REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mohamed Khider – BISKRA
Faculté des Sciences Exactes, des Sciences de la Nature et de la Vie
Département d'informatique

N° d'ordre :.....
Série :.....

Thèse

Présentée en vue de l'obtention du diplôme de docteur en sciences

Option : **Informatique**

Titre

Un middleware pour la gestion de la sensibilité au contexte dans les systèmes de l'Internet des Objets

Par :
M. KERTIOU Ismail

Soutenue le :

Devant le jury :

KAZAR Okba	Professeur	Université de Biskra	Président
BENHARZALLAH Saber	Professeur	Université de Batna 2	Rapporteur
KAHLOUL Laid	Professeur	Université de Biskra	Co-Rapporteur
BENMERZOUG Djamel	Professeur	Université de Constantine 2	Examinateur
MOUMEN Hamouma	MCA	Université de Batna 2	Examinateur
REZEG Khaled	Professeur	Université de Biskra	Examinateur

Année universitaire : 2020 – 2021

Dédicaces

Tout d'abord, je veux rendre grâce à Dieu, le Clément et le Très Miséricordieux pour son amour éternel. C'est ainsi que je dédie cette thèse à:

ma mère pour sa tendresse, patience et encouragement,
la mémoire de mon père, Que Dieu vous bénisse et fasse de vous le paradis,
ma chère épouse pour son soutien,
mes très chers enfants, que Dieu les protège,
mes sœurs pour leurs conseils,
tous les membres de ma famille,
tous mes amis,
tous ceux que j'aime.

Remerciements

En premier lieu, je remercie le bon Dieu de m'avoir donné la force, la santé, la volonté et la patience nécessaire pour achever ce travail de thèse.

C'est avec un énorme plaisir que je remercie aujourd'hui toutes ceux qui m'ont soutenue durant la préparation de cette thèse.

Je tiens à remercier très sincèrement mon directeur de thèse Pr.Benharzallah Saber, professeur à l'Université de Batna 2, pour l'aide qu'il a fournie et les connaissances qu'il a su me transmettre. Je le remercie aussi pour sa disponibilité et la qualité de ses conseils. Qu'il trouve ici l'expression de ma très grande gratitude. Je remercie également Pr. Kahloul Laid, professeur à l'Université de Biskra et co-encadreur pour ses conseils pertinents et son soutien.

Je remercie vivement Monsieur le Pr.Kazar Okba, de m'avoir fait l'honneur d'être président de mon jury.

Je tiens également à remercier Mr. Benmerzoug Djamel, Professeur à l'université de Constantine 2, Mr. Moumen Hamouma, Maître de conférence à l'université de Batna 2 et Mr. Rezeg Khaled, Professeur à l'université de Biskra pour l'intérêt qu'ils ont porté à mes travaux en examinant ce mémoire et pour l'honneur qu'ils me font en participant à ce jury.

Ma gratitude s'adresse également au Dr. Beggas Mounir, Docteur à l'Université d'El Oued et au Dr. Laouid Abdelkader, Docteur à l'Université d'El Oued; vous m'avez guidé avec un enthousiasme permanent; vous m'avez transmis les notions nécessaires et indispensables à ma recherche.

Je désire aussi remercier Dr. Abasse Massaoud, Docteur à l'Université d'El Oued, et Madame Gaia Sana, Enseignante à l'Université d'Eloued; pour la richesse et la pertinence de vos remarques, tant sur la forme que sur le fond, ont contribué à améliorer de manière significative le document que je vous soumets aujourd'hui.

Enfin, je remercie tous ceux qui m'ont soutenu, encouragé et m'ont donné l'envie de mener à terme ce travail.

Résumé

L'intégration du paradigme de l'Internet des objets (IdO) dans les services Web et le cloud computing nous permet de gérer des milliers de capteurs conjointement avec leurs données. À cet égard, le modèle "détection en tant que service" (Sensing as a Service) a récemment émergé. Les données générées par ces capteurs peuvent être réutilisées par différents utilisateurs et applications à l'aide des solutions middleware IdO. En fait, l'énorme nombre de capteurs disponibles dans l'environnement IdO, les ressources limitées et le facteur coût rendent impossible de collecter les données de tous les capteurs disponibles. Dans cette situation, le défi crucial est de savoir comment rechercher et sélectionner efficacement les meilleurs capteurs en fonction des besoins des utilisateurs dans un délai raisonnable. Notre objectif est de proposer une méthode efficace et sensible au contexte, pour rechercher et sélectionner les meilleurs capteurs afin d'aider les utilisateurs à acquérir les informations souhaitées.

Dans ce travail, nous proposons une méthode sensible au contexte adaptable aux différents middlewares IdO. La méthode proposée permet de réduire considérablement le temps de recherche et de sélection de capteurs. Nous exploitons la puissance de l'opérateur Skyline dynamique dans le domaine de la prise de décision multicritères, pour réduire l'espace de recherche. Le but est d'améliorer l'efficacité de la sensibilité au contexte et de sélectionner les meilleurs capteurs selon les besoins des utilisateurs. L'architecture adoptée dans cette proposition est composée de plusieurs passerelles réparties dans le réseau et connectées à un serveur. Les capteurs sont connectés aux passerelles gérant chacune son propre réseau local. Les passerelles doivent également répondre localement aux requêtes des utilisateurs. Par la suite, le serveur regroupera les résultats de toutes les passerelles et donnera la réponse finale. De plus, notre système permet de classer les capteurs sélectionnés par l'opérateur Skyline dynamique en fonction des exigences et des poids imposés par l'utilisateur. Pour classer les différents capteurs Skyline sélectionnés, nous utilisons la distance euclidienne. L'expérimentation montre l'efficacité de notre méthode par rapport à celles existantes.

Mots Clés: Internet des objets, Middleware, Recherche de capteur, Sensibilité au contexte, Skyline dynamique, Prise de décision multi-critères.

Abstract

The integration of the Internet of Things (IoT) paradigm into web services and cloud computing allows us to handle thousands of sensors together with their data. In this regard, "Sensing as a service" model has recently emerged. The data generated by its sensors can be reused by different users and applications within IoT middleware solutions. Due to the huge number of sensors available in the IoT environment, limited resources and cost factors, it is not possible to collect data from all available sensors. In this situation, the crucial challenge is how to effectively search and select the best sensors according to the users' requirements within a reasonable delay. Thus, our goal is to provide an efficient method for finding and selecting the best sensors to help users to acquire the desired information.

In this work, we propose a context-aware method that can be adopted by different IoT middlewares. The proposed method reduces significantly the time for searching and selecting sensors. We exploit the power of a dynamic Skyline operator for multi-criteria decision making and for search space reducing. The goal is to improve the efficiency of context-awareness and to select the best sensors according to users' needs. The architecture adopted in this proposal is composed by several gateways distributed in the network and connected to a server. Each gateway manage its own local network of sensors. Getways must also answer user requests locally. Then, the server collects the results of all gateways and gives the final answer. In addition, our system enables the classification of selected sensors according to users requirements and weights. To classify the various Skyline selected sensors, we use the Euclidean distance. The experimentation shows the efficiency of our method compared to existing ones.

Keywords: Internet of things, Middleware, Sensor search, Context Awareness, Dynamic skyline, Multi-criteria decision making.

الملخص

يسمح لنا دمج نموذج إنترنت الأشياء في خدمات الويب والحوسبة السحابية التعامل مع آلاف أجهزة الاستشعار وبياناتها. في هذا الصدد، ظهر مؤخرًا نموذج الاستشعار كخدمة حيث يمكن إعادة استخدام البيانات التي تم انتاجها بواسطة هذه المستشعرات بواسطة مستخدمين وتطبيقات مختلفة تستخدم حلول البرامج الوسيطة لإنترنت الأشياء. نظرًا للعدد الهائل من أجهزة الاستشعار المتوفرة في بيئة إنترنت الأشياء والموارد المحدودة وعوامل التكلفة، لا يمكن جمع البيانات من جميع أجهزة الاستشعار المتاحة. في هذه الحالة، يتمثل التحدي في كيفية البحث الفعال واختيار أفضل أجهزة الاستشعار وفقا لاحتياجات المستخدمين في إطار زمني معقول. وبالتالي، فإن هدفنا هو توفير طريقة فعالة للعثور على أفضل أجهزة الاستشعار واختيارها لمساعدة المستخدمين في الحصول على المعلومات المطلوبة.

في هذا العمل، نقترح طريقة حساسة للسياق يمكن اعتمادها من قبل مختلف البرامج الوسيطة لإنترنت الأشياء لتصميم حل ذي صلة يمكن أن يوفر مستوى عالٌ من الدقة ويقلل إلى أدنى حد من وقت البحث واختيار أجهزة الاستشعار. سنقوم باستغلال قوة مشغل سكايلين (Skyline)الديناميكي في مجال اتخاذ القرار متعدد المعايير، لتقليل مساحة البحث بهدف تحسين كفاءة حساسية السياق واختيار أفضل أجهزة الاستشعار حسب احتياجات المستخدم. تتكون البنية المعتمدة في هذا الاقتراح من عدة بوابات موزعة في جميع أنحاء الشبكة ومتصلة بخادم، ويتم توصيل المستشعرات بالبوابات التي تدير كل منها شبكتها المحلية، والتي يجب أن تجيب محليا على طلبات المستخدمين. بعد ذلك، يقوم الخادم بجمع ودمج نتائج جميع البوابات ويعطي الإجابة النهائية. بالإضافة إلى ذلك، يسمح نظامنا بترتيب المستشعرات التي يختارها مشغل سكايلين الديناميكي وفقا للمتطلبات والأوزان التي يفرضها المستخدم. من اجل ترتيب المستشعرات المتحصل عليها نستخدم المسافة الإقليدية. مختلف التجارب المطبقة تظهر فعالية طريقتنا مقارنة بالطرق الحالية.

كلمات مفتاحية: إنترنت الأشياء ، البرامج الوسيطة ، بحث على أجهزة الاستشعار، حساسية السياق ، سكايلين (Skyline) الديناميكي، اتخاذ القرار متعدد المعايير .

Table des matières

Liste des figures

Liste des tableaux

Liste des algorithmes

Introduction

Au cours des dernières années, l'Internet a connu une énorme évolution, elle permet de connecter des milliards de dispositifs à travers le monde. Ces dispositifs ont des tailles, capacités et puissances de traitements différents et hébergent différents types d'applications. Ainsi, l'Internet classique se développe vers l'Internet intelligent, appelé internet des objets (Internet of Things). Le terme internet des objets a été inventé pour la première fois en 1999 par Kevin Ashton dans le contexte de la gestion de la chaîne d'approvisionnement [8]. Selon Guillemin [46], on peut définir l'Internet des objets comme étant l'infrastructure qui permet aux personnes et aux objets d'être connectés à tout moment, en tout lieu, avec n'importe quoi et n'importe qui, en utilisant n'importe quel réseau et n'importe quel service. L'Internet des objets (IdO) permet d'interconnecter des personnes et des objets physiques, par exemple des capteurs, des smartphones ou des tablettes via internet. Il permet également le développement de nouveaux types de services et d'applications tels que la surveillance de la sécurité publique intérieure [131], les soins à distance [97, 130], le traitement de la pollution de l'environnement [108, 36], la gestion des villes intelligentes et big data [112, 60]. Selon le site web de statistiques Statista [95], le nombre d'objets connectés dans le monde a dépassé 20 milliards en 2017 et il est estimé à atteindre 75 milliards en 2025. Un nombre énorme de capteurs génèrent en permanence un grand volume de données. De plus, l'IdO permet aux propriétaires de capteurs de publier leurs données de capteurs gratuitement ou en payant des frais (détection en tant que service: Sensing as a service). Ces capteurs génèrent des données qui peuvent être partagées par plusieurs applications via des solutions middleware IdO [94]. Les middlewares IdO aident les utilisateurs à collecter des données à partir d'un grand nombre de capteurs pour différentes applications [53].

Les systèmes sensibles au contexte sont des solutions permettant de superviser la façon dont les utilisateurs interagissent avec l'environnement ubiquitaire et pour automatiser les actions répétitives des utilisateurs. La sensibilité au contexte est devenue plus populaire avec l'introduction du terme «informatique ubiquitaire» par Mark Weiser dans son article «l'ordinateur pour le 21ème siècle» en 1991 [127]. Ensuite, le terme «sensibilité au contexte» a été utilisé pour la première fois par Schilit et

Theimer en 1994 [113]. Depuis cette date, la notion de sensibilité au contexte est une caractéristique essentielle dans le développement des systèmes informatiques ubiquitaires et omniprésents. Les applications sensibles aux contextes sont évoluées des applications de bureau, des applications Web, de l'informatique mobile, de l'informatique ubiquitaire et omniprésente à l'Internet des objets au cours de la dernière décennie. Selon Dey [34], le contexte est toute information qui peut être utilisée pour caractériser la situation d'une entité. Une entité est une personne, un lieu ou un objet considéré comme pertinent pour l'interaction entre un utilisateur et une application, y compris l'utilisateur et les applications elles-mêmes. Les systèmes sensibles au contexte sont constitués de plusieurs composants distribués tels que des capteurs, des actionneurs, des dépôts d'informations contextuelles, des processeurs d'informations contextuelles, etc. Aujourd'hui, il est largement admis que, pour réduire la complexité des applications sensibles au contexte et encourager leur réutilisation, des composants d'infrastructure supplémentaires sont souhaitables [107]. Généralement, un middleware masque les complexités du système ou du matériel, permettant au développeur d'applications de concentrer tous ses efforts sur la tâche à résoudre, sans la distraction de problèmes orthogonaux au niveau du système ou du matériel. Issarny [55] définit un middleware comme une couche logicielle qui se situe entre le système d'exploitation et l'application et fournit des solutions réutilisables bien connues aux problèmes fréquemment rencontrés comme l'hétérogénéité, l'interopérabilité, la sécurité, la fiabilité. Un middleware fournit une couche logicielle entre les applications, le système d'exploitation et les couches de communication réseau, ce qui facilite et coordonne certains aspects du traitement coopératif.

Dans l'utilisation classique des réseaux de capteurs sans fil, les capteurs souffrent particulièrement d'un manque de localisation physique, d'un approvisionnement énergétique limité et d'un faible niveau de sécurité [66, 110, 67, 76]. L'intégration des capteurs sans fil dans le contexte de l'IdO permet de gagner certains avantages, mais avec certaines limitations [80]. De plus, le modèle capteur en tant que service (SaaS: Sensing as a service) vu une croissance considérable et les données générées par ces capteurs peuvent être réutilisées par différents utilisateurs et applications au sein des solutions middlewares dans l'IdO. Par conséquent, la sélection des capteurs dans le modèle SaaS va être l'un des défis les plus difficiles des middlewares dans l'IdO. Cependant, tous les middlewares existants possèdent une couche responsable de l'acquisition des données à partir des capteurs. En raison de la grande échelle des réseaux IdO, des ressources limitées et de facteur coût [84, 58, 6], il n'est pas possible de collecter les données de tous les capteurs disponibles. Dans cette situation, le défi essentiel des utilisateurs est de savoir comment rechercher et sélectionner les capteurs appropriés pour résoudre leurs problèmes dans un délai raisonnable. Ainsi, notre objectif est de proposer une méthode efficace pour sélectionner les meilleurs capteurs afin d'aider les utilisateurs à acquérir les informations souhaitées.

Zhou et al. [132] ont classé les méthodes existantes de recherche et de sélection de capteurs en deux catégories: Les méthodes basées sur le contenu et les méthodes sensibles au contexte. Les méthodes de la première catégorie trouvent des capteurs qui génèrent certaines valeurs. Mais le grand nombre de capteurs rend impossible le traitement de toutes les données collectées par ces capteurs. Tandis que les méthodes de la deuxième catégorie permettent de sélectionner des capteurs possédant certaines propriétés contextuelles. Les premières méthodes sensibles au contexte telles que Snoogle et Global Sensor Networks (GSN) [126, 3] sont basées sur une sélection manuelle et prennent en compte un nombre limité de propriétés contextuelles. En fait, le contexte peut inclure des informations relatives à la qualité des capteurs telles que la fiabilité, précision, coût et autres.

Pour résoudre le problème mentionné ci-dessus et publier les informations des capteurs dans un format uniforme, les dernières méthodes sensibles au contexte ont modélisé les capteurs avec l'ontologie du réseau de capteurs sémantique (ontologie SSN) [32]. Une technique d'indexation basée sur la distance euclidienne pondérée a été adoptée par Perera pour mesurer la similitude entre les capteurs et les préférences de l'utilisateur [94]. Ebrahimi et al. [37] ont proposé un algorithme métaheuristique pour regrouper les capteurs avec des informations de contexte similaires en un seul cluster. Suite à la requête soumise et en fonction des choix de l'utilisateur, le cluster contenant les capteurs les plus proches aux besoins de l'utilisateur est sélectionné. Nunes et al. [82] ont proposé un algorithme de sélection par élimination pour rechercher et découvrir des ressources dans des environnements IdO. Au début, toutes les options sont classées par l'algorithme TOPSIS. Par la suite, un algorithme de tri rapide non dominé est exécuté pour assurer la sélection des meilleures options. Dans le travail de Hsu et al. [53], une architecture de service de détection a été proposée comprenant une méthode de recherche et de sélection de capteur pour sélectionner efficacement les capteurs pertinents parmi un grand nombre de capteurs disponibles. Il faut également noter que plusieurs approches utilisent des méthodes heuristiques pour filtrer et sélectionner les capteurs en question. Cependant, le temps de recherche et de sélection est considérablement long pour la plupart des méthodes existantes. De plus, la plupart entre elles ne prenant pas en compte la nature distribuée des systèmes IdO.

Récemment, la filtration des opérateurs Skyline a été adoptée pour réduire l'espace de recherche dans le but de sélectionner certaines entités en fonction de certaines propriétés. Le Skyline a été largement utilisé dans de nombreuses applications de prise de décision multicritères telles que la recherche de systèmes de bases de données [11], la composition de services Web [128] et le routage dans les réseaux Ad Hoc sans fil [4]. Dans ce travail, nous nous concentrons sur le principe du Skyline dans le processus de recherche et de sélection contextuelle des meilleurs capteurs. Les capteurs non Skyline étant dominés par les capteurs Skyline, les capteurs Skyline ont de meilleures propriétés

contextuelles.

Cette thèse propose une méthode qui peut être adoptée par différents middlewares IdO pour concevoir une solution pertinente qui peut offrir un haut niveau de précision et réduire au minimum le temps de recherche et de sélection. La nouveauté de cette contribution est de savoir comment utiliser le principe de filtration Skyline pour sélectionner les capteurs dominés et les classer de la même manière que celle proposée par Perera dans [94]. Le but de l'exploitation du principe Skyline est de sélectionner le meilleur capteur avec un haut niveau de précision. Cependant, comme le Skyline est trop lent, nous proposons une architecture distribuée dans laquelle le filtrage Skyline sera exécuté à deux niveaux: au niveau des passerelles et au niveau du serveur. Les contributions de ce travail peuvent être résumées comme suit:

- Nous proposons une méthode sensible au contexte qui peut être adoptée par différents middlewares IdO. La méthode proposée permet de concevoir une solution pertinente qui peut offrir un haut niveau de précision et réduire considérablement le temps de recherche et de sélection des capteurs.
- Nous exploitons la puissance de l'opérateur Skyline dynamique dans le domaine de la prise de décision multicritères pour réduire l'espace de recherche. Le but est d'améliorer l'efficacité de la recherche contextuelle et de sélectionner les meilleurs capteurs en fonction des besoins des utilisateurs. En particulier, nous nous concentrons sur le Skyline dynamique dans le processus de sélection des capteurs sensibles au contexte. Dans ce dernier processus, les capteurs non Skyline dynamique sont dominés par des capteurs Skyline dynamique. Autrement dit, les capteurs Skyline dynamique ont des meilleures propriétés contextuelles par rapport aux requêtes des utilisateurs. Selon la requête des utilisateurs et le nombre de capteurs disponibles, le Skyline dynamique permet de donner le nombre idéal de capteurs (les meilleurs capteurs) pour répondre aux utilisateurs.
- Pour assurer la nature parallèle des architectures IdO, nous adoptons l'architecture présentée dans [53]. Dans notre proposition, le système est composé de plusieurs passerelles (gateways) réparties dans le réseau avec un serveur. Les capteurs sont connectés aux passerelles. Chaque passerelle dispose son propre réseau local, et elle doit répondre localement aux requêtes des utilisateurs. Ensuite, le serveur va regrouper les résultats de toutes les passerelles pour remettre une réponse finale. En utilisant notre architecture, nous profitons de l'énorme capacité de calcul et de stockage au niveau des passerelles pour éviter la duplication des informations au niveau du serveur et la mise à jour permanente entre les passerelles et le serveur.
- Dans le cas où le nombre de capteurs Skyline est très élevé, notre système permet de classer les capteurs sélectionnés en fonction des exigences et des poids imposés par

l'utilisateur. Pour classer les différents capteurs Skyline sélectionnés, nous utilisons la distance euclidienne.

- Nous avons évalué notre méthode proposée en utilisant une combinaison de données réelles et synthétiques, en comparant notre proposition avec d'autres travaux voisins. Le but est de montrer la précision et l'efficacité de notre démarche.

Dans le cadre de ce travail de thèse, nous avons renforcé notre proposition avec la publication scientifique suivante:

Ismail Kertiou, Saber Benharzallah, Laid Kahloul, Mounir Beggas, Reinhardt Euler, Abdelkader Laouid et Ahcène Bounceur « A dynamic skyline technique for a context-aware selection of the best sensors in an IoT architecture » L'article est accepté dans le journal : Ad Hoc Networks Volume 81, December 2018, Pages 183-196 (le journal est de classe A: indexed isi thomson IF= 3.643).

Cette thèse comporte cinq chapitres :

- **Chapitre 1**: Nous présentons un aperçu global sur le concept de l'internet des objets. Au début, nous citons les phases importantes qui ont contribué au développement de la technologie de l'Internet des objets. Ensuite, pour clarifier le concept de l'IdO, nous donnons un ensemble de définitions montrant les différentes visions de ce concept. Après, nous définissons les différentes couches architecturales de l'IdO. Nous présentons ainsi les différents domaines d'application de l'IdO. Enfin, nous expliquons les problèmes de recherche les plus importants qui doivent être traités pour répondre aux exigences caractérisants l'IdO.

- **Chapitre 2**: Dans ce chapitre, nous détaillons le concept de sensibilité au contexte. Au début, nous clarifions la notion des informations de contexte dans le domaine informatique. Ensuite, nous donnons un ensemble de définitions montrant les différentes visions de la sensibilité au contexte. Après, nous présentons les différentes couches architecturales des systèmes sensibles au contexte. Par la suite, nous citons les célébrés middlewares sensibles aux contextes les plus représentatifs et populaires pour l'IdO. Nous expliquons ainsi la notion d'aide à la décision multicritère. Enfin, nous terminons par la définition de la problématique de cette thèse.

- **Chapitre 3**: Nous présentons un état de l'art sur les travaux intéressés aux recherche et sélection des meilleurs capteurs répondant aux besoins des applications. Ainsi, nous critiquons leurs limites et manques afin de proposer des solutions efficaces.

- **Chapitre 4**: Nous présentons notre méthode de recherche et de sélection sensible au contexte des capteurs selon les besoins des utilisateurs. Nous avons basé sur les informations de contexte des capteurs pour rechercher et sélectionner les meilleurs capteurs pour une requête utilisateur. Nous discutons d'abord les différents concepts de la technique proposée. Par la suite, nous expliquons les différents composants de

l'architecture proposée. Après, nous détaillons les déférentes phases de recherche et sélection des meilleurs capteurs selon les besoins des utilisateurs. Enfin, nous fournissons un exemple pour illustrer et justifier les différentes étapes du processus proposé.

- **Chapitre 5**: Nous implémentons un prototype pour l'architecture proposée et nous définissons les jeux de données. Nous également discutons les résultats obtenus qui sont extraits de différentes études de cas. Enfin, nous comparons les résultats obtenus avec les algorithmes CASSARAM, AntClust et E-S à fin de montrer l'efficacité, la robustesse et l'influence de la technique proposée.
- **Conclusion**: Nous terminons ce travail de thèse par une conclusion générale mettant en évidence une synthèse globale, les résultats acquis et les perspectives de recherches.

Chapitre 1

Internet des objets (IdO)

Introduction

Récemment, il y a un intérêt croissant pour les dispositifs intelligents équipés par des terminaux tels que les capteurs et les actionneurs, également appelés objets. Ils peuvent se connecter à Internet pour coopérer et créer de nouveaux services pour le monde réel. Selon le site web de statistiques Statista [95], le nombre d'objets connectés dans le monde passera de 20 milliards en 2017 à 75 milliards en 2025. L'Internet des Objets (IdO) est la technologie qui vise à connecter, via Internet, des objets hétérogènes pour collecter et échanger les données et permettre ainsi des communications entre objets ou entre personne et objet. Dans ce chapitre, nous allons présenter un aperçu global sur l'Internet des objets, à travers son historique, sa définition, son architecture, ses différents domaines d'application et ses challenges imposés.

1.1 De l'internet classique vers l'IdO

L'Internet des objets permet de connecter des objets entre eux via un réseau internet. Au début, l'IdO a été utilisé pour décrire des objets identifiés de manière unique par une étiquette RFID (Radio Fréquence IDentification). Dans ce qui suit, on va citer les phases importantes qui ont contribué au développement de la technologie de l'Internet des objets.

Au cours des dernières décennies, la nécessité d'échanger des informations entre les dispositifs connectés à internet sans l'interaction humaine augmente, et il est prévu d'étendre à la vie quotidienne des personnes et aux processus industriels. Le passage de l'Internet classique à l'Internet des objets (IdO) requis de multiples changements dans les paradigmes de communication. En revanche, les avancées technologiques dans les protocoles de communications et les technologies radio à haute efficacité énergétique, constituent les briques essentielles pour la conception de modules de communication autonomes de petite taille, capable de surveiller et d'agir sur le monde physique. Les

réseaux de capteurs sans fil [7] ont été le premier pas dans cette direction.

Les avancées technologiques ont permis de fabriquer des capteurs de petite taille et faible coût, capable de faire des traitements et des communications sans-fil. Ces capteurs prennent des mesures sur l'environnement, puis transforment ces mesures en signaux pouvant être traités pour découvrir les caractéristiques des phénomènes situés dans la zone entourant ces capteurs. Un réseau de capteurs sans fil ou Wireless Sensor Network (WSN) est un réseau informatique composé d'un ensemble de capteurs capable de communiquer entre eux de manière multi-sauts, déployés dans une zone d'intérêt, pour surveiller des conditions environnementales ou physiques, comme la température, le son, les vibrations, la pression, le mouvement, etc. En fait, les WSN sont assez utilisés dans de nombreuses applications de surveillance militaires, environnementales, de santés, domestiques et industrielles. En général, les capteurs sont des dispositifs de petits tailles alimentés par des batteries, ce qui rend leurs capacités en termes d'énergie et de ressources informatiques très limitées. En conséquence, le grand challenge des chercheurs dans le domaine des WSN est l'optimisation de la consommation d'énergie des capteurs.

La communication machine à machine (M2M) est une nouvelle technologie de communication qui permet aux différents dispositifs physiques de communiquer entre eux de manière autonome et prendre des décisions en collaboration sans intervention humaine [29]. L'origine de communication M2M revient aux systèmes de supervision et d'acquisition de données, dans lesquels des réseaux des capteurs et d'autres périphériques sont utilisés avec des ordinateurs pour surveiller et contrôler les processus industriels [123]. Les réseaux M2M héritent les caractéristiques des réseaux de capteurs (limités en ressources, non protégés et déployés en masse) tout en développant la capacité de prise de décision et le contrôle autonome. L'environnement M2M peut être marqué par trois caractéristiques principales [109]. Premièrement, il s'agit d'un ensemble de composants hétérogènes, allant des capteurs à faibles ressources aux serveurs puissants, ces composants étant déployé sur une large zone géographique. Deuxièmement, il souligne l'augmentation des capacités de prise de décision et de contrôle autonome. En plus, tous les systèmes M2M sont conçus pour assurer la décentralisation et la minimisation de besoin d'intervention humaine. Enfin, les systèmes M2M adoptent un modèle de communication distribuée dans lequel deux nœuds quelconques peuvent établir une connexion, à condition que l'un offre le service nécessaire à l'autre.

La communication M2M devrait développer la technologie afin de créer des applications et des services intelligents évolutifs, fiables et intégrés, donc le terme M2M est étroitement lié à la technologie IdO. M2M et l'IdO devraient permettre l'automatisation et la gestion autonome des réseaux, nécessaires pour prendre en charge le grand nombre d'objets connectés.

La figure 1.1 illustre la transition des réseaux de capteurs sans fil vers l'Internet des objets et souligne l'existence d'une étape intermédiaire, à savoir les communications

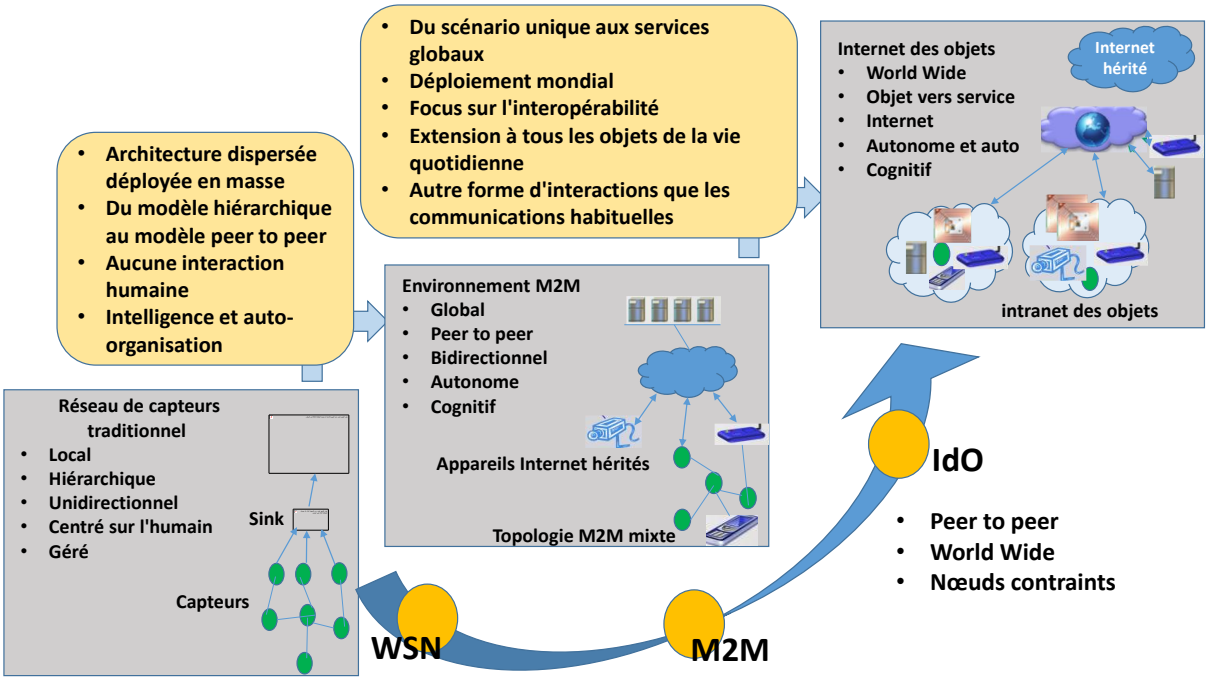

Figure 1.1: Transition des réseaux de capteurs sans fil (WSN) vers l'Internet des objets (IdO) [109].

machine à machine (M2M). Le paradigme M2M considère que tous les nœuds peuvent communiquer entre eux, mais limite l'application de telles communications à un seul scénario (par exemple, domotique ou gestion de l'énergie).

1.2 Définitions et caractéristiques de l'IdO

Au cours des dernières années, une attention considérable a été accordée aux recherches et développements dans le domaine de l'IdO. L'IdO permet aux objets du monde de connecter à Internet et interagir avec les autres avec très peu ou pas d'intervention humaine. L'objectif final est de créer une confortable vie pour les humains, où les objets qui nous entourent savent ce que nous aimons, ce que nous voulons et ce dont nous avons besoin et agissons en conséquence sans nos instructions explicites. [100]

Dans ce qui suit, nous présentons quelques définitions et caractéristiques de l'IdO.

1.2.1 Définitions de l'IdO

L'internet des objets est considéré comme étant la prochaine génération de l'Internet. Le terme internet des objets a été inventé pour la première fois en 1999 par Kevin Ashton dans le contexte de la gestion de la chaîne d'approvisionnement [8]. Selon les visions des auteurs, on peut trouver plusieurs définitions de l'IdO.

L'EPoSS (2008) (European Platform on Smart Systems Integration) donne trois définitions pour trois visions différentes [17]. Premièrement, considérant la fonctionnalité, l'IdO peut être défini comme étant des objets ayant des identités et personnalités

virtuelles opérant dans des espaces intelligents utilisant des interfaces intelligentes pour se connecter et communiquer dans des contextes sociaux, environnementaux et utilisateur. Deuxièmement, considérant l'intégration, l'IdO peut être défini comme étant des objets interconnectés jouant un rôle actif dans ce que l'on pourrait appeler le futur Internet. Enfin, considérant la sémantique de l'expression, l'IdO défini comme étant un réseau mondial d'objets interconnectés uniquement adressables, basés sur des protocoles de communication standard [17].

L'IERC (2010) (European Research Cluster on the Internet of Things) considère que l'IdO est «une infrastructure de réseau global dynamique avec des capacités d'autoconfiguration, où les objets physiques et virtuels ont des identités, des attributs physiques et des personnalités virtuelles, et utilisent des interfaces intelligentes pour se connecter entre elles et au réseau de données» [16].

Selon Chen (2012) l'Internet des objets donne aux dispositifs quotidiens les capacités de détection, communication et traitement numériques. Dans ce nouveau paradigme, les appareils intelligents vont collecter des données, relier les informations ou le contexte les uns aux autres, et traiter les informations de manière collaborative en utilisant le cloud computing et des techniques similaires. Enfin, soit les humains seront invités à agir, soit les machines elles-mêmes agiront automatiquement [30].

Vermesan (2014) vu que l'Internet des objets est un paradigme qui prend en compte la présence omniprésente dans l'environnement d'une variété d'objets qui, grâce à des connexions filaires et sans fil et à des systèmes d'adressage uniques, peut interagir les uns avec les autres et coopérer avec d'autres objets pour créer de nouvelles applications/services et atteindre des objectifs communs [124].

Guinard (2016) dit que l'Internet des objets est un système d'objets physiques pouvant être découvert, surveillé, contrôlé et avec des appareils électroniques peut communiquer via différentes interfaces de réseau et éventuellement être connecté à Internet [47].

Selon Guillemin [46] l'Internet des objets permet aux personnes et aux objets d'être connectés à tout moment, en tout lieu, avec n'importe quoi et n'importe qui, en utilisant n'importe quel réseau et n'importe quel service.

Dans cette thèse, nous adoptons la définition donnée par Guillemin, car nous croyons que cette définition résume la vision globale de l'IdO. La figure 1.2 illustre plus clairement la définition.

1.2.2 Caractéristiques de l'IdO

Selon Razzaque [100], les principales caractéristiques de l'IdO sont regroupées en deux catégories: infrastructures et applications.

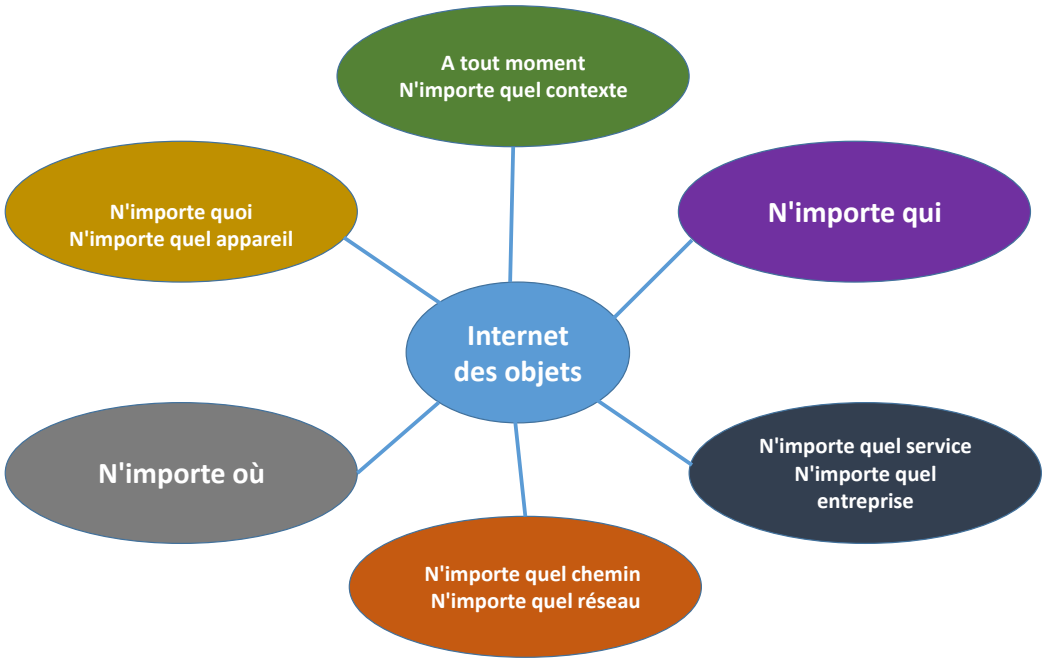

Figure 1.2: Définition de l'Internet des objets [46].

1.2.2.1 Caractéristiques de l'infrastructure IdO

Dans ce qui suit, on va citer les caractéristiques de l'infrastructure de l'IdO: [100]

- **Hétérogénéité des dispositifs**: L'IdO est composé d'un ensemble d'appareils et de capteurs intégrés et à faibles ressources. De plus, IdO aura également besoin d'appareils informatiques d'ordre supérieur pour effectuer des tâches plus lourdes (routage, commutation, traitement des données, etc.). L'hétérogénéité des dispositifs émerge non seulement des différences de capacité et de fonctionnalités, mais aussi pour d'autres raisons, notamment les différents fournisseurs de produits, les exigences des applications, etc.

- **Ressources limitées**: Les dispositifs de l'IdO se caractérisent par leur petite taille et leur faible consommation d'énergie, ce qui limite leur capacité de traitement, de mémoire et de communication.

- **Interaction spontanée**: Des interactions soudaines peuvent se produire lorsque des objets se déplacent et entrent dans la zone de communication d'autres objets, conduisant à la génération spontanée d'événements. Par exemple, un utilisateur de smartphone peut entrer en contact étroit avec un téléviseur ou réfrigérateur à la maison et générer des événements sans la participation de l'utilisateur.

- **Réseau à très grande échelle et grand nombre d'événements**: Dans un environnement de l'IdO, des milliers d'appareils ou d'objets peuvent interagir les uns avec les autres, même au même endroit (par exemple, dans un bâtiment, un

supermarché, une université). Selon le site Web de statistiques Statista [95], le nombre d'objets connectés dans le monde dépassera 75 milliards en 2025. En conséquent, les interactions spontanées entre un très grand nombre de dispositifs produisent un nombre énorme d'événements. Ce nombre incontrôlable d'événements peut entraîner des problèmes tels que l'encombrement des événements et une capacité de traitement d'événements réduite.

- **Réseau dynamique**: La plupart des appareils de l'IdO sont mobiles, connectés sans fil et limités en ressources. De plus, les nœuds du réseau IdO peuvent être déconnectés à tout moment en raison de mauvaises liaisons sans fil ou d'un manque de batterie. Ces facteurs rendront le réseau de l'IdO très dynamique.
- **Sensibilité au contexte**: Le contexte joue un rôle essentiel dans l'IdO et ses applications. La grande quantité de données générée par les capteurs, n'aura pas de valeur que si elle est analysée, interprétée et comprise. L'informatique sensible au contexte permet de stocker les informations de contexte et simplifier leur interprétation.
- **Intelligence**: Selon Intel, les appareils et les systèmes intelligents sont les deux éléments-clés de l'IdO. Dans le réseau dynamique et ouvert de l'IdO, ces entités intelligentes seront interopérables et pourront agir indépendamment en fonction du contexte, des circonstances ou des environnements.
- **Sensibilité aux localisations**: La localisation et les informations spatiales des objets jouent un rôle vital dans l'informatique contextuelle. Dans un réseau d'objets à grande échelle, les interactions dépendent fortement de leur emplacement, de leur environnement et de la présence d'autres entités (par exemple, les objets et les personnes).
- **Distribution**: L'Internet traditionnel lui-même est un réseau distribué à l'échelle mondiale, tout comme l'IdO. La forte dimension spatiale au sein de l'IdO rend leur réseau distribué à différentes échelles.

1.2.2.2 Caractéristiques des applications de l'IdO

Dans ce qui suit, nous allons citer les caractéristiques des applications de l'IdO: [100]

- **Diverses applications**: L'IdO peut offrir des services à un grand nombre d'applications dans de nombreux domaines. Ces domaines peuvent être regroupés en catégories telles que: transport et logistique, santé et environnement intelligent (maison, bureau, usine).
- **Temps réel**: La plupart des applications de l'IdO peuvent être largement classées dans la catégorie des applications en temps réel. Par exemple, les applications

de soins de santé exigent une livraison des données et services en temps réel. La livraison retardée des données peut rendre l'application ou le service inutile et même dangereux dans les applications critiques.

- **Tout en tant que service (XaaS)**: Le modèle tout en tant que service est très efficace, évolutif et facile à utiliser. Le modèle XaaS a été inspiré de l'approche Sensing as a Service dans les WSN et cela peut inévitablement conduire l'IdO vers un modèle XaaS.
- **Attaque de sécurité accrue**: Bien qu'il existe un énorme potentiel pour l'IdO dans différents domaines, il existe également des préoccupations pour la sécurité des applications et des réseaux. L'IdO exige une connectivité et une accessibilité mondiales, ce qui signifie que tout le monde peut y accéder à tout moment et de toute façon. Cela augmente considérablement les possibilités d'attaque des applications et des réseaux de l'IdO.
- **Confidentialité**: En utilisant l'IdO, les applications peuvent collecter des informations sur les activités quotidiennes des personnes (par exemple, les itinéraires de voyage, les habitudes d'achats, etc.). Ces informations sont considérées par de nombreuses personnes comme informations privées, l'exposition de ces informations pourrait avoir un impact sur la vie privée de ces personnes. En conséquent, toute application IdO non conforme aux exigences de confidentialité doit être interdite par la loi car elle porte atteinte à la vie privée des citoyens.

1.3 Architecture de l'IdO

L'Internet des objets peut être envisagé comme un réseau complet composé de divers objets du monde réel connectés, qui dépendent des technologies des objets, de communication, de mise en réseau et de traitement de l'information. L'IdO implique un nombre croissant de dispositifs intelligents interconnectés fournissant des services à tout moment, n'importe où et qui sont deviennent beaucoup plus décentralisés et complexes. Pour gérer cette complexité, une architecture de l'IdO est nécessaire. Dans ce contexte, Shivangi [122] définit l'architecture comme une structure permettant de spécifier les composants physiques d'un réseau, leur organisation et configuration fonctionnelles, ainsi que ses principes, procédures et les formats de données utilisés pour son fonctionnement. L'architecture basique de l'IdO décrite ci-après est issue d'une comparaison entre plusieurs travaux dans la littérature [122, 48, 59, 90]. En général, la structure de l'IdO est divisée en quatre couches, comme illustré dans la figure 1.3. La fonctionnalité des couches est décrite brièvement ci-dessous:

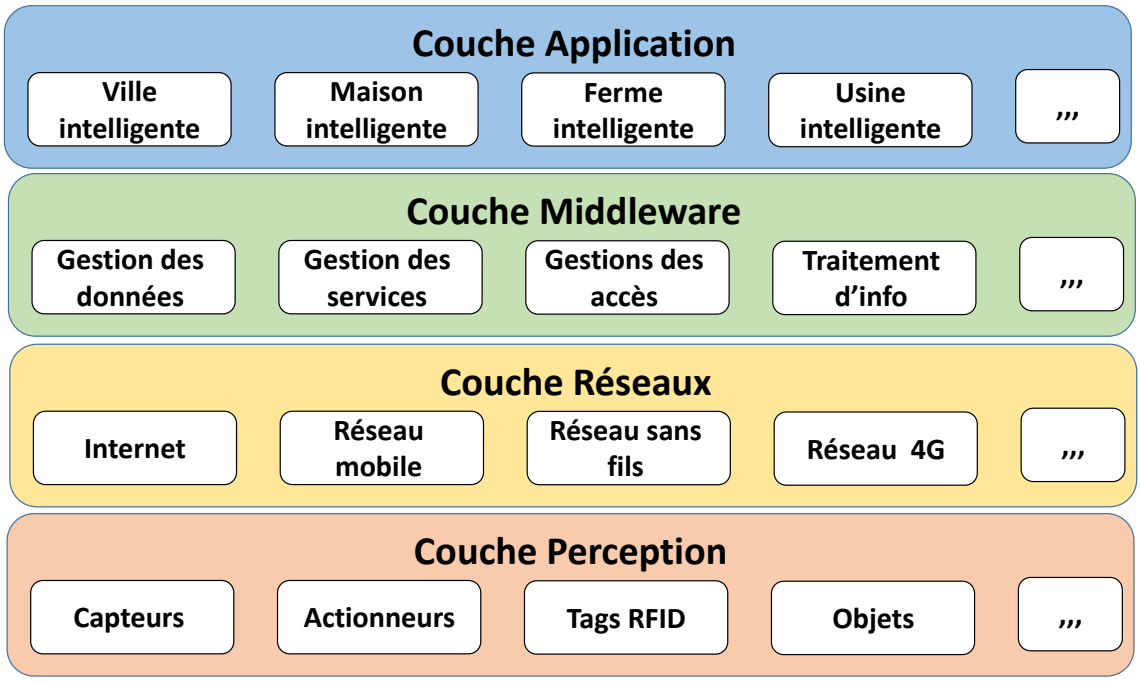

Figure 1.3: Architecture de l'IdO.

1.3.1 Couche perception

La couche perception est similaire à la couche physique dans le modèle OSI (Open Systems Interconnexion) qui comprend essentiellement les différents objets physiques (entités du monde réel) et objets virtuels (entités virtuelles). Les objets de l'IdO comprennent des logiciels embarqués (système d'exploitation, application embarquée, etc.) et du matériel (composants électriques et mécaniques avec capteurs embarqués, processeurs, antennes de connectivité, etc.). Le rôle principal de cette couche est la gestion des objets physiques pour assurer l'identification et la collecte d'informations des capteurs. Les capteurs sont des objets qui détectent et mesurent certains événements dans son environnement tel que la température, la vitesse du vent, le niveau de pH, l'humidité, la quantité de poussière dans l'air, etc. Dans certains cas, les capteurs peuvent aussi avoir une mémoire leur permettant d'enregistrer un certain nombre de mesures. Pour contrôler et suivre les objets déployés dans un système à grande échelle telle que l'IdO, chaque objet doit avoir une identité numérique unique. La technique qui consiste à attribuer une identité unique à un objet est appelée identificateur universel unique (UUID) [69]. En particulier, l'UUID est essentiel au succès du déploiement des services dans un immense réseau comme l'IdO. De nombreux aspects importants doivent être pris en compte lors de la détermination de cette couche, tels que le coût, la taille, les ressources, la consommation d'énergie, la manière de déployer les objets, l'hétérogénéité, la topologie du réseau, la communication et les protocoles. Les informations recueillies par les capteurs sont ensuite envoyées à la couche réseau pour une transmission sécurisée vers le système de traitement d'informations.

1.3.2 Couche réseaux

La couche réseau peut également être appelée couche de transmission. Cette couche joue un rôle important dans le transfert sécurisé des informations entre les dispositifs de capteurs et le système de traitement d'informations. La couche réseau comprend le matériel, les logiciels, les technologies et les protocoles qui permettent aux objets de se connecter, d'identifier leurs environnements et de partager des données avec les autres objets. Pour assurer un service fiable, la Qualité de Service (QoS) doit être prise en compte dans la communication dans le réseau. Plusieurs technologies de communication peuvent être utilisées dans cette couche telle que 3G, 4G, UMTS, Wi-Fi, WiMAX, RFID, infrarouge, Bluetooth, ZigBee, satellite, etc. Par conséquent, cette couche est principalement responsable du transfert des informations de la couche perception à la couche middleware.

1.3.3 Couche middleware

La communication entre les dispositifs du système IdO permettant de générer plusieurs types de services. Le rôle principal de la couche middleware est la gestion des services et le stockage des informations générées par la couche perception dans des bases de données. De plus, cette couche a la capacité de l'agrégation, du filtrage et du traitement des données reçues des dispositifs IdO. Ainsi que la découverte d'informations, l'apprentissage automatique, la modélisation prédictive et le contrôle d'accès de n'importe quelle application aux appareils de l'IdO. Puis prendre des décisions automatiques en fonction des résultats obtenus. Ainsi, du point de vue fonctionnel, il sert de couche d'intégration entre les dispositifs de capteurs et les applications. Certains middlewares IdO fournissent une gestion du système d'exploitation et des API tout en permettant aux applications IdO à communiquer sur des interfaces hétérogènes. Les solutions les plus courantes avec les fonctionnalités middleware IdO reposent sur le cloud computing. Les plateformes cloud permettent un développement des services IdO et un traitement des données indépendamment de la plateforme matérielle.

1.3.4 Couche application

La couche application est responsable de la gestion globale des applications en fonction des informations traitées dans la couche middleware. Les applications de l'IdO peuvent être: la santé intelligente, l'agriculture intelligente, le transport intelligent, les lunettes intelligentes, les villes intelligentes, la vie autonome intelligente, le transport intelligent, etc. Une application de l'IdO est un logiciel qui utilise les informations de couche middleware pour mieux comprendre l'environnement physique en demandant des données des capteurs ou pour contrôler des actions physiques à l'aide d'actionneurs. Par exemple, un système logiciel qui contrôle la température d'un bâtiment représente une application

de l'IdO. Selon Colakovic [69], les problèmes clés de développement des applications de l'IdO sont liés au déploiement dans divers cas d'utilisation, à la disponibilité, à la gestion, à la fiabilité, à l'interopérabilité, à l'évolutivité (déploiement et intégration à grande échelle), à la sécurité (authentification, contrôle d'accès, gestion de la configuration, protection antivirus, cryptographie, etc.) et la confidentialité. Les défis de recherche comprennent la création d'algorithmes et de schémas pour présenter, analyser et traiter les données collectées par les capteurs.

1.4 Domaines d'application de l'IdO

L'Internet des objets, avec sa vision d'objets connectés à Internet dotés de capacités de communication, pourrait renforcer le rôle des technologies de l'information et de la communication (TIC) en tant que moteur de l'innovation sur divers marchés d'applications. Aujourd'hui de nombreux secteurs sont concernés par l'IdO. Comme le montre la figure 1.4, les applications de l'IdO sont très nombreuses, elles impliquent pratiquement tous les domaines de la vie quotidienne. Les potentialités offertes par l'IdO permettent le développement d'un très grand nombre d'applications, dont seule une très petite partie est actuellement disponible pour notre société. Dans le futur, il y aura des applications intelligentes pour des maisons plus intelligents, des systèmes de transport plus intelligentes, des hôpitaux plus intelligents, des entreprises plus intelligentes. Dans ce qui suit, nous allons présenter quelques exemples de domaines d'application de l'IdO.

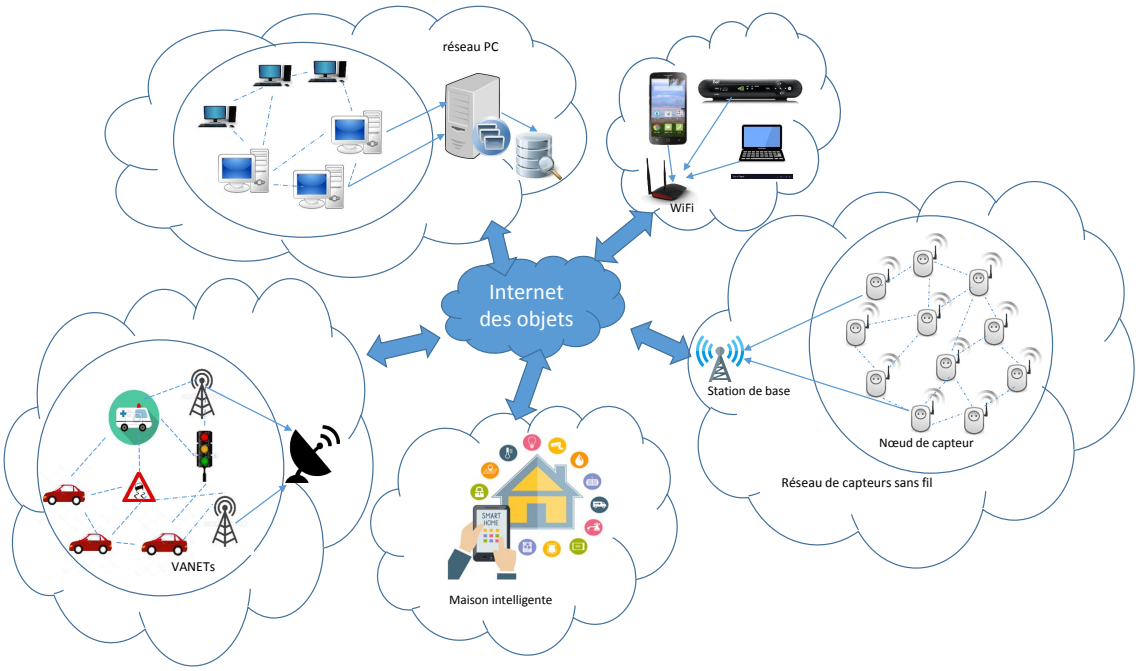

Figure 1.4: Domaines d'application de l'IdO.

1. **Surveillance de l'environnement**: Actuellement, l'Internet des objets contribue

de manière significative dans le développement des applications de surveillance de l'environnement. Elle permet de surveiller les phénomènes naturels (température, humidité, vent, précipitations, etc.), de manière distribuée et autogérée, et d'intégrer les données des capteurs aux applications globales. La capacité de communication entre les objets et le traitement de l'information en temps réel, permet de détecter les risques pouvant menacer la vie humaine et la vie animale. Le déploiement des dispositifs de capteurs donne la possibilité d'accéder à des zones critiques (par exemple zones volcaniques, zones isolées), à partir de laquelle les informations capturées peuvent être transmises au point de décision afin de détecter des conditions anormales. Dans cette perspective, les technologies de l'IdO donnent la possibilité de développer une nouvelle génération de systèmes de surveillance et d'aide à la décision améliorant les solutions actuelles [78].

2. **Agricultures intelligentes**: L'agriculture et la civilisation modernes exigent une production accrue d'aliments pour nourrir la population mondiale. De nouvelles technologies et solutions doivent être appliquées dans le domaine agricole pour fournir une alternative optimale à la collecte et au traitement de l'information tout en améliorant la productivité [99, 85]. L'Internet des objets permet de découvrir une nouvelle direction de recherche innovante dans le domaine agricole. Un réseau de capteurs permet de collecter des données, effectuer des traitements sur ces données et informer l'agriculteur par un moyen de communication (par exemple un message texte sur téléphone mobile) des parties de la ferme nécessitant une attention particulière. Ainsi, un système agricole intelligent aidera les agronomes à mieux comprendre les modèles de croissance des plantes et d'adopter des pratiques agricoles efficaces grâce à la connaissance des conditions du sol et de la variabilité climatique. Cela augmente considérablement la productivité agricole en évitant les conditions de production inappropriées. L'agriculture intelligente permet d'optimiser la consommation de l'eau, et la planification des travaux agricoles, et par conséquent économiser les ressources et préserver l'environnement en diminuant la pollution.

3. **Transport et logistique**: Aujourd'hui, les moyens de transport sont de plus en plus équipés par des capteurs et des actionneurs et possédants une puissance de traitement et de communication. En plus, les routes elles-mêmes et les marchandises transportées sont également équipées d'étiquettes et de capteurs. Elles envoient des informations aux systèmes de contrôle de la circulation et aux véhicules de transport afin de mieux orienter le trafic, de bien guider les touristes et de surveiller le statut des marchandises transportées. L'utilisation des technologies internet des objets pour la gestion des bagages et des passagers dans les aéroports et les opérations aériennes permettra le suivi et le tri automatisés des sacs et une sécurité accrue des voyageurs. L'IdO permettra de développer un système de transport plus intelligent et efficace

garantissant une meilleure sécurité, plus de confort et une facilité de déplacement, tout en privilégiant l'économie d'énergie et de temps.

4. **Villes intelligentes**: L'une des applications les plus prometteuses de l'Internet des objets pourrait bien être l'émergence de villes intelligentes. Alors que de plus en plus d'humains quittent les zones rurales pour se rendre dans les villes, il devient évident que des changements sont nécessaires dans la conception et le fonctionnement des grandes villes pour assurer la sécurité et le bien-être de leurs habitants. Le terme ville intelligente est utilisé pour désigner l'écosystème cyber-physique émergent en déployant une infrastructure de communication avancée et des services innovants à l'échelle de la ville. Grâce aux services avancés, il est en effet possible d'optimiser l'utilisation des infrastructures physiques de la ville (réseaux routiers, réseau électrique, etc.) et la qualité de vie des citoyens.

5. **Maisons et bâtiments intelligents**: L'utilisation des technologies avancées de l'internet des objets dans les bâtiments peut aider à économiser les ressources (électricité, eau) et à améliorer le niveau de satisfaction des habitants. L'impact est à la fois économique (réduction des dépenses) et environnemental (réduction de pollution). Des capteurs et des actionneurs répartis dans les maisons peuvent rendre notre vie plus confortable à plusieurs égards: le chauffage peut être adapté à nos préférences et au climat; l'éclairage peut changer en fonction de l'heure du jour; les incidents domestiques peuvent être évités avec des systèmes de surveillance et d'alarme; et l'énergie peut être économisée en éteignant automatiquement les équipements électriques lorsque vous n'en avez pas besoin. Dans cette application, les capteurs jouent un rôle essentiel, ils servent à surveiller la consommation des ressources et à détecter de manière pro-active les besoins actuels des utilisateurs [9, 69].

6. **Gestion intelligente des entreprises**: Les entreprises recherchent toujours à améliorer l'efficacité de production grâce à la réduction de coûts et de temps. L'intervention des techniques de l'IdO dans le processus de fabrication dans les entreprises peut apporter une vraie valeur ajoutée [78, 16]. Les technologies RFID sont déjà utilisés dans de nombreux secteurs pour la gestion des stocks, tout au long de la chaîne d'approvisionnement et de livraison. Les étiquettes RFID est habituellement utilisée pour surveiller et gérer le mouvement des produits dans une chaîne d'approvisionnement. Aussi, les technologies IdO permettent aux clients de consulter automatiquement toutes les informations relatives aux produits qu'ils ont achetés. De plus, les technologies d'identification peuvent aider à limiter les vols et à lutter contre la contrefaçon en fournissant des produits munis d'un identifiant unique comprenant une description complète et fiable du produit lui-même. Enfin, la connexion des machines à Internet permet de générer une quantité énorme de

données en temps réel et de fournir des systèmes industriels plus transparents et plus efficaces.

7. **Soins de santé**: Le monde de la médecine est en progression constante. Aujourd'hui, grâce à l'Internet des objets, il est possible de maintenir les personnes âgées à leur domicile tout en ne négligeant pas leur sécurité ou le lien social. L'intégration des technologies de l'IdO dans le secteur de la santé, permet de développer plusieurs applications. L'utilisation des téléphones mobiles avec des capteurs RFID donne la possibilité de créer des plates-formes de surveillance des paramètres médicale et de délivrance de médicaments [14]. Les patients porteront des capteurs médicaux pour surveiller des paramètres tels que la température corporelle, la pression artérielle et l'activité respiratoire. Les informations générées seront agrégées localement et transmises à des centres médicaux distants, qui pourront effectuer une surveillance à distance, assureront des soins préventifs et seront capables de réagir rapidement en cas d'urgence. Cela permet aux patients de réduire le temps et les risques car ils peuvent rester et guérir à la maison tout en bénéficiant d'une supervision et d'une assistance en temps réel, et aux hôpitaux de traiter plus de patients sans avoir besoin de plus de lits et de personnels [16].

8. **Sécurité et surveillance**: La surveillance de la sécurité est devenue une nécessité pour les entreprises, les centres commerciaux, les usines, les parkings et de nombreux autres lieux publics. Les technologies de l'IdO peuvent être utilisés pour améliorer considérablement les performances des solutions actuelles et offrant des alternatives moins coûteuses. Des capteurs et des caméras surveillants le comportement des personnes peuvent être utilisés pour évaluer la présence de personnes agissant de manière suspecte. Donc, des systèmes d'alerte précoce efficaces et des systèmes d'identifications personnelles utilisent les étiquettes RFID et des technologies similaires peuvent être construits. Cependant, de nombreuses associations mondiales protestent fortement contre la violation de la vie privée qui pourrait résulter de l'adoption généralisée d'une telle technologie.

Certainement, la portée de l'IdO est extrêmement large [78, 93, 14, 9, 69]. Cependant, les applications basées sur l'IdO peuvent améliorer de manière constante la compétitivité des solutions disponibles. L'adoption de l'IdO devrait donc être fortement influencée par les besoins du marché.

1.5 Challenges imposés par l'IdO

Généralement, les systèmes IdO sont complexes en raison de leur impact considérable sur tous les aspects de la vie humaine (sécurité, sureté, santé, mobilité, efficacité énergétique, durabilité de l'environnement, etc.), ainsi que de ses divers technologies déployés pour

permettre un échange de données autonome entre les différents objets. Bien que la grande révolution dans le domaine de la technique des dispositifs intelligents et des réseaux de communication rend le concept de l'IdO réalisable, un effort de recherche important est encore nécessaire. Dans cette section, nous expliquons les problèmes de recherche les plus importants qui doivent être traités pour répondre aux exigences qui caractérisent l'IdO.

1. **Sécurité et confidentialité**: Le problème majeur qui peut vraiment avoir un impact négatif sur les systèmes internet des objets est les attaques de sécurité. Plusieurs raisons rendent les systèmes de l'internet des objets extrêmement vulnérable aux attaques [9, 50]. Premièrement, les composants de l'IdO passent la plupart du temps sans surveillance ce qui facilite les attaques physiques. Deuxièmement, la plupart des communications sont sans fil, ce qui rend l'espionnage extrêmement facile. Enfin, la plupart des composants de l'IdO sont caractérisés par une faible capacité en matière d'énergie et de ressources informatiques, par conséquent, ils ne peuvent pas implémenter des systèmes complexes prenants en charge la sécurité. L'acceptation sociale des nouveaux technologies et services de l'IdO dépendra fortement de la fiabilité des informations et de la protection des données privées. Bien qu'un certain nombre de projets aient été développés pour la sécurité et la protection de la vie privée [73, 117, 120], un mécanisme fiable de protection de la sécurité pour l'IdO reste toujours demandé. Techniquement, les défis suivants devraient être abordés [14, 69]: (1) la définition de la sécurité et de la vie privée du point de vue social, juridique et culturel; (2) le mécanisme de confiance; (3) la sécurité des communications, services et applications ; (4) la confidentialité des communications et des données des utilisateurs.

2. **Standardisation**: La standardisation est considérée comme un élément principal qui contribue au développement des systèmes de l'internet des objets. La standardisation de l'IdO vise à rapprocher les fournisseurs de services au clients, améliorer l'interopérabilité et augmenter le niveau de concurrence des produits et services [56]. Cependant, l'émergence rapide de l'IdO rend la standardisation difficile. Les problèmes courants de la standardisation de l'IdO sont: l'interopérabilité et la sécurité sémantiques, les problèmes de niveau d'accès radio et les problèmes de confidentialité. De plus, les API sont très importants pour l'interconnexion entre les objets intelligents hétérogènes. Les interfaces Web sont considérées comme la solution parfaite qui facilite ces communications, en particulier avec les dispositifs de l'IdO et le cloud computing lors de la transmission de données pour le stockage [19]. Le principal problème réside dans l'absence de conception spécifique pour une communication efficace machine à machine. Les standards ouverts de l'IdO, tels que les standards de sécurité, les standards de communication et les standards d'identification, pourraient aider à la croissance des technologies de l'IdO. [69]

3. **Performances réseau et QoS**: La capacité de gérer un nombre élevé de dispositifs connectés avec divers services et processus dépend totalement des performances du système de l'internet des objets. Les principaux problèmes sont liés à la charge de trafic et aux divers modèles de trafic qui ont un impact considérable sur les performances des réseaux et la QoS (Qualité de Service). Certains problèmes importants liés à la QoS incluent la bande passante, le débit, la latence, etc. De plus, la croissance de nombre des appareils connectés et l'augmentation des débits de données dans les réseaux sans fil ont augmentés les problèmes d'utilisation du spectre radioélectrique [129]. Certains pays pourraient avoir du mal à trouver un spectre supplémentaire car l'efficacité spectrale des réseaux radio atteint ses limites physiques. Tous ces faits influencent les performances des systèmes IdO qui sont considérés comme extrêmement importants dans la plupart des applications. La QoS dans le système de l'IdO dépend des technologies déployés, des protocoles, des demandes de trafic, etc. Plusieurs études ont été réalisées pour évaluer les performances des services de l'IdO mais cela reste un problème ouvert. [31]

4. **Interopérabilité**: L'interopérabilité est la capacité d'interagir entre différents appareils et systèmes indépendamment du matériel et des logiciels déployés [31]. Une variété de standards et de technologies utilisées pour le développement de l'IdO ainsi que diverses solutions de différents fournisseurs conduisent à une hétérogénéité massive qui provoque des problèmes d'interopérabilité. Pour cette raison, l'interopérabilité doit être envisagée à la fois par les développeurs d'applications et par les fabricants d'appareils IdO pour garantir la prestation de services pour tous les clients. Les développeurs de l'IdO devraient créer des applications évolutives permettant d'ajouter de nouvelles fonctions sans problème tout en maintenant l'intégration avec différentes technologies de communication [50].

5. **Puissance et consommation d'énergie**: L'énorme quantité de données générées par les capteurs de surveillances de l'environnement urbain affecte négativement le trafic réseau, le stockage des données et l'utilisation de l'énergie [50]. Les technologies de stockage d'énergie doivent répondre aux exigences des systèmes de l'Internet des objets telles que la fourniture de sources d'énergie pour les petits appareils intégrés. Aujourd'hui, parmi les majors problèmes technologiques est de fournir des alimentations fiables aux capteurs et aux dispositifs de l'IdO ainsi que le développement de chipsets de faible consommation énergétique. Un autre défi important consiste à intégrer des calculs de grande taille dans les petits appareils intégrés ayant faible source d'énergie, en particulier dans le cas d'applications basées sur le traitement d'images et de vidéos. Ainsi, un ensemble de techniques capables d'améliorer l'efficacité énergétique sont nécessaires. Bien qu'il existe de nouvelles techniques telles que la détection compressive [45] et l'utilisation de méthodes

d'échantillonnage [28], mais la puissance de calcul et la consommation d'énergie restent un problème ouvert.

6. **Stockage, traitement et visualisation des données**: Les données générées par les dispositifs de l'internet des objets sont généralement en grand volume, sous diverses formes, et sont générées à différentes vitesses. Les applications de l'IdO prennent souvent des décisions critiques en fonction des données collectées. Parfois, ces données peuvent être corrompues pour diverses raisons telles que la défaillance d'un capteur, l'introduction de données non valides par un utilisateur malveillant, un retard dans la livraison des données ou un format de données incorrect. Par conséquent, les développeurs d'applications IdO sont confrontés au défi de développer des méthodes qui prennent en considération la présence de données invalides et de nouvelles techniques qui permettent de prendre des décisions sur les données collectées [121]. Plusieurs méthodes de Data mining telles que l'IA (intelligence artificielle), l'apprentissage automatique et d'autres algorithmes de prise de décision intelligents permettent de faire des calculs sur les grands ensembles de données [31]. Ces techniques peuvent être utilisées pour organiser les données brutes ainsi que pour extraire des informations et des connaissances, mais leur coût représente un major obstacle. Réellement, pour gérer une quantité de données en augmentation constante, il semble que seule la technologie Cloud puisse répondre efficacement à ces exigences.

7. **Problèmes environnementaux**: L'Internet des objets a des impacts positifs et négatifs sur l'environnement. Chaque jour de nouveaux appareils sont déployés, le respect de l'environnement est un sujet qui devrait faire l'objet d'une plus grande attention dans les recherches futures. La durabilité environnementale est l'une des plus grandes préoccupations en raison de l'augmentation de la demande d'énergie et des déchets électroniques. Plusieurs axes de recherche dans cette direction doivent être ouverts tels que la réduction de consommation d'énergie, l'utilisation des sources d'énergies renouvelables, etc. De nouvelles technologies de TIC vertes doivent être déployées dans le développement de systèmes IdO. [31]

8. **Disponibilité**: La disponibilité des services est l'un des principaux problèmes à résoudre pour gérer correctement la dynamicité des systèmes de l'IdO. La disponibilité signifie que les applications de l'IdO doivent être disponibles partout et à tout moment pour chaque objet autorisé. Les objets qui vont être connectés doivent être adaptatifs et intelligents pour prendre en charge la connectivité transparente et la disponibilité souhaitée. Le principal défi dans la conception de systèmes extrêmement disponibles est de tolérer les défaillances et de supporter ses effets. Les appareils IdO étant susceptibles de tomber en panne, ainsi des techniques de tolérance aux pannes et de redondance pour les appareils et les services sont

nécessaires [45].

9. **Architecture**: En présence d'un grand nombre d'objets connectés à Internet, il est indispensable d'avoir une architecture adéquate qui supporte et permette une connectivité facile, un contrôle serré, une communication idéale, une intégration de diverses technologies et prendre en charge l'interopérabilité totale [114]. En conséquence, l'un des principaux défis pour les systèmes de l'IdO est de construire une architecture ouverte, intégrée et normalisée. Les exigences de conception de l'architecture pour l'IdO sont l'évolutivité, l'interopérabilité, l'ouverture et la modularité dans un environnement hétérogène.

10. **Évolutivité**: L'évolutivité est la capacité d'ajouter de nouveaux appareils et services au système de l'IdO sans dégrader les performances des services existants. Le major défie lié à l'évolutivité et de prendre en charge un grand nombre de périphériques avec mémoire, traitement, bande passante et autre contraintes de ressources [92]. Des mécanismes évolutifs doivent être déployés pour une découverte efficace des appareils mais aussi pour permettre leur interopérabilité. Pour permettre l'évolutivité ainsi que l'interopérabilité, il faut utiliser une architecture en couches [111]. Ces architectures doivent gérer de nombreux périphériques connectés au système, qui expriment les problèmes d'évolutivité. L'une des solutions possibles consiste à utiliser des plates-formes cloud hautement évolutives avec la possibilité de stocker une énorme quantité de données collectées. Par conséquent, un cloud des objets [1, 105] peut être utilisé comme une architecture globale qui intensifie le cloud computing.

11. **Gestion et autoconfiguration**: La gestion des applications et des dispositifs de l'internet des objets est un facteur très important pour le succès des systèmes IdO. Les fonctionnalités de gestion telles que la surveillance, le contrôle et la configuration sont un défi majeur en raison de la complexité, l'hétérogénéité, le grand nombre d'appareils déployés et les demandes de trafic de l'IdO. Les applications de l'IdO doivent être capable d'identifier différents objets intelligents et d'interagir avec eux pour fournir une gestion efficace et des fonctionnalités d'autoconfiguration. L'autoconfiguration signifie que le système de l'IdO a des capacités d'adoption dynamique des changements dans son environnement. Par exemple, si les appareils électriques peuvent être éteints lorsque les gens ne sont pas à la maison, ils augmenteraient l'efficacité de la consommation d'énergie. Selon [31], on peut distinguer trois niveaux de gestion: gestion de données, gestion de réseau et gestion des appareils.

12. **Identification unique**: L'internet des objets vise à connecter des millions et des milliards d'objets physiques qui devraient être identifiables de manière unique sur Internet [77]. Chaque objet doit avoir un identifiant unique tel que l'adresse IP ou l'URI (Uniform Resource Identifier) et cela est considéré comme l'un des facteurs les

plus importants pour le succès de l'IdO. Si chaque objet possède un identifiant unique et une connexion à Internet, les objets peuvent être surveillés, contrôlés et gérés tout au long du cycle de vie. Ainsi, un schéma de gestion d'identité approprié est nécessaire qui attribuera et gérera dynamiquement des noms uniques pour une large gamme de périphériques physiques. Une gestion appropriée de l'identité avec des identifiants uniques et des schémas de distribution de clés efficaces est des problèmes mis en évidence dans certains travaux [104, 118], mais le problème reste ouvert.

Conclusion

Dans ce chapitre, nous avons présenté un aperçu global sur le concept de l'internet des objets. Au début, nous avons cité les phases importantes qui ont contribué au développement de la technologie de l'Internet des objets. Ensuite, pour clarifier le concept de l'IdO, nous avons donné un ensemble de définitions montrant les différentes visions de ce concept. Après, nous avons présenté les différentes couches architecturales de l'IdO. Par la suite, nous avons présenté quelques exemples de domaines d'application de l'IdO. Enfin, nous avons expliqué les problèmes de recherche les plus importants qui doivent être traités pour répondre aux exigences caractérisantes l'IdO. Dans le prochain chapitre, nous abordons la notion de sensibilité au contexte.

Chapitre 2

Sensibilité au contexte et Aide à la décision multicritère

Introduction

Dans le chapitre précédent, nous avons présenté un aperçu global sur l'internet des objets, nous avons vu son historique, sa définition, son architecture, ses différents domaines d'application et ses challenges imposés. Dans ce chapitre, nous présentons les fondements de la notion de sensibilité au contexte.

Les systèmes sensibles au contexte sont des solutions permettant de superviser la façon dont les utilisateurs interagissent avec l'environnement ubiquitaire et d'automatiser les actions répétitives des utilisateurs. La sensibilité au contexte est devenue plus populaire avec l'introduction du terme «informatique ubiquitaire» par Mark Weiser dans son article «l'ordinateur pour le 21ème siècle» en 1991 [127]. Ensuite, le terme «sensibilité au contexte» a été utilisé pour la première fois par Schilit et Theimer en 1994 [113]. Depuis cette date, la notion de sensibilité au contexte est une caractéristique essentielle dans le développement des systèmes informatiques ubiquitaires et omniprésents. Les applications sensibles aux contextes sont évoluées des applications de bureau, des applications Web, de l'informatique mobile, de l'informatique ubiquitaire et omniprésente à l'Internet des objets au cours de la dernière décennie. Pour cela, la recherche sur la sensibilité au contexte est devenue un domaine de recherche bien connu en informatique.

L'optimisation multicritère permet de résoudre des problèmes d'optimisation combinatoire liés à des problématiques réelles. Depuis les années 80, le domaine de l'aide à la décision multicritère permet de trouver des solutions de consensus pour des problèmes comportant plusieurs critères a connu un vif intérêt. [65]

Dans ce chapitre, nous détaillons la notion de contexte et la notion de sensibilité au contexte. Ensuite, nous présentons l'architecture générale d'un système sensible au contexte. Ainsi, nous citons quelques middlewares sensibles aux contextes les plus

représentatifs et populaires pour l'IdO. Nous présentons ainsi la notion d'aide à la décision multicritère. Nous terminons par la définition du problème qui va être traité dans cette thèse.

2.1 Notion de contexte

Les nouvelles sources d'information telles que les réseaux sociaux, le cloud computing, les réseaux de capteurs et notamment l'Internet des objets sont de plus en plus nombreuses, diverses et hétérogènes. Les informations ainsi produites sont appelées des informations de contexte. Dans cette section, nous présentons les définitions des notions de contexte et de la qualité de contexte. Par la suite, nous décrivons les caractéristiques et les catégories des informations contextuelles.

2.1.1 Définition du contexte

De nombreux chercheurs ont donné des définitions de la notion de contexte dans l'environnement informatique. Selon le dictionnaire collégial de Merriam-Webster [27], le mot «contexte» est défini comme les conditions interdépendantes dans lesquelles quelque chose existe ou se produit. Schilit et Theimer [113] se réfèrent au contexte comme étant l'emplacement, l'identité des personnes et des objets à proximité et les modifications apportées à ces objets. Par la suite, Schilit divise le contexte en trois catégories [113]. Premièrement, contexte informatique, comme la connectivité réseau, les coûts et la bande passante de communication, et les ressources à proximité telles que les imprimantes, les écrans et les postes de travail. Deuxièmement, contexte de l'utilisateur, tel que le profil de l'utilisateur, l'emplacement, les personnes à proximité, même la situation sociale actuelle. Finalement, contexte physique, comme l'éclairage, les niveaux de bruit, les conditions de circulation et la température. Selon Chen [27], le contexte est «l'ensemble des états et des paramètres environnementaux qui déterminent le comportement d'une application ou dans lesquels un événement d'application se produit et qui sont intéressants pour l'utilisateur». Chaari définit le contexte comme « l'ensemble des paramètres externes à l'application pouvant influer sur le comportement d'une application en définissant de nouvelles vues sur ses données et ses services. Ces paramètres ont un aspect dynamique qui leur permet d'évoluer durant le temps d'exécution. Ils ne sont pas « parlants » à l'utilisateur final et doivent donc lui être transparents. Une nouvelle instance de ces paramètres caractérise une nouvelle situation contextuelle qui ne modifie pas les données de l'application mais qui peut mener à les traiter d'une façon différente» [24]. Selon El Ghayam [38], le contexte est toute information qui décrit, à un moment donné, l'environnement dans lequel l'action aura lieu. Cependant, la définition la plus complète et la plus adoptée par les chercheurs est celle de Dey qui a considéré que « le contexte est toute information qui peut être utilisée pour caractériser la situation d'une

entité. Une entité est une personne, un lieu ou un objet considéré comme pertinent pour l'interaction entre un utilisateur et une application, y compris l'utilisateur et les applications elles-mêmes» [34].

2.1.2 Définition de la qualité du contexte

Il existe un certain nombre de définitions qui ont été proposés dans la littérature concernant la qualité du contexte (QoC). Buchholz considère que la qualité du contexte est « toute information qui décrit la qualité de l'information qui est utilisée comme information de contexte. Ainsi, la QoC fait référence aux informations et non au processus ni au composant matériel qui fournit éventuellement les informations» [20]. Selon Krause et Hochstatter [64], la qualité du contexte est toute information inhérente qui décrit les informations de contexte et peut-être utilisées pour déterminer la valeur des informations pour une application spécifique. Cela inclut des informations sur le processus de provisionnement que les informations ont subi (historique, âge), mais pas des estimations sur les étapes de provisionnement futurs qu'elles pourraient parcourir. Fanelli définit la QoC comme «l'ensemble des paramètres utiles pour exprimer les propriétés et les exigences de qualité sur les données de contexte, par exemple, la précision, l'actualité, la fiabilité, etc. De plus, la QoC ne s'agit pas d'avoir des données de contexte parfaites, telles que des données sans erreur, mais d'avoir une caractérisation correcte de la qualité des données» [40].

2.1.3 Caractéristiques de l'information de contexte

Selon El Ghayam, quatre caractéristiques techniques de l'information de contexte: [38]

1. L'information de contexte est changeable avec le temps: Au fil du temps, les informations contextuelles prennent différentes valeurs. Par exemple, différentes heures de la journée ont des températures différentes.

2. L'information de contexte est hétérogène: Le contexte peut être capturé par des capteurs physiques ou récupérés à partir de composants logiciels, donné par l'utilisateur ou bien dérivé a partir de plusieurs sources. Il est évident que les sources variés de contexte fourniront des informations contextuelles hétérogènes.

3. L'information de contexte est imparfaite: Les caractéristiques et la qualité de source d'information vont influer directement sur la qualité de l'information contextuelle. Par conséquent, le contexte peut être imprécis ou bien erroné ou même inconnu.

4. L'information de contexte est interdépendante: Les informations contextuelles peuvent dépendre d'autres informations de contexte. La modification de la valeur des informations de contexte peut affecter une autre valeur de contexte.

2.1.4 Catégorisation du contexte

L'hétérogénéité, la diversité et la qualité des informations contextuelles, obligent les chercheurs à faire une classification ou bien une catégorisation pour faciliter la collection et la présentation de ces informations dans un système d'adaptation. Plusieurs catégorisations de contextes ont été proposés par les chercheurs. Dey [34] définies deux catégories de contexte: le contexte primaire qui incluent les informations sur la localisation, l'identité, le temps et l'activité; le contexte secondaire qui peut être déduit du contexte primaire. Selon Chen [27], le contexte est divisé en deux classes: contexte actif qui influence les comportements d'une application, et le contexte passif qui est pertinent mais non critique pour une application. L'auteur dans [61], propose deux catégories de contexte: contexte physique qui décrit les caractéristiques de l'utilisateur; le contexte organisationnel de l'utilisateur qui décrit les informations relatives aux processus de collaboration dans lequel l'utilisateur est impliqué. Six catégories de contexte sont considérés par Razzaque: [101]

1. Contexte utilisateur: définir le profil de l'utilisateur (identifications, relations avec les autres, listes de tâches, etc).
2. Contexte physique: représenter l'environnement physique (humidité, température, niveau de bruit, etc).
3. Contexte du réseau: décrire l'environnement du réseau (connectivité, bande passante, protocole, etc).
4. Contexte d'activité: répertorier les évènements qui se sont déroulés dans l'environnement (entrer d'une personne, sortir, etc.).
5. Contexte matériel: définir le profil et les activités des appareils (identifications, emplacement, durée de vie de la batterie, etc.).
6. Contexte de service: présenter les informations sur les fonctions que le système peut fournir (format de fichier, affichage, etc).

Selon El Ghayam [38], on peut classer le contexte en quatre catégories:

1. Contexte utilisateur: décrire les préférences de l'utilisateur (la résolution de l'écran voulu, le terminal choisi, etc).
2. Contexte physique: définir les paramètres physiques de l'environnement du terminal (mémoire disponible, vitesse du processeur, etc.) et du réseau de connexion (bande passante, mode de connexion, etc.). Aussi, représenter la mesure de l'environnement physique (humidité, température, niveau de bruit, etc).

3. Contexte spatio-temporel: représenter les informations de l'emplacement de l'utilisateur et leurs voisinages. De plus, garder une référence temporelle des actions courantes de l'utilisateur et garder la trace.

4. Contexte organisationnel: définir les informations relatives à la collaboration dans laquelle un utilisateur est engagé (le groupe auquel il appartient, le rôle, l'activité en cours, etc).

2.2 Sensibilité au contexte

La sensibilité au contexte est un domaine qui étudie les méthodes et les outils pour découvrir, modéliser et consommer des informations contextuelles. Ces informations peuvent inclure toute information affectant l'interaction d'un utilisateur avec un système, comme l'emplacement de l'utilisateur, l'heure de la journée, les personnes et les appareils à proximité, l'activité de l'utilisateur, les conditions d'éclairage ou de bruit, etc [89]. En général, la sensibilité au contexte (Context Awareness) consiste à réagir proprement en prenant en compte l'information de contexte. Le terme "Context-Awareness" est utilisé pour la première fois en 1994 par Schilit et al [113]. Ils considèrent que les applications sensibles au contexte sont « des applications ayant des mécanismes de changer dynamiquement ou d'adapter leurs comportements en se basant sur le contexte de l'application ou de l'utilisateur » [38]. Selon Dey [34], un système est sensible au contexte s'il utilise le contexte pour fournir des informations et/ou des services pertinents à l'utilisateur, dont la pertinence dépend de la tâche de l'utilisateur. Ryan et al. [38] définissent la sensibilité au contexte en tant que « l'aptitude de capturer, interpréter et répondre aux aspects de l'environnement local de l'utilisateur et de terminal». Selon la façon de l'utilisation de contexte, Chen [27] donne deux définitions de la notion de sensibilité au contexte. Premièrement, sensibilité active au contexte: une application qui s'adapte automatiquement au contexte découvert, en modifiant le comportement de l'application. Deuxièmement, sensibilité passive au contexte: une application présente le nouveau contexte ou celui mis à jour à un utilisateur intéressé où rend le contexte persistant pour que l'utilisateur puisse le récupérer ultérieurement. Selon Rohn, une application sensible au contexte diffère d'une application classique car ce sont des: [102]

- **Systèmes adaptatifs**: ceux-ci apprennent les préférences de leurs utilisateurs et s'adaptent en conséquence.
- **Systèmes réactifs**: ils anticipent les besoins de l'utilisateur dans un environnement instable.
- **Systèmes pro-actifs**: ceux-ci sont dirigés par les objectifs, capables de prendre l'initiative, plutôt que de simplement réagir à l'environnement.

- **Systèmes autonomes**: ceux-ci peuvent agir indépendamment, sans intervention humaine.

Miraoui et Tadj [79] considère qu' « un système est dit sensible au contexte s'il peut changer automatiquement ces formes des services ou déclencher un service comme réponse au changement de la valeur d'une information ou d'un ensemble d'informations qui caractérisent le service.». El Ghayam définit la sensibilité au contexte comme étant «la réaction du système suite à un contexte survenu. Cette réaction se fait en réalisant une compatibilité entre le sens de l'action et le sens particulier apporté par le contexte. Le sens de l'action peut être défini par l'objet de l'action, son objectif et les moyens disponibles pour invoquer cette action. La compatibilité est faite en contraignant le processus de construction et/ou d'exécution de ses actions» [38]. Pascoe propose un ensemble de capacités génériques de base qu'on peut utiliser comme vocabulaire pour identifier et décrire un système sensible au contexte [88]:

- **Détection contextuelle**: La détection contextuelle est le niveau le plus bas de sensibilité au contexte. Le système identifie différents états environnementaux et les propose aux utilisateurs.
- **Adaptation contextuelle**: Ce n'est pas seulement l'utilisateur qui peut être intéressé par les données contextuelles d'un tel système. Les applications peuvent utiliser ces connaissances contextuelles en adaptant leur comportement pour s'intégrer de manière plus transparente à l'environnement de l'utilisateur.
- **Détection de ressources contextuelles**: Par l'utilisation de ces informations, le système peut trouver d'autres ressources dans le même contexte que lui et utiliser ces ressources tout en restant dans le même contexte.
- **Augmentation contextuelle**: Les informations contextuelles peuvent être utilisées pour détecter, réagir et interagir avec l'environnement. L'augmentation contextuelle étend ses capacités en augmentant l'environnement avec des informations supplémentaires.

2.3 Architecture générale d'une application sensible au contexte

La conception des applications sensibles au contexte est bien différente de celle des applications classiques. Les applications classiques traitent des variables internes ou des données explicites des utilisateurs. Pour les applications sensibles au contexte en plus des variables et données des utilisateurs, elles sont obligées de manipuler aussi les informations de contexte. Par conséquent, il faut gérer les difficultés de capture du contexte qui est généralement distribué et mène à des conceptions complexes. Plusieurs architectures pour

les applications sensibles au contexte ont été proposé dans la littérature. Cependant, elles se basent toutes sur l'architecture proposée par Dey [35]. En effet Dey a été le premier qui a proposé la séparation entre l'acquisition du contexte et son utilisation dans les applications. En général, les déférentes architectures proposées assurent la séparation entre la capture des informations de contexte et l'utilisation de ces informations afin d'assurer l'extensibilité et la réutilisation du système [38]. Comme le montre la figure 2.1, nous remarquons que la majorité des architectures proposées se basent sur quatre couches [35, 23, 61, 79, 13]: Capture de contexte, Interprétation de contexte, Gestion de contexte et Adaptation au contexte.

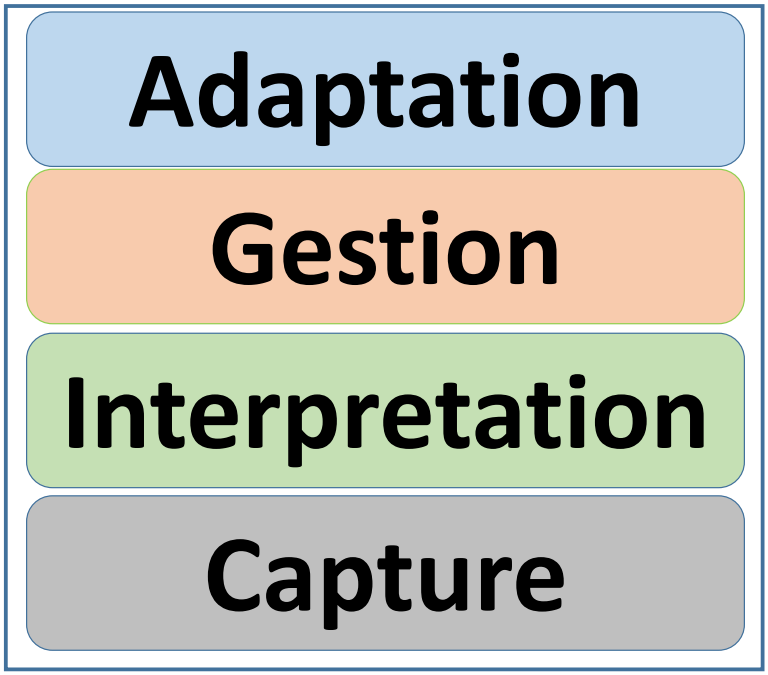

Figure 2.1: Architecture de systémes sensibles au contexte.

2.3.1 Capture de contexte

Les systèmes sensibles au contexte sont créés pour être à l'écoute aux changements de l'environnement du système. La première couche d'une architecture sensible au contexte est composée d'un ensemble de capteurs. Il existe différents types de capteurs qui peuvent être utilisés pour acquérir le contexte. Selon Indulska [54], les capteurs peuvent être divisés en trois catégories: physique, virtuel et logique.

1. **Capteur physique**: Les capteurs physiques sont des dispositifs matriciels capables de mesurer des grandeurs physiques et générer eux-mêmes des données de capteurs. Ce sont les types de capteurs les plus utilisés et ils sont tangibles. La plupart des appareils que nous utilisons aujourd'hui sont équipés d'une variété de capteurs (par exemple la température, l'humidité, le microphone, GPS). Les données récupérées des capteurs physiques sont appelées contexte de bas niveau.

2. **Capteur virtuel**: Ces capteurs n'ont pas de présence physique et ne génèrent pas eux-mêmes des données de capteurs. Les capteurs virtuels se basent sur les applications et les services pour fournir des informations contextuelles. Ils récupèrent les données de nombreuses sources et les publient en tant que données de capteurs (par exemple, calendrier, répertoire des numéros de contact, statuts Twitter, applications de chat et de messagerie électronique). Ils utilisent généralement la technologies des services Web pour envoyer et recevoir des données.

3. **Capteur logique**: Ils combinent des capteurs physiques et des capteurs virtuels afin de produire des informations plus significatives. Un service Web dédié à la fourniture d'informations météorologiques peut être considéré comme capteur logique. Les stations météorologiques utilisent des milliers de capteurs physiques pour collecter des informations météorologiques. Ils collectent également des informations à partir de capteurs virtuels tels que des cartes, des calendriers et des données historiques. Enfin, les informations météorologiques sont produites en combinant les capteurs physiques et virtuels. Les capteurs logiques permettent de fournir des informations de haut niveau et faciles à manipuler par l'application.

2.3.2 Interprétation de contexte

Cette couche offre des moyens de transformation et d'interprétation des données contextuelles générées par la couche de capture. Les données contextuelles brutes sont analysées et transformées en paramètres de plus haut niveau qui sont plus faciles à utiliser par l'application. Par exemple, une adresse postale peut être beaucoup plus significative que des coordonnées GPS brutes. L'interprétation peut être variée de simple agrégation de plusieurs données générées par les capteurs vers une analyse statistique complexe. Dans cette couche, plusieurs opérations peuvent être appliquées sur les données brutes tel que l'extraction, quantification, raisonnement, agrégation. Cependant, l'utilisation de plusieurs sources de contexte peut donner des résultats contradictoires ou des situations imprécises. Cette couche doit donc avoir des moyens et des techniques pour surmonter ces conflits. [23]

2.3.3 Gestion de contexte

Après avoir capturé et interprété le contexte dans les couches précédentes, dans cette couche, les informations contextuelles sont organisées, stockées et représentées formellement. Selon El Ghayam [38], le stockage peut être centralisé ou distribué. Le stockage centralisé est le plus facile et le plus utilisé puisqu'il permet de faire des mises à jour et des modifications simples sur les informations contextuelles. Mais, le problème de ce type de stockage réside dans les limitations de ressources de stockage caractérisants les équipements et les appareils mobiles utilisés dans la sensibilité au contexte. Le

stockage distribué permet de résoudre le problème de l'espace de stockage des appareils utilisés. Cependant, il impose des fonctions additionnelles et complexes relatives à la synchronisation et à l'actualisation des valeurs de contexte.

Avant de stocker une information contextuelle, il faut définir un modèle de représentation. Perera [93] présente les six techniques de modélisation de contexte les plus populaires: attribut-valeur, schémas de balisage, graphiques, basés sur des objets, basés sur la logique et basés sur l'ontologie.

1. **Attribut-Valeur**: Il s'agit de la forme la plus simple de représentation du contexte parmi tous les autres techniques. L'attribut représente un élément de contexte tel que la température. La valeur est la valeur actuelle de cette information. Ce modèle est facile à gérer lorsqu'il existe une plus petite quantité de données.

2. **Schémas de balisage**: Il modélise les données à l'aide de balises. L'avantage d'utiliser des balises est qu'il permet une récupération efficace des données. Les schémas de balisage tels que XML sont largement utilisés dans presque tous les domaines d'application pour stocker des données temporairement et transférer des données entre les applications.

3. **Graphiques**: Ces modèles consistent à modéliser les informations contextuelles selon un graphe conceptuel. Il modélise le contexte avec les relations. Quelques exemples de cette technique de modélisation sont le langage de modélisation unifié (UML) et la modélisation de rôle d'objet (ORM).

4. **Basée sur des objets**: La représentation basée sur des objets (ou orientés objet) est utilisée pour modéliser des données à l'aide des hiérarchies et des relations de classe. Le paradigme orienté objet favorise l'encapsulation et la réutilisation.

5. **Basés sur la logique**: Les faits, les expressions et les règles sont utilisées pour représenter des informations sur le contexte. La modélisation basée sur la logique permet d'extraire de nouvelles informations de contexte de haut niveau à l'aide d'un contexte de bas niveau. Par conséquent, il a la capacité d'améliorer d'autres techniques de modélisation de contexte en agissant comme un supplément.

6. **Basé sur l'ontologie**: Le contexte est organisé en ontologies utilisant des technologies sémantiques. Un certain nombre de normes (RDF, RDFS, OWL) et de capacités de raisonnement sont disponibles pour être utilisées en fonction des besoins.

2.3.4 Adaptation au contexte

L'adaptation au contexte est l'ensemble des mécanismes de réaction prévue suite aux changements de contexte. Pour assurer l'adaptation, il faut définir un ensemble de

règles d'adaptation. Ces règles sont implémentées selon des langages de programmation traditionnelles ou bien en utilisant la logique de prédicats. [38]

Selon Mansoor [72], l'adaptation de l'application au contexte peut être pilotée par quatre approches:

1. **Cadre conceptuel**: se concentre sur l'aspect architectural des systèmes sensibles au contexte et fournir des moyens pour faciliter la capture, l'interprétation et la transmission des données de contexte aux parties intéressées.

2. **Plateformes de services**: visent à fournir les services pertinents à l'utilisateur en fonction du contexte. Cela comprend la découverte dynamique de services et le déploiement dynamique de services adaptatifs répondant aux problèmes d'évolutivité, de sécurité et de confidentialité.

3. **Environnements des appareils**: essayent de donner des solutions au problème d'hétérogénéité en fournissant des techniques et des cadres d'interopérabilité.

4. **Environnements informatiques**: pour les applications omniprésentes, les environnements informatiques concentrent sur la conception de l'infrastructure physique et logique pour contenir les systèmes ubiquitaires.

2.4 Middlewares sensibles aux contextes pour l'IdO

Les systèmes sensibles au contexte sont constitués de plusieurs composants distribués tels que des capteurs, des actionneurs, des dépôts d'informations contextuelles, des processeurs d'informations contextuelles, etc. Aujourd'hui, il est largement admis que, pour réduire la complexité des applications sensibles au contexte et encourager leur réutilisation, des composants d'infrastructure supplémentaires sont souhaitables [107]. Généralement, un middleware masque les complexités du système ou du matériel, permettant au développeur d'applications de concentrer tous ses efforts sur la tâche à résoudre, sans la distraction de problèmes orthogonaux au niveau du système ou du matériel. Issarny [55] définit un middleware comme « une couche logicielle qui se situe entre le système d'exploitation et l'application et fournit des solutions réutilisables bien connues aux problèmes fréquemment rencontrés comme l'hétérogénéité, l'interopérabilité, la sécurité, la fiabilité». Un middleware fournit une couche logicielle entre les applications, le système d'exploitation et les couches de communication réseau, ce qui facilite et coordonne certains aspects du traitement coopératif. Un middleware pour l'IdO est requis pour les raisons suivantes: [15, 100]

- La difficulté de définir et d'appliquer une norme standard pour les différents appareils appartenant aux différents domaines de l'IdO.
- Le middleware agit comme un lien reliant les composants hétérogènes entre eux.

- Les applications de divers domaines nécessitent une couche d'abstraction et d'adaptation.
- Le middleware fournit une API pour les communications de la couche physique et les services requis pour les applications, cachant tous les détails de la diversité.

Avant de présenter les middlewares sensibles aux contextes les plus représentatifs et populaires, nous citons les exigences nécessaires que devrait prendre en charge par un middleware pour l'IdO.

2.4.1 Exigences de middleware pour l'IdO

Razzaque catégorise les exigences de middleware sensible au contexte pour l'IdO en deux classes [100]: service et architecture.

2.4.1.1 Exigences de service middleware IdO

Les exigences de service middleware pour l'IdO peuvent être classées comme fonctionnelles et non fonctionnelles.

1. **Exigences fonctionnelles**:
    - Découverte de ressources: L'hétérogénéité et la dynamicité de l'infrastructure et l'environnement de l'IdO rendent la découverte manuelle des ressources irréalisables, et par conséquent une exigence importante est la découverte automatique des ressources.
    - Gestion des ressources: toutes les applications exigent une QoS acceptable, et dans un environnement où les ressources sont limitées et ont un impact sur la QoS, comme l'IdO, il est important que les applications soient fournies avec un service qui gère ces ressources.
    - Gestion des données: Un middleware pour l'IdO doit fournir des services de gestion de données aux applications, y compris l'acquisition de données, le traitement de données et le stockage de données.
    - Gestion des événements: Les applications de l'IdO génèrent un énorme nombre d'événements, qui devraient être gérés en tant que partie intégrante d'un middleware de l'IdO. La gestion des événements transforme les événements simples observés en événements significatifs.
    - Gestion du code: le déploiement du code dans un environnement IdO est difficile et devrait être directement pris en charge par le middleware. En particulier, des services d'allocation et de migration du code sont nécessaires.
2. **Exigences non fonctionnelles**:

- Évolutivité: un middleware pour l'IdO doit être évolutif pour s'adapter à la croissance du réseau, des applications et des services de l'IdO.
- Temps réel ou opportunité: un middleware doit fournir des services en temps réel lorsque l'exactitude d'une opération qu'il prend en charge dépend non seulement de son exactitude logique, mais également de l'heure à laquelle elle est exécutée.
- Fiabilité: un middleware doit rester opérationnel pendant la durée d'une mission, même en présence de pannes.
- Disponibilité: un middleware prenant en charge les applications de l'IdO doit être disponible à tout moment. Même s'il y a une défaillance, son temps doit être suffisamment petits pour atteindre la disponibilité souhaitée.
- Sécurité et confidentialité: dans le middleware de l'IdO, la sécurité doit être prise en compte dans tous les blocs fonctionnels et non fonctionnels, y compris l'application au niveau utilisateur. Comme la sécurité, chaque bloc de middleware, qui utilise des informations personnelles, doit préserver la confidentialité du propriétaire.
- Déploiement simple: étant donné qu'un middleware IdO est généralement déployé par l'utilisateur, le déploiement ne doit pas nécessiter de connaissances ou de support technique complexe.

2.4.1.2 Exigences architecturales de middleware IdO

Les développeurs d'applications de l'IdO doivent prendre en charge les exigences architecturales. Dans ce qui suit, nous allons lister les exigences architecturales:

- Abstraction de programmation: Fournir une API aux développeurs d'applications est une exigence fonctionnelle importante pour tout middleware. Pour le développeur d'applications, les interfaces de programmation de haut niveau doivent séparer entre le développement des applications et les opérations fournies par les infrastructures hétérogènes de l'IdO.
- Interopérabilité: un middleware doit fonctionner avec des appareils, des technologies et des applications hétérogènes, sans effort supplémentaire de la part du développeur de l'application ou du service. Les composants hétérogènes doivent pouvoir échanger des données et des services. L'interopérabilité dans un middleware peut être vue à partir de perspectives réseau, syntaxiques et sémantiques, chacune devant être prise en charge dans l'IdO.
- Basée sur les services: une architecture de middleware doit être basée sur les services pour offrir une grande flexibilité lorsque des nouvelles fonctions doivent être ajoutées au middleware IdO. Un middleware basé sur les services fournit des abstractions pour le matériel complexe via un ensemble de services requis par les applications.

- Adaptation: un middleware doit être adaptatif pour pouvoir évoluer afin de s'adapter aux changements de son environnement ou de ses circonstances. Pour garantir la satisfaction des utilisateurs et l'efficacité de l'IdO, un middleware doit s'adapter dynamiquement ou s'ajuster pour s'adapter à toutes ces variations.
- Sensibilité au contexte: la sensibilité au contexte est une exigence clé dans la construction de systèmes adaptatifs. L'architecture middleware de l'IdO doit être sensible au contexte des utilisateurs, des appareils et de l'environnement et les utiliser pour proposer des services efficaces aux utilisateurs.
- Autonomie: les appareils, les technologies et les applications sont des participants actifs dans l'IdO et ils devraient être capables d'interagir et de communiquer entre eux sans intervention humaine directe.
- Distribution: généralement les composants de l'infrastructure de l'IdO sont distribués géographiquement, et donc une vue centralisée ne sera pas suffisante pour prendre en charge de nombreux services ou applications distribuées. Une implémentation de middleware doit prendre en charge des fonctions réparties sur l'infrastructure physique de l'IdO.

2.4.2 Middlewares sensibles aux contextes

Middleware dans l'IdO est un domaine de recherche très actif. De nombreuses solutions ont été proposées et mises en œuvre, notamment ces dernières années. Cette section fournit une vue d'ensemble des architectures middlewares sensibles aux contextes les plus représentatifs et populaires.

1. **AURA:** est une architecture appropriée pour l'informatique ubiquitaire. Il est basé sur l'idée d'Aura Personnelle et agit comme un proxy pour l'utilisateur qu'il représente. Lorsque l'environnement de l'utilisateur change, Aura prend en charge les tâches d'utilisateurs en s'adaptant aux ressources locales. Aura fourni des services pour la gestion des tâches, des applications et du contexte. Aura se compose de quatre composants principaux: observateur de contexte (collecte le contexte et l'envoie aux gestionnaires de tâches), gestionnaire de tâches (responsable du contrôle et migration des tâches), gestionnaire d'environnement (gère les fournisseurs de contexte et les services connexes) et les fournisseurs de contexte (fournit des informations de contexte). [62, 107, 93]
2. **CARISMA:** est un middleware informatique mobile qui exploite le principe de la réflexion pour améliorer la construction d'applications mobiles adaptatives et sensible au contexte. Dans CARISMA, les profils existent en tant que métadonnées du middleware pour chaque application. Les profils comprennent des parties passives et actives. Dans les parties passives, les actions que le middleware doit prendre en

réponse à des événements de contexte spécifiques sont décrites. Les informations actives spécifient les relations entre les services utilisés par l'application et les règles qui doivent être appliquées pour fournir ces services. Différentes conditions environnementales peuvent être spécifiées pour déterminer comment un service doit être fourni à l'application demandée. La réflexion peut être utilisée par l'application à tout moment pour modifier le profil conservé par le middleware. De plus, un mécanisme de résolution des conflits est également introduit dans CARISMA, basé sur des techniques macroéconomiques. [22]

3. **Gaia:** est une infrastructure middleware distribuée qui coordonne les entités logicielles et le réseau des appareils hétérogènes existe dans un espace physique. Gaia est conçu pour prendre en charge le développement et l'exécution d'applications portables pour les environnements informatiques ubiquitaires dans lesquels les utilisateurs interagissent avec plusieurs appareils et services. Gaia exporte des services pour interroger, accéder et utiliser les ressources et le contexte existants, et fournit un cadre pour développer des applications mobiles centrées sur l'utilisateur, sensibles aux ressources, multi-appareils et sensible au contexte. Les ontologies sont utilisées pour représenter les informations de contexte. L'architecture de Gaia comprend six composantes clés: fournisseur de contexte (acquisition de données à partir de capteurs ou d'autres sources de données), consommateur de contexte (différentes parties intéressées par le contexte), synthétiseur de contexte (générer des informations de contexte de haut niveau à l'aide de contextes brut de bas niveau), service de recherche du fournisseur de contexte (maintient un registre détaillé des fournisseurs de contexte afin que les fournisseurs de contexte appropriés puissent être trouvés en fonction de leur capacité si nécessaire), service d'historique de contexte (stocke l'historique du contexte) et serveur d'ontologie (maintient différentes ontologies). [103, 93]

4. **SOCAM:** est un middleware sensible au contexte basé sur l'ontologie. Il sépare les ontologies en deux niveaux: l'ontologie de niveau supérieur pour les concepts généraux et l'ontologie de niveau inférieur pour les descriptions spécifiques. Les principales tâches incluses sont l'obtention du contexte à partir de diverses sources, l'interprétation et le partage du contexte. L'architecture SOCAM comprend plusieurs composantes clés: fournisseur de contexte (acquiert des données de capteurs et d'autres sources de données internes et externes et convertit le contexte en représentation OWL), interpréteur de contexte (effectue le raisonnement à l'aide du moteur de raisonnement et stocke les informations de contexte traitées dans la base de connaissances), services contextuels (consommateurs de contexte), et services de localisation de services (les fournisseurs et interpréteurs de contexte sont autorisés à s'enregistrer pour que d'autres composants puissent rechercher des fournisseurs et

des interpréteurs appropriés en fonction de leur capacité). [62, 107, 93, 44]

5. **GSN:** Le middleware Global Sensor Networks (GSN) fournit une plate-forme uniforme pour l'intégration et le déploiement de réseaux de capteurs hétérogènes en introduisant l'abstraction de capteurs virtuels, qui spécifie toutes les informations nécessaires à son utilisation et à son déploiement. GSN utilise des capteurs virtuels pour contrôler la priorité de traitement, la gestion des ressources et des données stockées. À l'aide de spécifications déclaratives, les capteurs virtuels peuvent être déployés et reconfigurés dans des conteneurs GSN au moment de l'exécution. GSN crée des environnements de traitement hautement dynamiques et permet au système de réagir rapidement à l'évolution des besoins de traitement et des conditions environnementales. La gestion dynamique des ressources accomplit trois tâches principales: le partage des ressources, la gestion des pannes et le contrôle explicite des ressources. Si le nombre de clients augmente, le temps de traitement moyen pour chaque client diminue, ce qui assure l'évolutivité. L'architecture de GSN suit un modèle basé sur les conteneurs, dans lequel chaque conteneur héberge et gère un certain nombre de capteurs virtuels simultanément. GSN fournit un accès simple et uniforme à la multitude de technologies hétérogènes disponibles et est facile à déployer. GSN est adaptatif, mais il n'est pas autonome et n'offre pas de support pour l'interopérabilité, la sécurité ou la confidentialité. [100, 2, 26]

6. **e-SENSE:** présente une architecture configurable pour un système WSN qui est capable de fournir des informations de contexte des capteurs à l'utilisateur dans différents environnements d'application. Il permet aussi son intégration efficace dans des systèmes de communication mobiles de troisième génération (3G), en particulier dans le sous-système multimédia IP. e-SENSE permet aux systèmes ambiants intelligents d'utiliser les réseaux multiplicateurs sans fil afin de mettre des informations contextuelles à la disposition des applications et des services. e-SENSE combine les réseaux de capteurs corporels (BSN), les réseaux de capteurs d'objets (OSN) et les réseaux de capteurs d'environnement (ESN) pour capturer le contexte dans le paradigme de l'IdO. Les fonctionnalités requises par les solutions middleware sensible au contexte pour l'IdO sont identifiées comme la capture de données de capteurs, préfiltrage des données, intégration de la source de données d'abstraction de contexte, extraction de contexte, moteur de règles, et l'adaptation. [42, 93]

7. **DMS-CA:** L'architecture de contexte du système de gestion des données (DMS-CA) a été conçue pour fournir des services contextuels dans un environnement du bâtiment intelligent. C'est une partie spécialisée d'un système général de gestion des données (DMS) qui gère les données des réseaux sans fil dans un environnement omniprésent. XML est utilisé pour définir des règles, des contextes et des services. En outre, une technique de vérification des règles pilotées par les événements est utilisée

pour raisonner le contexte. Les règles peuvent être configurées par des appareils mobiles et les envoyer au serveur pour être utilisées par le moteur de vérification des règles. Fournir une interface mobile pour créer des règles et des requêtes est important dans un environnement dynamique et mobile tel que l'IdO. L'architecture de DMS-CA comprend quatre couches: La couche la plus basse contient les contextes et services réels qui sont enregistrés pour une application particulière; la couche des appareils mobiles contient tous les appareils mobiles interagissant avec le serveur; la couche serveur des appareils mobiles est comme un contrôleur qui interagit avec les clients distribués; la couche du moteur de règles vérifie le contexte en fonction de toutes les règles des utilisateurs et offre des services aux utilisateurs une fois que les conditions des règles sont respectées. [51, 93]

8. **SOCRADES:** (Service-Oriented Cross-layer Infrastructure for Distributed smart Embedded devices) est un projet de recherche européen portant sur le paradigme de fabrication basé sur SOA. Le middleware SOCRADES abstract les objets physiques en tant que services à l'aide de profils de périphérique pour les services Web (DPWS). Son objectif principal est de développer une plate-forme de conception, d'exécution et de gestion pour la prochaine génération des systèmes industriels, en exploitant l'architecture orientée services au niveau des appareils et des applications. SOCRADES créera de nouvelles méthodologies, technologies et outils pour la modélisation, la conception, la mise en œuvre et l'exploitation de systèmes en réseau constitués de dispositifs embarqués intelligents. Son architecture se compose d'une couche pour les services d'application (par exemple, le stockage d'événements) et d'une couche pour les services d'appareils (par exemple, le gestionnaire et le moniteur d'appareils, la découverte de services, la gestion du cycle de vie des services). [21]

9. **HYDRA:** est un middleware pour l'IdO qui vise à intégrer des appareils sans fil et des capteurs dans les systèmes ambiants intelligents. Hydra comprend un cadre contextuel (CAF). CAF offre les capacités d'un raisonnement puissant et de haut niveau, basé sur l'utilisation d'ontologies et d'un traitement sémantique de bas niveau basé sur une approche orientée objet et valeur-clé. Les CAF se composent de deux composants principaux: le composant d'acquisition de données (responsable de la connexion et de la récupération des données des capteurs) et le gestionnaire de contexte (responsable de la gestion du contexte, de la sensibilité du contexte et de l'interprétation du contexte). Un moteur de règles appelé plate-forme Drools a été utilisé comme mécanisme principal de raisonnement contextuel. CAF modélise trois types de contexte distincts: les contextes de périphériques (par exemple, la source de données), les contextes sémantiques (par exemple l'emplacement, l'environnement et l'entité) et les contextes d'application (par exemple, spécifiques au domaine). Hydra identifie le moteur de règles de raisonnement contextuelles, le stockage de contexte,

l'interrogation de contexte et la gestion des événements et des actions comme les composantes clés d'un cadre sensible au contexte. [93, 12]

10. **Ubiware:** est un middleware qui répond directement aux exigences de l'IdO et qui prend en charge la création de systèmes industriels autonomes, complexes, flexibles et extensibles. Leur principe est de prendre en charge la découverte, la surveillance, la composition, l'appel et l'exécution automatiques des ressources de différentes applications. Un agent Ubiware est réparti sur trois couches: un moteur de comportement implémenté en Java, une couche intermédiaire déclarative (modèles de comportement correspondant aux rôles d'agent) et une troisième couche, qui contient des ressources partagées et réutilisables interprétées comme des composants Java (capteurs, actionneurs, machines et appareils intelligents, RFID, services Web, etc.). L'interopérabilité est obtenue par l'adaptation sémantique et par l'affectation d'un agent pro-actif à chacune des ressources. Ceci est pris en charge en utilisant des métadonnées et des ontologies. Cependant, la prise en charge de l'interopérabilité est limitée. Par exemple, il ne couvre pas l'interopérabilité entre différents protocoles de découverte de ressources. [100, 26]

11. **UbiROAD:** est un middleware sémantique pour les environnements de routes intelligentes sensibles au contexte. Il traite de l'interopérabilité entre les voitures et les dispositifs hétérogènes en bordure de route. L'interopérabilité sémantique est réalisée par deux couches: l'interopérabilité au niveau des données et l'interopérabilité et la coordination au niveau du protocole fonctionnel. UbiROAD est une plate-forme spécialisée pour les environnements de trafic intelligents, mais peut également servir comme protocole intelligent entre la couche d'appareils routiers intelligents et les futures architectures orientées services. Il est hétérogène en ce qui concerne les composants, les normes, les formats de données et les protocoles. Il est auto-adaptatif en déployant des agents distribués et garantit la sensibilité du contexte et la composition adaptative et reconfigurable. Ces exigences sont atteintes par la customisation, la personnalisation, le comportement dynamique et l'autonomie des services. La gestion de confiance autonome est réalisée via une annotation sémantique. UbiROAD garantit un haut niveau de sécurité. [100, 116]

12. **Octopus:** est un système open source extensible dynamiquement qui prend en charge la gestion et la fusion des données pour les applications de l'IdO. Octopus développe des abstractions de middleware et des modèles de programmation pour l'IdO. Il permet aux développeurs non spécialisés de déployer des capteurs et des applications sans connaissances détaillées des techniques et des réseaux. Octopus se concentre sur le domaine des maisons et bureaux intelligents et son composant principal est Solver. Solver est un module qui effectue des opérations de fusion de données de capteurs. Solvers peuvent être ajoutés et supprimés du système

à tout moment en fonction des besoins. D'autres Solvers peuvent être combinés dynamiquement pour créer des opérations complexes. [41, 93]

2.5 Aide à la décision multicritère

Dans cette section nous présentons quelques éléments de base de l'aide à la décision multicritère ainsi que les différentes catégories des méthodes d'optimisation multicritères.

2.5.1 Définition de l'aide à la décision

Selon Mintezberg [115], la décision est définie comme «l'engagement dans une action, c'est a dire une intention explicite d'agir». Le but est d'aider les personnes et les organisations de résoudre leurs problèmes. Dans ce qui suit, nous allons expliquer la notion d'aide à la décision.

La plupart des travaux d'aide à la décision adoptent la définition de Roy (1985). Il définit l'aide à la décision comme «l'activité de celui qui, prenant appui sur des modèles clairement explicités mais non nécessairement complètement formalisés, aide à obtenir des éléments de réponse aux questions que se pose un intervenant dans un processus de décision, éléments concourant à éclairer la décision et normalement à prescrire, ou simplement à favoriser, un comportement de nature à accroître la cohérence entre l'évolution du processus d'une part, les objectifs et le système de valeurs au service desquels cet intervenant se trouve placé d'autre part.» [106]. Selon cette définition, Chakhar [25] a remarqué qu'une démarche d'aide à la décision s'articule autour d'un processus de décision. Dans le même sens, Landry (1998) a estimé que le succès d'une activité d'aide à la décision nécessite une compréhension de l'ensemble du processus décisionnel dans lequel elle se déroule. Cela signifie être capable de bien comprendre le problème qui justifie l'origine et qui alimente ce processus plus tard [25]. Martel (1999) soutient la définition de Roy et dit qu'une activité d'aide à la décision "implique un minimum d'insertion dans le processus de décision: elle se fait essentiellement avec les acteurs du processus dans l'établissement d'une véritable relation d'aide" [10].

2.5.2 Problématique de la décision

Roy (1985) a été précisé quatre problèmes fondamentaux que tout problème de décision multicritères doit se ramener nécessairement à l'un d'entre eux. Les problématiques peuvent être classées comme suit: [106, 25]

- **Problématique du choix (P.α)**: Elle consiste à indiquer un sous-ensemble de l'ensemble A, contenant les meilleures options. L'idéal est d'obtenir la meilleure option. Mais à cause de la nature conflictuelle des critères, il est mieux d'offrir au décideur quelques options qui donnent différentes variantes de la meilleure option.

À la fin, le décideur doit raffiner le résultat final. Formellement, le résultat est un sous-ensemble $A_0 \subset A$. Exemple: choix d'un service web où chaque service représente une action.

- **Problématique de tri (P.β)**: Elle consiste à classer chaque option à une classe prédéfinie. Cette formulation est réalisable lorsque le problème de décision va traiter chaque option séparément des autres. L'objectif est de définir une suggestion parmi un ensemble des suggestions spécifiées en avance. Chaque suggestion peut être rattachée à une classe. La procédure de tri doit être établie de telle sorte que chaque option est placée à une et seule classe. Formellement, une prescription consiste à une partition de A. Exemple: Attribuer des images à différentes catégories de maladies.
- **Problématique de rangement (P.γ)**: Elle permet d'ordonner les différentes options de la meilleure option à la moins bonne. L'idéal est de faire un ordre total. Cependant, à cause de la nature conflictuelle des critères, à l'imprécision, à l'existence de systèmes de valeurs différents, il est généralement plus pratique d'offrir au décideur un ordre partiel. Formellement, la prescription est un ordre partiel, i.e., une relation transitive définie sur A (ou un sous-ensemble de A). Exemple : ordonner les différents services web répondants à une requête utilisateur.
- **Problématique de description (P.δ)**: Elle permet de présenter les options et leurs conséquences. Alors, il n'existe pas une prescription et la procédure d'investigation est cognitive. Exemple : déterminer et dénoncer, dans le seul objectif de découvrir et se de préparer à des potentiels événements, les différentes options possibles pour lutter contre la pollution.

Les caractéristiques des quartes types de problématiques sont résumées dans le tableau 2.1.

Problème	**Objectif**	**Procédure**	**Prescription**
P.α	Sélectionner un sous ensemble	Sélection	Sous ensemble $A_0 \subset A$
P.β	Affecter chaque action de A à une catégorie	Segmentation	Partition de A
P.γ	Ordonner les actions de A	Classement	Ordre partiel sur A
P.δ	Décrire les actions et leurs conséquences	Cognitive	Aucune prescription

Tableau 2.1: Les différentes problématiques de décision [25]

2.5.3 Définition de l'aide à la décision multicritère

L'aide à la décision multicritère est élaborée pour fournir une démarche et des outils permettant de résoudre les problèmes décisionnels complexes. Ainsi, l'analyse multicritère représente l'un des axes les plus intéressants de la recherche opérationnelle et des théories de la décision. En effet, on parle de l'aide à la décision multicritère lorsque la démarche de l'aide à la décision se base sur une approche multicritère. Pratiquement, l'aide à la

décision multicritère est inventée pour traiter plusieurs types de problèmes de décision (choix, tri, description, rangement...) tout en considérant plusieurs critères (attributs), souvent conflictuels et non commensurables. [10]

Vincke définit l'aide à la décision multicritère comme étant « L'aide multicritère à la décision vise, comme son nom l'indique, à fournir à un décideur des outils lui permettant de progresser dans la résolution du problème de décision ou plusieurs points de vue, souvent contradictoires, doivent être pris en compte » [125]. Selon Maystre [75], l'aide à la décision multicritère «est une analyse ayant pour but d'expliciter une famille cohérente de critères permettant d'appréhender les différentes conséquences d'une action».

2.5.4 Méthodes d'optimisation multicritère

Les approches de résolution des problèmes multicritères peuvent être réparties en classes suivantes :

2.5.4.1 Les méthodes Métaheuristiques

Les métaheuristiques sont des algorithmes d'optimisation. Elles sont généralement inspirées par des phénomènes naturels, qu'ils soient pris en physique (cas du recuit simulé), en biologie de l'évolution (cas des algorithmes génétiques) ou encore en éthologie (cas des algorithmes de colonies de fourmis). Elles visent à résoudre les problèmes d'optimisation difficiles. Dans ce qui suit, nous allons présenter les méthodes métaheuristiques de base les plus populaires.[52, 96]

- **Méthode de Recuit Simulé**: est une méthode inspirée de la métallurgie. Cette métaheuristique est fondée sur une technique employée par les métallurgistes qui, pour aboutir un alliage sans défaut, faisant alterner les cycles de réchauffage (ou de recuit) et de refroidissement lent des métaux (évolution d'un système thermodynamique). Il permet de recherche locale normale, suivant une approche pour échapper les minima locaux. L'idée du recuit simulé est d'explorer de façon itérative l'espace des solutions.[52, 96]

- **Méthode de recherche Tabou**: est une métaheuristique permet de résoudre des problèmes d'optimisation combinatoire. Elle découverte d'une manière flexible un compromis entre la qualité de la solution et le temps de calcul. L'objectif de cette méthode est d'offrir au processus de recherche une mémoire flexible qui permet de réaliser une sélection plus "intelligente" dans l'espace des solutions. Elle regroupe une démarche de sélection locale avec un ensemble de règles et de mécanismes pour éviter l'obstacle des optima locaux.[49]

- **Algorithmes Génétiques**: sont des algorithmes de recherche inspirés des mécanismes de l'évolution naturelle des êtres vivants et de la génétique. Ils sont

des algorithmes d'optimisation stochastique basés sur les mécanismes de la sélection naturelle et de la génétique. Leur fonctionnement est extrêmement simple, on part d'une population de solutions potentielles (chromosomes) initiales, arbitrairement choisies. On évalue leur performance (Fitness) relative. Sur la base de ces performances on crée une nouvelle population de solutions potentielles en utilisant des opérateurs évolutionnaires simples : la sélection, le croisement et la mutation.[52]

D'autres métaheuristiques ont montré leurs puissances pour l'optimisation multiobjectif telle que: les colonies de fourmis, l'optimisation par essaim de particules et les réseaux de neurones artificiels. [49, 52, 96]

2.5.4.2 Les méthodes non Pareto

Les méthodes non Pareto traitent les problèmes comme des problèmes mono-objectif. Elles consiste à transférer le problème initial à un ou plusieurs problèmes mono-objectifs. Elles permettent de traiter les objectifs séparément. Parmi ces approches nous citons :

- **Les méthodes Agrégées**: Ces méthodes convertirent un problème multiobjectif en un problème mono-objectif. Dans ce qui suit, nous allons mentionner les plus célèbres:[49]
  - Méthodes d'agrégation par pondération: Cette méthode permet de rassembler tous les objectifs en attribuant à chacun d'eux un poids de pondération. Ce poids indique l'utilité relative que le décideur affecté à l'objectif.
  - Méthode ϵ-contraintes: Elle convertit un problème d'optimisation multiobjectif en un problème d'optimisation mono-objectif. Au début, il faut sélectionner un objectif à optimiser prioritairement. Par la suite, un vecteur de contraintes initiales est sélectionné. Enfin, le problème est transformé en gardant l'objectif prioritaire et en transformant les autres objectifs en contraintes d'inégalités.
  - Méthode de but à atteindre: Dans cette approche le décideur détermine un ensemble des objectifs qu'il désire réaliser et les poids associés. L'algorithme essaye de réduire l'intervalle entre la solution actuel et ses objectifs. Cette méthode employe un point de référence pour diriger la recherche.
  - Méthode de min-max: Cette méthode permet de minimiser l'intervalle relatif par rapport à un point de référence nommé objectif défini par le décideur.
  - Méthode de Goal programming: Dans cette méthode le décideur définit un but à réaliser pour chaque objectif. Ces buts sont ainsi insérés au problème comme des conditions supplémentaires. La nouvelle fonction objective est optimisée de façon à minimiser la somme des écarts entre les résultats et les buts à atteindre.
- **Méthode lexicographique**: c'est une méthode dans laquelle les buts sont initialement classés par ordre de priorité par le décideur. Les fonctions sont

optimisées progressivement, suivant le classement prédéfini. Ce classement permet de préciser le poids des objectifs. Ensuite, le minimum est atteint en optimisant tout d'abord la fonction objective la plus importante puis la deuxième et ainsi de suite. Plusieurs métaheuristiques ont été développées pour la résolution des problèmes multiobjectifs à choix lexicographique.[49, 96]

- **Vector evaluated genetic algorithm**: est une progression d'un algorithme génétique simple pour la résolution d'un problème multiobjectif dénommée. La seule différence avec un algorithme génétique simple est la façon dont se réalise le choix. L'idée est simple. Si nous disposons K objectifs et une population de N individus, une sélection de N/K individus est faite pour chaque objectif. Ainsi K sous populations vont être développés, chacune d'entre elles contient les N/K meilleurs individus pour un objectif particulier. Les K sous populations sont par la suite fusionnées afin d'atteindre une nouvelle population de taille N. L'algorithme s'achever par l'exécution des opérateurs génétiques de modification (croisement et mutation).[49, 96]

2.5.4.3 Les méthodes Pareto

La plupart des problèmes d'optimisation multicritère issue du monde réel possèdent plusieurs objectifs à optimiser simultanément et qui sont généralement conflictuels. Pour ces problèmes multiobjectifs, il n'y a pas, une meilleure solution, mais plutôt, un ensemble de solutions qui sont meilleures à d'autres en considérant tous les critères[65]. L'approche Pareto se base sur la notion de dominance pour sélectionner des solutions. Cette méthode offre une gestion équilibrée de chaque critère, car il n'y a pas d'ordre a priori de l'importance des critères. Ainsi, enfin de traitement, l'algorithme fournit un ensemble de solutions qui approchent le front de Pareto[96]. Un point est dit Pareto optimal s'il n'est dominé par aucun autre point. Ces points sont également appelés solutions non inférieures ou non dominées. La sélection de la solution optimale revient donc à l'utilisateur, qui doit identifier parmi l'ensemble obtenu la solution qui lui corresponde le mieux. Ces méthodes se sont constatées être les plus fiables donc, de nos jours, la plupart des algorithmes appliquent une méthode Pareto pour résoudre les problèmes multicritères. Les algorithmes Pareto les plus connus sont:[71]

- MOGA: Multiple Objective Genetic Algorithm.
- NPGA: Niched Pareto Genetic Algorithm.
- NSGA: Nondominated Sorting in Genetic Algorithm.
- SPEA: Strength Pareto Evolutionnary Algorithm.
- MOEA: Multi-Objective Evolutionary Algorithm.

2.5.4.4 Les méthodes hybrides

Afin d'optimiser les performances d'un algorithme, on essaye de le combiner avec d'autre algorithmes ou méthodes. Ce principe général appelé hybridation, peut s'appliquer à plusieurs méthodes. Un exemple spécifique de l'hybridation entre deux approches permet de fusionner un algorithme génétique avec un processus de recherche locale. Dans cette hybridation on remplace généralement la mutation par une méthode de recherche locale. Dans le cas des problèmes multicritères, on peut citer les méthodes hybrides suivantes:

- La méthode MOTS fusionnant une population et une recherche Tabou,
- La méthode PSA fusionnant un algorithme génétique et le recuit simulé,
- La méthode M-PAES utilisant un schéma généralisant l'implémentation d'un grand nombre d'algorithmes hybrides pour l'optimisation multicritère.

2.6 Problématique de la thèse

Bien que les middlewares présentés dans la section précédente répondent à de nombreux problèmes et exigences de l'IdO, il existe encore des défis de recherche ouverts. En particulier, des recherches sont nécessaires dans le domaine de la découverte et de la composition dynamique des ressources hétérogènes, de l'évolutivité, de la fiabilité, de l'interopérabilité, de la sensibilité au contexte, de la sécurité et de la confidentialité avec le middleware de l'IdO. Il est important de noter que la plupart des middlewares actuels traitent les WSN, tandis que d'autres perspectives (par exemple, M2M, RFID et IdO) sont rarement abordées [100]. Dans cette section, nous présentons la problématique que nous allons traiter dans cette thèse.

Récemment, le modèle capteur en tant que service (SaaS: Sensing as a service) a connu une croissance considérable et les données générées par ces capteurs peuvent être réutilisées par différents utilisateurs et applications au sein des solutions middlewares dans l'IdO. Par conséquent, la sélection des capteurs dans le modèle SaaS va être l'un des défis les plus difficiles des middlewares dans l'IdO. Cependant, tous les middlewares présentés dans la section précédente, possèdent une couche qui est responsable d'acquérir les données à partir des capteurs. Il est clair que nous allons avoir accès à des milliards de capteurs. Dans un tel environnement, il pourrait y avoir de nombreux capteurs alternatifs différents à utiliser. Par exemple, considérons une situation où un spécialiste de l'environnement veut mesurer la température à Alger. Il y a deux problèmes principaux: (1) «quels capteurs fournissent des informations sur la température?» (2) quand il y a plusieurs capteurs qui peuvent mesurer le même paramètre, «quel capteur devrait être utilisé?». Afin de répondre à la question (1), la connaissance du domaine doit être intégrée à la solution middleware de l'IdO. Il est évident que la sélection manuelle des capteurs qui fourniront des informations sur la température est impossible dans l'IdO en raison

de son échelle. Afin de répondre à la question (2), des cadres de qualité doivent être définis et utilisés. Une telle technique devrait être utilisée pour classer les capteurs en fonction de facteurs tels que la précision, la pertinence, les commentaires des utilisateurs, la fiabilité, le coût et l'exhaustivité. Des défis similaires ont été relevés dans le domaine des services web au cours de la dernière décennie, où nous pouvons tirer des leçons de ces efforts. Dans cette situation, le défi essentiel des utilisateurs est de savoir comment rechercher et sélectionner les capteurs appropriés pour résoudre leurs problèmes dans un délai raisonnable. Par conséquent, notre objectif est de proposer une méthode efficace pour rechercher et sélectionner les meilleurs capteurs afin d'aider les utilisateurs à acquérir les informations souhaitées.

Conclusion

Dans ce chapitre nous avons présenté un aperçu global sur la notion de sensibilité au contexte. Au début, nous avons clarifié la notion des informations de contexte dans le domaine informatique. Ensuite, nous avons donné un ensemble de définitions montrant les différentes visions de la sensibilité au contexte. Après, nous avons présenté les différentes couches architecturales des systèmes sensibles au contexte. Par la suite, nous avons présenté quelques middlewares sensibles aux contextes les plus représentatifs et populaires pour l'IdO. Nous avons ainsi expliqué la notion d'aide à la décision multicritère . Enfin, nous avons terminé par la définition de la problématique de cette thèse.

Dans le prochain chapitre, nous présenterons les différents travaux liés à notre travail. Nous allons expliquer les différentes phases et techniques de recherche et de sélection des capteurs utilisées dans les différentes approches existant dans la littérature. Ainsi, nous signalerons les limites et les différents défauts de chaque approche.

Chapitre 3

Travaux connexes

Introduction

Dans le chapitre précédent, nous avons vu un aperçu global sur la notion de sensibilité au contexte. Nous avons également expliqué les notions de l'information contextuelle, la sensibilité au contexte et les middlewares sensibles aux contextes. Enfin, nous avons précisé la problématique traitée dans cette thèse.

Année après année, l'Internet des objets (IdO) devient plus populaire. En raison de l'augmentation de la puissance de calcul, des capacités de communication et du faible coût, le nombre de capteurs déployés a augmenté et génère un énorme volume de données. Selon le site web de statistiques Statista [95], le nombre d'objets connectés dans le monde passera de 20 milliards en 2017 à 75 milliards en 2025. Le rôle principal de l'IdO est de permettre l'accès via Internet à un énorme nombre de capteurs hétérogènes qui fournissent une variété de données. De plus, une fois que nous pouvons rendre les données des capteurs disponibles via une plate-forme cloud interopérable (capteur en tant que service), le plus grand défi est de rechercher et sélectionner les capteurs pertinents et adéquats en fonction des besoins des utilisateurs au cas où il y aurait une grande quantité de capteurs disponibles à choisir et, souvent, avec des caractéristiques hétérogènes [70]. En fait, aujourd'hui, la plupart des moteurs de recherche Web n'intègrent pas les données IdO sur le Web, car les objets IdO peuvent avoir des descriptions complexes et leurs données générées peuvent avoir des propriétés représentées selon des dimensions thématiques, spatiales et temporelles [132]. De nombreuses solutions système middleware proposées [126, 3, 74] pour l'IdO ont été mises en place dans des environnements de recherche et industriels pour répondre à ce besoin. Tous les middlewares proposés doivent respecter les données pertinentes des capteurs sans demander aux utilisateurs de choisir manuellement les capteurs qui correspondent à leurs besoins. À cette fin, la recherche, la sélection et l'interaction avec les dispositifs restent un défi critique.

L'extraction des données de tous les capteurs existants est une opération coûteuse

en matière de calcul et du temps, particulièrement dans les applications qui nécessitent des données continues. Dans ce cas, le défi est d'identifier les meilleurs capteurs qui peuvent fournir les donnés répondants aux besoins utilisateurs. L'objectif des méthodes de recherche et de sélection de capteurs est de sélectionner les capteurs appropriés aux utilisateurs en fonction de leurs besoins. Suite à cette vision, Zhou et al. [132] ont classé les méthodes existantes en deux groupes: basées sur le contenu (par exemple, recherchent les capteurs qui génèrent une telle valeur) et basées sur le contexte (par exemple, la recherche est basée sur l'emplacement et le type des capteurs).

Dans ce chapitre, nous citons les métriques à utiliser pour l'évaluation des méthodes de recherche des capteurs. Ensuite, nous présentons les différentes méthodes et techniques de recherche et de sélection des capteurs existants dans la littérature. Ainsi, nous précisons leurs limites et manques afin de proposer des solutions plus efficaces.

3.1 Métriques d'évaluation des méthodes de recherche

L'Internet des objets est un environnement dynamique et évolutif composé de milliards d'objets inter connectés, par conséquent, la portée des méthodes de recherche sur l'IdO est large. Les travaux de recherche peuvent simplement se concentrer sur un aspect limité de ce vaste domaine, par exemple, l'évolutivité, la représentation des connaissances (pour permettre la recherche sémantique), la gestion des flux, la dynamicité, etc. Pour obtenir un aperçu rapide et approfondir des méthodes de recherche, il faut définir des métriques permettent d'évaluer l'efficacité des différentes méthodes de recherches proposées [132]. Ces métriques sont présentées comme suit:

- Format de données: indique la représentation des données.
- Approche d'accès: fait référence à la manière dont les clients de la fonctionnalité de recherche peuvent accéder aux résultats de la recherche.
- Type de recherche: fait référence aux techniques de recherche fondamentales sur la base desquelles les systèmes de recherche sont développés, telles que la recherche par mot-clé, la requête du type SQL (Structured Query Language), l'indexation, la recherche spatiale ou la requête continue.
- Échelle des expériences: indique l'échelle des expériences réalisées dans le travail de recherche particulier, par exemple, le nombre de capteurs et d'entités, ou la quantité de données, etc., si les informations sont disponibles.
- Dynamicité: il est utilisé pour indiquer si le mécanisme de recherche fournit un support pour gérer les problèmes causés par la haute dynamicité de l'environnement IdO.

- Architecture: indique si une plateforme de recherche soit conçue ou si les expériences sont réalisées de manière centralisée ou distribuée.
- Implémentation: indique quels langages et modèles de programmation sont utilisés pour implémenter les techniques de recherche.

3.2 Recherche basée sur le contenu

De nombreuses méthodes ont été développées pour la recherche des capteurs basée sur le contenu. En effet, une telle recherche conduit à découvrir des capteurs à partir des valeurs générées par les capteurs. Dans cette section, nous présentons les approches les plus célèbres de recherche basée sur le contenu.

3.2.1 Méthode de Elahi

Elahi et al. [39] ont essaie de résoudre le problème principal de réalisation d'un moteur de recherche pour l'IdO qui est la rechercher des capteurs basés sur le contenu. Était donné un grand nombre de capteurs, comment trouver efficacement un sous-ensemble de capteurs qui émettent une valeur recherchée à un moment donné. Ils introduisent une primitive appelée classement des capteurs qui permet la mise en œuvre d'une recherche efficace de capteurs basée sur le contenu. L'idée de base est de calculer une liste classée de capteurs de telle sorte que plus le rang d'un capteur dans cette liste est élevé, plus ce capteur correspond à la requête. De cette façon, un moteur de recherche peut traiter les capteurs dans l'ordre de leurs rangs, en effectuant des recherches d'abord sur les capteurs les plus susceptibles de correspondre à la requête. Elahi et al. [39] ont calculé la probabilité d'estimation pour trouver les capteurs dont les sorties produisent les mêmes résultats et pour traiter les capteurs dans l'ordre de probabilité décroissante. L'idée clé du classement des capteurs est d'exploiter la nature périodique des capteurs centrés sur les personnes en utilisant des modèles de prédiction appropriés. En utilisant ces modèles de prédiction, les capteurs peuvent être classés en fonction de leur probabilité de correspondre à une recherche de capteurs basée sur le contenu. L'utilisation de deux ensembles de données du monde réel, permet de montrer que le classement des capteurs peut améliorer considérablement les performances d'un moteur de recherche par rapport à une méthode de base.

3.2.2 Méthode Dyser

Le principal défi à relever lors de la construction d'un moteur de recherche pour l'IdO est la taille énorme et la dynamique extrême de l'espace de recherche. En revanche, la grande majorité du Web d'aujourd'hui est statique en ce sens que les pages Web sont modifiées à des intervalles de temps supérieurs à la vitesse de mise à jour des capteurs. Ainsi, un

moteur de recherche pour l'IdO doit prendre en charge la recherche du contenu structuré et en évolution rapide. Ostermaier et al. [86] ont proposé un moteur de recherche en temps réel pour l'IdO nommé Dyser. Il prend en charge la recherche d'entités du monde réel qui présente un certain état actuel tel qu'il est perçu par les capteurs. Le moteur de recherche Dyser répond au défi majeur de la recherche évolutive de contenu en changement rapide tout en tirant parti de l'infrastructure Web existante. Au cas où un utilisateur soumettrait une demande et afin d'identifier l'état réel, Dyser utilise les dernières données pour décider si elles correspondent ou non à la requête de l'utilisateur. Pour atteindre cet objectif, les modèles de prédiction permettent de trouver des capteurs adéquats avec un nombre minimum de données de capteurs visité. Par exemple, Dyser pourrait être utilisé pour rechercher des chambres dans un grand bâtiment actuellement occupé, des stations de location de vélos qui disposent actuellement de vélos, des endroits actuellement calmes au bord de l'eau ou des embouteillages actuels dans une ville.

3.2.3 Méthode de Truong

Truong et al. [119] ont exploité la théorie des ensembles flous pour proposer un algorithme capable de calculer un score de similarité pour une paire de capteurs qui sont utilisés pour obtenir une liste classée des capteurs correspondants. Dans cette approche, les auteurs adoptent la recherche, par exemple utilisé par le moteur de recherche TinEye, pour la sélection des capteurs fournissant des sorties similaires. Ce service est appelé service de recherche de similarité de capteur. Il peut être utilisé à des fins différentes. Premièrement, il pourrait être utilisé pour trouver des lieux aux propriétés physiques similaires. Deuxièmement, il pourrait être utilisé pour aider les utilisateurs à formuler une description de métadonnées d'un capteur nouvellement déployé. Dans ce travail, les auteurs proposent un service de recherche de similarité de capteur qui permette de trouver des capteurs avec des sorties similaires que la sortie passée d'un capteur. Une approche efficace est conçue pour calculer un score de similarité pour une paire de capteurs. Tous les capteurs calculent des ensembles flous qui représentent leur sortie passée en utilisant seulement quelques dizaines d'octets. Ces ensembles flous sont indexés dans une base de données. Compte tenu de la sortie d'un autre capteur, des scores de similarité sont calculés pour chaque capteur indexé, les capteurs sont classés par ce score et renvoyés à l'utilisateur. Enfin, un prototype fonctionnel a été construit pour démontrer la fonctionnalité de service et pour soutenir l'expérimentation dans des environnements réalistes.

Cependant, la recherche de capteurs basés sur le contenu n'est pas utile pour les applications car le grand nombre de capteurs rend impossible le traitement de toutes les données collectées par ces capteurs.

En raison du grand nombre de fournisseurs de contexte IdO, la sensibilité au

contexte jouera un rôle essentiel pour proposer de nouvelles méthodes de recherche et de sélection des capteurs en fonction de la qualité de leurs données détectées. Le contexte a été défini dans [5] comme toute information pouvant être utilisée pour caractériser la situation d'une entité. Une entité est une personne, un lieu ou un objet considéré comme pertinent pour l'interaction entre un utilisateur et une application, y compris l'utilisateur et l'application eux-mêmes. Certains travaux dans le domaine de recherche et sélection de capteurs sensibles au contexte peuvent être trouvés dans la littérature.

Pour des raisons objectives, dans cette section, nous divisons ces approches en deux catégories: Les premières méthodes sensibles au contexte qui n'utilise qu'un nombre limité de propriétés contextuelles et les dernières méthodes sensibles au contexte qui utilise un grand nombre de propriétés contextuelles.

3.3 Premières méthodes basées sur le contexte

Dans ce qui suit, nous allons présenter les premières célébrées méthodes de recherche et de sélection de capteurs sensible au contexte.

3.3.1 Méthode GSN

La diminution rapide des prix des technologies de capteurs sans fil permet d'augmenter le nombre de réseaux de capteurs autonomes. Ces réseaux de capteurs ne resteront généralement pas isolés, mais il sera nécessaire de les interconnecter au niveau du réseau pour permettre un traitement intégré des données, réalisant ainsi la vision globale d'un «Internet de capteurs». Cela nécessite une couche middleware flexible qui fait l'abstraction des technologies de réseau de capteurs hétérogènes sous-jacents et prend en charge le déploiement et l'ajout rapide et simple de nouveaux capteurs. De plus, cette couche doit faciliter le traitement efficace des requêtes distribuées et la combinaison des données des capteurs, prendre en charge la mobilité des capteurs, et permettre l'adaptation dynamique de la configuration du système pendant l'exécution avec un minimum d'effort (zéro programmation). La plate-forme Global Sensor Networks (GSN) [3] vise à fournir un middleware flexible pour atteindre les objectifs cités ci-dessus et relever les défis de l'intégration des données des capteurs et du traitement distribué des requêtes. En effet, cette plate-forme répertorie tous les capteurs disponibles pour le besoin des utilisateurs dans une combo-box à sélectionner.

Le GSN suppose le modèle simple illustré à la figure 3.1: un réseau de capteurs interne peut utiliser des algorithmes de routage ad hoc et multi-sauts arbitraires pour fournir des lectures de capteurs à un ou plusieurs nœuds récepteurs (sink node). Un nœud récepteur est un nœud connecté à un ordinateur de base plus puissant qui à son tour exécute le middleware GSN et peut participer à un réseau (à grande échelle) d'ordinateurs de base, chacun exécutant GSN et desservant un ou plusieurs réseaux de capteurs.

GSN ne fait aucune hypothèse sur les composants internes d'un réseau de capteurs sauf le fait que le nœud récepteur est connecté à l'ordinateur de base via un wrapper logiciel conforme à l'API GSN. En plus de cette couche d'accès physique, le GSN fournit des capteurs virtuels qui font l'abstraction des détails de mise en œuvre de l'accès aux données des capteurs et définissent le traitement du flux de données à effectuer.

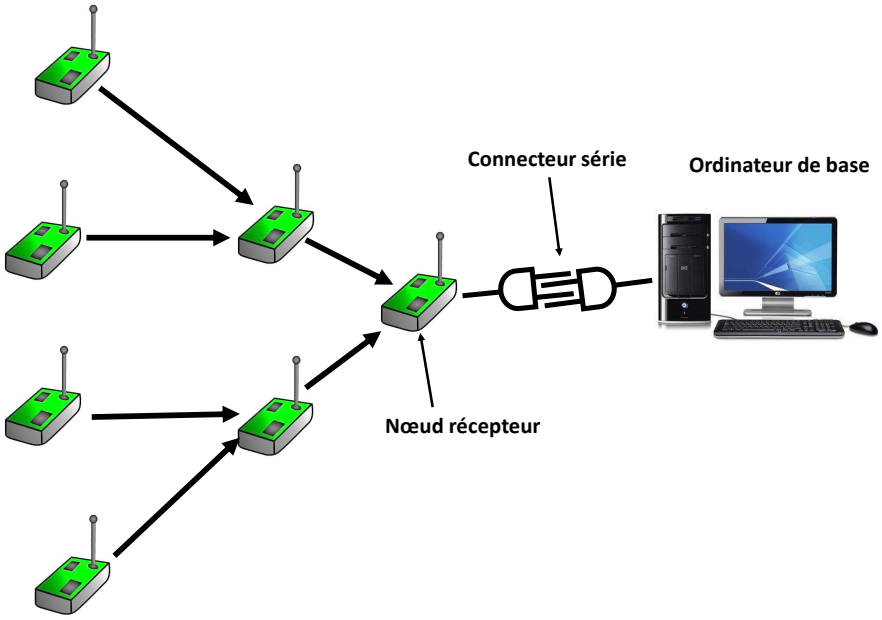

Figure 3.1: Modèle GSN.

3.3.2 Méthode Snoogle

Bien que la recherche d'informations (IR) sur Internet soit bien établie, l'adoption de l'IR dans un réseau de capteurs pose plusieurs défis. Premièrement, un réseau de capteurs doit limiter la communication pour économiser l'énergie. Pour cette raison, de nouvelles techniques de collecte, de stockage et de recherche de données sont nécessaires. Deuxièmement, lorsque l'on considère que les capteurs sont connectés à des objets physiques, ces capteurs peuvent être mobiles et les données stockées peuvent changer rapidement. Troisièmement, la sécurité et la confidentialité présentent des défis plus importants pour la recherche dans les réseaux de capteurs que la recherche sur Internet. Pour résoudre les problèmes de recherche signalés précédemment, Snoogle [126] est un moteur de recherche conçu pour permettre aux utilisateurs de rechercher soit un objet mobile, soit une liste d'objets répondant à leurs préférences. Snoogle est un système de recherche d'informations basé sur des réseaux de capteurs pour le monde physique. Comme il est illustré dans la figure 3.2, Snoogle se compose de trois composants:

- **Capteur d'objet**: est un atome attaché à un objet physique, et il contient une description textuelle de l'objet physique.

- **Point d'index (IP)**: est un dispositif de capteur statique associé à un emplacement physique. Les IPs sont responsables de la collecte et de la gestion des données des capteurs d'objet dans leurs voisinages.
- **Point d'index clé (KeyIP)**: est un super nœud qui gère et collecte les données à partir de différentes IPs du réseau.

Snoogle adopte une architecture hiérarchique à deux niveaux comme elle est illustrée dans la figure 3.2. Le niveau inférieur implique des capteurs d'objets et des IPs. Chaque IP gère une certaine zone dans sa plage de transmission. Les capteurs d'objets s'enregistrent et transmettent les métadonnées de leur description à l'IP spécifique. Les IPs sont responsables de la création des index inversés pour la recherche locale. Au niveau supérieur, les IPs ont deux rôles. Tout d'abord, les IPs transmettent les informations d'objet agrégées au KeyIP afin que le KeyIP puisse renvoyer une liste d'IP les plus pertinentes pour une certaine requête utilisateur. Deuxièmement, les IPs acheminent également le trafic entre les IPs, le KeyIP et les objets. Le KeyIP sont considérés comme le "puits du réseau", qui contient les informations d'agrégation d'objets globales rapportées par chaque IP. Les utilisateurs interrogent Snoogle à l'aide d'appareils portables tels que des smartphones ou des PDA. Snoogle fournit deux types de requêtes, une requête locale et une requête distribuée. La requête locale est effectuée lorsqu'un utilisateur dirige sa requête vers une IP spécifique. Ce type de requête se produit lorsqu'un utilisateur souhaite limiter sa recherche à des objets situés à un emplacement spécifique. Un utilisateur exécute une requête distribuée lorsqu'il interroge le KeyIP. La capacité de requête distribuée permet une évolutivité puisque les utilisateurs n'ont pas besoin d'inonder chaque IP pour trouver un objet particulier.

Les contributions de Snoogle sont résumées comme suit:

- Il est le premier travail de recherche qui permet de construire un système de recherche d'informations pour le monde physique basé sur des réseaux de capteurs.
- Il examine les techniques de compression telles que les Bloom filtres et les Bloom filtres compressés pour réduire les données transmises au sein du réseau de capteurs.
- Il développe un algorithme de requête top-k distribué pour réduire le coût de communication des requêtes distribuées par l'utilisateur.
- Il propose un cadre de sécurité et de confidentialité plus flexible pour qu'un utilisateur puisse rechercher un objet dans un réseau de capteurs.

3.3.3 Méthode Mayer

L'augmentation du nombre de requêtes et de commandes qui seront produites lors de l'intégration d'un très grand nombre d'objets numériques, va ralentir et surcharger

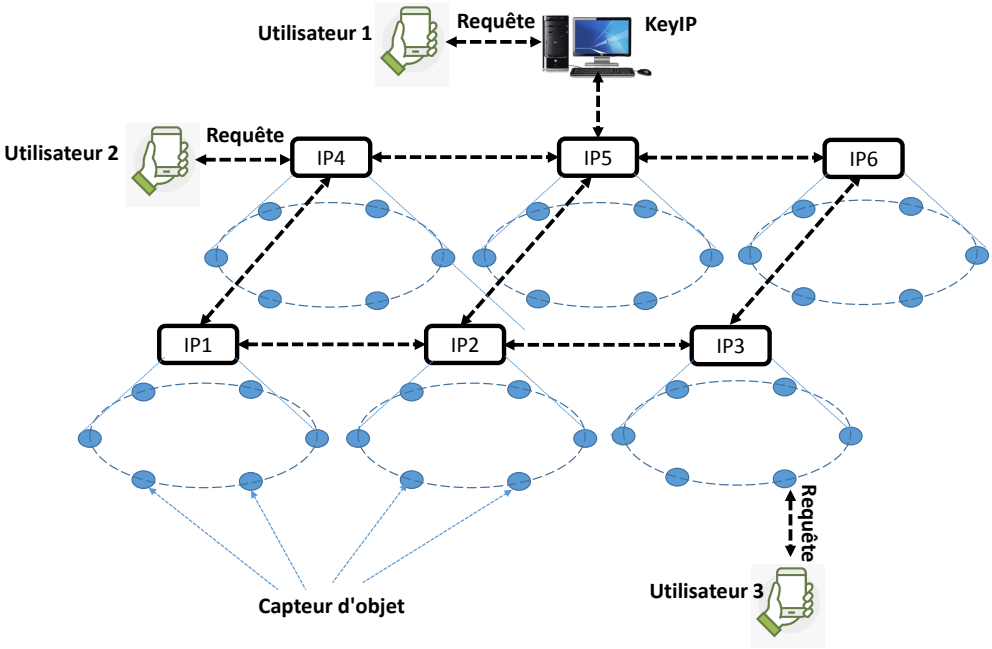

Figure 3.2: Vue d'ensemble de l'architecture de Snoogle.

le réseau Internet. Pour résoudre ce problème, il est nécessaire de développer une infrastructure de gestion distribuée pour assurer l'accès aux appareils intelligents. Une telle infrastructure devrait prendre en charge la description, la découverte, la recherche et l'interaction des dispositifs intelligents et de leurs services de détection et d'actionnement associés pour les utilisateurs humains et les machines. Cela devrait permettre aux objets intelligents de communiquer et de coopérer dans des contextes étendus concernant leur distribution spatiale ainsi que leur nombre. Selon Mayer et al. [74], il faut assurer les facteurs suivants pour le succès d'une infrastructure d'objets intelligents:

- **Évolutivité**: Le système doit être conçu de manière à permettre la gestion d'interaction et les tâches de communication entre un nombre élevé d'objets intelligents.
- **Emplacement et équilibrage de charge**: Les appareils intelligents sont censés interagir plus souvent avec d'autres objets à proximité et donc présenter un certain degré de localité. Une infrastructure pour environnements intelligents doit exploiter cette propriété et éviter ainsi le routage global autant que possible. En outre, la grande disponibilité des informations sur l'emplacement des objets intelligents est une condition essentielle pour permettre des services contextuels.
- **Autogestion**: Le système doit être conçu de manière à ce que les efforts d'administration manuelle soient réduits au minimum. Cela implique d'automatiser la découverte d'objets intelligents par l'infrastructure ainsi que de minimiser les travaux manuels de configuration et de maintenance sur l'infrastructure elle-même.

- **Convivialité**: Le système doit exposer des interfaces facilement compréhensibles pour les utilisateurs humains et les machines, pour rechercher des services fournis par des objets intelligents.

Mayer et al. [74] développent un mécanisme de recherche d'objets intelligents qui permet à ses utilisateurs d'interroger le monde réel et qui est intégré dans une infrastructure pour les appareils intelligents. La tâche de trouver des objets intelligents pertinents est beaucoup plus compliquée que la recherche de documents, non seulement parce que les objet intelligents doivent être identifiés en fonction d'informations dynamiques et contextuelles, mais aussi en raison de l'absence d'une manière uniforme de décrire les objets, leurs propriétés et les services qu'elles offrent: un objet intelligent n'exprime pas nécessairement sa fonctionnalité de telle sorte qu'elles peuvent être trouvées par les moteurs de recherche traditionnels qui visent à trouver des documents textuels. En raison du degré élevé de localisation des interactions entre les utilisateurs humains et les objets intelligents, Mayer et al. [74] proposent de faire l'emplacement d'un objet intelligent comme étant l'une de ses propriétés de contexte principal et d'utiliser des identificateurs de lieu logiques pour structurer les nœuds de l'infrastructure. Afin de prendre en charge l'évolutivité, les interactions entre les nœuds voisins sont limitées aux communications directes. Le système proposé consiste d'une hiérarchie de nœuds de gestion interconnectée structurée en fonction des identifiantes logiques des lieux qu'elle couvre. Ensuite, les capteurs sont recherchés par un nœud local ou distant à l'aide des techniques de recherche arborescente.

3.3.4 Méthode de Ramachandran

L'Internet des objets contient des milliards de dispositifs interconnectés tels que des objets physiques, des animaux ou des êtres humains qui donnent le pouvoir de transférer des informations sur un réseau sans interaction humaine. L'énorme quantité de données générées en continu par ces dispositifs de détection soulève le défi de rechercher les données de capteurs les plus pertinentes en fonction de la requête utilisateurs. De plus, les requêtes spécifiées par les utilisateurs sont en langage humain naturel qui ne peut pas être traité par des capteurs. Ramachandran et al. [98] ont tenté de réduire l'espace de recherche en regroupant les capteurs en cluster en fonction de leur niveau d'énergie et de leur proximité. Ils traitent les demandes des utilisateurs, qui sont formulées dans un langage humain naturel, et les transforment en une forme reconnue par les capteurs. Ces demandes sont ensuite mises en correspondance avec les capteurs pour effectuer la recherche, et les données qui seront renvoyées à l'utilisateur. La figure 3.3 donne une vue globale de l'architecture proposée dans [98]. Les déférentes phases de cette approche sont résumées comme suit:

- **Formation de cluster**: Tout d'abord, chaque périphérique doit enregistrer son

adresse auprès de la station de base afin de savoir le type de périphérique. Puisque l'application proposée ne sera présente dans aucune zone éloignée, les capteurs peuvent être remplacés au fur et à mesure que le niveau de la batterie baisse. Par conséquent, l'énergie n'est pas considérée comme un facteur majeur lors de la formation de cluster. La station de base lancera la communication en envoyant un message de diffusion aux dispositifs à sa portée. Il choisit le premier niveau de contrôleurs en fonction de la région dans laquelle le contrôleur est présent, de sorte que chaque région aura un contrôleur qui peut communiquer avec la station de base. Des identifiants uniques sont également attribués à chaque dispositif au fur et à mesure de la formation des clusters. Si deux nœuds ont le même nombre de voisins, alors le nœud dont la distance est éloignée de la station de base, mais toujours accessible au contrôleur est sélectionnée.

- **Classification des dispositifs**: Lors de la formation de clusters, il est attribué des identifiants au dispositif simultanément de manière hiérarchique.
- **Décomposition des requêtes**: L'utilisateur entrera la requête dans une langue anglaise naturelle complexe et abstraite. Il devrait découvrir des capteurs associés afin de répondre à la requête. Il est supposé qu'une requête est une combinaison de mots-clés, d'attributs et de l'activité à effectuer sur l'ensemble de capteurs donné. Tout mot-clé donné dans la requête est mappé avec une ontologie correspondante définie qui contient le mot-clé, sa priorité et la forme de bas niveau correspondante.
- **Recherche**: La recherche peut être effectuée en comparant les bits de la requête avec les identifiants de dispositif. Il s'agit d'un processus récursif et la formation de clusters réduira considérablement l'espace de recherche. La requête transformée et l'adresse de l'expéditeur sont utilisées pour aller à la station de base spécifique. Ensuite, la requête est comparée aux identifiants de contrôleur attachés aux stations de base. Si cela correspond, accédez à ce contrôleur spécifique. Ce processus se poursuit jusqu'à ce que le nœud requis soit atteint. Si l'utilisateur demande une valeur agrégée, le système garde une trace de toutes les données pertinentes lors de la recherche et renvoyons la valeur agrégée.

Tous les travaux cités ci-dessus ne prennent en compte qu'un nombre limité de propriétés contextuelles (localisation, type et niveau d'énergie des capteurs), mais réellement le contexte peut inclure des informations liées à la qualité de service des capteurs telle que la précision, la fiabilité, le coût, etc.

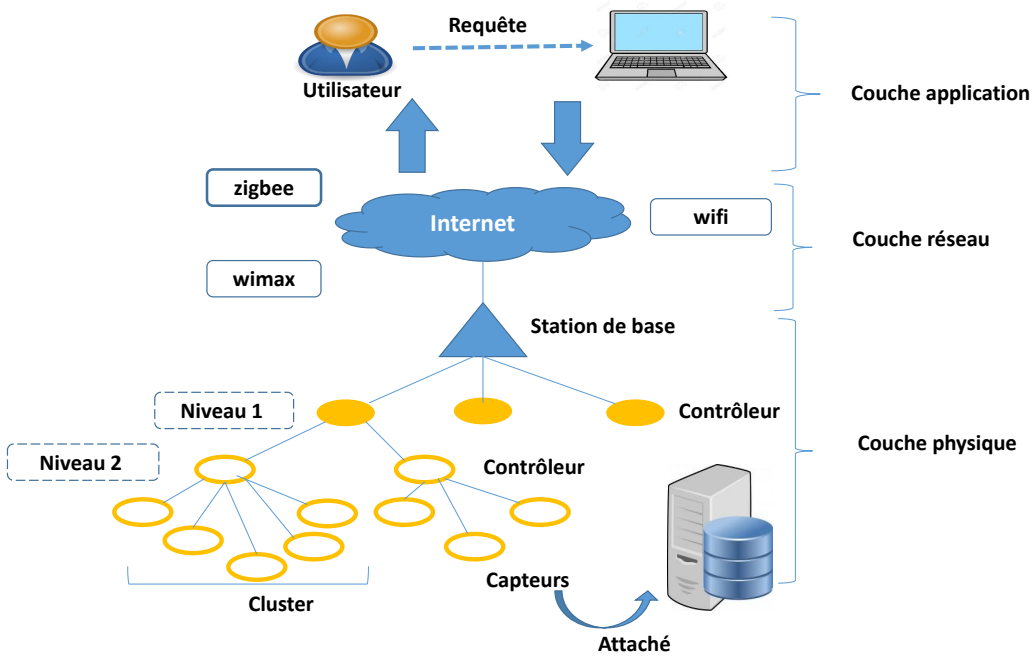

Figure 3.3: Architecture de l'approche de Ramachandran .

3.4 Méthodes récentes de recherche basées sur le contexte

Pour manipuler plusieurs propriétés contextuelles, Perera et ses collaborateurs [94, 37] ont modélisé les capteurs avec l'ontologie du réseau de capteurs sémantiques (ontologie SSN). Le SSN permet la modélisation des descriptions de capteurs, des propriétés de contexte et publie les informations des capteurs dans un format uniforme. Dans cette section, on va présenter les méthodes récentes les plus connues de recherche et sélection de capteurs sensibles aux contextes qui utilisent plusieurs propriétés contextuelles.

3.4.1 Méthode CASSARAM

En raison du nombre croissant de capteurs disponibles, la recherche et la sélection devront être focalisées sur les capteurs qui fournissent des données qui aideront à résoudre le problème en question de la manière la plus efficace. Le framework proposé dans le travail de Perera et al. [94] est nommé CASSARAM, il effectue la recherche et la sélection des capteurs en fonction des préférences et des priorités des utilisateurs. Il utilise la qualité de service qui est liée au modèle de sélection et de classement des données. La technique de distance euclidienne pondérée basée sur l'indexation est utilisée pour évaluer la similitude entre les préférences de l'utilisateur et les capteurs. En fait, une méthode heuristique est appliquée pour diminuer la quantité de données nécessaires à traiter lors de la découverte. Pour activer une recherche distribuée sur différents nœuds de serveur, le système applique

une méthode de traitement parallèle optimisée.

Les étapes principales de CASSARAM sont présentées dans la figure 3.4. L'objectif de cette méthode est de permettre aux utilisateurs de rechercher et de sélectionner les capteurs qui correspondent le mieux à leurs besoins. Dans ce modèle, les exigences des utilisateurs sont divisées en deux catégories (du point de vue de l'utilisateur): les exigences basées sur des points (non négociables) et les exigences basées sur la proximité (négociable). Au début, CASSARAM identifie les exigences basées sur les points, les exigences basées sur la proximité et les priorités des utilisateurs. Tout d'abord, les utilisateurs doivent sélectionner les exigences basées sur des points. Ensuite, les utilisateurs peuvent définir les exigences basées sur la proximité. L'outil prototype CASSARAM fournit une interface utilisateur pour exprimer les besoins des utilisateurs via des requêtes SPARQL. Toutes les propriétés contextuelles sont disponibles pour être définies de manière comparative en définissant les priorités via une interface utilisateur. Ensuite, chaque capteur est tracé dans un espace multidimensionnel où chaque dimension représente une propriété contextuelle (par exemple, précision, fiabilité, latence). Chaque dimension est normalisée entre [0,1]. Ensuite, une technique d'indexation basée sur la distance euclidienne pondérée, appelée l'indice pondéré basé sur la priorité comparative (CPWI) est générée pour chaque capteur en combinant les priorités de l'utilisateur et les valeurs des propriétés de contexte. Les capteurs sont classés à l'aide du CPWI et le nombre de capteurs requis par l'utilisateur est sélectionné en haut de la liste.

Par la suite, les auteurs présentent trois approches qui améliorent l'efficacité et la capacité de CASSARAM:

- Ils proposent une technique appelée filtrage heuristique basée sur la priorité comparative (CPHF) pour rendre CASSARAM plus efficace. L'idée de base est de supprimer les capteurs qui sont loin du capteur idéal défini par l'utilisateur et de réduire le nombre de capteurs qui doivent être indexés et classés.
- Ils introduisent une technique de filtrage basée sur l'expression relationnelle qui économise des ressources de calcul. Cette technique définit comment les ressources de calcul peuvent être sauvegardées et comment accélérer le processus de recherche et de sélection des capteurs en permettant aux utilisateurs de définir les valeurs de propriétés de contexte préférées à l'aide d'opérateurs relationnels.
- Ils développent trois techniques pour relever le défi de la recherche et de la sélection de capteurs distribués: traitement en chaîne, traitement parallèle et traitement hybride.

3.4.2 Méthode Antclust

La recherche et la découverte de capteurs sont l'une des fonctionnalités les plus importantes requises dans l'IdO. En raison du nombre croissant de capteurs disponibles,

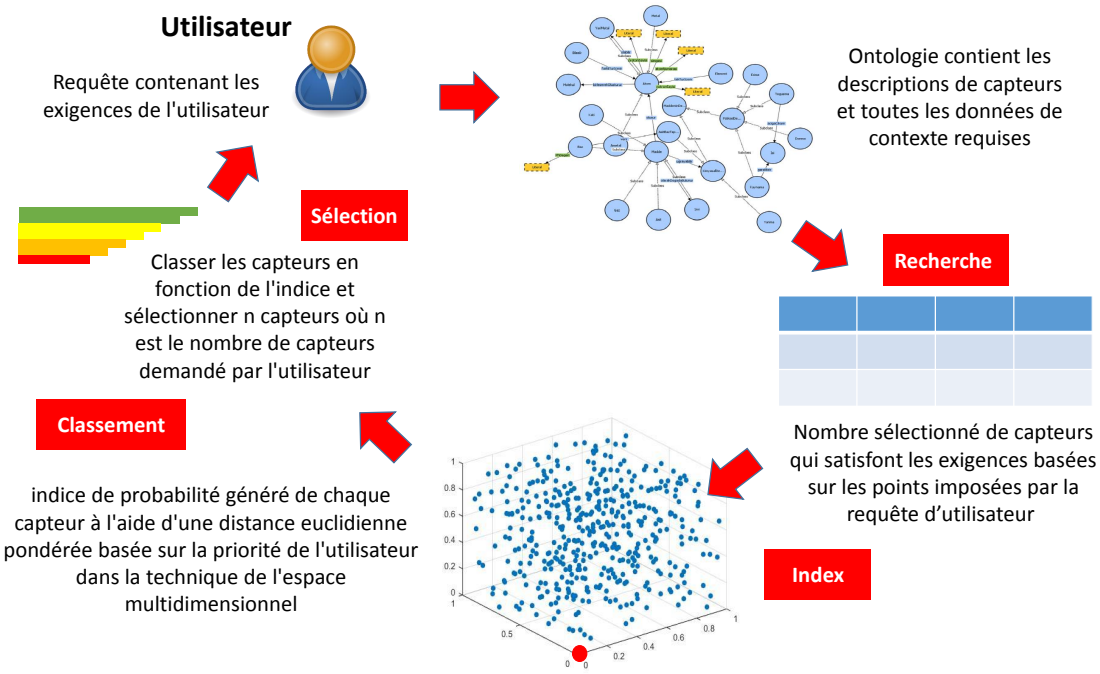

Figure 3.4: Vue d'ensemble de haut niveau de CASSARAM.

les auteurs se focalisent sur la recherche et la sélection des capteurs les plus associés de manière efficace et efficiente. Afin d'améliorer l'efficacité et l'évolutivité de la recherche de capteurs dans l'IdO, Ebrahimi et al. [37] proposent un algorithme métaheuristique (Antclust) afin de regrouper les capteurs avec des informations de contexte similaires (comme le trafic, l'activité routière, la météo, etc.) en un seul cluster SSONs (Réseaux de superposition sémantique de capteur). Dans chaque SSON, les capteurs sont regroupés par un algorithme de clustering de fourmis en fonction de leurs propriétés de contexte (par exemple, précision, fiabilité, énergie, disponibilité, coût, etc.). Ainsi, les requêtes des utilisateurs ne seront transmises qu'aux SSON pertinentes et par conséquent aux clusters associés aux attributs qui satisfont les contraintes du contenu de la requête. Cela réduit l'espace de recherche et augmente l'efficacité du processus de recherche. En fonction de la requête soumise et en fonction des choix des utilisateurs, le principe est de sélectionner les clusters contenant les capteurs les plus appropriés. En outre, des ajustements utiles sont appliqués pour réduire le coût du processus de recherche de capteurs et proposer une stratégie adaptative pour maintenir les performances du système par rapport à la dynamicité de l'IdO. La figure 3.5 illustre une vue d'ensemble de ce système. Afin de modéliser les propriétés et les descriptions du contexte des capteurs, Antclust utilise l'ontologie sémantique du réseau de capteurs (SSN). Le SSN fournit un modèle de bout en bout indépendant du domaine pour les applications de capteurs et couvre les sous-domaines spécifiques des capteurs tels que les principes et les capacités des capteurs.

Les principales contributions de l'algorithme Antclust sont résumées comme suit:

- Clustering des capteurs pour créer des SSON: les capteurs de même type tel que la météo, le trafic, l'activité routière et autres, sont regroupés en fonction de leurs propriétés de contexte afin de créer le SSON pertinent. Après avoir reçu et analysé la requête de l'utilisateur, les SSON prévues seront sélectionnés. Ensuite, la combinaison des clusters de capteurs les plus pertinents sera choisie parmi les SSON sélectionnées pour fournir le meilleur service à l'utilisateur.
- Algorithme de clustering basé sur les fourmis pour créer des SSON: Antclust introduit un algorithme inspiré des algorithmes de clustering de fourmis, pour regrouper les capteurs de grande similitude dans un même groupe. Alors que les capteurs de différents groupes sont différents que possible.
- Analyse des requêtes utilisateur et processus de recherche: Lorsqu'une requête est soumise par un utilisateur, les valeurs des attributs sont extraites du contenu de la requête. En fonction des priorités de l'utilisateur ou de la situation spécifique, le poids de chaque attribut sera attribué. Un poids est calculé pour chaque propriété de contexte et une priorité plus élevée signifie un poids plus élevé. Ensuite, la distance euclidienne entre les attributs de la requête et chaque Cluster Head sera calculée. Le Cluster Head est un capteur virtuel qui indique la valeur moyenne des attributs des capteurs dans le cluster concerné. Enfin, les clusters contenant les capteurs les plus appropriés qui aideront les utilisateurs à résoudre leurs propres problèmes seront sélectionnés à l'issue du processus de recherche.
- Maintenir les performances de clustering dans les réseaux de capteurs dynamiques: Dans le contexte d'un environnement IdO dynamique, l'adaptabilité devient une caractéristique clé des services car elle permet à une application de s'adapter aux nouvelles exigences contextuelles. Afin de maintenir les performances de clustering, les clusters peuvent avoir besoin d'être modifiés dynamiquement lorsque le réseau de capteurs change, c'est-à-dire après l'arrivée et le départ de certains capteurs. Antclust propose un algorithme qui est déclenché lorsqu'un changement se produit dans le SSON.

3.4.3 Méthode de Nunes

Le nombre de dispositifs connectés à Internet a augmenté rapidement, créant une quantité de données qui ne peut pas être gérée dans un environnement IdO autonome et limité en énergie. De plus, L'intégration de l'IdO au cloud computing, appelée Cloud of Things (CoT), peut offrir des services de détection omniprésents sans précédent et des ressources puissantes pour traiter les flux de données de détection au-delà de la capacité des objets individuels. Cependant, le CoT pose de nouveaux défis car il doit combiner différents types de services fournis par plusieurs parties prenantes et prendre en charge un grand

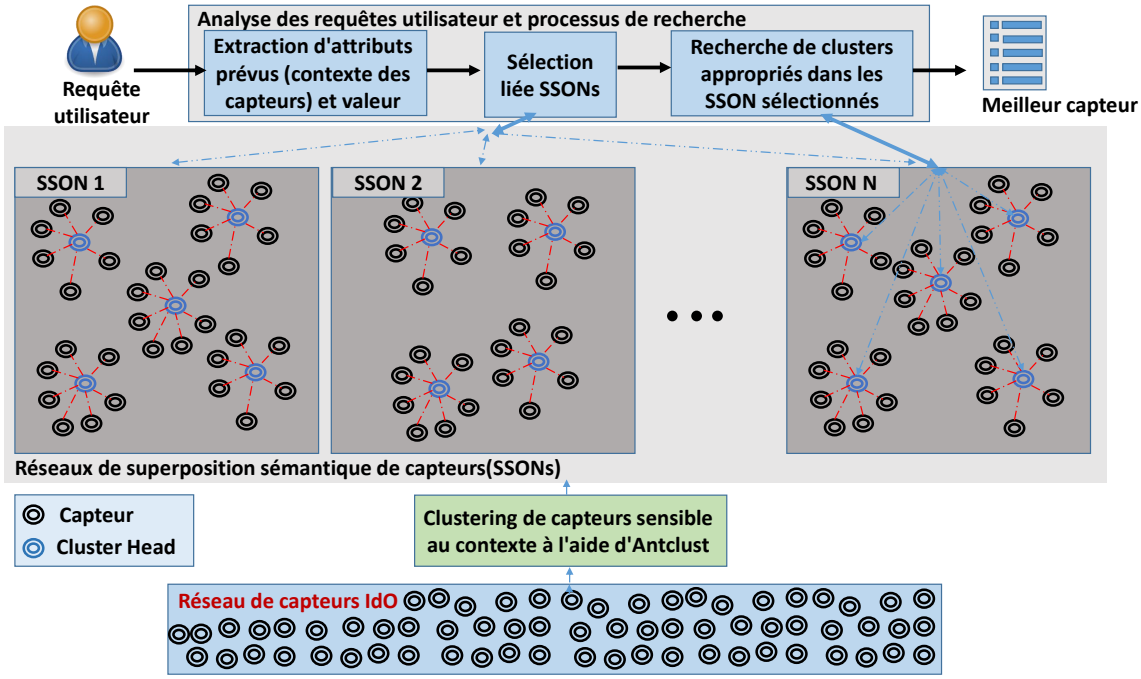

Figure 3.5: Vue d'ensemble du système Antclust.

nombre d'utilisateurs et d'appareils. L'un de ces défis est de fournir un ensemble d'outils et d'environnements pour le développement d'applications dynamiques et d'assurer leur exécution transparente pour répondre aux exigences de qualité de contexte (QoC) et de qualité de service (QoS) imposées par différents types d'applications.

Dans le travail de Nunes et al. [83], les auteurs présentent une étude qualitative de trois méthodes d'analyse de décision multicritères (MCDA) pour effectuer la recherche et la sélection des capteurs afin d'obtenir le meilleur compromis entre les propriétés QoS et QoC. Ils évaluent et comparent la qualité globale de la recherche de capteurs entre trois méthodes de décision multicritère: la méthode du poids additif simple (SAW), la technique pour l'ordre de priorité par similarité à la solution idéale (TOPSIS) et VIseKriterijumska Optimizacija Je Kompromisno Resenje (VIKOR). Ils analysent également la qualité des solutions proposées par ces méthodes en les comparant aux solutions optimales de Pareto. Les contributions scientifiques de ce travail peuvent être résumées comme suit:

- Ils ont proposé une méthodologie qui peut être utilisée pour comparer différentes techniques de recherche de capteurs du point de vue de la qualité de la recherche. Ils ont démontré le fonctionnement de leur méthode proposée en utilisant trois techniques de recherche de capteurs différents.
- Ils ont examiné l'impact du «nombre de propriétés de contexte» sur la qualité globale de la recherche de capteurs.
- Ils ont examiné l'impact du «nombre de capteurs devant être sélectionnés» sur la

qualité globale de la recherche de capteurs.

- Ils ont évalué et comparé la qualité globale de la recherche de capteurs entre trois techniques d'analyse décisionnelle multicritère SAW, TOPSIS et VIKOR.

L'approche d'évaluation suivie dans ce travail est basée sur un ensemble de données de capteurs qui seront classées selon une méthode MCDA et des propriétés de contexte. Le nombre souhaité de capteurs est extrait du haut de la liste classée et les fronts Pareto-optimaux sont calculés. La figure 3.6 synthétise l'ensemble de la proposition de traitement pour l'évaluation des MCDA. Le critère de Pareto-optimalité est utilisé pour comparer la qualité des solutions obtenues par chaque méthode. Il utilise le concept de dominance pour déterminer quand une solution est meilleure qu'une autre.

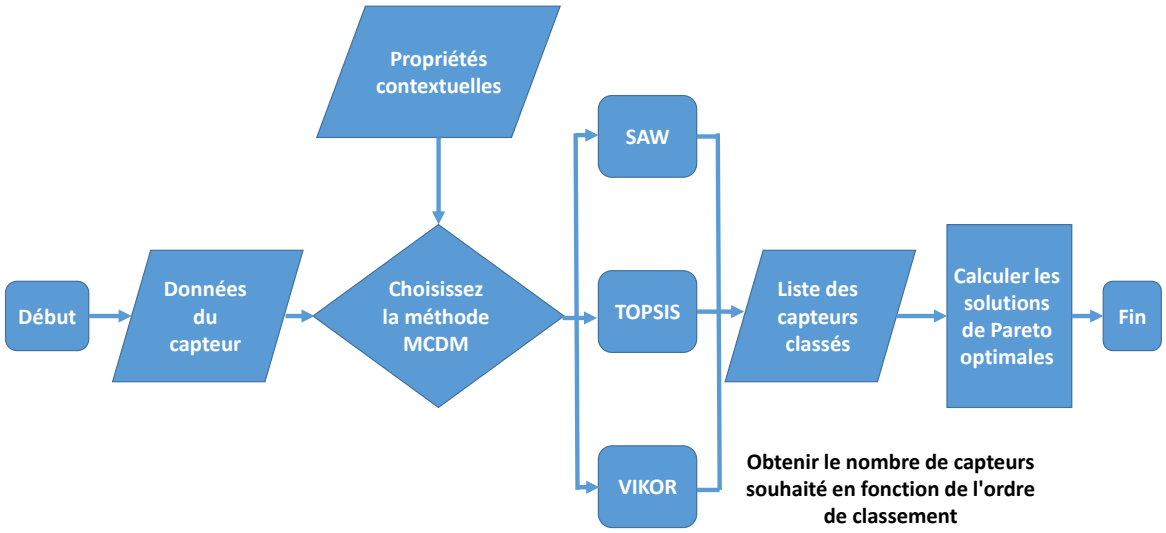

Figure 3.6: Flux de travail d'évaluation suivi par Nunes.

3.4.4 E-S algorithme

L'un des principaux défis des plates-formes IdO est la recherche et la découverte de ressources dans des environnements à grande échelle et hétérogènes pour une réutilisation par d'autres applications pour répondre à leurs besoins spécifiques. Fondamentalement, le processus de recherche et découverte peut être divisé en deux phases: (i) utilisation d'une requête statique pour trouver les ressources concernant un ensemble d'exigences spécifiques et (ii) appliquer un algorithme d'analyse de décision multicritères (MCDA) selon les priorités relatives à chaque exigence pour classer les ressources disponibles. Les travaux existants concernent uniquement le temps de recherche et de découverte des

ressources et n'évaluent pas correctement la qualité de la solution proposée, ce qui peut affecter la qualité d'expérience (QoE) d'un utilisateur.

Nunes et al. [82] ont proposé un algorithme d'élimination-sélection (E-S algorithme) qui améliore la qualité de la solution en fonction de leurs travaux antérieurs. La proposition a été effectué dans la deuxième phase du processus de recherche et de découverte. Ils ajoutent une petite surcharge pour calculer la solution. L'E-S algorithme combine deux algorithmes: une sélection rapide en utilisant la méthode de décision à critères multiples TOPSIS, qui est utilisée dans un outil ViSIoT [81] et la meilleure option en utilisant le tri Rapide-Non-Dominé. En conséquence, ils appliquent l'E-S algorithme dans une étude de cas agricole en utilisant un ensemble de données réel. L'E-S algorithme peut être appliqué à n'importe quel système pour obtenir le meilleur compromis entre le processus de recherche et de découverte des ressources et la QoE offerte à un utilisateur concernant la qualité et le temps de la solution proposée.

Dans ce qui suit, nous expliquons les différentes étapes de l'E-S algorithme:

- Premièrement, l'algorithme TOPSIS est utilisé pour classer et trier l'ensemble d'options disponibles.
- Ensuite, l'algorithme de tri rapide non dominé est exécuté en considérant les premières options $N \times SR$. Le paramètre SR (Valeur du taux de recherche) joue un rôle clé dans cette étape, car il est utilisé pour augmenter les chances d'obtenir une solution optimale proposée par l'algorithme TOPSIS et minimiser le temps et la complexité de stockage pour exécuter l'algorithme de tri rapide non dominé. N représente le nombre d'options à sélectionner.
- Enfin, les N options appartenant aux premiers fronts sont renvoyées comme résultat.

3.4.5 Méthode de Hsu

Bénéficiant des solutions middleware IdO, les utilisateurs peuvent facilement collecter des données à partir d'un grand nombre de capteurs pour diverses applications. Cependant, dans ces middlewares, les utilisateurs doivent déterminer eux-mêmes quels capteurs sont appropriés à leurs fins, ce qui n'est pas convivial. Par conséquent, afin de gérer des milliers voire des millions de capteurs et de fournir une utilisation intuitive aux utilisateurs, Hsu et al. [53] ont proposé une architecture de service de détection avec une nouvelle méthode de recherche et de sélection de capteurs pour offrir une réutilisabilité des capteurs et sélectionner efficacement les capteurs pertinents pour l'utilisateur parmi un large ensemble de capteurs disponibles. Dans l'architecture proposée, les fonctionnalités des capteurs de même finalité (par exemple, température, humidité) sont logiquement liées à un service applicatif, et le nombre total de services peut donc être réduit, ce qui permet une gestion efficace. De plus, les valeurs des propriétés de contexte des capteurs modifiées en continu (par exemple, latence, temps de réponse) sont stockées uniquement dans la passerelle

locale, qui ne sont pas globalement mises à jour, de sorte que le trafic réseau peut être réduit.

Grâce à l'architecture de service de détection ainsi qu'à la méthode de recherche et de sélection des capteurs proposée, les capteurs peuvent être facilement réutilisés même par des utilisateurs non professionnels. Par conséquent, les expériences d'interaction peuvent être améliorées. Pour l'intégration de capteurs hétérogènes, l'ontologie du réseau de capteurs sémantiques (SSN) est adoptée pour permettre l'utilisation d'un format de données uniforme afin d'assurer l'interopérabilité dans l'IdO. Elle définit également les propriétés de contexte des capteurs. Dans le prototype développé, chaque requête utilisateur contient les propriétés de contexte des capteurs pour la recherche et la sélection des capteurs.

Pour réaliser une utilisation intuitive pour les utilisateurs non professionnels, un service d'application est implémenté en tant que service Web. Il permet aux utilisateurs d'obtenir les informations souhaitées via des interfaces Web. Cette méthode détermine d'abord les poids des propriétés de contexte des capteurs par le biais de questions et réponses avec un utilisateur. Puis classe les capteurs pour déterminer quels capteurs sont appropriés à l'utilisateur avec des pondérations. L'architecture permet une gestion efficace des capteurs hétérogènes et réduit le trafic réseau. À l'aide de la méthode de recherche et de sélection des capteurs, les utilisateurs peuvent définir des conditions implicites dans leurs demandes d'utilisation intuitive. De plus, il peut sélectionner des capteurs appropriés et prolonger la durée de vie des réseaux de capteurs.

L'architecture de service de détection proposée se compose de trois composants: capteur, passerelle et serveur, comme illustré sur la figure 3.5. La passerelle se connecte et gère un réseau local composé de plusieurs capteurs à proximité déployés dans la zone d'intérêt. Le serveur se connecte à plusieurs passerelles pour communiquer avec les capteurs. Les détails de trois composants sont décrits ci-dessous:

- **Capteur**: Un capteur envoie d'abord un message d'enregistrement de capteur (c'est-à-dire un fichier de description de capteur) à la passerelle, puis passe en mode actif s'il reçoit une demande de la passerelle. Dans ce mode, un capteur transmet ses données de détection (par exemple, température, humidité) à la passerelle. La qualité des données de détection est également considérée dans cette architecture pour offrir une meilleure expérience utilisateur.

- **Passerelle**: La passerelle est le responsable du traitement des requêtes envoyées par le serveur et des données de détection des capteurs. La requête du serveur contient la durée d'abonnement aux capteurs, qui indique un intervalle de temps pendant lequel le capteur doit signaler les données de détection. Par conséquent, les capteurs peuvent passer en mode veille hors la durée d'abonnement pour économiser l'énergie. Pour enregistrer les informations du capteur et accéder aux capteurs, la passerelle crée un service générique pour chaque capteur qui stocke les informations

de chemin de routage et toutes les propriétés de contexte d'un capteur, y compris les propriétés de contexte hautement dynamiques (par exemple, latence, temps de réponse, énergie résiduelle) et des informations de base (par exemple, emplacement, ID, types de détection) contenues dans le fichier de description du capteur. Ensuite, pour chaque service générique, la passerelle envoie un message d'enregistrement de service comprenant des informations de base au serveur.

- **Serveur**: c'est le responsable de traitement des demandes des utilisateurs et de la gestion des passerelles. Une fois le serveur reçoit un message d'enregistrement de service, il extrait tous les types de détection dans le service générique, crée un service d'application pour chaque type de détection et stocke les informations de base du service générique dans le service d'application. Si le service d'application d'un type de détection a existé dans le serveur, le service générique est alors enregistré auprès du service d'application. Par conséquent, un service d'application peut contrôler/accéder à plusieurs services génériques de passerelles. Le nombre de services dans le serveur est nettement inférieur au nombre total de services génériques, ce qui permet une gestion efficace. Du fait que le service d'application ne stocke que les informations de base des capteurs, la passerelle n'a pas besoin d'envoyer fréquemment des messages de mise à jour, ce qui réduit le trafic réseau.

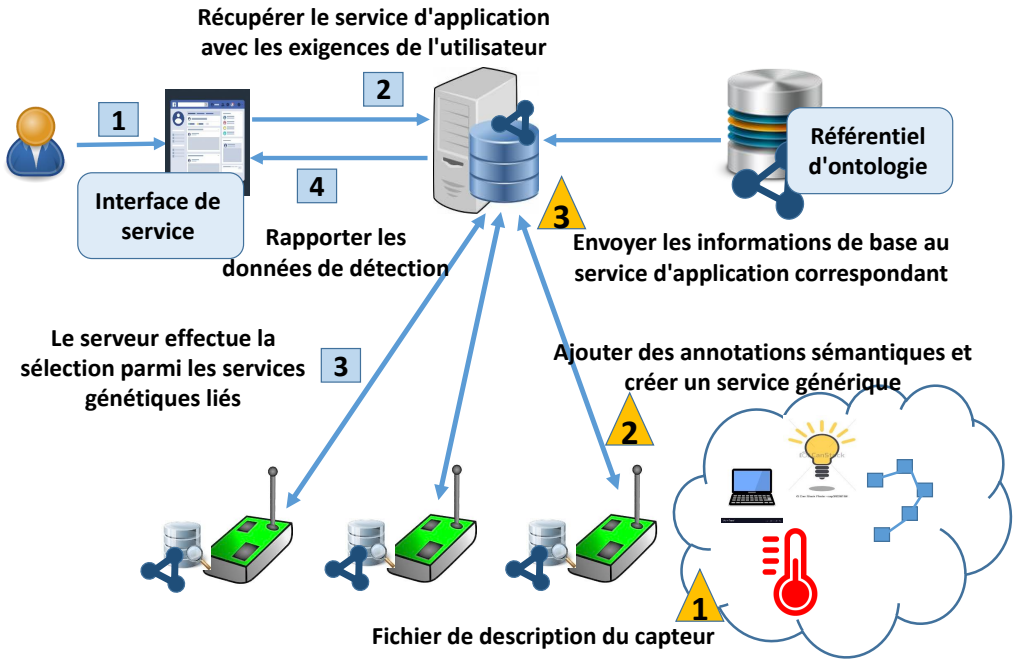

Figure 3.7: L'architecture de service de détection proposée par Hus.

Les étapes décrivant comment un utilisateur demande un service d'application et obtient finalement les informations souhaitées sont les suivantes:

1. Il se connecte au système proposé et détermine ses besoins via une interface web.

2. L'interface envoie la demande d'informations, y compris l'ID de l'utilisateur et les exigences de l'utilisateur, au serveur.

3. Le serveur extrait d'abord le type de détection et les informations de base de la demande de l'utilisateur. Puis, il sélectionne le service d'application correspondant en fonction du type de détection. Il utilise ensuite des informations de base pour rechercher les services génériques appartenant au service d'application afin de déterminer quelles passerelles peuvent fournir les données de détection appropriées à l'utilisateur. Ensuite, le serveur exécute la méthode de recherche et de sélection de capteur proposée pour attribuer des tâches aux passerelles. La passerelle qui reçoit les tâches calcule les scores des services génériques et renvoie ces scores au serveur. Enfin, le serveur détermine un ensemble de services génériques appropriés en fonction des scores.

4. Le serveur demande aux passerelles pertinentes de rapporter les données de détection de ses capteurs par la connexion à leurs services génériques correspondants.

3.5 Synthèse des travaux existants

Le nombre de capteurs déployés dans le monde augmente à un rythme rapide. Ces capteurs génèrent en permanence d'énormes quantités de données. Cependant, la collecte de données à partir de tous les capteurs disponibles ne crée pas de valeur supplémentaire à moins qu'ils ne soient capables de fournir des informations précieuses qui aideront à relever les défis auxquels nous sommes confrontés chaque jour. De plus, il est inutile en raison de la grande taille, des ressources limitées et de facteur coût caractérisant l'IdO. Lorsqu'un grand nombre de capteurs est disponible parmi lesquels choisir, il devient un défi et une tâche fastidieuse de sélectionner les capteurs appropriés qui aideront les utilisateurs à résoudre leurs propres problèmes.

Le modèle de détection en tant que service (Sensing as a service) devrait être construit au sommet de l'infrastructure et des services IdO. Il envisage également que les capteurs soient disponibles pour être utilisés sur Internet, soit gratuitement, soit en payant des frais via des solutions middleware. Actuellement, plusieurs solutions middleware qui facilitent un tel modèle sont développés. Ces solutions middleware se concentrent fortement sur la connexion des dispositifs de détection aux systèmes logiciels et aux fonctionnalités associées. Cependant, lorsque de plus en plus de capteurs se connectent à Internet, la fonctionnalité de recherche devient critique.

Zhou et al. [132] ont classé les méthodes de recherche et de sélection des capteurs en deux groupes: basées sur le contenu (par exemple, recherche les capteurs qui génèrent une telle valeur) et basées sur le contexte (par exemple, la recherche est basée sur l'emplacement et le type des capteurs).

Les deux tableaux 3.1 et 3.2 présentent une étude comparative entre les différentes méthodes existantes basées sur un ensemble de métriques bien ciblés. Les métriques considérées dans cette étude sont: format de données, approche d'accès, type de recherche, échelle des expériences, dynamicité, architecture, implémentation. Après avoir étudié et examiné les différents travaux existants dans la littérature, nous constatons que:

- La recherche de capteurs basés sur le contenu n'est pas utile pour les applications car le grand nombre de capteurs rend impossible le traitement de toutes les données collectées par ces capteurs. De plus, les contraintes matérielles des capteurs en matière d'énergie, mémoire et puissance de calcul rendent la visite permanente et périodique de tous les capteurs pratiquement impossible. Bien que la recherche dans cette ligne ne soit pas aussi populaire que celle basée sur le contexte, elle représente une approche utile dans certaines applications, par exemple, surveiller si les capteurs fonctionnent correctement, identifier les capteurs nécessitant une maintenance, déployer de nouveaux capteurs, ou définir de nouveaux services basés sur des capteurs particuliers.
- En raison du grand nombre de fournisseurs de contexte IdO, la sensibilité au contexte jouera un rôle essentiel pour proposer de nouvelles méthodes de recherche et de sélection des capteurs en fonction de la qualité de leurs données détectées. La recherche de capteurs basée sur le contexte aussi souffre de nombreux problèmes et limites. Les premiers travaux ne prennent en compte qu'un nombre limité de propriétés contextuelles (localisation, type et niveau d'énergie des capteurs), mais réellement le contexte peut inclure des informations liées à la qualité de service des capteurs telle que la précision, la fiabilité, le coût, etc. Bien que les derniers travaux considèrent un grand nombre de propriétés contextuelles, de nombreuses lacunes restent à soulever. La plupart des travaux cités utilisent des méthodes heuristiques pour filtrer et sélectionner les capteurs en question, ce qui va diminuer la qualité de recherche et de sélection et le niveau de satisfaction des utilisateurs. De plus, ces méthodes ne prennent pas en compte la nature distribuée des systèmes IdO. Bien que l'on puisse trouver des progrès à cet égard, ces systèmes sont encore assez compliqués. En outre, certaines recherches ne prennent pas en compte le grand dynamisme de l'IdO et la mobilité des capteurs. En fait, nous proposons d'utiliser une méthode plus simple pour améliorer l'évolutivité du réseau.

S'inspirant de l'architecture principale discutée dans [53], notre architecture se compose de plusieurs passerelles (gateways) réparties dans le réseau et d'un serveur. Le serveur est connecté aux passerelles qui gèrent à leurs rôles leurs propres réseaux locaux des capteurs hétérogènes. Les utilisateurs envoient leurs requêtes au serveur qui communique avec les différentes passerelles pour apporter une réponse. Contrairement à Hsu et al.[53], nous profitons de l'énorme capacité de calcul et de stockage au niveau des

Technique de recherche	Classification	Format de donnée	Approche d'accès	Type de recherche
Elahi [39]	Basé sur le contenu	Capteur avec sorties intégrées	Liste classée des capteurs	Classement basé sur la prédiction
Dyser [86]	Basé sur le contenu	Capteurs virtuels (microformat)	URL/HTTP/ HTML	Basé sur des mots-clés
Truong [119]	Basé sur le contenu	Flux de données du capteur	-	Recherche de capteurs basée sur le flux de données
GSN [3]	Basé sur le contexte	Capteurs virtuels en XML	HTTP REST	Recherche par mot-clé, requête SQL
Snoogle [126]	Basé sur le contexte	Communication directe avec les capteurs	Capteurs virtuels au format intégré	Recherche top-k basée sur des mots-clés
Mayer [74]	Basé sur le contexte	Microformats/ microdonnées	HTTP REST	Correspondance des mots clés avec la portée
Ramachandran [98]	Basé sur le contexte	Capteurs virtuels en XML	Réponse à la requête	Regroupement des fourmis en fonction du contexte
CASSARAM [94]	Basé sur le contexte	Ontologie SSN	Réponse à la requête SPARQL	Requête SPARQL
Antclust [37]	Basé sur le contexte	Ontologie SSN	Réponse à la requête	Regroupement des fourmis en fonction du contexte
Nunes [83]	Basé sur le contexte	Ontologie SSN	Réponse à la requête	Requête
E-S algorithme [82]	Basé sur le contexte	Ontologie SSN	Réponse à la requête	Requête
Hsu [53]	Basé sur le contexte	Ontologie SSN	Réponse à la requête	Requête
Méthode proposée	Basé sur le contexte	Ontologie SSN	Réponse à la requête	Requête

Tableau 3.1: Étude comparative 1 selon différentes métriques.

Technique de recherche	Échelle des expériences	Dynamicité	Architecture	Implémentation
Elahi [39]	20 capteurs	Oui	Centralisé	C++
Dyser [86]	385 capteurs	Oui	Centralisé	Java/PHP
Truong [119]	1500 points de données	Oui	Distribué	Java
GSN [3]	-	Oui	Centralisé	Java/MySQL
Snoogle [126]	-	Oui	Distribué	TelosB motes
Mayer [74]	600 capteurs simulés	Non	Distribué	ApacheBench
Ramachandran [98]	100000 capteurs	Oui	Centralisé	-
CASSARAM [94]	1000000 capteurs	Oui	Distribué	Apache Jena API
Antclust [37]	100000 capteurs	Oui	Centralisé	-
Nunes [83]	100000 capteurs	Oui	Centralisé	Java
E-S algorithme [82]	100000 capteurs	Oui	Centralisé	Java
Hsu [53]	50 capteurs	Oui	Distribué	Java
Méthode proposée	100000 capteurs	Oui	Distribué	Java/MySQL

Tableau 3.2: Étude comparative 2 selon différentes métriques.

passerelles pour éviter la duplication des informations au niveau du serveur et la mise à jour permanente entre les passerelles et le serveur. Chaque passerelle doit répondre aux demandes des utilisateurs localement et le serveur doit regrouper les résultats des

différentes passerelles pour donner les réponses finales aux utilisateurs. De plus, pour profiter de l'efficacité du dynamique Skyline dans le domaine de la prise de décision multicritères, un Skyline dynamique est appliqué pour réduire la quantité de métadonnées à traiter lors de la recherche et de la sélection des meilleurs capteurs. Enfin, pour ordonner les capteurs résultants par ordre croissant, nous utilisons la distance euclidienne.

Conclusion

Dans ce chapitre, nous avons présenté des travaux de recherche sur la recherche et la sélection des meilleurs capteurs répondant aux besoins des applications. Nous avons divisé les différents travaux en deux catégories: basé sur le contenu et basé sur le contexte. Nous avons constaté que pour la recherche de capteurs basée sur le contenu n'est pas utile pour les applications car le grand nombre de capteurs rend impossible le traitement de toutes les données collectées par ces capteurs. Bien que des progrès acceptables dans la recherche et la sélection de capteurs aient été réalisés par les méthodes basées sur le contexte, il y a encore de nombreux défis restent à faire face.

Dans le prochain chapitre, nous présenterons notre proposition en terme de méthode pour la recherche et la sélection contextuelle des meilleurs capteurs en fonction des besoins de l'utilisateur.

Chapitre 4

Méthode proposée pour sélectionner les meilleurs capteurs

Introduction

Dans le chapitre précédent, nous avons présenté les différents travaux existants dans la littérature pour la recherche et la sélection des capteurs dans le contexte de l'IdO. Plusieurs travaux de recherche ont été présentés tels que Snoogle [126], Global Sensor Networks [3], CASSARAM [94], Antclust [37] et E-S algorithme [82]. Nous avons constaté que la plupart des approches utilisent des méthodes heuristiques pour filtrer et sélectionner les capteurs en question. Cependant, le temps de recherche et de sélection est considérablement long. De plus, la plupart des approches proposées ne considèrent pas la nature distribuée des systèmes IdO. De nouveaux travaux supplémentaires sont nécessaires pour résoudre et combler les lacunes des travaux actuels.

Récemment, l'opérateur Skyline a été adopté pour réduire l'espace de recherche dans le but de sélectionner certaines entités en fonction de certaines propriétés. Le Skyline a été largement utilisé dans de nombreuses applications de prise de décision multicritères tels que la recherche dans les bases de données [11], la composition des services Web [128] et le routage dans les réseaux Ad Hoc sans fil [4]. Dans le cadre de cette thèse, nous exploitons le principe de Skyline dans le processus de recherche et de sélection des meilleurs capteurs.

Notre objectif consiste à proposer une méthode sensible au contexte qui peut être adoptée par différents middlewares IdO pour concevoir une solution pertinente. Cette dernière permettra un haut niveau de précision et réduira le temps de recherche et de sélection. La nouveauté de cette contribution est de savoir comment utiliser le principe de filtration Skyline pour sélectionner et classer les meilleurs capteurs répondant aux besoins des applications. Le but d'exploitation de Skyline consiste à sélectionner les meilleurs capteurs avec un haut niveau de précision. Cependant, le Skyline est considérablement

lent, surtout dans le cas d'une explosion combinatoire de capteurs. Nous proposons une architecture distribuée dans laquelle la filtration du Skyline sera exécutée au niveau des passerelles et au niveau du serveur. La contribution de ce travail est résumée comme suit:

- Nous exploitons la puissance de l'opérateur Skyline dynamique dans le domaine de la prise de décision multicritères pour réduire l'espace de recherche. Le but est d'améliorer l'efficacité de la recherche et de la sélection contextuelle des meilleurs capteurs en fonction des besoins utilisateurs.
- Pour assurer la nature parallèle des architectures IdO, notre architecture est composée de plusieurs passerelles réparties dans le réseau avec un serveur. Les capteurs sont connectés aux passerelles. Chaque passerelle gère son propre réseau de capteurs local, comme elle doit répondre localement aux requêtes des utilisateurs. Par la suite, le serveur regroupera les résultats de toutes les passerelles et donnera les réponses finales aux utilisateurs.
- Dans le cas où le nombre de capteurs Skyline est très élevé, notre système permet de classer les capteurs sélectionnés en fonction des exigences et des poids imposés par l'utilisateur. Pour classer les différents capteurs Skyline sélectionnés, nous utilisons la distance euclidienne.

Notre proposition a été publiée dans le journal Ad Hoc Networks Volume 81, Décembre 2018, Pages 183-196 [57].

4.1 Motivation et modélisation de l'architecture

Dans cette section, nous expliquons et discutons les différents concepts de la technique proposée. L'objectif est de présenter une vue d'ensemble de l'architecture proposée pour la découverte des capteurs. En outre, nous décrivons également les flux d'exécution globale et les étapes critiques de la proposition.

4.1.1 Architecture de découverte des capteurs

En règle générale, chaque instance de middleware IdO garde une trace des capteurs qui lui sont spécifiquement connectés. Cela signifie que chaque serveur ne connaît qu'un certain nombre de capteurs. Cependant, afin de répondre aux besoins complexes des utilisateurs, nous devons interroger plusieurs instances du middleware IdO pour trouver et sélectionner les capteurs les plus adaptés aux besoins des utilisateurs. Ces middlewares IdO permettent de partager des capteurs entre plusieurs utilisateurs et applications pour collecter des données via un abonnement à des services Web. Dans cette contribution, l'architecture utilisée dans [53] est adoptée de telle manière que plusieurs passerelles (gateways) sont réparties dans le réseau et connectées à un serveur. L'utilisateur envoie

sa demande au serveur pour fournir une réponse. Contrairement à Hsu et al. [53], nous pouvons profiter des énormes capacités de calcul et stockage au niveau des passerelles. En effet, cette situation peut être atteinte soit en évitant la duplication d'informations au niveau serveur, soit en mettant à jour périodiquement les informations des capteurs entre les passerelles et le serveur. Pour atteindre cet objectif, chaque passerelle doit répondre aux demandes des utilisateurs localement et le serveur agrégera les résultats des différentes passerelles pour fournir la réponse finale aux utilisateurs.

L'architecture proposée (voir la figure 4.1) contient trois composants:

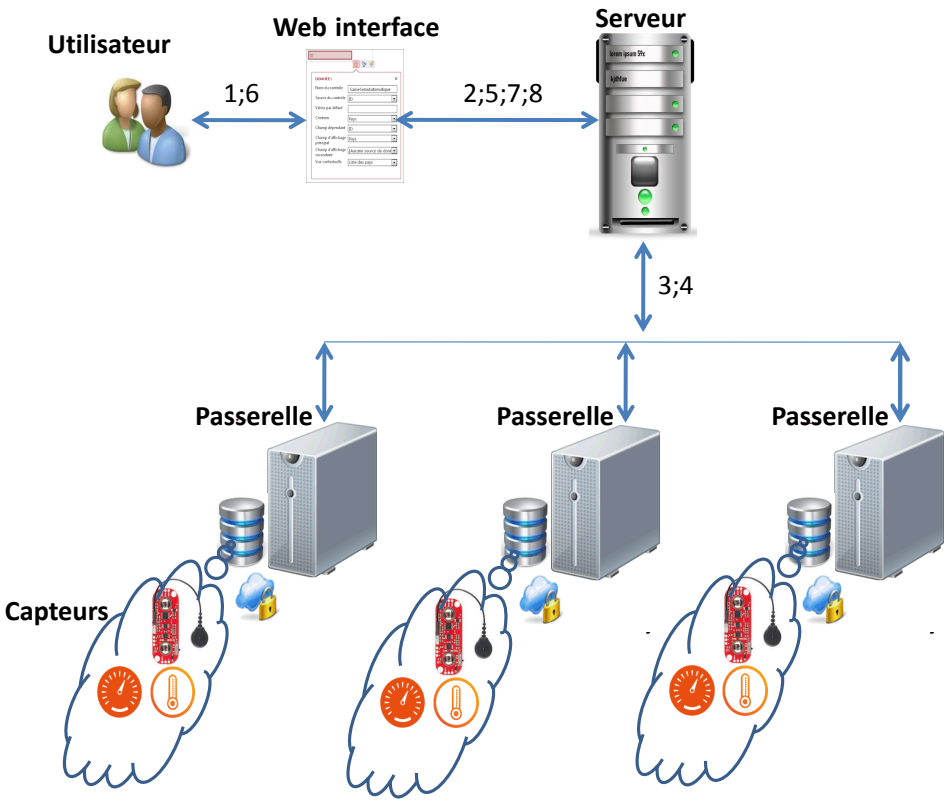

Figure 4.1: Architecture de découverte des capteurs.

- **Serveur**: le serveur est chargé de traiter les demandes des utilisateurs, de transférer les requêtes des utilisateurs vers différentes passerelles, d'agréger les résultats de toutes les passerelles et de fournir la réponse finale à l'utilisateur.
- **Passerelles (gateways)** : la passerelle est chargée au traitement des requêtes du serveur et de collecter les données des capteurs. De plus, la passerelle gère les différentes mises à jour des informations des capteurs.
- **Capteurs**: les informations et les propriétés contextuelles des capteurs sont enregistrées au niveau des passerelles. Les capteurs transfèrent leurs mesures vers leurs passerelles.

La figure 4.1 décrit un aperçu de haut niveau de notre architecture en indiquant les flux d'exécution et les étapes critiques qui peuvent être expliquées comme suit:

- **(1)**: L'utilisateur se connecte au système proposé et formule sa demande via une interface web .
- **(2)**: Le gestionnaire d'interface envoie la demande au serveur.
- **(3)**: Le serveur transmet la requête aux différentes passerelles.
- **(4)**: Chaque passerelle calcule la dynamique Skyline local et renvoie le résultat au serveur.
- **(5)**: Le serveur calcule la dynamique Skyline global et affiche le résultat à l'utilisateur.
- **(6)**: Si l'utilisateur souhaite obtenir le résultat de manière ordonnée, l'utilisateur affecte les valeurs de pondération des propriétés contextuelles via l'interface Web.
- **(7)**: Le gestionnaire d'interface envoie la demande au serveur.
- **(8)**: Le serveur classe les capteurs et affiche le résultat à l'utilisateur.

4.1.2 Description et modélisation du système

Ce travail aborde le problème de la sélection d'un sous-ensemble des meilleurs capteurs à partir d'un très grand nombre de capteurs IdO déployés. En effet, proposer une technique efficace pour rechercher et sélectionner un sous-ensemble de capteurs présente de nombreux avantages tels que la garantie d'un temps de réponse rapide, d'une gestion efficace de l'énergie, d'un coût de service réduit et d'une faible surcharge du réseau. De nombreux paramètres doivent être pris en compte lors de la modélisation de ce problème. Une modélisation adéquate nous amène à représenter efficacement le problème de la recherche et de la sélection des meilleurs éléments dans un grand pool de nœuds de capteurs. Par la suite, nous pouvons proposer une technique efficace pour résoudre le problème abordé. Nous proposons un modèle tête-base (modélisation verticale), comme le montre la figure 4.2, qui représente la manière dont l'utilisateur envoie une requête et comment il propose de rechercher et de sélectionner une meilleure solution de capteurs. Dans ce modèle triangulaire, la tête représente le niveau utilisateur et ses requêtes et la base représente le niveau des nœuds déployés et leurs propriétés. Le modèle triangulaire proposé est divisé verticalement en deux parties côté droit et côté gauche. Le côté droit examine les capteurs déployés en fonction de la demande utilisateur, en partant de la tête vers la base, c'est le processus d'enquête. Cependant, le côté gauche commence de rechercher les meilleurs capteurs en fonction de la demande de l'utilisateur en partant de la base vers la tête, c'est le processus de sélection. Toutefois, ce modèle est également divisé horizontalement en quatre couches de différentes tailles:

- La couche de base est la couche du pool qui représente l'ensemble des nœuds de capteurs déployés.
- La couche Skyline représente les nœuds sélectionnés après le processus de filtration par Skyline. Elle est placée au-dessus de la couche de pool.
- La troisième couche est la couche de classement. Elle contient les capteurs choisis après le processus de classement.
- La dernière couche est placée au-dessus de la tête du triangle. Elle représente le résultat de la demande.

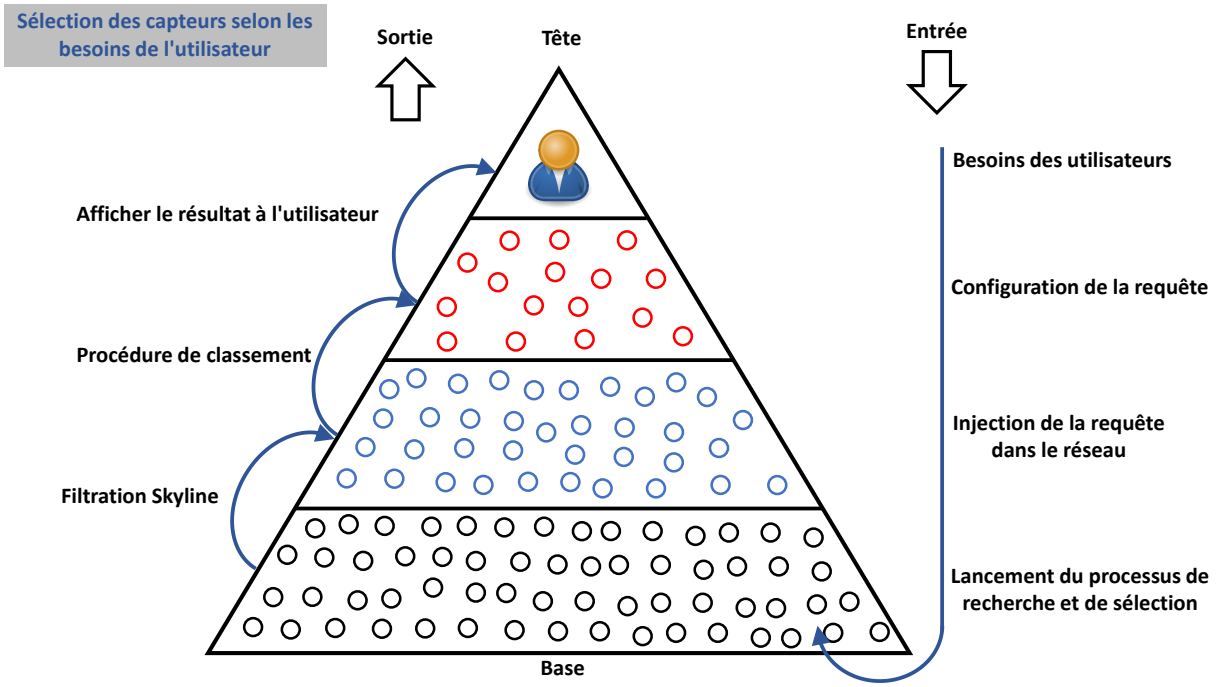

Figure 4.2: Le modèle Tête-Base pour la configuration des requêtes, la recherche et la sélection des meilleurs capteurs.

Des travaux connexes peuvent être trouvés dans la littérature [53, 37]. Certaines approches exploitent la spécification du côté gauche du modèle présenté dans la figure 4.2, pour proposer une solution [94, 83]. En effet, les solutions proposées sont uniquement applicables sur des scénarios particuliers, selon la conception de l'architecture et le modèle proposé. Notre modèle suppose le scénario dans lequel les clients sont autorisés à concevoir des capteurs comme ils les souhaitent et en fonction des informations de contexte. Par exemple, certains clients peuvent être en mesure de payer plus pour l'exactitude des données (c.-à-d. des capteurs de haute précision) tandis que d'autres peuvent être en mesure de payer moins, en fonction de leurs besoins, de leur situation et de leurs préférences. La figure 4.3 montre en détail comment filtrer un sous-ensemble de capteurs à partir d'un grand pool de capteurs IdO déployés, puis classer le sous-ensemble filtré. Le résultat est de fournir un nombre acceptable des meilleurs capteurs sélectionnés qui

répondent aux besoins des utilisateurs soumis. En fait, la figure 4.3 est une modélisation horizontale de la solution proposée. Nous observons qu'il existe trois phases principales pour répondre aux besoins des utilisateurs. Il est important de noter ici que nous pouvons définir une spécification d'ontologie de réseau sémantique du système proposé dans le but de pouvoir comprendre les propriétés des capteurs et de fournir les résultats en conséquence.

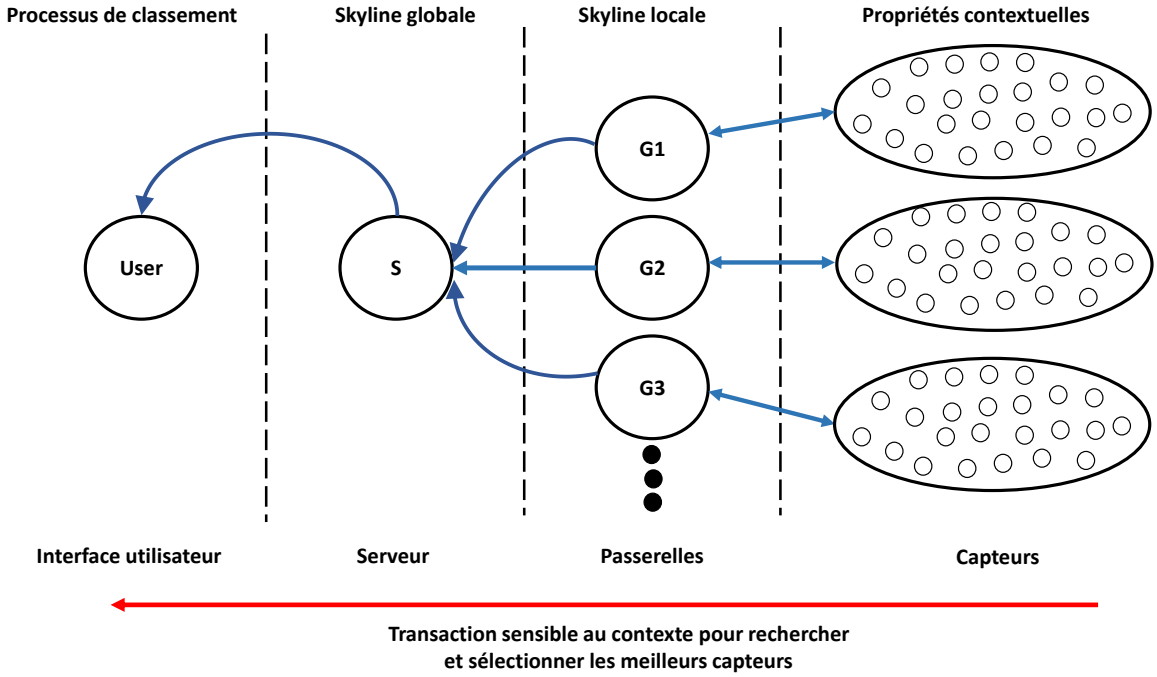

Figure 4.3: Les étapes proposées pour sélectionner les meilleurs capteurs en fonction de leurs propriétés contextuelles.

4.1.3 Ontologie de réseau de capteurs sémantique

Dans le contexte de l'IdO, l'un des principaux défis est de collecter différents types d'informations à partir d'objets différents et hétérogènes. Ces objets doivent être sémantiquement homogène pour permettre l'intégration des données dans différentes sources et pour soutenir des mécanismes de raisonnement et de prise de décision autonome.

Pour modéliser la description des capteurs et les propriétés de contexte, et pour publier les informations des capteurs dans un format uniforme, nous avons opté pour l'ontologie de réseau de capteurs sémantique (Semantic sensor network ontology: SSN) [32]. L'utilisation de l'ontologie dans l'IdO et l'interopérabilité conduit les chercheurs à accepter l'utilisation de l'ontologie SSN. Cette dernière montre un modèle d'architecture de haut niveau pour définir les caractéristiques du capteur (telles que les capacités des capteurs, les performances des capteurs et les conditions dans lesquelles ils peuvent être utilisés). L'ontologie SSN exploite les propriétés de contexte les plus

courantes, telles que l'exactitude, le coût, la précision, la sensibilité, la sélectivité, la plage de mesure, la limite de détection, le temps de réponse, la fréquence et la latence. L'ontologie peut être décrite par quatre aspects principaux:

- **Le capteur**: le rôle du capteur tel que quel type d'événement et quoi sentir;
- **L'observation**: la nature des informations observées;
- **Le système**: les systèmes de communications et de déploiements;
- **Les caractéristiques et les propriétés**: quelles informations pertinentes et propriétés particulières détectées.

Ce travail se concentre sur la perspective du capteur pour décrire les informations des capteurs comme illustré dans la figure 4.4.

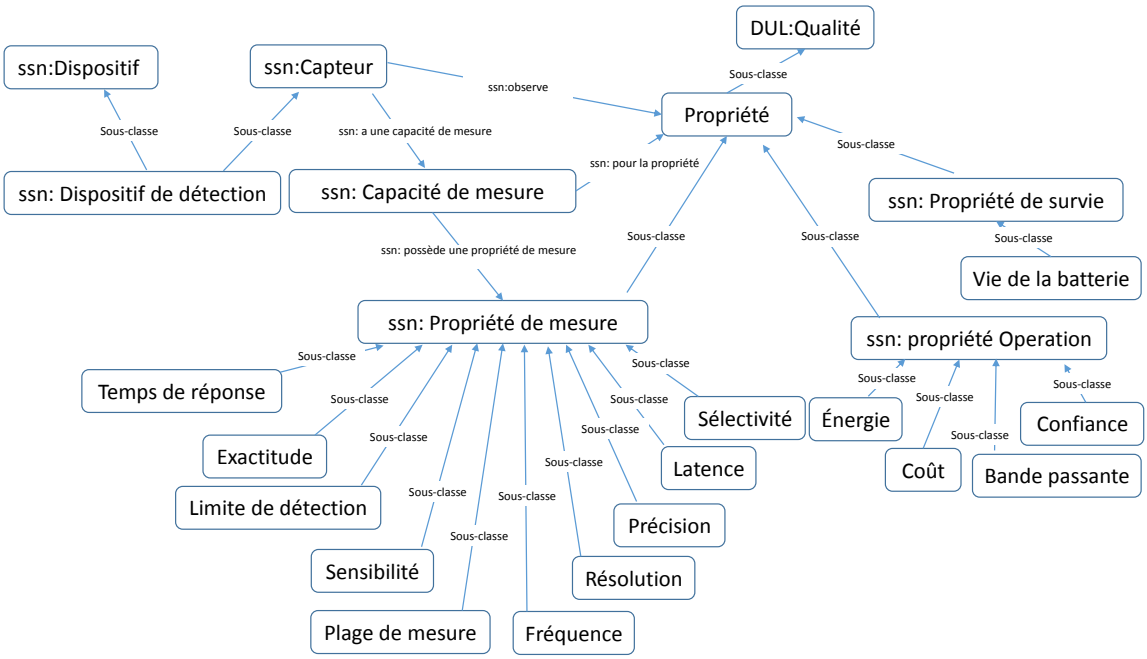

Figure 4.4: Ontologie SSN décrite du point de vue du capteur.

4.2 Méthode proposée pour sélectionner les meilleurs capteurs

Afin de gérer des milliers voire des millions de capteurs et de fournir des résultats intuitifs aux utilisateurs, nous proposons une méthode de recherche et de sélection des meilleurs capteurs associés suivant les besoins des utilisateurs en fonction de leurs propriétés contextuelles. Sur la base du modèle proposé dans la figure 4.3, nous discuterons en détail dans cette section chaque tâche. En commençant par la soumission des exigences

de l'utilisateur jusqu'à ce que l'on réponde à l'utilisateur en sélectionnant les meilleurs capteurs en fonction de l'exigence soumise.

4.2.1 Skyline et Skyline dynamique

Depuis les années 60, les requêtes Skyline ont été étudiées dans le domaine de la théorie où les objets Skyline sont connus sous le nom d'ensemble Pareto. L'opérateur Skyline [18] et ses variantes comme Skyline inversé [33] et Skyline dynamique [87] ont récemment attiré l'attention des chercheurs dans le domaine de la prise de décision à critères multiples. Dans cette section, nous présenterons la requête Skyline et Skyline dynamique et décrivons comment lier le Skyline dynamique à l'approche proposée pour résoudre le problème de la recherche et de la sélection de capteurs en fonction du contexte.

L'exemple typique de Skyline, tel que présenté dans [18], est la sélection de l'hôtel disposant le meilleur prix et le plus proche de la plage. Un hôtel h_1 avec un prix de 500$ et une distance de 1 mile de la plage est meilleur que l'hôtel h_2 avec un prix de 600$ et une distance de 2 miles de la plage. On dit que h_1 domine h_2. Dans la phase de sélection des meilleurs hôtels, h_2 est ignoré car il est dominé par h_1. En appliquant cette intuition, le principe du Skyline est utile pour la phase de sélection notamment en présence d'un grand nombre de points de données. Dans notre étude de cas, le fait que nous ayons un grand nombre de capteurs ainsi que leurs propriétés contextuelles (coût, précision, temps de réponse, etc.) rend préférable de tirer profit du principe du Skyline.

Skyline est l'ensemble de tous les points qui ne sont pas dominés par aucun autre point [18]. Supposons qu'on a un ensemble P de points de données dans un espace d-dimensionnel qui représentent les capteurs dans notre proposition. Chaque dimension représente une propriété contextuelle des capteurs attribuée avec des valeurs correctement ordonnées. Nous supposons que la valeur la plus petite est préférable dans chaque attribut. Ainsi, un capteur S_i donné est meilleur qu'un capteur S_j par rapport à P, si et seulement si S_i n'est pas supérieur à S_j dans toutes les dimensions. De plus, S_i doit être plus petit que S_j dans au moins une dimension et alors on dit que S_i domine S_j.

La figure 4.5 illustre un exemple en considérant les propriétés contextuelles bidimensionnelles dans l'espace P. P se compose d'un ensemble de capteurs qui sont présentés dans le tableau 4.1. Chaque capteur a deux caractéristiques contextuelles: la précision et le coût. Nous pouvons observer que S_1 est un meilleur choix que S_2 car S_1 possède les plus petites valeurs à la fois de coût et de précision par rapport à S_2, ce qui signifie que S_1 domine S_2. Sur la base de cette intuition, il est possible d'identifier les capteurs Skyline qui ne sont pas dominés par aucun autre capteur. Un utilisateur donné qui cherche à choisir les meilleurs capteurs peut considérer que les capteurs Skyline $\{S_1,S_3,S_5,S_7\}$, car le Skyline ne contient que les meilleurs capteurs.

En général, la sélection des meilleurs points dépend de la requête de l'utilisateur. Pour mieux comprendre le principe du Skyline dynamique, nous revenons à l'exemple de

ID Capteurs	Précision	Coût
S_1	15	85
S_2	85	95
S_3	55	35
S_4	80	55
S_5	60	15
S_6	70	40
S_7	40	70
S_8	65	90

Tableau 4.1: Un ensemble de capteurs avec deux propriétés contextuelles

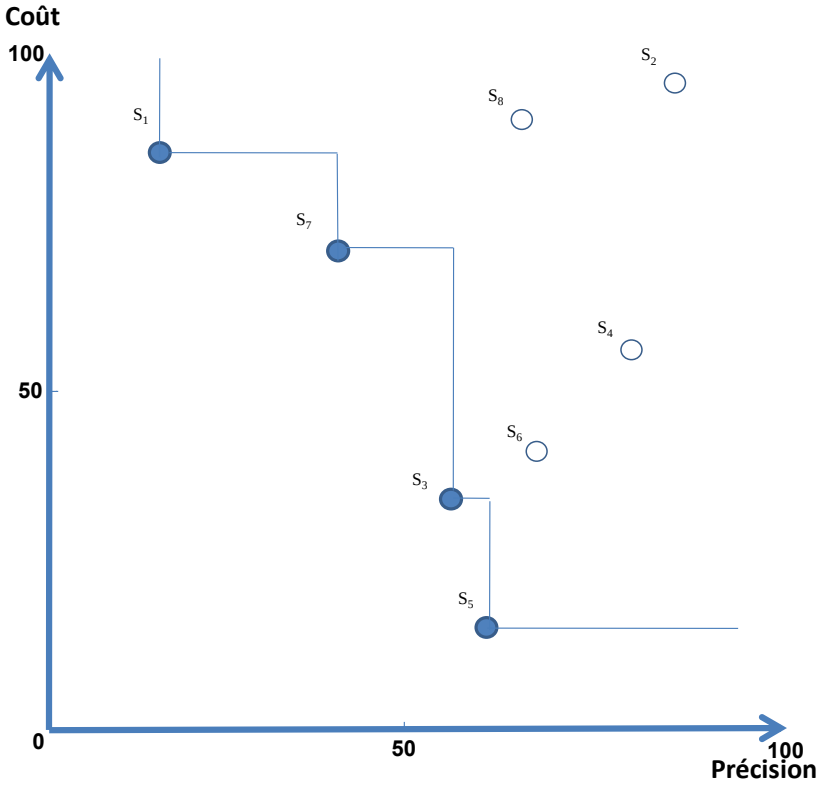

Figure 4.5: Sélection des capteurs Skyline.

sélection des meilleurs hôtels. Prenons le cas où un utilisateur donné souhaite sélectionner un hôtel dont le prix est d'environ 580$ et dont la distance est d'environ 1,5 km. Dans cette situation, hôtel h_2 est meilleur que h_1 selon la requête de l'utilisateur car h_2 est plus proche des exigences de l'utilisateur que h_1. On dit que h_2 domine dynamiquement h_1.

Le Skyline dynamique est l'ensemble de tous les points qui ne sont pas dominés par aucun autre point par rapport à un point de requête donné [87]. Nous acceptons la définition du Skyline dynamique présentée dans [33], qui spécifie un nouvel espace d'-dimensionnel basé sur l'espace de données original d-dimensionnel. Étant donné un ensemble P de capteurs dans un espace d-dimensionnel de propriétés de contexte et un capteur de requête q, une requête Skyline dynamique selon q récupère tous les capteurs de P qui ne sont pas dominés dynamiquement. Un capteur S_i domine dynamiquement S_j par rapport au capteur de requête q, noté $S_i <_q S_j$, si et seulement si:

$$|S_i(k) - q(k)| \leq |S_j(k) - q(k)| \tag{4.1}$$

$\forall\ k$ avec $1 \leq k \leq d$ et il existe k avec $1 \leq k \leq d$ tels que

$$|S_i(k) - q(k)| < |S_j(k) - q(k)| \tag{4.2}$$

où $S_i(k)$ représente la valeur de la k-ème propriété contextuelle du capteur S_i.

Par exemple, sur la figure 4.6, nous observons que S_6 est meilleur que S_1 selon la requête q (le coût et la précision sont d'environ 50), c'est-à-dire que S_6 domine dynamiquement S_1. Sur la base de cette intuition, il est possible d'identifier les capteurs Skylines dynamiques qui ne sont pas dynamiquement dominés par aucun autre capteur. Un utilisateur qui souhaite sélectionner un capteur considère les capteurs dans le Skyline dynamique $\{S_3,S_4,S_6\}$, car le Skyline dynamique contient les meilleurs capteurs selon q.

4.2.2 Recherche et sélection des meilleurs capteurs

Une précision, une efficacité et une évolutivité améliorées de la recherche de capteurs dans l'IdO peuvent être obtenues en utilisant la technique du Skyline proposée. L'objectif principal est de permettre à l'utilisateur de rechercher et de sélectionner les capteurs qui correspondent le mieux à ses besoins. La technique de recherche et de sélection des capteurs proposée aidera l'utilisateur à évaluer l'adéquation des capteurs pour toutes les applications.

L'approche Skyline et ses variantes comme le Skyline dynamique et le Skyline inversé ont récemment attiré l'attention des chercheurs en raison de leur large champ d'application. Nous adaptons la requête Skyline dynamique pour réduire l'espace de recherche dans le but d'améliorer l'efficacité de la sélection des capteurs en fonction des propriétés contextuelles. Le Skyline dynamique est l'ensemble de tous les capteurs qui ne sont pas dynamiquement dominés par aucun autre en respectant une distance à un

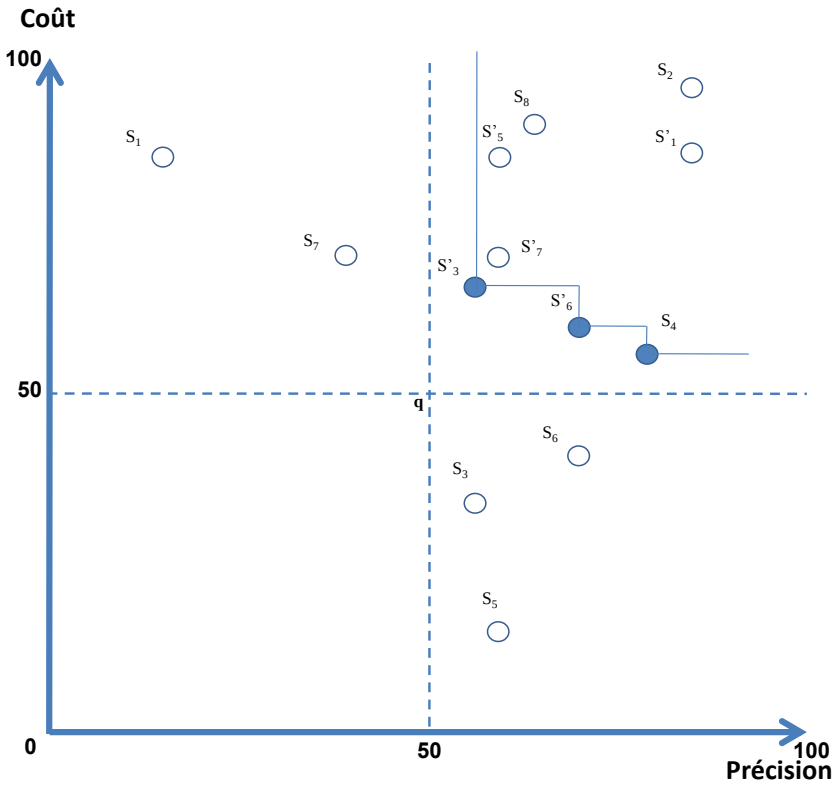

Figure 4.6: Sélection des capteurs Skyline dynamique.

capteur de requête donné. En particulier, nous nous concentrons sur les capteurs du Skylines dynamiques dans le processus de sélection des capteurs sensibles au contexte. Les capteurs Skylines non dynamiques sont dominés par les capteurs dynamiques car les capteurs du Skylines dynamiques ont de meilleures propriétés de contexte.

Les étapes d'exécution de la technique Skyline proposée sont détaillées dans l'algorithme 4.1, dont les abréviations utilisées sont présentées dans le tableau 4.2.

Algorithme 4.1 : Algorithme de recherche et de sélection des meilleurs capteurs

Algorithme *(Entré : $G(Passerelle), Q, ord, W$; Sortie : $GDS, GDSR$)*
Début
  Q := capture de la requête de l'utilisateur;
  PourChaque G_i **Faire**
    $LDS(G_i)$:= calcul du Skyline dynamique local (G_i, Q);
  FinPourChaque
  GDS := calcul du Skyline dynamique global à partir LDS;
  Si *ord* **Alors**
    W :=capture des poids des exigences approximatives imposés par l'utilisateur;
    $GDSR$:= classement des capteurs de GDS selon la distance euclidienne(GDS, W);
  FinSi
Fin

Comme il est mentionné dans la section 4.1.2, la procédure de recherche et de sélection des capteurs comprend quatre phases:

Abréviations	Définition
G_i	Passerelle i
SG_i	Ensemble de données de la passerelle i
S_i	i^{éme} capteur
Q	Requête utilisateur
$PoiR$	Exigences nécessaires
$ProR$	Exigences approximatives
$index_1$ and $index_2$	Liste des capteurs filtrés
$card(index)$	Nombre de capteurs dans l'index
$LDS(G_i)$	Skyline dynamique local de G_i
GDS	Skyline dynamique global
ord	Un booléen utilisé pour ordonner le résultat obtenu
W	Poids des exigences approximatives
MA	Matrice d'analyse
MAN	Matrice d'analyse normalisée
$GDSR$	GDS classé

Tableau 4.2: Liste d'abréviations utilisées

4.2.2.1 Phase 1: Capture de la requête de l'utilisateur

Une fois l'utilisateur connecté au système proposé, l'utilisateur formule ses besoins via une interface Web. Par la suite, le gestionnaire d'interface envoie la requête au serveur et le serveur joue son rôle en transférant la requête vers les différentes passerelles.

4.2.2.2 Phase 2: Calcul du Skyline dynamique local

L'algorithme 4.2 montre le processus de calcul du Skyline dynamique local après la réception de la requête de l'utilisateur. Chaque passerelle calcule le Skyline dynamique local comme suit:

- Les exigences des utilisateurs sont divisées en deux catégories: exigences nécessaires et exigences approximatives. Les exigences nécessaires sont les propriétés que le système doit respecter. Cependant, dans les exigences approximatives, l'utilisateur accepte les capteurs avec une légère variation dans les propriétés contextuelles, si ces capteurs n'affectent pas les performances de l'application.
- Filtrage horizontal: A partir de la liste des capteurs inclus dans la passerelle, nous créons un index des capteurs respectant les exigences nécessaires.
- Filtrage vertical: nous supprimons les colonnes de l'index créé qui ne sont pas incluses dans les exigences approximatives. Cette opération minimise la taille des données à transférer vers le serveur.
- Skyline dynamique local: nous utilisons les exigences approximatives pour calculer le Skyline dynamique local de la manière suivante: étant donné une requête q (valeur des exigences approximatives), nous jugeons que le capteur $S1$ domine

dynamiquement le capteur $S2$ par rapport à q, noté $S1 < q\ S2$, si et seulement si les équations (4.1) et (4.2) sont vérifiées.

- La liste des capteurs qui ne sont pas dominés dynamiquement par aucun autre capteur par rapport à la requête q est transférée au serveur.

Algorithme 4.2 : Skyline dynamique local

Procédure *(Entré : $SG(Ensemble - de - données), Q$; Sortie : LDS)*
Début
  $PoiR := extract1(Q)$;
  $ProR := extract2(Q)$;
  $index_1 := horizontal\_filter(SG, PoiR)$;
  $index_2 := vertical\_filter(Index1, ProR)$;
  Pour $i = 1$ **Jusqu'à** $card(index_2)$ **Faire**
    $bool := true$;
    $j := 1$;
    Tantque *($j \leq\ card(index2)$) & (bool)* **Faire**
      Si *(S_j domine dynamiquement S_i)* **Alors**
        $bool := false$;
      FinSi
      $j := j + 1$;
    FinTantque
    Si *bool* **Alors**
      Ajouter S_i to LDS;
    FinSi
  FinPour
Fin

4.2.2.3 Phase 3: Calcul du Skyline dynamique global

Lorsque le serveur reçoit les différents résultats du Skyline dynamique local, c'est-à-dire les listes des meilleurs capteurs de différentes passerelles. Il fusionne tous les capteurs et calcule le Skyline dynamique global. Le serveur affiche ensuite le résultat à l'utilisateur si ord = faux.

4.2.2.4 Phase 4: Classement des capteurs

Dans le cas où le nombre des exigences approximatives augmente, la probabilité qu'un capteur domine un autre va diminuer. En conséquence, le nombre de capteurs Skyline dynamiques devient trop énorme et ne fournit pas d'informations intéressantes. Pour cette raison, l'utilisateur peut demander au serveur de représenter la liste des capteurs résultante de manière ordonnée. Pour classer les capteurs, nous utilisons l'algorithme 4.3 qui décrit le processus de classement. Par conséquent, l'ordre des capteurs est obtenu comme suit:

- L'utilisateur introduit le poids de chaque exigence approximative via une interface Web. Le gestionnaire d'interface envoie les poids au serveur.

- Au niveau du serveur, la liste des capteurs du Skyline dynamique global est considérée comme une matrice d'analyse $MA=(q_{ij})$, $i = 1..n$, $j = 1..m$, où chaque ligne représente un capteur et chaque dimension (colonne) représente une propriété contextuelle (ex. précision, fiabilité ou latence), n est le nombre de capteurs et m est le nombre d'exigences approximatives. De plus, chaque élément de la matrice q_{ij} représente la valeur de la propriété contextuelle j du capteur i .
- Nous utilisons ensuite la formule suivante pour normaliser la matrice d'analyse sur $[0, 1]$:

$$q'_{ij} = \frac{q_{ij} - q_j^{min}}{q_j^{max} - q_j^{min}}$$

où q_j^{min} est la valeur minimale et q_j^{max} la valeur maximale de la colonne j.

- Par la suite, nous calculons la distance euclidienne entre chaque capteur de la matrice et le capteur idéal S_{ideal} comme suit:

$$d(S, S_{ideal}) = \sqrt{\sum_{j=1}^{m} [w_j (S^j - S_{ideal}^j)^2]}$$

où w_j représente le poids de la $j^{éme}$ propriété contextuelle des capteurs.

- Enfin, nous classons les capteurs par ordre croissant en fonction de la distance euclidienne avant de les envoyer à l'utilisateur.

Algorithme 4.3 : Classement des capteurs

Procédure *(Entré : GDS, W; Sortie : $GDSR$)*

Début

MA := créer la matrice d'analyse;
MAN := MA normalisé ;
Calcule de distance euclidienne entre chaque capteur (ligne) dans MAN et le capteur idéal;
$GDSR$:= Capteurs de MAN sont classés en fonction de la distance euclidienne.

Fin

4.2.3 Exemple d'illustration

Pour expliquer plus en détail la méthode de recherche et de sélection proposée, dans cette section, nous présentons un exemple pour illustrer et justifier les différentes étapes du processus. Supposant qu'il existe trois passerelles G_1, G_2 et G_3. Chaque passerelle gère un ensemble de capteurs (voir les tableaux 4.3, 4.4 et 4.5). Dans ces tableaux, chaque ligne représente un capteur et chaque colonne représente une propriété contextuelle des capteurs (c-à-d. ID de capteur, localité, type de capteur, précision, fiabilité et le coût). Nous utilisons $S_{i,j}$ pour désigner un capteur, où i représente le numéro de la passerelle et j l'identification du capteur dans la passerelle correspondante. Nous considérons l'exemple

de cas où un utilisateur demande les meilleurs capteurs de température dans la ville d'Alger avec une précision autour de 92, et un coût autour de 7. Afin de rechercher et sélectionner les meilleurs capteurs qui répondent aux besoins de l'utilisateur, nous appliquons les phases ci-dessus:

- **Phase 1**: Lorsque le serveur reçoit une requête utilisateur, il la transmet aux différentes passerelles.
- **Phase 2**: Chaque passerelle calcule le Skyline dynamique local comme suit: premièrement les besoins de l'utilisateur sont catégorisés en deux: «Alger» et «température» sont considérés comme des exigences nécessaires, et «précision» et «coût» comme des exigences approximatives. Ensuite, nous effectuons le filtrage horizontal et vertical. Ci-après, nous utilisons les exigences approximatives pour calculer le Skyline dynamique local (voir les tableaux 4.6, 4.7 et 4.8). La passerelle envoie ensuite le résultat au serveur.
- **Phase 3**: Lors de la réception des résultats locaux des passerelles, le serveur fusionne tous les résultats reçus et calcule le Skyline dynamique global comme présenté dans le tableau 4.9. Enfin, le résultat est affiché à l'utilisateur.
- **Phase 4**: Avant que l'utilisateur demande au serveur de représenter la liste des capteurs résultante de manière ordonnée, l'utilisateur doit introduire le poids de chaque exigence approximative via une interface Web. Le gestionnaire d'interface envoie les poids au serveur. Le serveur normalise la matrice d'analyse MA des capteurs. Ensuite, il calcule les distances euclidiennes entre les capteurs résultants et le capteur idéal. Si nous considérons les poids $0,7$ et $0,3$ pour la précision et le coût, le résultat obtenu est présenté dans le tableau 4.10.

ID capteur	localité	Type de capteur	Précision	Fiabilité	Coût
$S_{1,1}$	Alger	Température	93	90	16
$S_{1,2}$	Alger	Pression	85	55	10
$S_{1,3}$	Skikda	Température	94	84	9
$S_{1,4}$	Alger	Température	94	87	9
$S_{1,5}$	Alger	Température	85	100	5
$S_{1,6}$	Alger	Température	77	87	3

Tableau 4.3: Liste des capteurs de la passerelle 1.

Conclusion

Dans ce chapitre, nous avons présenté notre méthode de recherche et sélection sensible au contexte des capteurs selon les besoins des applications. Nous sommes basés sur les informations de contexte des capteurs pour rechercher et sélectionner les meilleurs capteurs pour une requête utilisateur. Nous avons profité de la puissance du Skyline

ID capteur	localité	Type de capteur	Précision	Fiabilité	Coût
$S_{2,1}$	Alger	Température	91	90	10
$S_{2,2}$	Alger	Pression	85	55	6
$S_{2,3}$	Oran	Température	77	84	9
$S_{2,4}$	Adrrar	Température	91	87	12
$S_{2,5}$	Alger	Température	95	100	6
$S_{2,6}$	Alger	Température	77	87	1

Tableau 4.4: Liste des capteurs de la passerelle 2.

ID capteur	localité	Type de capteur	Précision	Fiabilité	Coût
$S_{3,1}$	Alger	Température	99	90	2
$S_{3,2}$	El Oued	Pression	85	55	6
$S_{3,3}$	Alger	Température	88	84	7
$S_{3,4}$	Alger	Température	95	87	9
$S_{3,5}$	Skikda	Température	75	100	4
$S_{3,6}$	Alger	Pression	77	87	8

Tableau 4.5: Liste des capteurs de la passerelle 3.

ID capteur	Précision	Coût
$S_{1,1}$	93	16
$S_{1,4}$	94	9

Tableau 4.6: Liste des capteurs du Skyline dynamique local de la passerelle 1.

ID capteur	Précision	Coût
$S_{2,1}$	91	10
$S_{2,5}$	95	6

Tableau 4.7: Liste des capteurs du Skyline dynamique local de la passerelle 2.

ID capteur	Précision	Coût
$S_{3,3}$	88	7
$S_{3,4}$	95	9

Tableau 4.8: Liste des capteurs du Skyline dynamique local de la passerelle 3.

ID capteur	Précision	Coût
$S_{1,4}$	94	9
$S_{2,1}$	91	10
$S_{2,5}$	95	6
$S_{3,3}$	88	7

Tableau 4.9: Liste des capteurs du Skyline dynamique global.

ID capteur	Précision normalisée	Coût normalisé	Distance
$S_{1,4}$	$0,86$	$0,75$	$0,36$
$S_{2,5}$	$1,00$	$0,00$	$0,38$
$S_{2,1}$	$0,43$	$1,00$	$0,43$
$S_{3,3}$	$0,00$	$0,25$	$0,48$

Tableau 4.10: Liste des capteurs classés par distance.

dynamique dans le domaine de la prise de décision multicritères pour réduire l'espace de recherche et sélectionner les meilleurs capteurs en fonction des besoins des utilisateurs. De plus, pour assurer le caractère parallèle de l'IdO, notre architecture est composée de plusieurs passerelles réparties au sein du réseau et connectées à un serveur, chaque passerelle répond aux demandes des utilisateurs localement. Puis le serveur agrégera les résultants de toutes les passerelles pour donner la réponse finale. Finalement, et dans le cas où le nombre de capteurs Skyline dynamique est très élevé, notre système utilise la distance euclidienne pour présenter les capteurs résultants par ordre de priorité.

Dans le prochain chapitre, nous présenterons l'implémentation du prototype de l'architecture proposée. Nous également discuterons les différents résultats obtenus et comparerons les performances de notre approche avec d'autres travaux existants.

Chapitre 5

Expérimentations et analyses

Introduction

Dans le chapitre précédent, nous avons présenté notre méthode de recherche et de sélection sensible au contexte des capteurs selon les besoins des applications. Nous avons utilisé les informations contextuelles des capteurs pour découvrir et sélectionner les meilleurs capteurs pour une requête utilisateur. Nous avons profité de la puissance du Skyline dynamique dans le domaine de la prise de décision multicritères pour réduire l'espace de recherche, afin de sélectionner les meilleurs capteurs en fonction des besoins des utilisateurs.

Dans ce chapitre, nous implémentons un prototype de l'architecture proposée et les jeux de données décrits. Nous avons également discuté les résultats obtenus de différentes études de cas. Puis, ces résultats sont comparés aux méthodes CASSARAM, AntClust et E-S algorithme afin de montrer l'efficacité, la robustesse et l'influence de la technique proposée.

5.1 Implémentation

Pour évaluer et analyser la méthode proposée, un réseau privé composé d'un serveur et de trois nœuds de calcul a été utilisé (chaque nœud représente une passerelle). Nous avons créé un prototype en utilisant Java Eclipse Helios. Dans cette implémentation, nous avons calculé les services Skyline avec l'algorithme BNL (Block Nested Loop) [18]. Les données ont été stockées dans une base de données MySQL. Le prototype aide les utilisateurs à introduire leurs préférences et les priorités des différentes propriétés contextuelles de capteurs. Nous avons utilisé des ordinateurs avec un processeur Intel (R) Core i5-2557M 1,70 GHz et 4 Go de RAM pour évaluer la méthode proposée. Dans ce travail, nous avons supposé que les descriptions des capteurs, telles que les capacités et les mesures des capteurs, avaient déjà été récupérées auprès des fabricants de capteurs et fusionnées dans le SSN.

5.2 Collecte de données

Aucun ensemble de données (Dataset) public, à l'heure actuelle, ne fournit à une grande échelle de capteurs et leurs informations contextuelles. Pour cette raison, nous avons utilisé une combinaison de données réelles et synthétiques. Nous avons collecté les données des capteurs du projet Linked Sensor Middleware (LSM) [68], du Bureau de Meteorology [43] et du projet Phenonet [91]. Cette combinaison de données permet de fournir une grande quantité de données de capteurs et aide à mieux comprendre le comportement de notre proposition dans différents contextes du monde réel liés à l'IdO. Dans nos expériences, nous avons considéré 5 propriétés contextuelles (disponibilité, précision, fiabilité, temps de réponse et coût) des capteurs. Plus de propriétés contextuelles peuvent être ajoutées à cette liste si nécessaire. Pour une bonne analyse des résultats, nous avons pris les moyennes de plusieurs expériences.

5.3 Analyse de performance

Nous avons évalué la méthode proposée en utilisant différents paramètres tels que le nombre de capteurs, le temps de traitement et le nombre de capteurs sélectionnés. Cette section explique également les expériences menées dans le but de montrer l'efficacité de la méthode proposée et sa faisabilité en utilisant plusieurs paramètres.

La figure 5.1 montre le temps de traitement requis pour rechercher et sélectionner les meilleurs capteurs en fonction des exigences de l'utilisateur avec différents nombres de capteurs et propriétés contextuelles. On remarque que si on réduit le nombre total de capteurs à moins de 10000, le temps de recherche et sélection de la méthode proposée va diminuer. Par contre, le temps de traitement a considérablement augmenté au-dessus de 50000 capteurs.

D'autre part, nous avons extrait le nombre de capteurs résultants pour le cas d'un nombre croissant de propriétés contextuelles et de capteurs comme illustré dans la figure 5.2. Les résultats obtenus montrent que l'augmentation de ces nombres (propriétés contextuelles et des capteurs) conduit à une augmentation significative du nombre de capteurs résultants.

Les figures 5.3, 5.4, 5.5 et 5.6 montrent le temps nécessaire pour rechercher et sélectionner les capteurs en question pour un nombre croissant de capteurs et un nombre différent de propriétés contextuelles. Nous avons utilisé quatre cas différents illustrés dans la figure 5.3 avec deux propriétés contextuelles, la figure 5.4 avec trois propriétés contextuelles, la figure 5.5 avec quatre propriétés contextuelles et la figure 5.6 avec cinq propriétés contextuelles. Chaque phase (c'est-à-dire, Skyline dynamique local et Skyline dynamique global et le classement) a été mesurée séparément. La phase du Skyline dynamique local nécessite particulièrement plus de temps de traitement que les phases

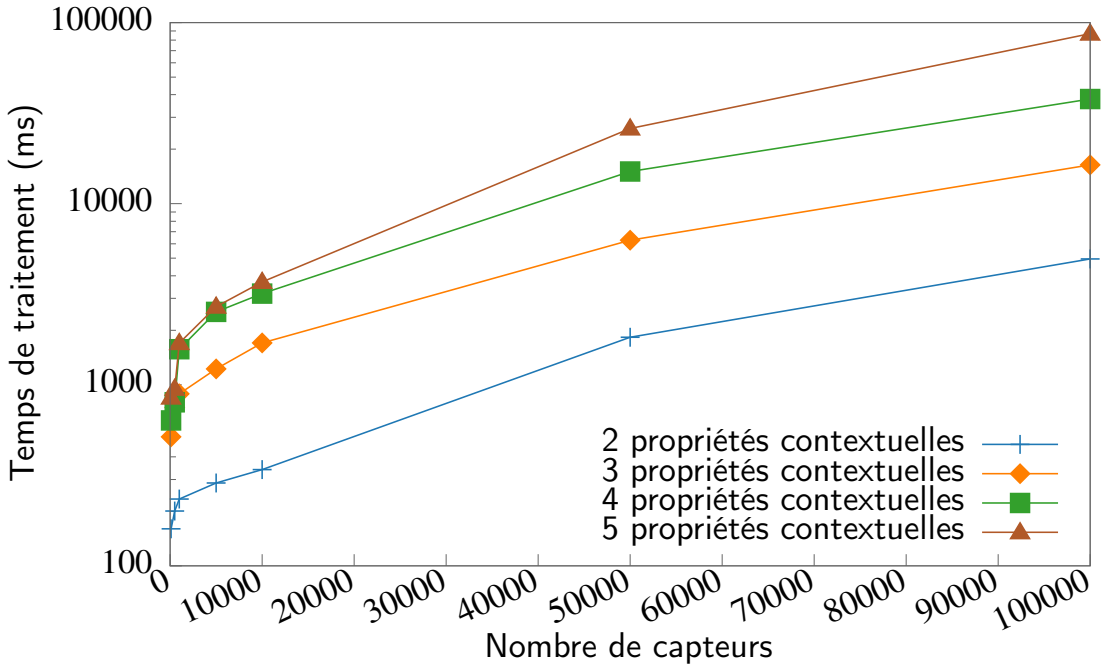

Figure 5.1: Temps de traitement pour rechercher et sélectionner les meilleurs capteurs pour un nombre croissant de propriétés contextuelles et de capteurs.

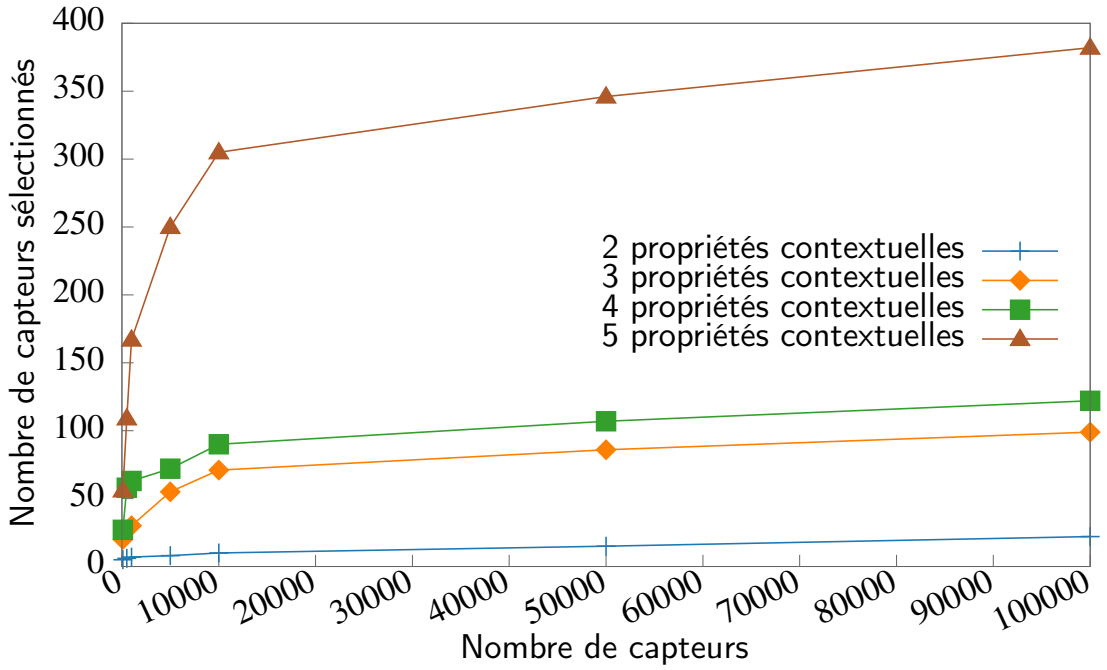

Figure 5.2: Nombre de capteurs sélectionnés pour un nombre croissant de propriétés contextuelles et de capteurs.

du Skyline dynamique global et le classement car le nombre de capteurs qui doivent être gérés au niveau des passerelles est généralement supérieur au nombre de capteurs qui doivent être gérés au niveau du serveur. De plus, l'augmentation du nombre de propriétés contextuelles requises par la requête augmentera considérablement le temps d'exécution.

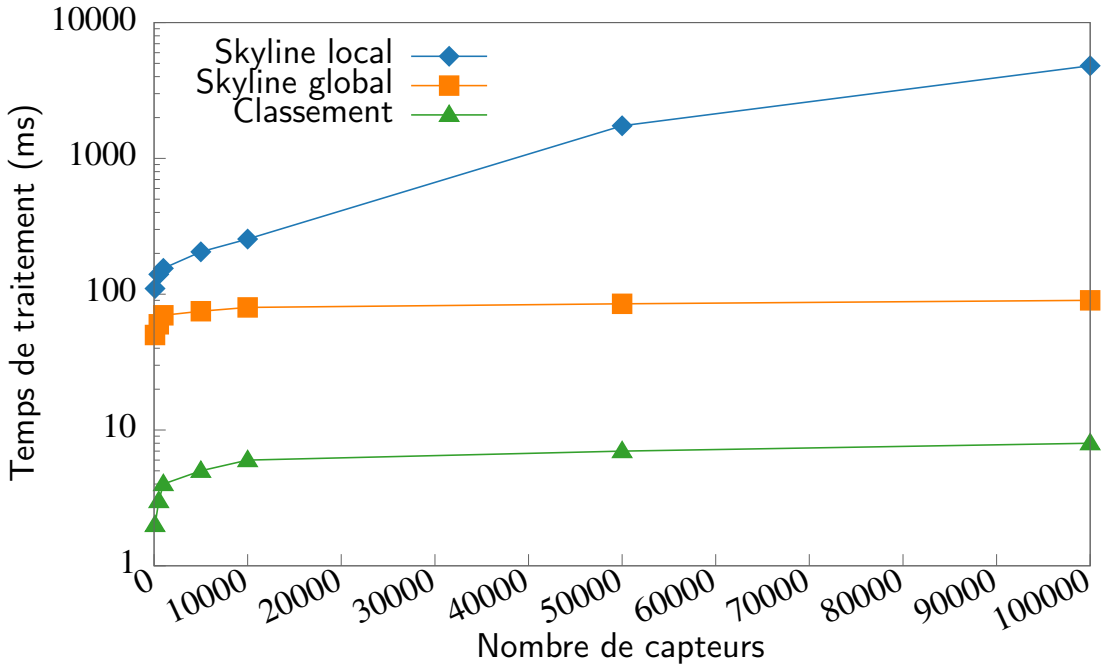

Figure 5.3: Temps de traitement du Skyline dynamique local, Skyline dynamique global et du classement pour deux propriétés contextuelles.

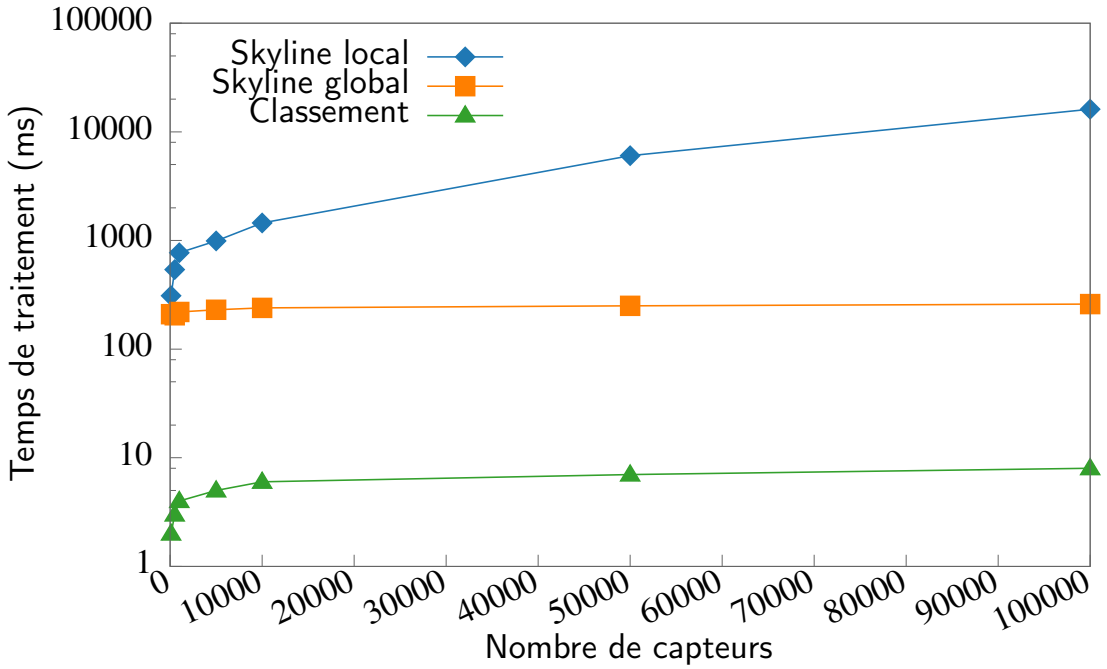

Figure 5.4: Temps de traitement du Skyline dynamique local, Skyline dynamique global et du classement pour trois propriétés contextuelles.

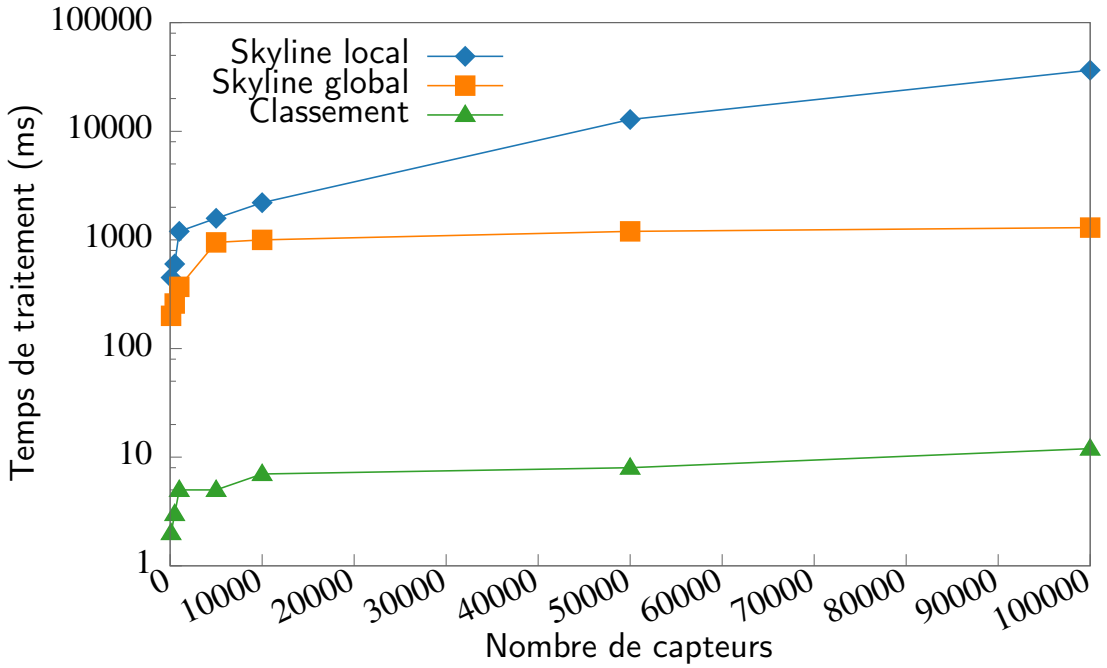

Figure 5.5: Temps de traitement du Skyline dynamique local, Skyline dynamique global et du classement pour quatre propriétés contextuelles.

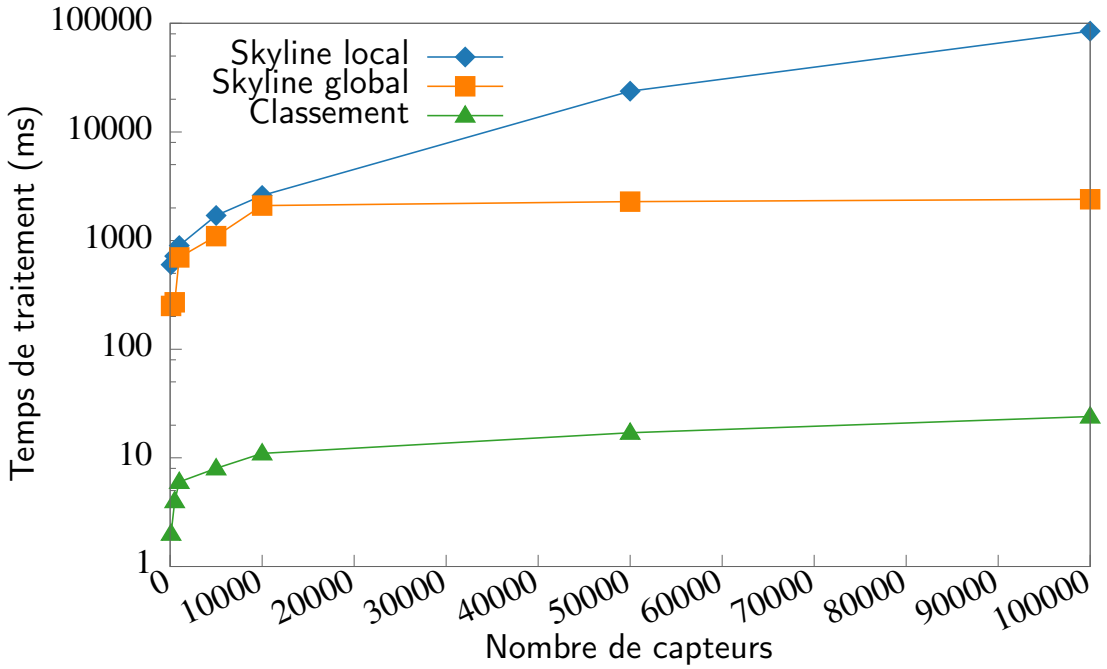

Figure 5.6: Temps de traitement du Skyline dynamique local, Skyline dynamique global et du classement pour cinq propriétés contextuelles.

La figure 5.7 montre le temps de traitement requis pour rechercher et sélectionner les meilleurs capteurs en fonction des besoins de l'utilisateur avec un nombre différent de capteurs dans le cas d'une seule et de trois passerelles. L'expérience démontre que le temps de traitement pour rechercher et sélectionner les meilleurs capteurs avec trois passerelles est toujours meilleur. Dans le cas où le nombre de capteurs est inférieur à 10000, l'intervalle de temps de traitement est significativement petit. Par contre, dans le cas où le nombre de capteurs dépasse 10000, l'écart de temps de traitement commence à être énorme. Ce résultat justifie notre choix d'une architecture à plusieurs passerelles.

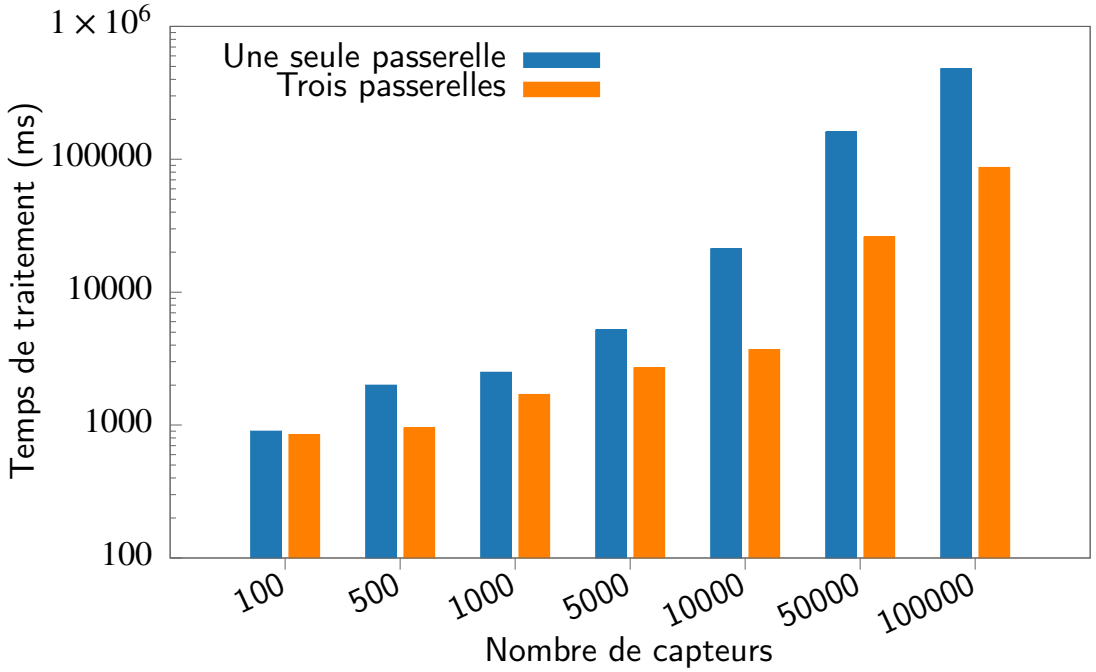

Figure 5.7: Comparaison des performances de la technique proposée dans le cas d'une seule passerelle et de trois passerelles.

L'évolutivité est la capacité de prendre en charge des réseaux plus importants. Une technique de filtration Skyline dynamique doit supporter des réseaux à grande échelle, et doit être flexible par rapport au augmentation substantielle de la taille du réseau même après le déploiement. On peut observer dans la discussion et l'analyse de la méthode de recherche et de sélection proposée que certains facteurs influencent l'évolutivité du réseau. La distribution des passerelles est le principal facteur qui joue un rôle important dans l'architecture proposée. Pour cela, nous avons proposé une solution pour éviter les goulots d'étranglement et améliorer le temps de réponse de l'utilisateur. Un paramètre de seuil prédéfini (le nombre de passerelle et le nombre de capteurs dans chaque passerelle) est utilisé pour résoudre le problème de charge utile, minimiser le goulot d'étranglement et également prendre en charge l'évolutivité du réseau.

5.4 Comparaison et résultats

Dans cette section, nous allons montrer l'efficacité et la faisabilité de notre méthode de recherche, de sélection et de classement des capteurs proposée en la comparant avec d'autres méthodes. Comme il est discuté dans la section des travaux connexes, les méthodes CASSARAM, AntClust et E-S algorithme sont les plus pertinentes pour faire une comparaison avec notre méthode proposée.

5.4.1 Comparaison avec CASSARAM et AntClust

Nous avons comparé le temps de traitement de la technique de filtration des capteurs Skyline dynamique proposé avec les approches CASSARAM et AntClust. La figure 5.8 montre le temps de traitement de la technique proposée par rapport à CASSARAM et AntClust. On constate que le temps de traitement du Skyline dynamique proposée est nettement meilleur que les autres, notamment lorsqu'il y a moins de 50000 capteurs. Cependant, le temps de traitement commence à augmenter par rapport aux autres lorsque le nombre de capteurs dépasse 50000 en raison du temps obtenu par l'algorithme BNL Skyline dynamique utilisé pour un grand nombre de capteurs.

En fait, d'autres algorithmes peuvent être implémentés tels que l'algorithme NN (Nearest Neighbor) [63] et BBS (Branch and Bound Skyline) [33], qui peuvent fournir moins de temps que BNL. Une autre solution consiste à partager l'ensemble de données des capteurs et à procéder à un calcul parallèle du Skyline dynamique dans chaque passerelle. Une architecture distribuée peut également minimiser considérablement le temps de traitement en fixant un seuil pour chaque passerelle de sorte qu'une autre passerelle devrait être créée au cas où le nombre de capteurs deviendrait supérieur au seuil prédéfini. La valeur affectée de ce seuil dépendrait du temps de réponse souhaité des utilisateurs. En revanche, la technique Skyline proposée ne permet pas de prédire le nombre de capteurs sélectionnés au départ, car nous ne pouvons produire que des capteurs qui intéressent les utilisateurs. Compte tenu de la précision, nous pouvons observer que la technique de sélection et de classement des capteurs proposée est meilleure que CASSARAM puisque dans notre proposition la phase de classement n'a été appliqué que sur les meilleurs capteurs sélectionnés en utilisant le Skyline dynamique.

L'approche de filtration Skyline dynamique se base sur le nombre de capteurs connectés, le nombre de passerelles utilisé et le nombre de propriétés contextuelles. L'idée est de fournir le nombre idéal de capteurs dans un temps raisonnable et dans des réseaux à grande échelle, où le temps de traitement peut être amélioré à l'aide de solutions matérielles ou logicielles.

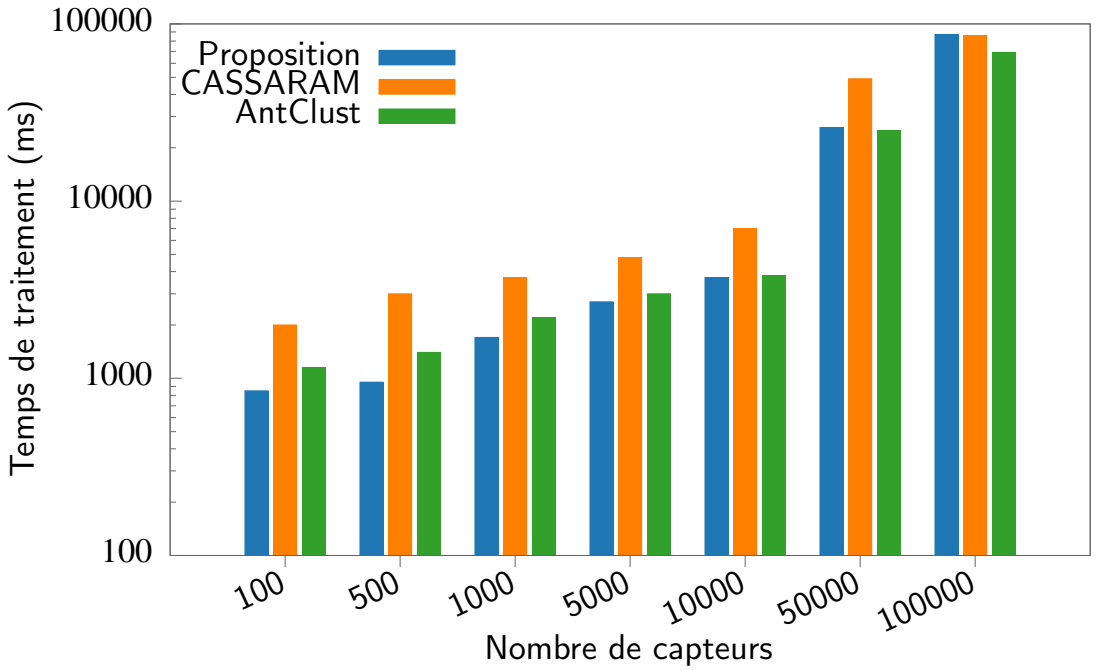

Figure 5.8: Comparaison des performances de notre proposition avec CASSARAM et AntClust.

5.4.2 Comparaison avec E-S algorithme

Avant de faire la comparaison, nous présentons les étapes de l'E-S algorithme comme suit:

- La technique pour l'ordre de priorité par similarité à la solution idéale (Technique for the Order of Prioritization by Similarity to Ideal Solution: TOPSIS) est utilisée pour classer les propriétés disponibles.
- L'algorithme rapide non dominé est exécuté. Il utilise les premières options $N \times SR$ (représentant le nombre de propriétés à utiliser par l'algorithme de tri non dominé), où N est le nombre d'options à sélectionner et SR est la valeur du taux de recherche.
- SR est le paramètre qui joue un rôle clé dans l'E-S algorithme. Il est utilisé pour augmenter les chances d'obtenir une solution optimale proposée par TOPSIS et minimiser le temps et la complexité de stockage. Le but est d'exécuter l'algorithme de tri rapide non dominé. La précision et le temps de traitement sont augmentés par une augmentation de SR.

Pour comparer l'E-S algorithme avec notre proposition, le nombre de capteurs utilisé est de 100000 et le nombre de propriétés contextuelles est de 5. Les figures 5.9 et 5.10 montrent que lorsque SR est inférieur à 70 pour cent, le temps de traitement de l'E-S algorithme est meilleur par rapport à notre proposition, mais la précision de notre proposition est meilleure. Lorsque le SR dépasse 70 pour cent, le temps de traitement de notre proposition commence à être meilleur, tandis que la précision reste toujours meilleure dans notre proposition. L'exécution de l'E-S algorithme est gérable pour une

seule passerelle mais dans le cas de plusieurs passerelles, il est extrêmement difficile à exécuter.

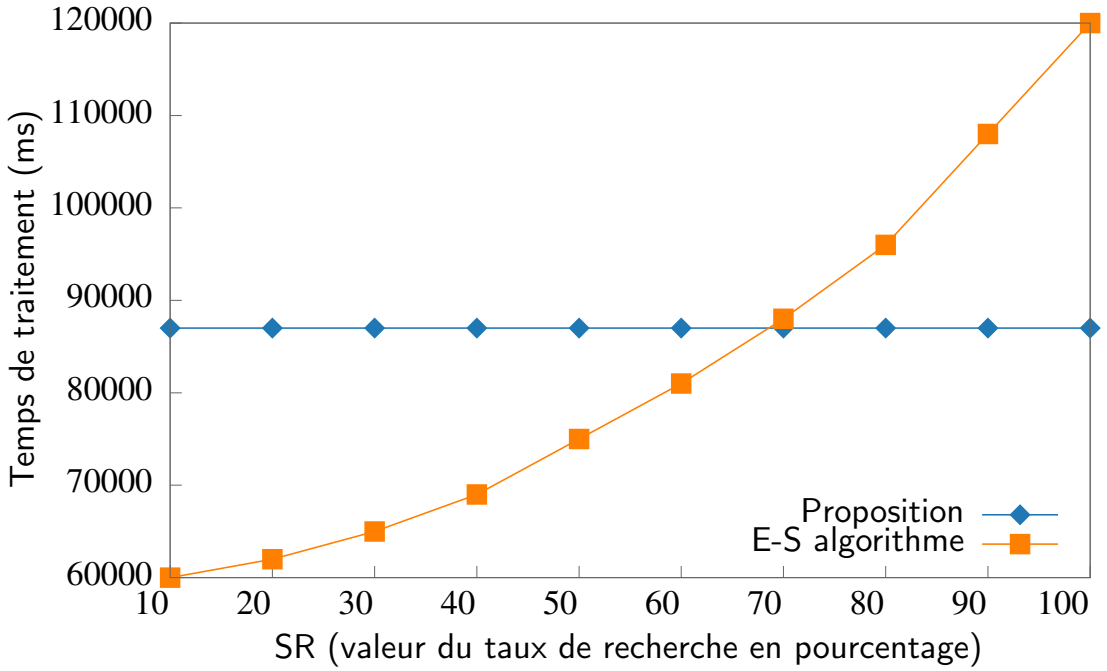

Figure 5.9: Comparaison de temps de traitement de notre proposition avec l'E-S algorithme selon le paramètre SR.

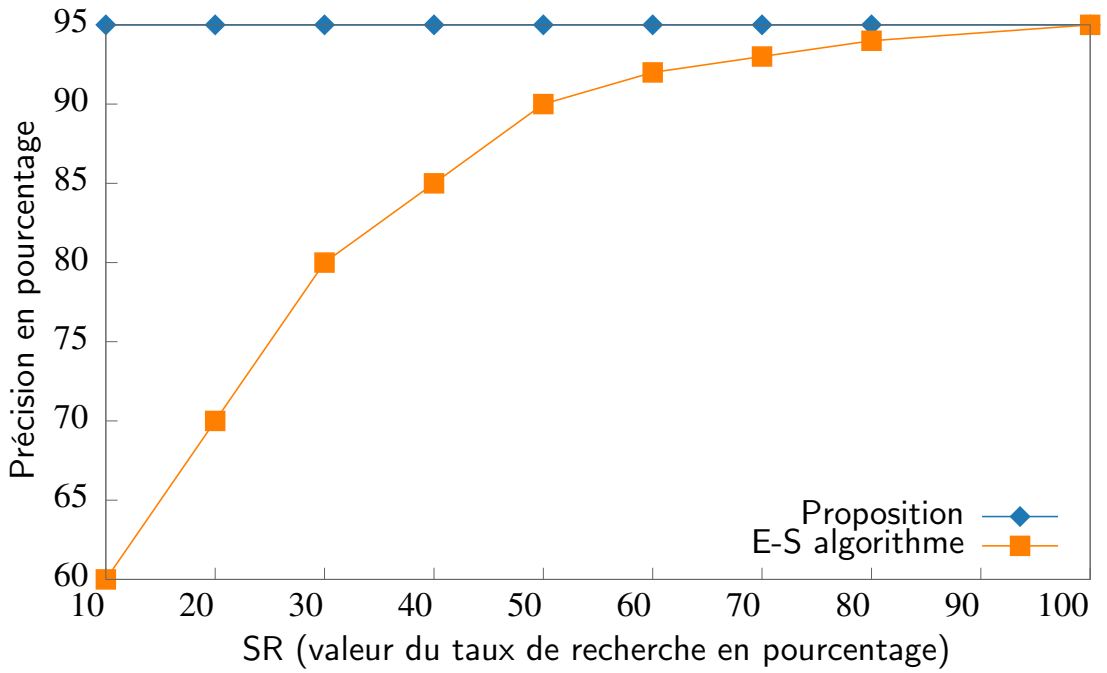

Figure 5.10: Comparaison de précision de notre proposition avec l'E-S algorithme selon le paramètre SR.

Conclusion

Dans ce chapitre, nous avons présenté l'implémentation d'un prototype de l'architecture proposée. Au début, nous avons décrit l'environnement de travail sur là quel le prototype est développé. Ensuite, nous avons donné l'enjeu de données utilisé dans les différentes expérimentations. Nous avons également discuté les différents résultats obtenus et nous avons comparé les performances de notre approche avec les méthodes CASSARAM, AntClust et l'E-S algorithme. Nous avons constaté que la précision de notre technique de sélection et de classement des capteurs est meilleure que CASSARAM et AntClust. De plus, on a remarqué que notre technique va avoir une petite dégradation dans le temps d'exécution si le nombre de capteurs à gérer dépasse le 50000. Mais ce problème peut être résolue par des solutions matérielles (augmentation de nombre de passerelles) ou des solutions logicielles (utilisation d'autre algorithmes Skyline tel que NN et BBS, faire le calcule parallèle du Skyline au niveau des passerelles). De plus, notre proposition donne des bons résultats par rapport à l'E-S algorithme lorsque le SR dépasse 70 pour cent. Enfin, l'expérimentation a démontré les bonnes performances de recherche et la grande évolutivité de notre proposition.

Dans la section prochaine, nous conclurons et discuterons les perspectives de ce travail.

Conclusion et perspectives

Le développement dans le domaine de l'IdO permet l'accessibilité à un grand nombre de capteurs hétérogènes pouvant fournir une variété de données. De plus, l'IdO permet aux propriétaires de capteurs de publier leurs données en tant que service. Les données générées peuvent être partagées par plusieurs applications via des solutions middlewares IdO. Les middlewares IdO permettent de collecter des données à partir d'un grand nombre de capteurs pour différentes applications. L'extraction des données de tous les capteurs existants est une opération coûteuse en terme de ressources comme en terme du temps de calcul, en particulier dans les applications qui nécessitent des données continues. Généralement, il n'est pas possible de collecter les données de tous les capteurs disponibles, ce qui nécessite le développement des méthodes et techniques pour rechercher et sélectionner les meilleurs capteurs en fonction des besoins des utilisateurs.

Le développement d'une méthode de recherche et de sélection de capteurs efficace revient à identifier les meilleurs capteurs qui peuvent fournir les données pertinentes dans un temps raisonnable. Dans ce travail de thèse, nous avons développé une méthode sensible au contexte et adaptable par différents middlewares IdO. La solution proposée permet un haut niveau de précision et réduit le temps de recherche et de sélection.

Cette thèse est composée de cinq chapitres. Les deux premiers chapitres représentent l'état de l'art. Dans le premier chapitre, nous avons présenté un aperçu global sur l'Internet des objets, à travers son historique, sa définition, son architecture, ses différents domaines d'application et ses challenges imposés. Dans le deuxième chapitre, nous avons d'abord introduit la notion de contexte et la notion de sensibilité au contexte. De plus, nous avons présenté l'architecture générale d'un système sensible au contexte. Ensuite, nous avons cité quelques middlewares sensibles aux contextes les plus représentatifs et populaires pour l'IdO, et nous avons terminé par la définition du problème étudié dans ce travail de thèse.

Dans le troisième chapitre, nous avons présenté les différentes méthodes et techniques de recherche et sélection des capteurs existants dans la littérature. Nous avons précisé leurs limites et manques afin de proposer une solution plus efficace.

Dans le quatrième chapitre, nous avons présenté notre méthode de recherche et

de sélection des capteurs selon les besoins des utilisateurs. Notre étude a été axée sur les propriétés contextuelles des capteurs pour rechercher et sélectionner les meilleurs capteurs à propos d'une requête utilisateur. Nous avons profité de la puissance du Skyline dynamique dans le domaine de la prise de décision multicritères pour réduire l'espace de recherche et sélectionner les meilleurs capteurs en fonction des besoins des utilisateurs. De plus, pour assurer le caractère parallèle de l'IdO, notre architecture est composée de plusieurs passerelles réparties au sein du réseau et connectées à un serveur. Chaque passerelle répond aux demandes des utilisateurs localement, et le serveur collecte les résultats de toutes les passerelles pour donner la réponse finale.

Dans le dernier chapitre, nous avons implémenté un prototype de l'architecture proposée. Nous avons également discuté les différents résultats obtenus et nous avons comparé les performances de notre approche avec les autres techniques existants. Les résultats d'expérimentation ont montré les bonnes performances de recherche et la grande évolutivité de notre proposition par rapport aux méthodes similaires.

Pour conclure, cette thèse nous a permis d'avoir un large éventail de concepts, de modèles et de technologies dans les domaines de l'Internet des objets et la sensibilité au contexte. Ainsi, nous avons proposé une nouvelle architecture parallèle basée sur l'opérateur Skyline. Nous avons montré aussi que le travail proposé rejoint une thématique de recherche riche et encourageante.

Suite au travail réalisé dans le cadre de cette thèse, plusieurs perspectives peuvent être planifiées afin d'optimiser et améliorer la démarche proposée:

- Des techniques parallèles, telles que MapReduce, peuvent être utilisées pour améliorer les performances de recherche et de sélection des capteurs.
- Après avoir sélectionné les capteurs qui répondent le mieux aux besoins de l'application, il reste à récupérer les données des capteurs sélectionnés. Pour cette raison, un protocole de routage sensible au contexte doit être développé pour déterminer le meilleur itinéraire en fonction des exigences de l'application et des métriques de routage.

Pour finir, l'Interne des objets est un domaine récent vis-à-vis des systèmes réseaux classiques. Ainsi, plusieurs axes de recherche restent à étudier. Leurs détails sont hors la limite de cette thèse.

Bibliographie

[1] S. Abdelwahab, B. Hamdaoui, M. Guizani et T. Znati – "Cloud of things for sensing-as-a-service: Architecture, algorithms, and use case", *IEEE Internet of Things Journal* **3** (2016), no. 6, p. 1099–1112.

[2] K. Aberer et M. Hauswirth – "Middleware support for the" internet of things"", (2006).

[3] K. Aberer, M. Hauswirth et A. Salehi – "Infrastructure for data processing in large-scale interconnected sensor networks", *Mobile Data Management, 2007 International Conference on*, IEEE, 2007, p. 198–205.

[4] M. Abourezq, A. Idrissi et F. Yakine – "Routing in wireless ad hoc networks using the skyline operator and an outranking method", *Proceedings of the International Conference on Internet of things and Cloud Computing*, ACM, 2016, p. 37.

[5] G. Abowd, A. Dey, P. Brown, N. Davies, M. Smith et P. Steggles – "Towards a better understanding of context and context-awareness", *Handheld and ubiquitous computing*, Springer, 1999, p. 304–307.

[6] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari et M. Ayyash – "Internet of things: A survey on enabling technologies, protocols, and applications", *IEEE Communication Surveys & Tutorials* **17** (2015), no. 4, p. 2347–2376.

[7] J. N. Al-Karaki et A. E. Kamal – "Routing techniques in wireless sensor networks: a survey", *IEEE wireless communications* **11** (2004), no. 6, p. 6–28.

[8] K. Ashton et al. – "That 'internet of things' thing", *RFID journal* **22** (2009), no. 7, p. 97–114.

[9] L. Atzori, A. Iera et G. Morabito – "The internet of things: A survey", *Computer networks* **54** (2010), no. 15, p. 2787–2805.

[10] D. Ayadi – "Optimisation multicritère de la fiabilité: application du modèle de goal programming avec les fonctions de satisfactions dans l'industrie de traitement de gaz", Thèse, Université d'Angers, 2010.

[11] G. Babanejad, H. Ibrahim, N. I. Udzir, F. Sidi et A. A. A. Aljuboori – "Finding skyline points over dynamic incomplete database", *Proceedings of Malaysian National Conference on Databases (MaNCoD)*, 2014.

[12] A. Badii, M. Crouch et C. Lallah – "A context-awareness framework for intelligent networked embedded systems", *2010 Third International Conference on Advances in Human-Oriented and Personalized Mechanisms, Technologies and Services*, IEEE, 2010, p. 105–110.

[13] M. Baldauf, S. Dustdar et F. Rosenberg – "A survey on context-aware systems", *International Journal of Ad Hoc and Ubiquitous Computing* **2** (2007), no. 4, p. 263–277.

[14] D. Bandyopadhyay et J. Sen – "Internet of things: Applications and challenges in technology and standardization", *Wireless personal communications* **58** (2011), no. 1, p. 49–69.

[15] S. Bandyopadhyay, M. Sengupta, S. Maiti et S. Dutta – "Role of middleware for internet of things: A study", *International Journal of Computer Science and Engineering Survey* **2** (2011), no. 3, p. 94–105.

[16] Y. Banouar – "Gestion autonomique de la qos au niveau middleware dans l'iot", Thèse, Université de Toulouse, Université Toulouse III-Paul Sabatier, 2017.

[17] A. Bassi et G. Horn – "Internet of things in 2020: A roadmap for the future", *European Commission: Information Society and Media* **22** (2008), p. 97–114.

[18] S. Borzsony, D. Kossmann et K. Stocker – "The skyline operator", *Data Engineering, 2001. Proceedings. 17th International Conference on*, IEEE, 2001, p. 421–430.

[19] A. Botta, W. De Donato, V. Persico et A. Pescapé – "Integration of cloud computing and internet of things: a survey", *Future generation computer systems* **56** (2016), p. 684–700.

[20] T. Buchholz, A. Küpper et M. Schiffers – "Quality of context information: What it is and why we need it", *Proceedings of the 10th HP–OVUA Workshop*, vol. 2003, 2003.

[21] A. Cannata, M. Gerosa et M. Taisch – "Socrades: A framework for developing intelligent systems in manufacturing", *2008 IEEE International Conference on Industrial Engineering and Engineering Management*, IEEE, 2008, p. 1904–1908.

[22] L. Capra, W. Emmerich et C. Mascolo – "Carisma: Context-aware reflective middleware system for mobile applications", *IEEE Transactions on software engineering* **29** (2003), no. 10, p. 929–945.

[23] T. Chaari – "Adaptation d'applications pervasives dans des environnements multi-contextes", *PhDthesis a l'institut national des sciences appliquees de lyon, laboratoire LIRIS* (2007).

[24] T. Chaari, F. Laforest et A. Flory – "Adaptation des applications au contexte en utilisant les services web", *Proceedings of the 2nd French-speaking conference on Mobility and ubiquity computing*, ACM, 2005, p. 111–118.

[25] S. Chakhar – "Cartographie décisionnelle multicritère: formalisation et implémentation informatique", Thèse, Université Paris Dauphine-Paris IX, 2006.

[26] M. A. Chaqfeh et N. Mohamed – "Challenges in middleware solutions for the internet of things", *2012 international conference on collaboration technologies and systems (CTS)*, IEEE, 2012, p. 21–26.

[27] G. Chen et D. Kotz – "A survey of context-aware mobile computing research", *Dartmouth Computer Science Technical Report TR2000-381* (2000).

[28] H.-C. Chen, H. Gulati, H. Kung et S. Teerapittayanon – "Compressive wireless pulse sensing", *2015 International Conference on Collaboration Technologies and Systems (CTS)*, IEEE, 2015, p. 5–11.

[29] M. Chen, J. Wan et F. Li – "Machine-to-machine communications: Architectures, standards and applications.", *Ksii transactions on internet & information systems* **6** (2012), no. 2.

[30] Y.-K. Chen – "Challenges and opportunities of internet of things", *17th Asia and South Pacific design automation conference*, IEEE, 2012, p. 383–388.

[31] A. Čolaković et M. Hadžialić – "Internet of things (iot): A review of enabling technologies, challenges, and open research issues", *Computer Networks* **144** (2018), p. 17–39.

[32] 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* **17** (2012), p. 25–32.

[33] E. Dellis et B. Seeger – "Efficient computation of reverse skyline queries", *Proceedings of the 33rd international conference on Very large data bases*, VLDB Endowment, 2007, p. 291–302.

[34] A. K. Dey – "Understanding and using context", *Personal and ubiquitous computing* **5** (2001), no. 1, p. 4–7.

[35] A. K. Dey, D. Salber, M. Futakawa et G. D. Abowd – "An architecture to support context-aware applications", Tech. report, Georgia Institute of Technology, 1999.

[36] A. Doni, C. Murthy et M. Kurian – "Survey on multi sensor based air and water quality monitoring using iot", *Indian J. Sci. Res* **17** (2018), no. 2, p. 147–153.

[37] M. Ebrahimi, E. ShafieiBavani, R. K. Wong, S. Fong et J. Fiaidhi – "An adaptive meta-heuristic search for the internet of things", *Future Generation Computer Systems* **76** (2017), p. 486–494.

[38] Y. EL GHAYAM – "La sensibilité au contexte dans un environnement mobile", *Thse* (2011).

[39] B. M. Elahi, K. Romer, B. Ostermaier, M. Fahrmair et W. Kellerer – "Sensor ranking: A primitive for efficient content-based sensor search", *Proceedings of the 2009 International Conference on Information Processing in Sensor Networks*, IEEE Computer Society, 2009, p. 217–228.

[40] M. Fanelli, L. Foschini, A. Corradi et A. Boukerche – "Qoc-based context data caching for disaster area scenarios", *2011 IEEE International Conference on Communications (ICC)*, IEEE, 2011, p. 1–5.

[41] B. Firner, R. S. Moore, R. Howard, R. P. Martin et Y. Zhang – "Poster: Smart buildings, sensor networks, and the internet of things", *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, ACM, 2011, p. 337–338.

[42] A. Gluhak et W. Schott – "A wsn system architecture to capture context information for beyond 3g communication systems", *2007 3rd International Conference on Intelligent Sensors, Sensor Networks and Information*, IEEE, 2007, p. 49–54.

[43] B. Government – "Experimental environmental linked-data published by the bureau of meteorology", URL http://lab.environment.data.gov.au/, 2012.

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

[45] J. Gubbi, R. Buyya, S. Marusic et M. Palaniswami – "Internet of things (iot): A vision, architectural elements, and future directions", *Future generation computer systems* **29** (2013), no. 7, p. 1645–1660.

[46] P. Guillemin, P. Friess et al. – "Internet of things strategic research roadmap", *The Cluster of European Research Projects, Tech. Rep* (2009).

[47] D. Guinard et V. Trifa – *Building the web of things: with examples in node. js and raspberry pi*, Manning Publications Co., 2016.

[48] J. Guth, U. Breitenbücher, M. Falkenthal, F. Leymann et L. Reinfurt – "Comparison of iot platform architectures: A field study based on a reference architecture", *2016 Cloudification of the Internet of Things (CIoT)*, IEEE, 2016, p. 1–6.

[49] H. Hachimi – "Hybridations d'algorithmes métaheuristiques en optimisation globale et leurs applications", Thèse, INSA de Rouen, 2013.

[50] S. Hammoudi, Z. Aliouat et S. Harous – "Challenges and research directions for internet of things", *Telecommunication Systems* **67** (2018), no. 2, p. 367–385.

[51] J. Herbert, J. O'Donoghue et X. Chen – "A context-sensitive rule-based architecture for a smart building environment", *2008 Second International Conference on Future Generation Communication and Networking*, vol. 2, IEEE, 2008, p. 437–440.

[52] R. E. R. Hervé – "Titre: Optimisations multicriteres de la production de l'energie electrique dans les reseaux ht/mt interconnectes aux systemes de productions decentralisees", Thèse, UNIVERSITÉ D'ANTANANARIVO, 2018.

[53] Y.-C. Hsu, C.-H. Lin et W.-T. Chen – "Design of a sensing service architecture for internet of things with semantic sensor selection", *In: International Conference on Ubiquitous Intelligence and Computing, on Autonomic and Trusted Computing, and on Scalable Computing and Communications, UTC-ATC-ScalCom*, IEEE, 2014, p. 290–298.

[54] J. Indulska et P. Sutton – "Location management in pervasive systems", *Proceedings of the Australasian information security workshop conference on ACSW frontiers 2003-Volume 21*, Australian Computer Society, Inc., 2003, p. 143–151.

[55] V. Issarny, M. Caporuscio et N. Georgantas – "A perspective on the future of middleware-based software engineering", *2007 Future of Software Engineering*, IEEE Computer Society, 2007, p. 244–258.

[56] H. Jiang, S. Zhao, S. Qiu et Y. Chen – "Strategy for technology standardization based on the theory of entropy", *Information Technology and Management* **13** (2012), no. 4, p. 311–320.

[57] I. Kertiou, S. Benharzallah, L. Kahloul, M. Beggas, R. Euler, A. Laouid et A. Bounceur – "A dynamic skyline technique for a context-aware selection of the best sensors in an iot architecture", *Ad Hoc Networks* **81** (2018), p. 183–196.

[58] J. A. Khan, H. K. Qureshi et A. Iqbal – "Energy management in wireless sensor networks: A survey", *Computers & Electrical Engineering* **41** (2015), p. 159–176.

[59] R. Khan, S. U. Khan, R. Zaheer et S. Khan – "Future internet: the internet of things architecture, possible applications and key challenges", *2012 10th international conference on frontiers of information technology*, IEEE, 2012, p. 257–260.

[60] A. Khelaifa, S. Benharzallah, L. Kahloul, R. Euler, A. Laouid et A. Bounceur – "A comparative analysis of adaptive consistency approaches in cloud storage", *Journal of Parallel and Distributed Computing* **129** (2019), p. 36–49.

[61] M. Kirsch-Pinheiro, M. Villanova-Oliver, J. Gensel et H. Martin – "Context-aware filtering for collaborative web systems: adapting the awareness information to the user's context", *Proceedings of the 2005 ACM symposium on Applied computing*, ACM, 2005, p. 1668–1673.

[62] K. E. Kjær – "A survey of context-aware middleware", *Proceedings of the 25th conference on IASTED International Multi-Conference: Software Engineering*, ACTA Press, 2007, p. 148–155.

[63] D. Kossmann, F. Ramsak et S. Rost – "Shooting stars in the sky: An online algorithm for skyline queries", *VLDB'02: Proceedings of the 28th International Conference on Very Large Databases*, Elsevier, 2002, p. 275–286.

[64] M. Krause et I. Hochstatter – "Challenges in modelling and using quality of context (qoc)", *International Workshop on Mobile Agents for Telecommunication Applications*, Springer, 2005, p. 324–333.

[65] M. Lachhab – "Proposition d'un outil d'aide à la décision multicritère sous incertitudes à base de colonies de fourmis: une approche intégrée appliquée à la gestion des risques dans les projets d'ingénierie système.", Thèse, 2018.

[66] A. Laouid, A. Dahmani, A. Bounceur, R. Euler, F. Lalem et A. Tari – "A distributed multi-path routing algorithm to balance energy consumption in wireless sensor networks", *Ad Hoc Networks* **64** (2017), p. 53–64.

[67] A. Laouid, A. Dahmani, H. R. Hassen, A. Bounceur, R. Euler, F. Lalem et A. Tari – "A self-managing volatile key scheme for wireless sensor networks", *Journal of Ambient Intelligence and Humanized Computing* (2018), p. 1–16.

[68] D. Le-Phuoc, H. N. M. Quoc, J. X. Parreira et M. Hauswirth – "The linked sensor middleware–connecting the real world and the semantic web", *Proceedings of the Semantic Web Challenge* **152** (2011), p. 22–23.

[69] S. Li, L. Da Xu et S. Zhao – "The internet of things: a survey", *Information Systems Frontiers* **17** (2015), no. 2, p. 243–259.

[70] W. T. Lunardi, E. de Matos, R. Tiburski, L. A. Amaral, S. Marczak et F. Hessel – "Context-based search engine for industrial iot: Discovery, search, selection, and usage of devices", *Emerging Technologies & Factory Automation (ETFA), 2015 IEEE 20th Conference on*, IEEE, 2015, p. 1–8.

[71] A. Makhlouf – "Méthodologie pour l'optimisation dynamique multicritère d'un procédé discontinu alimenté: application à la production bactérienne d'arômes laitiers", Thèse, Institut National Polytechnique de Lorraine, 2006.

[72] W. Mansoor, M. Khedr, D. Benslimane, Z. Maamar, M. Hauswirth, K. Aberer, T. Chaari, F. Laforest et A. Celentano – "Adaptation in context-aware pervasive information systems: the secas project", *International Journal of Pervasive Computing and Communications* (2007).

[73] B. Maram, J. Gnanasekar, G. Manogaran et M. Balaanand – "Intelligent security algorithm for unicode data privacy and security in iot", *Service Oriented Computing and Applications* **13** (2019), no. 1, p. 3–15.

[74] S. Mayer, D. Guinard et V. Trifa – "Searching in a web-based infrastructure for smart things", *Internet of Things (IOT), 2012 3rd International Conference on the*, IEEE, 2012, p. 119–126.

[75] L. Y. Maystre, J. Pictet et J. Simos – *Méthodes multicritères electre: description, conseils pratiques et cas d'application à la gestion environnementale*, vol. 8, PPUR presses polytechniques, 1994.

[76] S. Medileh, A. Laouid, E. Nagoudi, R. Euler, A. Bounceur, M. Hammoudeh, M. AlShaikh, A. Eleyan et O. A. Khashan – "A flexible encryption technique for the internet of things environment", *Ad Hoc Networks* (2020), p. 102240.

[77] R. Mehta, J. Sahni et K. Khanna – "Internet of things: Vision, applications and challenges", *Procedia computer science* **132** (2018), p. 1263–1269.

[78] D. Miorandi, S. Sicari, F. De Pellegrini et I. Chlamtac – "Internet of things: Vision, applications and research challenges", *Ad hoc networks* **10** (2012), no. 7, p. 1497–1516.

[79] M. Miraoui – "Architecture logicielle pour l'informatique diffuse: modélisation du contexte et adaptation dynamique des services", Thèse, École de technologie supérieure, 2009.

[80] M. K. Muruganandam, B. Balamurugan et S. Khara – "Design of wireless sensor networks for iot application: A challenges and survey", *International Journal of Engineering and Computer Science* **7** (2018), no. 03, p. 23790–23795.

[81] L. Nunes, J. Estrella, L. Nakamura, R. de Libardi, C. Ferreira, L. Jorge, C. Perera et S. Reiff-Marganiec – "A distributed sensor data search platform for internet of things environments", *arXiv preprint arXiv:1606.07932* (2016).

[82] L. H. Nunes, J. C. Estrella, C. Perera, S. Reiff-Marganiec et A. C. Delbem – "The elimination-selection based algorithm for efficient resource discovery in internet of things environments", *Consumer Communications & Networking Conference (CCNC), 2018 15th IEEE Annual*, IEEE, 2018, p. 1–7.

[83] L. H. Nunes, J. C. Estrella, C. Perera, S. Reiff-Marganiec et A. C. Botazzo Delbem – "Multi-criteria iot resource discovery: a comparative analysis", *Software: Practice and Experience* (2016).

[84] O. O. Ogundile et A. S. Alfa – "A survey on an energy-efficient and energy-balanced routing protocol for wireless sensor networks", *Sensors* **17** (2017), no. 5, p. 1084.

[85] T. Ojha, S. Misra et N. S. Raghuwanshi – "Wireless sensor networks for agriculture: The state-of-the-art in practice and future challenges", *Computers and Electronics in Agriculture* **118** (2015), p. 66–84.

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

[87] D. Papadias, Y. Tao, G. Fu et B. Seeger – "An optimal and progressive algorithm for skyline queries", *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, ACM, 2003, p. 467–478.

[88] J. Pascoe – "Adding generic contextual capabilities to wearable computers", *2nd international symposium on wearable computers*, Ieee Computer Soc, 1998, p. 92–99.

[89] N. Paspallis, A. Chimaris et G. A. Papadopoulos – "Experiences from developing a distributed context management system for enabling adaptivity", *IFIP International Conference on Distributed Applications and Interoperable Systems*, Springer, 2007, p. 225–238.

[90] K. K. Patel, S. M. Patel et al. – "Internet of things-iot: definition, characteristics, architecture, enabling technologies, application & future challenges", *International journal of engineering science and computing* **6** (2016), no. 5.

[91] H. Patni, C. Henson et A. Sheth – "Linked sensor data", *Collaborative Technologies and Systems (CTS), 2010 International Symposium on*, IEEE, 2010, p. 362–370.

[92] C. Pereira et A. Aguiar – "Towards efficient mobile m2m communications: Survey and open challenges", *Sensors* **14** (2014), no. 10, p. 19582–19608.

[93] C. Perera, A. Zaslavsky, P. Christen et D. Georgakopoulos – "Context aware computing for the internet of things: A survey", *IEEE communications surveys & tutorials* **16** (2013), no. 1, p. 414–454.

[94] — , "Sensing as a service model for smart cities supported by internet of things", *Transactions on Emerging Telecommunications Technologies* **25** (2014), no. 1, p. 81–93.

[95] T. S. Portal – "Internet of things (iot) connected devices installed base worldwide from 2015 to 2025 (in billions).", [Online] Available:https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide, 2017.

[96] M. M. Rafik – "Conception de métaheuristique d'optimisation pour la compression d'image".

[97] A. M. Rahmani, T. N. Gia, B. Negash, A. Anzanpour, I. Azimi, M. Jiang et P. Liljeberg – "Exploiting smart e-health gateways at the edge of healthcare internet-of-things: a fog computing approach", *Future Generation Computer Systems* **78** (2018), p. 641–658.

[98] N. Ramachandran, V. Perumal, S. Gopinath et M. Jothi – "Sensor search using clustering technique in a massive iot environment", Industry Interactive Innovations in Science, Engineering and Technology, Springer, 2018, p. 271–281.

[99] P. P. Ray – "Internet of things for smart agriculture: Technologies, practices and future direction", *Journal of Ambient Intelligence and Smart Environments* **9** (2017), no. 4, p. 395–420.

[100] M. A. Razzaque, M. Milojevic-Jevric, A. Palade et S. Clarke – "Middleware for internet of things: a survey", *IEEE Internet of things journal* **3** (2015), no. 1, p. 70–95.

[101] M. A. Razzaque, S. Dobson et P. Nixon – "Categorization and modelling of quality in context information", (2006).

[102] E. Rohn – "Predicting context aware computing performance", *Ubiquity* **2003** (2003), no. February, p. 1–17.

[103] M. Román, C. Hess, R. Cerqueira, A. Ranganathan, R. H. Campbell et K. Nahrstedt – "A middleware infrastructure for active spaces", *IEEE pervasive computing* **1** (2002), no. 4, p. 74–83.

[104] R. Roman, P. Najera et J. Lopez – "Securing the internet of things", *Computer* (2011), no. 9, p. 51–58.

[105] M. Roopaei, P. Rad et K.-K. R. Choo – "Cloud of things in smart agriculture: Intelligent irrigation monitoring by thermal imaging", *IEEE Cloud computing* **4** (2017), no. 1, p. 10–15.

[106] B. Roy – "The outranking approach and the foundations of electre methods", Readings in multiple criteria decision aid, Springer, 1990, p. 155–183.

[107] A. Saeed et T. Waheed – "An extensive survey of context-aware middleware architectures", *2010 IEEE International Conference on Electro/Information Technology*, IEEE, 2010, p. 1–6.

[108] A. K. Saha, S. Sircar, P. Chatterjee, S. Dutta, A. Mitra, A. Chatterjee, S. P. Chattopadhyay et H. N. Saha – "A raspberry pi controlled cloud based air and sound pollution monitoring system with temperature and humidity sensing", *Computing and Communication Workshop and Conference (CCWC), 2018 IEEE 8th Annual*, IEEE, 2018, p. 607–611.

[109] Y. B. Saied – "Collaborative security for the internet of things", Thèse, 2013.

[110] M. Saoudi, F. Lalem, A. Bounceur, R. Euler, M.-T. Kechadi, A. Laouid, M. Bezoui et M. Sevaux – "D-lpcn: A distributed least polar-angle connected node algorithm for finding the boundary of a wireless sensor network", *Ad Hoc Networks* **56** (2017), p. 56–71.

[111] C. Sarkar, A. U. N. SN, R. V. Prasad, A. Rahim, R. Neisse et G. Baldini – "Diat: A scalable distributed architecture for iot", *IEEE Internet of Things journal* **2** (2014), no. 3, p. 230–239.

[112] Z. Sayah, O. Kazar, B. Lejdel, A. Laouid et A. Ghenabzia – "An intelligent system for energy management in smart cities based on big data and ontology", *Smart and Sustainable Built Environment* (2020).

[113] B. N. Schilit et M. M. Theimer – "Disseminating active mop infonncition to mobile hosts", *IEEE network* (1994).

[114] J. A. Stankovic – "Research directions for the internet of things", *IEEE Internet of Things Journal* **1** (2014), no. 1, p. 3–9.

[115] A. TAIBI – "Fouille de données en épidémiologie spatial : Contribution à la sélection des sites industriels", Thèse, Université d'Oran, 2018.

[116] V. Terziyan, O. Kaykova et D. Zhovtobryukh – "Ubiroad: Semantic middleware for context-aware smart road environments", *2010 Fifth International Conference on Internet and Web Applications and Services*, IEEE, 2010, p. 295–302.

[117] R. T. Tiburski, C. R. Moratelli, S. F. Johann, M. V. Neves, E. de Matos, L. A. Amaral et F. Hessel – "Lightweight security architecture based on embedded virtualization and trust mechanisms for iot edge devices", *IEEE Communications Magazine* **57** (2019), no. 2, p. 67–73.

[118] M. Trnka et T. Cerny – "Identity management of devices in internet of things environment", *2016 6th international conference on it convergence and security (ICITCS)*, IEEE, 2016, p. 1–4.

[119] C. Truong, K. Romer et K. Chen – "Fuzzy-based sensor search in the web of things", *Internet of Things (IOT), 2012 3rd International Conference on the*, IEEE, 2012, p. 127–134.

[120] M. Uddin, A. S. Shanta, M. B. Majumder, M. S. Hasan et G. S. Rose – "Memristor crossbar puf based lightweight hardware security for iot", *2019 IEEE International Conference on Consumer Electronics (ICCE)*, IEEE, 2019, p. 1–4.

[121] I. S. Udoh et G. Kotonya – "Developing iot applications: challenges and frameworks", *IET Cyber-Physical Systems: Theory & Applications* **3** (2018), no. 2, p. 65–72.

[122] S. Vashi, J. Ram, J. Modi, S. Verma et C. Prakash – "Internet of things (iot): A vision, architectural elements, and security issues", *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)*, IEEE, 2017, p. 492–496.

[123] P. K. Verma, R. Verma, A. Prakash, A. Agrawal, K. Naik, R. Tripathi, M. Alsabaan, T. Khalifa, T. Abdelkader et A. Abogharaf – "Machine-to-machine (m2m) communications: A survey", *Journal of Network and Computer Applications* **66** (2016), p. 83–105.

[124] O. Vermesan, P. Friess et al. – *Internet of things-from research and innovation to market deployment*, vol. 29, River publishers Aalborg, 2014.

[125] P. Vincke – "Recent progresses in multicriteria decision-aid", *Rivista di Matematica per le scienze Economiche e Sociali* **17** (1994), no. 2, p. 21–32.

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

[127] M. Weiser – "The computer for the 21 st century", *Scientific american* **265** (1991), no. 3, p. 94–105.

[128] J. Wu, L. Chen, Q. Yu, L. Kuang, Y. Wang et Z. Wu – "Selecting skyline services for qos-aware composition by upgrading mapreduce paradigm", *Cluster computing* **16** (2013), no. 4, p. 693–706.

[129] Q. Wu, G. Ding, Z. Du, Y. Sun, M. Jo et A. V. Vasilakos – "A cloud-based architecture for the internet of spectrum devices over future wireless networks", *IEEE access* **4** (2016), p. 2854–2862.

[130] C. Xu, Y. Li, J. Bilbao et A. El Saddik – "Advances in next-generation networking technologies for smart healthcare", *IEEE Communications Magazine* (2018), p. 14.

[131] C.-T. Yang, S.-T. Chen, W. Den, Y.-T. Wang et E. Kristiani – "Implementation of an intelligent indoor environmental monitoring and management system in cloud", *Future Generation Computer Systems* (2018).

[132] Y. Zhou, S. De, W. Wang et K. Moessner – "Search techniques for the web of things: A taxonomy and survey", *Sensors* **16** (2016), no. 5, p. 600.