovery of web services," in *Web Services (ICWS), 2010 IEEE International Conference on*. IEEE, 2010, pp. 147–154.

[101] J. Ma, Y. Zhang, and J. He, "Efficiently finding web services using a clustering semantic approach," in *Proceedings of the 2008 international workshop on Context enabled source and service selection, integration and adaptation: organized with the 17th International World Wide Web Conference (WWW 2008)*. ACM, 2008, p. 5.

[102] Q. A. Liang and H. Lam, "Web service matching by ontology instance categorization," in *Services Computing, 2008. SCC'08. IEEE International Conference on*, vol. 1. IEEE, 2008, pp. 202–209.

[103] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM Comput. Surv.*, vol. 31, no. 3, pp. 264–323, Sep. 1999. [Online]. Available: http://doi.acm.org/10.1145/331499.331504

[104] R. Nayak and B. Lee, "Web service discovery with additional semantics and clustering," in *Web Intelligence, IEEE/WIC/ACM International Conference on.* IEEE, 2007, pp. 555–558.

[105] S. Rajagopal and S. Thamarai Selvi, "Semantic grid service discovery approach using clustering of service ontologies," in *TENCON 2006. 2006 IEEE Region 10 Conference.* IEEE, 2006, pp. 1–4.

[106] W. A. Wilson, "On quasi-metric spaces," *American Journal of Mathematics*, vol. 53, no. 3, pp. pp. 675–684, 1931. [Online]. Available: http://www.jstor.org/stable/2371174

[107] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM computing surveys (CSUR)*, vol. 31, no. 3, pp. 264–323, 1999.

[108] M. Ester, H.-P. Kriegel, J. Sander, M. Wimmer, and X. Xu, "Incremental clustering for mining in a data warehousing environment," in *Proceedings of the 24rd International Conference on Very Large Data Bases.* Morgan Kaufmann Publishers Inc., 1998, pp. 323–333.

[109] S. Lühr and M. Lazarescu, "Incremental clustering of dynamic data streams using connectivity based representative points," *Data & Knowledge Engineering*, vol. 68, no. 1, pp. 1–27, 2009.

[110] F. Can, "Incremental clustering for dynamic information processing," *ACM Transactions on Information Systems (TOIS)*, vol. 11, no. 2, pp. 143–164, 1993.

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

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

[113] D. H. Fisher, "Knowledge acquisition via incremental conceptual clustering," *Machine learning*, vol. 2, no. 2, pp. 139–172, 1987.

[114] T. Zhang, R. Ramakrishnan, and M. Livny, "Birch: an efficient data clustering method for very large databases," in *ACM SIGMOD Record*, vol. 25, no. 2. ACM, 1996, pp. 103–114.

[115] I. Navas, I. Sanz, J. F. Aldana, and R. Berlanga, "Automatic generation of semantic fields for resource discovery in the semantic web," in *Database and Expert Systems Applications.* Springer, 2005, pp. 706–715.

[116] C. E. S. Pires, R. M. L. Santiago, A. C. Salgado, Z. Kedad, and M. Bouzeghoub, "Ontology-based clustering in a peer data management system," *International Journal of Distributed Systems and Technologies (IJDST)*, vol. 3, no. 2, pp. 1–21, 2012.

[117] J. E. Dennis Jr and R. B. Schnabel, *Numerical methods for unconstrained optimization and nonlinear equations.* Siam, 1996, vol. 16.

[118] C. Gardiner, *Stochastic methods.* Springer-Verlag, Berlin–Heidelberg–New York–Tokyo, 1985.

[119] M. Compton and P. Barnaghi, "The SSN ontology of the W3C semantic sensor network incubator group," *J. Web Sem.*, pp. 25–32, 2012.

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

[121] M. Paolucci, T. Kawamura, T. R. Payne, and K. P. Sycara, "Semantic Matching of Web Services Capabilities," in *International Semantic Web Conference*, 2002, pp. 333–347.

[122] M. Paolucci, K. P. Sycara, T. Nishimura, and N. Srinivasan, "Using DAML-S for P2P Discovery," in *ICWS*, 2003, pp. 203–207.

[123] G. Cassar, P. M. Barnaghi, W. Wang, and K. Moessner, "A hybrid semantic matchmaker for iot services," in *GreenCom*, 2012, pp. 210–216.

Appendix A

Middleware prototype implementation

We present in this appendix, some elements about the implementation of the semantic-based middleware. The implementation integrates the mechanisms presented in this thesis in terms of service registration and search for services in a static context (Chapter 4) and in a dynamic context (Chapter 5).

The reminder of the appendix is structured as follows. First, we present the implementation environment in section A.1. Then, we describe the used data set during evaluations in section A.2. In section A.3, we present the different simulation components for graph generation, service distribution over lowest level nodes and service mobility. Finally, in section A.4, we detail the implementation of the different functions within the prototype.

A.1 Description of the implementation environment

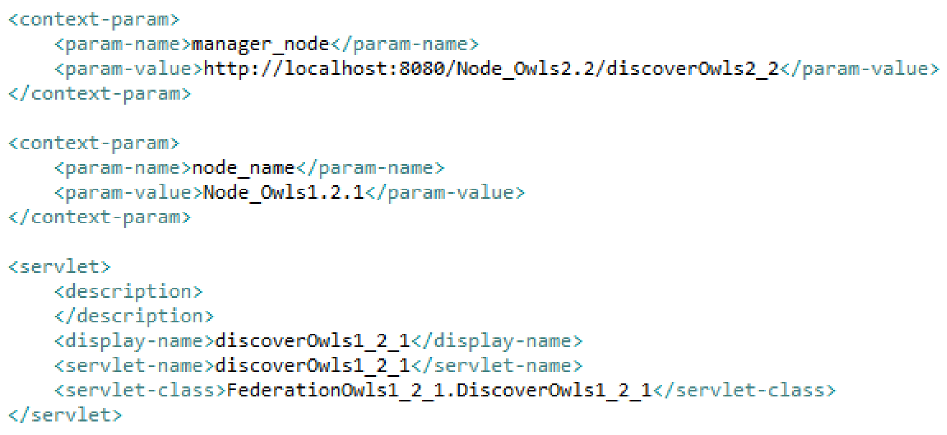
The middleware is implemented using Java 1.7. Each semantic gateway represented by a node in the tree hierarchy is embodied in a Java Web application deployed in a servlet container. The servlet container is a Tomcat Web server [1]. This Web application embeds the different functional blocs described in chapter 3, section 3.2.

The ontology and the semantic service repositories contain ontologies and OWL-S based semantic service descriptions from OWL-S service retrieval Test Collection (OWL-S TC v3.0 [2]) data set (see section A.2). Reading and processing these descriptions is performed using the OWL API [3] coupled with Pellet [4], a semantic engine capable of reasoning on OWL ontologies.

Interconnection of nodes in the tree hierarchy is achieved by attaching configuration parameters to each node. Amongst these parameters, one is an accessible endpoint of the manager (i.e., parent node) of a given node. As our nodes are embodied in Web applications, this accessible endpoint is a URL mapped on a piece of code able to process incoming requests. The following figure A.1 shows an extract of a *web.xml* document used to configure a Web application. We note that a node without the "manager" parameter is supposed to be the top node of the

1. Tomcat:http://tomcat.apache.org/
2. http://projects.semwebcentral.org/projects/owls-tc/
3. OWLAPI:http://owlapi.sourceforge.net/
4. Pellet:http://clarkparsia.com/pellet/

hierarchy. At initialization, a node is configured with the values of these parameters and becomes capable of contacting its manager.



```
<context-param>
    <param-name>manager_node</param-name>
    <param-value>http://localhost:8080/Node_Owls2.2/discoverOwls2_2</param-value>
</context-param>

<context-param>
    <param-name>node_name</param-name>
    <param-value>Node_Owls1.2.1</param-value>
</context-param>

<servlet>
    <description>
    </description>
    <display-name>discoverOwls1_2_1</display-name>
    <servlet-name>discoverOwls1_2_1</servlet-name>
    <servlet-class>FederationOwls1_2_1.DiscoverOwls1_2_1</servlet-class>
</servlet>
```

Figure A.1: Extract of a web.xml document used to configure node 1.2.1 having node 2.1 as a manager.

A.2 Data set description

As mentioned in the state of the art chapter (see section 2.2.1.2), a number of efforts to represent IoT resources and services using Semantic Web technologies have been made ([119], [120]). Furthermore, many existing works on semantic service matchmaking [121] and service search [122] are based on the OWL-S model for providing both rich, expressive descriptions and well-defined semantics. However, these models are heavyweight and complex to be directly applied to IoT resources. As a consequence, in [123], they adapted and refined the OWL-S model to their IoT model to make a trade off between being lightweight and providing sufficient modeling constructs for representing IoT services.

Although many efforts have been made to model IoT resources as we mentioned in 2.2.2, public real and rich data sets about IoT semantic service descriptions that can be used in experiments are still not very common today. Thus, we utilized for our experimental evaluation the widely used OWL-S service retrieval Test Collection (OWL-S TC v3.0 [5]). Besides, we should mention that the objective of our work is not to propose a semantic model for IoT service descriptions, but to show how semantic service descriptions can be used to search IoT services in an efficient way, and independently from the used service descriptions model.

The OWL-S TC data set comprises a set of ontologies derived from 7 different categories (see table A.1), a set of 1007 OWL-S different service descriptions and an accompanying set of 29 OWL-S sample requests. Based on this data set, we generated a set of approximately 45000 services to perform exhaustive tests, by creating copies of existing service descriptions. We also picked up service descriptions from the original data set and added them to the original 29 requests to create a sample of 100 requests.

5. http://projects.semwebcentral.org/projects/owls-tc/

Table A.1: Service categories in OWL-S TC

Category number	Category Name
C_1	communication
C_2	medical
C_3	weapon
C_4	food
C_5	economy
C_6	travel
C_7	education

A.3 Simulation components

We present in this part components which are not part of the middleware architecture. However, they are used respectively to generate the graph of nodes, distribute services over lowest level nodes and simulate service mobility.

NodeGenerator generates the graph of nodes that represents the semantic gateways. The node generator takes as inputs the different parameters that characterize the graph of the semantic gateways: the number of levels and the number of nodes at each level. NodeGenerator also generates automatically the *web.xml* configuration file of each node. This file contains the information related to the node : its name, address and also (if applicable) its parent node and its children nodes. The generated nodes communicate together based on these configuration files. Each node contains registries to host ontologies and semantic service descriptions. Besides, it implements OWL API for working with OWL ontologies.

NodeFilling fills lowest level nodes with services picked randomly from service categories of the data set (see table A.1). Service categories are represented in lowest level nodes and follow a non uniform distribution in order to reflect real use cases. Each node represents a subset of service categories. The different cases of service distributions over lowest level nodes considered during the experiments are detailed in chapters 4 and 5 respectively in 4.5 and 5.4.

ServiceMobility simulates a process of service arrivals and departures in each lowest level node of the tree hierarchy. 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. 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 an arrival rate η. We consider systems in which the request arrival process is faster than the service arrival process (i.e., $\eta >> \lambda$) for a given period of time T.

A.4 Prototype functions implementation

This section details the different implemented components within each servlet. Each node implements 3 main components:

NodeEntry: This component is the entry point to the node. All requests sent to the node are first received by this component before being processed. **NodeEntry** receives either requests about service search sent by a user or forwarded by a node, or notifications about service arrivals and departures sent by physical gateways upon the detection of device mobility. These requests are simulated by the **ServiceMobility** component.

The NodeEntry component extracts parameters from the request to determine whether it is a service search request or a service insertion/deletion request due to a new service arrival/departure. In the latter case, the service is sent to the **ServiceRegistry** component to be added or deleted from service registry. In the former case, the request is sent to the **ServiceSearch** component to find matching services in the semantic gateway .

ServiceRegistry: As mentioned above, this component is responsible for insertion/deletion of service descriptions resulting from service arrival and departure. It is also responsible for the following tasks: organizing services into clusters, aggregating clusters, creating and updating routing tables. Finally, it optimizes service repartition within clusters of a node through cluster merging and splitting mechanisms. Based on the type of service mobility (i.e., service arrival or departure), **ServiceArrivalManagement()** method is called to insert the new service into the node following the different cases detailed in 5.3.1.1 or **ServiceDepartureManagement()** method is called to delete the service from the node following the different cases detailed in 5.3.1.2. Both methods lead to incremental clusters creation and encompass a set of sub-methods:

- **ServiceMatching()**: To extract concepts from service and cluster representatives and match them together based on the quasi-metric Q (see definition 6).
- **RoutingTableUpdate()**: To aggregate cluster content by selecting a cluster representative and a radius and update the routing table if needed.
- **ClusteringOptimizing()**: This method is called at the end of a service insertion/deletion treatment to check whether it is needed to optimize the clustering. For that, the number of clusters and the size of the biggest cluster in terms of number of services are compared with the optimal model (see definition 2 in chapter 5) which is calculated via **GetPerfectModel()** method. Based on this comparison, **MergeCluster()** method or **SplitCluster()** method are called to optimize the clustering.

ServiceSearch:This component receives search requests, searches for registered services that match the request and forwards the request to other nodes if needed. A number of methods are called in this component:

- **FindClusterInCurrentLevel()**: Extracts concepts from request and cluster representatives in the routing table and matches them together based on the quasi-metric Q (see definition 6). It compares the matching results to the decision threshold (see theorem 3) of the considered cluster to decide whether it is pertinent for the request to visit the cluster and further match with its elements.
- **FindServices()**: Matches the request with the elements of a cluster and finds services that match the request based on either the matching threshold in the case of a lowest level node or the decision threshold of the represented cluster (i.e., the decision threshold of the cluster in a child node which is represented by a representative in the parent node) in the case of a higher level node.
- **FindNodesToForwardRequest()**: It is called at a higher level node to determine the nodes to which the request will be forwarded.

Intergiciel Sémantique pour la recherche des services de l'Internet des Objets

Sameh BEN FREDJ

RESUME : Avec l'avènement de l'Internet des Objets, nous sommes confrontés à une prolifération des appareils connectés répartis sur des emplacements physiques, appelés des espaces intelligents et qui offrent des services de l'Internet des Objets. La découverte simple et transparente de ces services est cruciale pour le succès de l'Internet des Objets. Les caractéristiques des services de l'Internet des objets, tels que leur nombre, leur hétérogénéité et leur dynamicité induite par la mobilité des appareils connectés, rendent leur découverte difficile. Dans cette thèse, nous proposons une architecture de système et ses mécanismes associés pour permettre une découverte efficace et scalable des services de l'Internet des Objets, en se basant sur le Web Sémantique et en supportant des contextes dynamiques. Notre approche repose sur les passerelles distribuées qui intègrent des mécanismes de regroupement, d'agrégation de l'information et de routage sémantique.

MOTS-CLEFS : Internet des Objets, intergiciel, services de l'Internet des Objets, Web Sémantique, la recherche des services, extensibilité, mobilité des services.

ABSTRACT : With the advent of the Internet of Things (IoT), we are facing a proliferation of connected devices distributed over physical locations, so called smart spaces and offering IoT services. Enabling an easy and seamless discovery of these IoT services is crucial for the success of the Internet of Things. The characteristics of IoT services, such as their sheer number, their heterogeneity and their dynamicity induced by the mobility of the related devices, make discovering them a challenge. In this thesis, we propose a system architecture and the associated mechanisms to enable efficient and scalable semantic-based IoT service discovery supporting dynamic contexts. Our approach relies on distributed semantic gateways that embed clustering, information aggregation and semantic routing mechanisms.

KEY-WORDS : Internet of Things, middleware, IoT services, Semantic Web, service search, scalability, service mobility.