UQAC
Université du Québec
à Chicoutimi

UNE NOUVELLE ARCHITECTURE DISTRIBUÉE POUR LA RECONNAISSANCE D'ACTIVITÉS AU SEIN D'UNE MAISON INTELLIGENTE

PAR VALÈRE PLANTEVIN

THÈSE PRÉSENTÉE À L'UNIVERSITÉ DU QUÉBEC À CHICOUTIMI COMME EXIGENCE PARTIELLE EN VUE DE L'OBTENTION DU GRADE DE PHILOSOPHIÆ DOCTOR (PH.D.) EN SCIENCES ET TECHNOLOGIES DE L'INFORMATION

QUÉBEC, CANADA

RÉSUMÉ

L'espérance de vie humaine n'a cessé de croître durant les dernières décennies. Ce phénomène, quoique bénéfique d'un certain point de vue, cause l'apparition de divers types de dégénérescences physiques et mentales au fur et à mesure du vieillissement. Ces dernières peuvent même entraîner la démence sénile dont la principale cause est la maladie d'Alzheimer dont une des conséquences est une perte de l'autonomie. Malgré celle-ci, les individus touchés désirent plus que tout rester chez eux. Cette situation force la mise en place d'une aide à domicile, onéreuse, donnée par la famille ou du personnel médical.

Depuis une trentaine d'années, les domaines de l'informatique et de la microélectronique ont connu un âge d'or sans précédent. On assiste à une augmentation exponentielle de la puissance des appareils pour un prix, une taille et une consommation énergétique qui baisse au même rythme permettant notamment l'émergence de l'Intelligence Ambiante (Amb.I) dont une des applications est l'habitat intelligent où l'environnement tente de reconnaître les activités réalisées par le résident afin d'aider ce dernier si le besoin s'en fait sentir. Malheureusement, la reconnaissance des dites activités, la fiabilité ainsi que le coût des installations restent, encore aujourd'hui, des défis majeurs auxquels il convient de répondre.

Dans cette thèse, nous apportons des réponses au problème de la fiabilité de ces environnements en introduisant une nouvelle façon de concevoir ceux-ci. Ainsi, en utilisant les transducteurs déjà présents dans l'environnement, nous avons réussi à construire une infrastructure distribuée, peu onéreuse, extrêmement fiable et permettant, autant que les anciennes architectures, de reconnaître les activités. Afin d'atteindre cet objectif, nous avons réalisé trois contributions principales dans différents domaines. La première est un nouveau protocole de communication appelé « Light Node Communication Framework » permettant de communiquer au sein d'un environnement intelligent sans aucun point central, travail publié dans un journal spécialisé. La seconde, objet principal d'un article soumit dans un journal, est une architecture, facile à reproduire et assurant trois points importants qui sont la fiabilité, la mise à l'échelle et le faible coût. Pour finir, nous introduisons dans cette thèse, une nouvelle façon de reconnaître les activités de manière distribuée qui est le cœur d'un papier de conférence soumis. Toutes ces contributions mises ensemble répondant au problème de fiabilité dont souffraient les précédents travaux dans le domaine.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ABRÉVIATIONS

ADL	Activity of Daily Living
AES	Advanced Encryption Standard
Amb.I	Intelligence Ambiante
AMQP	Advanced Message Queuing Protocol
ANN	Artificial Neural Network
API	Application Programming Interface
CoAP	Constrained Application Protocol
DDM	Distributed Data Mining
DOMUS	Laboratoire de Domotique et informatique Mobile à l'Université de Sherbrooke
DTLS	Datagram Transport Layer Security
EADL	Enhanced Activity of Daily Living
EEPROM	Electrically Erasable Programmable Read-Only Memory
FTP	File Transfer Protocol
GPIO	General Purpose Input Output
HMAC	Hash Message Authentication Code
HMM	Hidden Markov Model
HTTP	Hyper-Text Transfer Protocol
IADL	Instrumental Activity of Daily Living
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
IP	Internet Protocol
IV	Initialization Vector
JSON	JavaScript Object Notation
KDEC	Kernel Density Estimation based Clustering
KDF	Key Derivation Function
LIARA	Laboratoire d'Intelligence Ambiante pour la Reconnaissance d'Activités
LNCF	Light Node Communication Framework
LSB	Least Significant Bit
LWT	Last Will and Testament
mDNS	multicast Domain Name System
MQTT	Message Queuing Telemetry Transport
MSB	Most Significant Bit
ONU	Organisation des Nations Unies

OSGI	Open Service Gateway Initiative
Pub/Sub	Publisher/Subscriber
QoS	Quality of Service
SoC	System on Chip
SPoF	Single Point of Failure
SSV	Semi-column Separated Values
SVM	Support Vector Machine
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
URI	Unified Resource Identifier
UUID	Universally Unique Identifier
VPN	Virtual Private Network
XMPP	eXtensible Messaging and Presence Protocol
ZMQ	ZeroMQ

DÉDICACE

Je dédie ce travail à ma famille et mes amis

Merci à tous pour votre aide et soutien

REMERCIEMENTS

Cette thèse aurait été totalement impossible sans le support et l'aide de nombreuses personnes. Il m'est malheureusement impossible de toutes les remercier ici par écrit, mais je tiens tout de même à mentionner les plus importantes d'entre elles.

Pour commencer un grand merci à l'ensemble de ma direction, les professeurs Sebastien Gaboury, Bruno Bouchard et Abdenour Bouzouane. Ils m'ont permis, au travers de conseils et de nombreuses rencontres, de terminer cette thèse et de publier mes travaux et sans leur patience et leur aide, ce travail n'aurait jamais pu être mené à bien.

Ensuite, un grand merci à tous mes collègues qui sont devenus, avec le temps et les crises, de bons amis. Merci à vous les gars, car sans nos discussions interminables et nos débats animés aucune des avancées présentées ici n'aurait vu le jour. Mais surtout merci de tous ces bons moments qui ont permis de décompresser un peu et par conséquent de ne pas devenir fou.

Pour finir, un dernier remerciement à ma famille qui malgré la grande distance entre nous a su rester une source d'inspiration, de soutien, d'amour et de fierté, composants essentiels permettant de garder le cap.

CHAPITRE I

INTRODUCTION

1.1 CONTEXTE DE LA RECHERCHE

L'espérance de vie humaine n'a cessé de croître durant les dernières décennies. Selon le rapport mondial sur le vieillissement de la population réalisé en 2015 par l'Organisation des Nations Unies (ONU) (United Nations *et al.*, 2015), 901 millions de personnes sont âgées de plus de 60 ans soit une augmentation de 48 % depuis 2000. Et la tendance n'est pas près de s'inverser avec des prévisions de 1,4 milliard d'individus (56 % de plus qu'en 2015) en 2030 et 2,1 milliards en 2050 (plus du double de 2015). Ce phénomène, quoique bénéfique d'un certain point de vue, cause l'apparition de divers types de dégénérescence physiques et mentales au fur et à mesure du vieillissement. Ces dernières peuvent même entraîner la démence sénile dont la principale cause est la maladie d'Alzheimer (Alzheimer's Association, 2016). Ainsi, selon le rapport mondial sur l'Alzheimer (Prince *et al.*, 2016), cette démence toucherait environ 47 millions de personnes et les projections tendent à montrer une croissance significative pour s'élever à plus de 131 millions en 2050.

La démence est un phénomène qui peut durer plusieurs années voire plusieurs décennies et son évolution peut être très lente. Pendant le développement de la maladie, la personne atteinte va, progressivement, perdre son autonomie pour finir par devenir totalement dépendante de sa famille ou de son personnel aidant. Malgré cette perte d'autonomie, les individus touchés désirent plus que tout rester chez eux. Cette situation

force la mise en place d'une aide à domicile, onéreuse, donnée par la famille ou du personnel médical. À titre d'exemple, le coût mondial de la démence pour l'année 2015 est estimé à 818 milliards de dollars soit une fois et demi le budget militaire américain et dix fois le chiffre d'affaires du géant Microsoft. Et tout comme le vieillissement, ce phénomène ne va pas ralentir avec une prévision de 1000 milliards de dollars en 2018 (Prince *et al.*, 2016). Ajouté à cet impact financier mondial, le personnel aidant subit un stress énorme qui n'est pas prêt de s'atténuer. Il est donc crucial de mettre au point des technologies permettant de contribuer à réduire les impacts de ce phénomène.

Depuis une trentaine d'années, les domaines de l'informatique et de la microélectronique ont connu un âge d'or sans précédent. On assiste à une augmentation exponentielle de la puissance des appareils pour un prix, une taille et une consommation énergétique qui baisse au même rythme permettant notamment l'émergence de l'Intelligence Ambiante (Amb.I)(Cook *et al.*, 2009). Ce concept consiste à enrichir l'environnement avec de la technologie afin de construire un système capable de ressentir des caractéristiques de son milieu dans le but de prendre des décisions en vue d'assister l'utilisateur. Une des applications de l'Intelligence Ambiante est l'habitat intelligent où l'environnement tente de reconnaître les activités réalisées par le résident afin d'aider ce dernier si le besoin s'en fait sentir. Malheureusement, la reconnaissance des dites activités ainsi que le coût des installations restent, aujourd'hui, des défis majeurs auxquels il convient de répondre.

1.2 L'ACTIVITÉ HUMAINE

Dans le cadre de la recherche proposée, la notion d'activité revient constamment. Par conséquent, il convient de commencer par définir ce terme de manière adéquate. Les activités de la vie quotidienne (Activity of Daily Living (ADL)) ont été originellement proposées par Katz *et al.* (1963). Elles regroupent un ensemble d'activités qu'un individu doit réaliser dans le but de maintenir sa santé et son autonomie. Ainsi, la réussite ou l'échec dans l'accomplissement de ses activités permet de mesurer l'autonomie d'un patient atteint de déficience cognitive. Selon Lawton et Brody (1969) il existe deux types d'activités :

Activités basiques (ADL) : il s'agit des activités minimales qu'un humain doit réaliser pour répondre aux besoins vitaux et être ainsi considéré comme semi-autonome. On y trouve, par exemple, le fait de se nourrir, se laver ou même s'habiller. Ces activités sont composées de très peu d'étapes et ne requièrent aucune planification particulière.

Activités instrumentalisées (Instrumental Activity of Daily Living (IADL)) : elles permettent à une personne d'être considérée comme autonome dans une vie en société. Les activités telles que téléphoner, cuisiner ou gérer son argent en font notamment partie. Ces dernières sont constituées de plusieurs étapes et requièrent par conséquent un plus haut niveau de planification que les activités basiques.

Cette définition a été étendue par Rogers *et al.* (1998) afin de prendre en compte un troisième type d'activité les Enhanced Activity of Daily Living (EADL). Cette dernière catégorie contient l'ensemble des activités demandant une adaptation, voire un apprentissage, pour être exécutées. Cela peut être l'adaptation à une nouvelle tablette

intelligente ou à un nouvel ordinateur.

Les activités de la vie quotidiennes ainsi définies, il est désormais possible de revenir sur l'objectif principal des environnements intelligents : la reconnaissance d'activités afin d'assister l'habitant de l'environnement.

1.3 LA RECONNAISSANCE D'ACTIVITÉS

La reconnaissance d'activités fait partie du domaine de l'intelligence artificielle. Elle connaît un intérêt grandissant depuis une quarantaine d'années (Cook et Krishnan, 2014; Ziaeefard et Bergevin, 2015; Roy *et al.*, 2017), date de sa première définition donnée par Schmidt *et al.* (1978). Celle-ci décrit le problème comme étant la découverte du but visé par un acteur (e.g. l'habitant de l'environnement) à partir de la séquence d'actions réalisées par ce dernier. Ainsi, la reconnaissance d'activités est la reconnaissance d'une structure d'activité (i.e. une suite d'actions dans le temps) choisie et réalisée par une entité (e.g. l'habitant de l'environnement) qu'un agent observateur (e.g. l'habitat intelligent) va tenter de reconnaître.

Plus récemment, Roy *et al.* (2013) ajoutent qu'il est possible de caractériser la reconnaissance d'activités par la relation existant entre l'observateur et l'observé. Cela permet d'établir un classement suivant que l'observé aide, empêche ou reste neutre envers le processus de reconnaissance de l'agent observateur. Dans le premier cas, l'observé va tout faire pour faciliter le travail fait par l'agent observateur. Ce dernier peut donc présumer une collaboration pleine et entière dans son processus de reconnaissance et peut, par exemple, poser des questions en cas de doute. Dans le cas où l'observé est

une personne atteinte d'Alzheimer, il est impossible de réaliser ce type de reconnaissance. En effet, adapter son comportement dans le but d'aider un agent demande une forte augmentation de la charge cognitive du patient. Or dans notre cas, la démence empêche cette surcharge cognitive. Le deuxième cas de figure possible tire son origine des mondes militaire et vidéoludique où l'observé va agir comme un ennemi de l'observateur et va adapter son comportement pour délibérément empêcher le bon déroulement de la reconnaissance. Bien que dans notre cas, l'habitat ne reçoive aucune aide de la part de l'habitant, ce dernier ne tente pas volontairement d'empêcher la reconnaissance. Pour finir, il est possible que l'observé ignore tout de l'agent observateur. Dans ce dernier cas, l'observé n'agit ni pour aider ni pour contrecarrer la reconnaissance de l'observateur ce qui correspond au cas d'un malade atteint d'Alzheimer.

La reconnaissance d'activités ainsi définie, il convient désormais de se pencher sur la façon dont celle-ci s'effectue dans un environnement intelligent.

1.4 LA RECONNAISSANCE D'ACTIVITÉS DANS LES ENVIRONNEMENTS INTELLIGENTS

Récemment, une nouvelle extension de la définition de reconnaissance d'activités spécialement appliquée aux environnements intelligents a été formulée (Patterson *et al.*, 2003). Elle indique qu'il s'agit d'inférer les activités réalisées à partir de l'observation des capteurs de plus bas niveau. Cette interprétation plus récente du problème, s'inscrivant dans la lignée de l'informatique ubiquitaire (Weiser, 1991), a transformé la reconnaissance d'activités dans les environnements intelligents en un contexte applicatif bien concret. Dans ce dernier, l'agent observé évolue dans un environnement peuplé

de capteurs dont les valeurs changent en fonction des interactions entre l'acteur et son environnement et ce sont ces changements qui vont permettre aux agents observateurs de reconnaître l'activité réalisée (Roy *et al.*, 2013).

En partant de la définition de Patterson *et al.* (2003), Roy *et al.* (2013) ont énuméré quatre défis qui constituent la reconnaissance d'activités dans les environnements intelligents. Le premier est la récupération uniforme des valeurs de capteurs hétérogènes. Cette tâche consiste à fournir une seule et même interface pour récupérer toutes les données de tous les capteurs et ce peu importe la façon dont ces derniers communiquent (e.g. I2C, RFID, Ethernet). Le second défi est d'interpréter ces valeurs pour en récupérer de l'information intelligible qui peut être la localisation du patient, la quantité d'eau dans une casserole, etc. Pour cela, le système doit être capable d'interpréter les valeurs brutes des capteurs et même en fusionner certaines. Ensuite, le troisième défi consiste à identifier les actions réalisées à partir de ces données. Par exemple, l'environnement doit être en mesure de reconnaître que l'utilisateur fait bouillir de l'eau à partir des positions de la casserole, le poids de celle-ci et la consommation électrique de la cuisinière. Il s'agit là d'un problème de classification qui doit, à partir des modifications de l'environnement entre le temps $t-1$ et le temps t, reconnaître l'action effectuée par l'habitant (Roy *et al.*, 2010). Pour finir, il convient d'interpréter la suite d'actions en une activité de plus haut niveau.

Pour répondre à ces quatre problématiques, Roy *et al.* (2013) proposent un modèle de reconnaissance en quatre couches chacune constituée d'un des défis susmentionnés. Ce modèle, rappelé en Figure 1.1, stipule que chaque couche se doit d'être

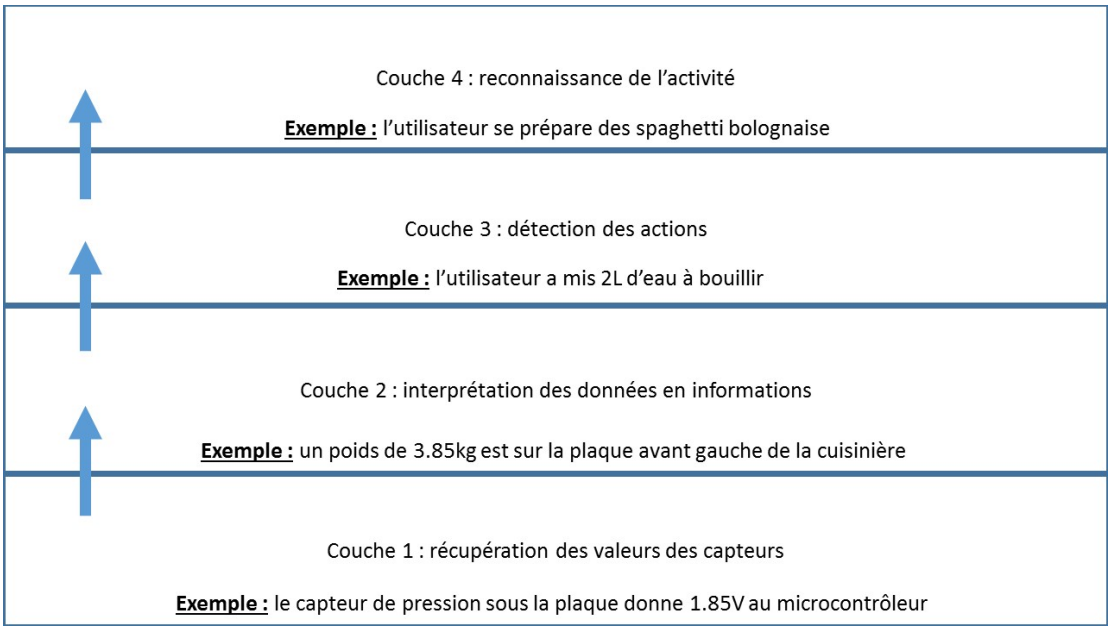

Figure 1.1 : Représentation multicouches du problème de la reconnaissance d'activités

indépendante et ne répondre qu'à une problématique en particulier tout en fournissant aux couches supérieures les informations désirées. Ainsi, chaque problème peut être traité par un agent bien spécifique qui communiquera avec les autres pour leur fournir des données sur lesquelles ils pourront exécuter des traitements de plus haut niveau jusqu'à reconnaître l'activité.

C'est donc à partir des capteurs de bas niveau et en répondant aux différents défis énumérés que la reconnaissance d'activités s'effectue dans un environnement intelligent. Il convient donc de s'intéresser aux architectures existantes qui se sont crées pour implémenter ce paradigme.

1.5 LES ENVIRONNEMENTS INTELLIGENTS

La grande majorité des habitats existants se sont construits dans les dix dernières années et ont tous adopté le même type d'architecture matérielle pour reconnaître des activités (Hu *et al.*, 2016; Bouchard *et al.*, 2014). Celle-ci centralise toutes les valeurs des capteurs sur une entité unique (un serveur) qui va, à elle seule, exécuter toutes les couches qui constituent la reconnaissance d'activités. Bien que tous les habitats aboutissent au même résultat, ils peuvent être découpés en trois familles distinctes suivant la méthode utilisée. Ainsi, le Laboratoire d'Intelligence Ambiante pour la Reconnaissance d'Activités (LIARA) et le Laboratoire de Domotique et informatique Mobile à l'Université de Sherbrooke (DOMUS) (Giroux *et al.*, 2009), représentants connus de la famille des maisons héritées du monde de l'industrie, implémentent une récupération des valeurs des capteurs par du matériel industriel qui envoie ces données de bas-niveaux vers un serveur centralisé exécutant les autres couches de la reconnaissance d'activité. D'une autre manière, les familles basées Open Service Gateway Initiative (OSGI) et mesh respectivement représentées par les habitats Gator Tech et CASAS (Cook *et al.*, 2012; King et Jansen, 2005) ont implémenté des capacités de communication dans leurs capteurs pour que ceux-ci se chargent d'envoyer leurs valeurs vers l'entité centralisée qui va elle aussi terminer le processus de reconnaissance.

Cette architecture, centralisée, facilite les algorithmes qui n'ont pas besoin d'être répartis sur plusieurs machines. Néanmoins, elle comporte plusieurs défauts importants auxquels il convient de répondre. Le premier problème inhérent à ce type d'architecture est qu'elle comporte un goulot d'étranglement qui est aussi un point de rupture unique.

En effet, l'entièreté des données transite par le serveur et si cette entité venait à tomber, c'est l'intégralité de la reconnaissance qui tombe avec. Et dans un contexte comme celui de l'assistance aux malades, c'est un événement qui ne doit jamais arriver. Ensuite, puisque toutes les données se retrouvent immanquablement sur le même serveur, une entité malicieuse qui voudrait les récupérer n'aurait qu'un point à attaquer plutôt que plusieurs. Cette centralisation facilite donc énormément une potentielle attaque et crée, du coup, un important problème de sécurité. Pour finir, un autre défaut est la nécessité de posséder un serveur qui comporte un coût d'acquisition et d'entretien certain.

1.6 PROBLÉMATIQUES INVESTIGUÉES DANS LA THÈSE

Cette thèse apporte des réponses aux problématiques que nous venons tout juste de citer et cela de plusieurs façons différentes. Nous y définissons un tout nouveau type d'architecture moins onéreux tout en nous adaptant mieux aux contraintes présentes dans les environnements intelligents. De plus, nous montrons qu'il est toujours possible, sur cette nouvelle architecture, de pratiquer la reconnaissance d'activités dans le but d'assister l'habitant. Nous répondons ainsi aux questions suivantes :

— Comment décentraliser, à bas coûts, l'architecture d'une maison intelligente en utilisant les capteurs déjà présents dans l'environnement comme support ?

— Comment transformer la reconnaissance d'activités d'intelligence centralisée en une intelligence répartie en utilisant le forage de données distribué ?

Cette thèse est principalement axée autour de trois contributions majeures qui font l'objets de trois articles dans des publications scientifiques.

1. Un nouveau moyen de communiquer au sein des environnements intelligents en utilisant un protocole créé spécifiquement pour cela. L"article portant sur cette contributions est publié dans le journal *Sensors* (Plantevin *et al.*, 2017).
2. Un nouveau type d'architecture des habitats intelligents, moins onéreux, plus fiable et facilitant la mise à l'échelle. L'article est soumis et en cours de révision dans un journal scientifique (« Journal of Ambient Intelligence and Humanized Computing ») (Plantevin *et al.*, 2018b).
3. Une nouvelle méthode pour reconnaître les activités au sein d'un environnement intelligent distribué. L'article est soumis et en cours de révision dans une conférence scientifique (« The 15th IEEE International Conference on Ubiquitous Intelligence and Computing ») (Plantevin *et al.*, 2018a).

1.7 MÉTHODOLOGIE DE LA RECHERCHE

Dans cette thèse, nous allons tenter de répondre à deux questions. Tout d'abord, nous cherchons à démontrer qu'il est possible de décentraliser, en utilisant les capteurs déjà présents dans l'environnement, l'architecture d'une maison intelligente. Ensuite, nous allons montrer que la reconnaissance d'activités est toujours possible sur cette nouvelle architecture. Notre méthodologie pour arriver à ces objectifs s'est découpée en deux phases distinctes. Ces deux phases ont été mises en pratique dans notre laboratoire le LIARA qui nous fournissait les infrastructures matérielles nécessaires pour réaliser de tels tests.

La première phase a consisté à élaborer et implémenter une nouvelle infrastruc-

ture distribuée sur les transducteurs de la maison. Cette architecture permet l'accès facile aux différents capteurs tout en ne possédant aucun point chaud. Nous pensons que l'implémentation dans un contexte réel d'une telle architecture permet de démontrer sa faisabilité. De plus, les méthodes et outils développés lors de sa mise en œuvre pourront aider la communauté scientifique à la reproduction de cette infrastructure dans d'autres laboratoires.

La seconde phase visait à montrer qu'il est toujours possible de réaliser la reconnaissance d'activités sur ladite architecture, et ce même si cette dernière est désormais distribuée. Nous avons proposé ici une nouvelle méthode pour reconnaître des activités sur une infrastructure distribuée et nous avons comparé ses résultats avec d'autres méthodes, centralisées. Cette comparaison des mesures de précision nous a permis de confirmer la viabilité de notre avancée architecturale pour son utilisation, dans le cadre de la reconnaissance d'activités, dans les laboratoires de recherches et même certaines implémentations concrètes dans des maisons existantes.

La méthodologie maintenant présentée, il convient d'expliquer comment ce document s'organise.

1.8 ORGANISATION DU DOCUMENT

Cette thèse se découpe en sept chapitres. En premier lieu, le Chapitre 2 présente un état de l'art sur les architectures existantes et l'évolution des capteurs intelligents. Lui succédant, le Chapitre 3 décrit les méthodes de reconnaissance d'activités et les techniques de forage de données distribuées. Le Chapitre 4 concerne la première contri-

bution faite dans cette thèse en explicitant un nouveau protocole de messagerie créée pour les besoins de notre architecture qui constituera le point principal du Chapitre 5. Suite à la définition de celle-ci, nous montrerons dans le Chapitre 6 que la reconnaissance d'activités est toujours faisable même si l'infrastructure la supportant est distribuée. Pour finir, une conclusion générale viendra clore cette thèse et dégager les pistes d'investigations futures.

CHAPITRE II

L'ARCHITECTURE D'UN ENVIRONNEMENT INTELLIGENT

Avant d'introduire la reconnaissance d'activités dans un habitat intelligent, il convient d'étudier cet environnement et d'en comprendre les tenants et aboutissants. La forte concentration en transducteurs (c.-à-d. capteurs et effecteurs) hétérogènes de ces milieux a forcé les auteurs à redoubler d'ingéniosité. En effet, dans le but d'exécuter une forme d'intelligence, il convient d'avoir un accès facile aux données et ce peu importe leur provenance. Dans ce domaine, de nombreux travaux existent (Cook *et al.*, 2012; Chen *et al.*, 2015; Al-Shaqi *et al.*, 2016). Ainsi, dans un premier temps, ce chapitre se portera sur l'étude des architectures existantes. Puis, dans un second temps, une étude portant sur les capteurs intelligents qui sont les composants de base des travaux modernes dans ce domaine sera présentée. Pour finir, nous passons en revue certains protocoles de communications déjà couramment utilisés dans le domaine et permettant d'échanger des messages entre des entités dans le but de créer un échange de données.

2.1 LES ARCHITECTURES EXISTANTES

Au cours des dix dernières années, de nombreux habitats intelligents se sont créés (Bouchard *et al.*, 2014; Hu *et al.*, 2016; Giroux *et al.*, 2009). Chacun de ces projets a appliqué, à sa manière, les principes de la reconnaissance d'activités multicouches présentés en introduction (Roy *et al.*, 2013). Cette absence d'harmonisation a causé l'apparition de différents types d'architecture ayant chacun ses avantages et ses inconvénients qu'il convient d'étudier. Cette partie se découpe en trois sous-parties. La première est

dédiée aux architectures industrielles comme celle du LIARA (Bouchard *et al.*, 2014) ou du laboratoire de DOMUS (Giroux *et al.*, 2009) qui sont des architectures similaires héritées du monde de l'automatisation. La seconde concerne les architectures basées sur OSGI (pour Open Service Gateway Initiative) telles que Gator Tech (King et Jansen, 2005). Pour finir, la troisième sous partie passe en revue les architectures utilisant le ZigBee comme CASAS (Cook *et al.*, 2012) l'habitat intelligent en boîte.

2.1.1 LES ARCHITECTURES INDUSTRIELLES

Dans les débuts de la maison intelligente, certains environnements de tests se sont créés en utilisant des technologies héritées du monde de l'industrie. Ces architectures sont présentes dans différents types d'environnement allant de la voiture intelligente avec le projet LiSA (Ked, 2014) aux maisons intelligentes avec, entre autres, les exemples du DOMUS (Giroux *et al.*, 2009), LIARA (Bouchard *et al.*, 2014) et House_n (Intille *et al.*, 2006). Ces architectures sont toutes très semblables, et ce si l'on considère leur réalisation, qualités ou défauts. Par conséquent, nous allons ici nous intéresser en détail qu'au DOMUS et LIARA.

Les laboratoires LIARA (Bouchard *et al.*, 2014) et DOMUS (Giroux *et al.*, 2009) possèdent chacun un habitat expérimental dans lequel ils étudient comment les environnements intelligents peuvent assister les personnes souffrant de déficit cognitif (e.g. Alzheimer). Leur architecture (résumée en Figure 2.1) est centralisée et basée sur un automate industriel chargé de récupérer les valeurs des capteurs divisés en îlots. Ces valeurs sont ensuite enregistrées dans une base de données SQL Server afin d'offrir une

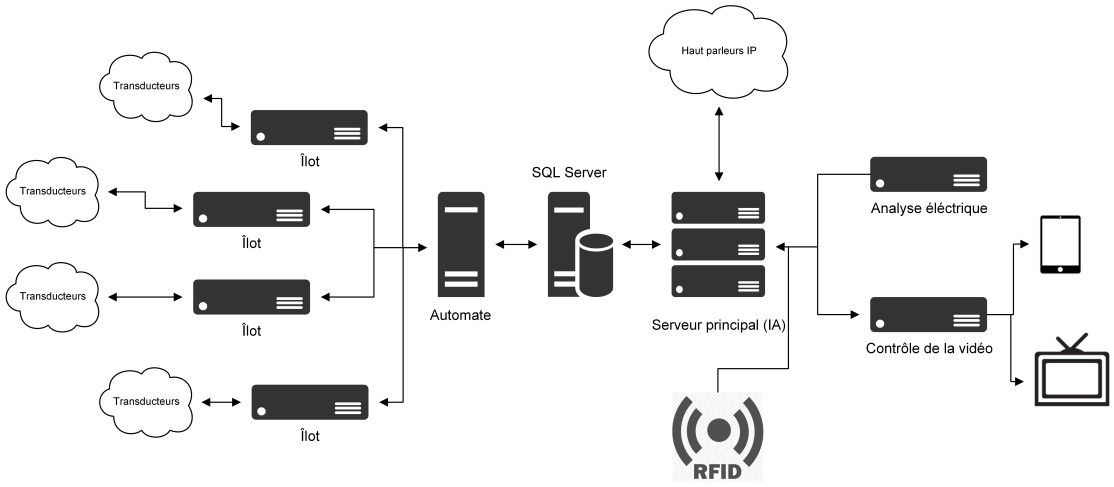

Figure 2.1 : Représentation de l'architecture des habitats intelligents LIARA, DOMUS et LiSA

même interface pour récupérer toutes les valeurs et ainsi effectuer les couches supérieures de la reconnaissance d'activités sur un serveur.

Ce type d'architecture présente plusieurs avantages. Tout d'abord, le matériel utilisé est industriel (un Apax pour l'automate et des boîtes Adam pour les îlots) (Bouchard *et al.*, 2012). Il a donc été testé pour une utilisation en continu dans des usines automatisées qui constituent des environnements bien plus stressants pour les infrastructures qu'un habitat intelligent. Par conséquent, c'est un matériel très fiable, conçu et testé pour un fonctionnement sans interruption. Le deuxième avantage, directement issu de la centralisation, est la facilité d'accès aux données de l'environnement. En effet, toutes les valeurs des capteurs sont situées dans une même base de données et uniquement dans celle-ci et il est très facile d'interroger une base de données à partir d'un autre programme qui pourrait être une intelligence artificielle.

Néanmoins, si la centralisation et l'utilisation de matériel industriel apportent des avantages, cela implique aussi des inconvénients. Le premier concerne le coût global très élevé d'une telle infrastructure puisque les composants qui la composent sont individuellement très onéreux. Un résumé des prix moyens (Advantech, 2016; Dell, 2016) est présenté en Table 2.1. Dans ce tableau, la première ligne est dédiée aux prix de chacun des éléments en dollars américains. La seconde est constituée des quantités de ces éléments qu'il est nécessaire d'avoir pour finaliser l'infrastructure. Enfin, la dernière ligne rappelle le prix total (prix unitaire par la quantité) de chacune des parties de l'habitat. Avec un prix total de 13 500 dollars américains sans capteurs ni infrastructure pour soutenir l'environnement (réseautique, batterie de secours, main-d'œuvre et garantie, câblage, etc.) l'investissement est bien trop élevé si l'on considère qu'une majorité des personnes âgées en perte d'autonomie vivent dans des conditions financières précaires (Alzheimer's Association, 2017). Un autre inconvénient majeur de ce type d'habitat est sa centralisation. En effet, si l'un des îlots venait à tomber c'est un quart de l'environnement qui tombe avec. Dans le cas de l'automate ou du serveur, ce serait l'entièreté de l'habitat qui cesserait de répondre et donc d'assister l'habitant. Pour finir, ce type d'architecture est monolithique et par conséquent complexe à faire évoluer. Le fonctionnement des îlots ainsi que de la base de données relationnelle implique que chaque capteur doit être connu et saisi manuellement à la main dans le système. De plus il est extrêmement complexe d'ajouter de nouveaux types de capteurs qui ne seraient pas connectés par câble aux îlots (e.g. capteurs sans fil ZigBee ou WiFi).

Cette famille d'habitats (e.g. LIARA, DOMUS) présente ainsi une architecture basée sur des composants de l'industrie. Ces habitats comportent deux avantages ma-

Tableau 2.1 : Résumé des prix moyens des éléments des architectures du LIARA et du DOMUS

	Îlot	Automate	Serveur
Prix(USD)	2000 $	1500 $	4000 $
Quantité	4	1	1
Total	8000 $	1500 $	4000 $

jeurs qui sont la fiabilité du matériel choisi et la facilité d'accès aux données. Mais ces avantages ne viennent pas sans inconvénient qui sont le prix, la présence de nombreux points chauds qui, en cas de panne, mettent à mal l'assistance et donc le but principal de l'environnement ainsi que l'absence d'évolutivité dans un domaine en constante amélioration.

2.1.2 LES ARCHITECTURES BASÉES OSGI

Certains auteurs ont tenté de répondre au problème de l'évolutivité en utilisant une architecture basée sur OSGI (Lin *et al.*, 2008; Novák et Binas, 2011; King et Jansen, 2005). Ces architectures partagent toutes les mêmes caractéristiques et ce qu'elles aient été créées il y a vingt ans avec GatorTech (King et Jansen, 2005) ou plus récemment avec les initiatives de Lin *et al.* (2008) ou Novák et Binas (2011). Ainsi, nous détaillons dans cette partie uniquement le cas GatorTech, mais les conclusions s'y appliquant s'appliquent de la même manière aux autres habitats du même type.

Gator Tech est un habitat intelligent fondé dans les années 2000 en Floride (King et Jansen, 2005). L'objectif principal de cet environnement est de prouver la faisabi-

lité d'une maison intelligente à faible coût où l'intégration de nouveaux capteurs serait facile. Pour ce faire, les auteurs présentent une architecture basée sur OSGI (Alliance, 2016). Cette infrastructure, représentée en Figure 2.2, est composée de capteurs intelligents chargés de s'enregistrer de manière autonome dans le système. Pour cela, chaque capteur et effecteur de l'environnement est composé d'une mémoire morte (de type EEPROM permettant la conservation des données même quand le capteur n'est plus alimenté) contenant le code exécutable du pilote permettant la communication. Ainsi, quand un de ces dispositifs est mis en fonction, il s'enregistre auprès d'une définition de service OSGI avec son logiciel pilote et cette définition va servir de couche d'abstraction pour créer des services basiques. Ces derniers offrent aux développeurs la possibilité de consommer des données fortement abstraites ou de combiner plusieurs services de base en un service composite. Un exemple d'une donnée abstraite serait « ensoleillé » quand le capteur de luminosité retourne 10 000 lumens. Concernant les services composites, un exemple serait de combiner tous les services basiques fournissant des flux audio pour effectuer une reconnaissance vocale sur tout l'environnement. Ainsi, les développeurs peuvent concevoir des programmes intelligents d'assistance sans jamais se préoccuper de la façon de communiquer avec les capteurs ni du format de données que ces derniers transmettent.

Ce type d'architecture procure trois avantages majeurs. Le premier d'entre eux est l'ajout automatique de capteurs et effecteurs permettant de facilement mettre l'environnement à l'échelle (e.g. rajouter des capteurs ou en remplacer certains car obsolètes ou défectueux). Le second avantage présenté par cette infrastructure est la haute abstrac-

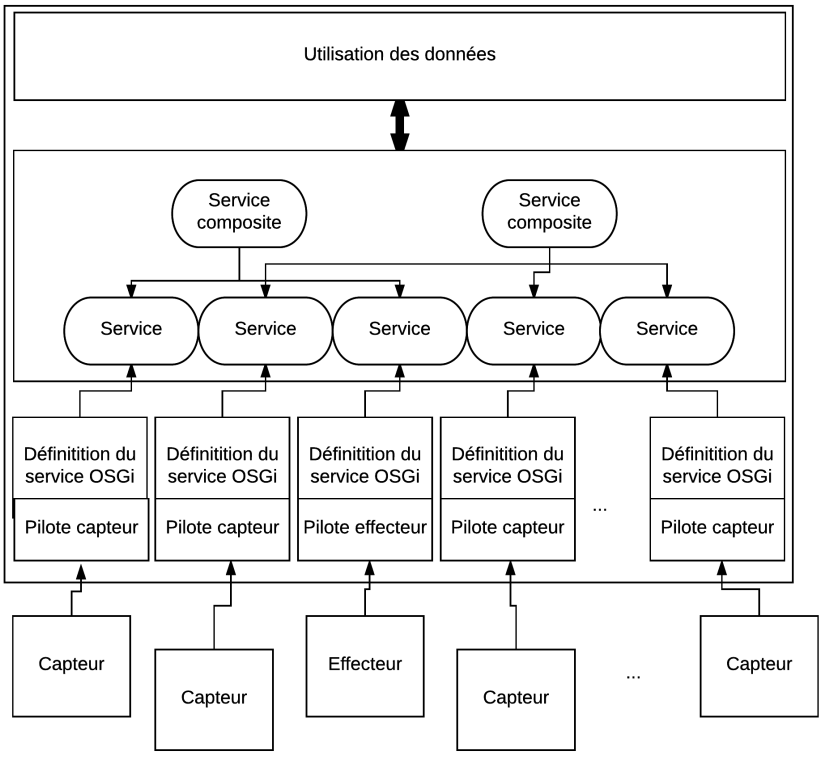

Figure 2.2 : Architecture de l'habitat intelligent Gator Tech

tion des données finales. En effet, cela facilite grandement la tâche des développeurs de haut niveau (e.g. intelligence artificielle, assistance) puisqu'il est plus facile de raisonner sur des données abstraites que sur des données brutes (King et Jansen, 2005). Par exemple, il est trivial de dire qu'il convient d'allumer la climatisation quand la température de l'habitation est « chaude » ou de baisser les volets quand le temps est très ensoleillé et que l'habitant regarde la télévision. De tels liens logiques sont plus complexes à établir quand il s'agit de données à faible abstraction (e.g. la luminosité est de 10 000 lumens) voire de données brutes (e.g. la valeur du capteur de luminosité est de 1023). Pour finir, le prix de cet habitat est réduit. En effet, un simple serveur est nécessaire car les capteurs sont sans fil et s'enregistrent seuls à l'infrastructure (King et Jansen, 2005). Quant au prix des capteurs eux-mêmes, les auteurs précisent que ceux-

ci sont basés sur un Atmega 128, qui est une plateforme à faible coût (Drumea *et al.*, 2005). Par conséquent, mis à part le serveur central, cette architecture présente un coût moyennement élevé. Malgré ces nombreux avantages, Gator Tech, comme toute architecture, n'est pas exempt de défauts. L'environnement ne s'exécutant que sur un seul serveur son fonctionnement est profondément centralisé et si cette pièce maîtresse venait à tomber c'est l'entièreté de la maison qui cesserait de fonctionner.

L'habitat Gator Tech, tout comme la majorité des architectures basées sur OSGI, démontre trois points importants. Tout d'abord, il est possible avec les technologies actuelles de réaliser une maison intelligente à relativement faible coût. Ensuite, il semble être bénéfique pour l'environnement d'embarquer un peu de puissance de calcul dans les capteurs afin que ceux-ci puissent au moins s'enregistrer par eux-mêmes dans l'architecture. Pour finir, l'abstraction des données semblerait faciliter grandement le travail de l'intelligence artificielle. Malgré ces points positifs, cet environnement intelligent conserve tous les défauts inhérents à la centralisation dont le plus important, dans ce cas, est la création d'un goulot d'étranglement constituant aussi un point de rupture unique. Ainsi, toutes les données transitant par le serveur, celui-ci a de grandes chances d'être surchargé si celles-ci venaient à augmenter ralentissant voire stoppant l'assistance.

2.1.3 LES ARCHITECTURES BASÉES MESH

Depuis quelques années, de nombreux travaux se sont portés sur les topologies réseaux mesh et notamment la norme IEEE-802.15.4 (Callaway *et al.*, 2002; Gutier-

rez *et al.*, 2001; Montenegro *et al.*, 2007). La maison intelligente avec ses réseaux de capteurs n'est pas une exception. Ainsi, plusieurs environnements se sont crées autour de ces technologies avec notamment CASAS (Hu *et al.*, 2016), Smart* (Barker *et al.*, 2012) et d'autres travaux (Zhihua, 2016; Zou *et al.*, 2011). Tous ces travaux se ressemblant particulièrement nous nous intéressons ici au cas de CASAS qui est, de loin, le plus connu.

CASAS (Hu *et al.*, 2016), ou la maison intelligente en boîte, est une architecture où l'accent a été mis sur le prix et la facilité d'installation. Représentée en Figure 2.3, l'architecture se décompose en quatre éléments principaux. Tout d'abord, le « mesh » qui est un réseau dans lequel chaque nœud collabore dans le relai de l'information, est constitué de tous les capteurs de l'habitat reliés entre eux à travers un réseau sans fil ZigBee. Ce dernier, communique avec un service de messagerie au travers d'un pont ZigBee qui est chargé de convertir les éléments matériels (e.g. données des capteurs) en message XMPP (XMPP, 2016) de plus haut niveau. Le service de messagerie de type « publish/subscribe » permet ainsi à d'autres services de se connecter afin d'envoyer ou recevoir des données. Par défaut, il existe deux autres services, le premier est une application d'archivage chargée d'enregistrer dans une base de données les événements (qu'elle récupère grâce au pont « Scribe ») qui se produisent dans l'environnement. Le second est l'intelligence en elle-même, reliée à travers le pont applicatif, elle se décompose en trois éléments : la surveillance énergétique, la découverte d'activités chargée de reconnaître si le flux de données en contient ou non et enfin, la reconnaissance d'activités, chargée de reconnaître celles qui ont été découvertes.

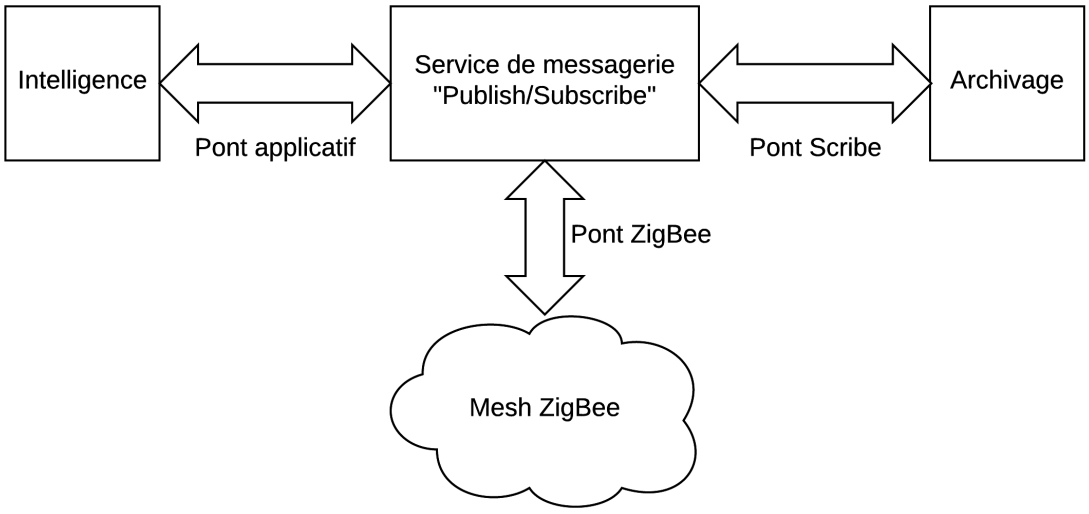

Figure 2.3 : Architecture de l'habitat intelligent CASAS

CASAS, et de manière générale toute cette famille d'architecture, possède deux avantages majeurs : le prix et la facilité d'installation. En effet le prix de l'infrastructure au complet s'élève à 2 765 dollars américains (Cook *et al.*, 2012) ce qui est un point important considérant le fait que d'autres montent à plus de 13 000 $ et que la majorité de la population atteinte par les déficiences visées par ce type d'architecture a déjà de nombreux autres frais à payer et ne peut pas forcément s'offrir une architecture coûteuse (Alzheimer's Association, 2017; Bureau, 2016) (cf. 2.1.1). De plus, les auteurs illustrent la facilité d'installation de leur système en réalisant une étude sur 20 participants âgés de 21 à 62 ans. Le temps moyen d'installation de l'infrastructure par les participants étant d'une heure il est évident que l'installation est aisée. Néanmoins, l'infrastructure présente trois points individuels de défaillance qui font tomber toute l'architecture en cas de panne. Le premier est le pont ZigBee reliant tous les capteurs au service de messagerie, le second est le service de messagerie lui-même et pour finir le pont applicatif qui est le seul point reliant l'intelligence artificielle au reste de

l'infrastructure.

Bien que les familles d'architectures présentées diffèrent toutes les unes des autres, certains points communs émergent. Tout d'abord, la majorité des éléments de leur architecture sont des capteurs et il semblerait, en considérant les familles d'architectures OSGI et ZigBee, qu'embarquer un peu d'intelligence et de communication dans ces capteurs facilite la réduction des coûts, la mise en place et l'évolution de l'habitat. Pour finir, force est de constater que toutes ces architectures partagent le même défaut qui est la centralisation sur une seule entité. Ce phénomène amène la création de points individuels de défaillance qui en tombant (e.g. coupure de courant, problème logiciel ou matériel) force l'habitat au complet à arrêter son assistance. Dans le domaine de l'informatique d'entreprise et plus encore du Web, ce problème a déjà été traité à maintes reprises avec l'utilisation quasi systématique de grappes de serveurs et de redondance (Guo et Yang, 2015; Tan *et al.*, 2018; Anderson *et al.*, 2017).

L'architecture idéale semble donc être composée d'une multitude de capteurs intelligents facilitant grandement l'évolutivité de l'environnement. Ce point particulier rapproche l'environnement d'un autre grand domaine de l'informatique moderne : l'internet des objets (Atzori *et al.*, 2010; Li *et al.*, 2015). Dans ce dernier, déjà utilisé dans le domaine de la maison intelligente (Ghayvat *et al.*, 2015), une multitude « d'objets intelligents » communiquent de manière uniforme entre eux générant une grande quantité de données, tellement grande qu'elle est souvent associée au « Big Data » (Chen *et al.*, 2014). Dans ce dernier cas, il n'est plus pensable de traiter l'information de manière centralisée, et ce même en utilisant de gros clusters de serveurs. Il convient d'utiliser

des méthodes décentralisées relocalisant l'intelligence et le forage de données au plus proche des unités composant le réservoir de données (Chen *et al.*, 2014; Hey *et al.*, 2009).

2.2 CAPTEURS INTELLIGENTS ET ÉVOLUTION DU MATÉRIEL

Les capteurs et effecteurs sont la base de tout habitat intelligent puisqu'ils permettent d'agir et de ressentir l'environnement. Les environnements CASAS et Gator Tech (Cook *et al.*, 2012; King et Jansen, 2005) ont démontré qu'embarquer de l'intelligence et de la communication dans ces transducteurs permettait non seulement de diminuer les coûts, mais aussi de faire évoluer l'habitat plus facilement. Afin d'étudier au mieux ces capteurs intelligents, cette partie est décomposée en deux sous parties. La première apporte une définition générique du capteur intelligent. La seconde présente l'évolution matérielle que le domaine du traitement embarqué a subie depuis la création des premiers habitats intelligents.

2.2.1 LE CAPTEUR INTELLIGENT

Selon la norme IEEE 1451.2 (Mark, 2004), un capteur intelligent est un capteur fournissant des fonctionnalités au-delà de celles nécessaires pour générer une représentation correcte d'une quantité contrôlée ou perçue. Lewis (2005) définit justement ces fonctionnalités supplémentaires ainsi que les objectifs visés par les capteurs intelligents. Pour l'auteur, les fonctionnalités importantes à ajouter sont notamment l'identification des capteurs, un processus facilitant l'installation, l'autodiagnostic, des protocoles standards de contrôle, des interfaces réseau, la coordination et la synchronisation

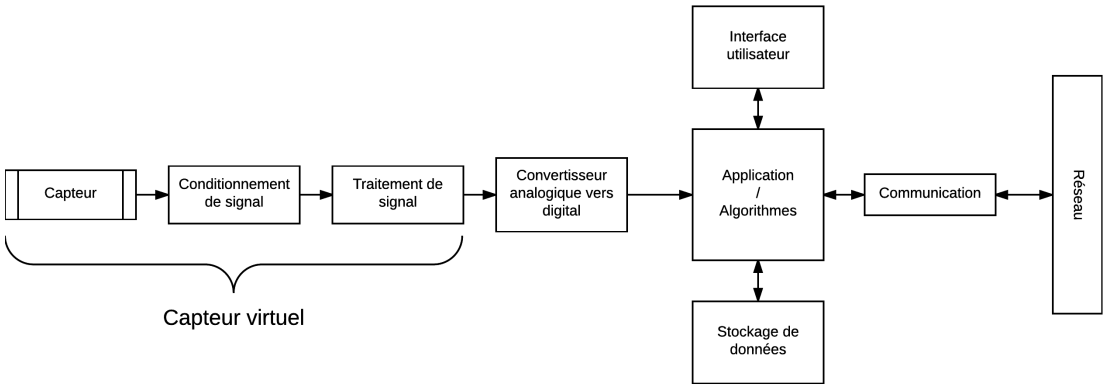

Figure 2.4 : Architecture générique d'un capteur intelligent

avec d'autres capteurs. Concernant les objectifs, il en propose trois. Le premier d'entre eux est de déplacer l'intelligence le plus proche possible du point de mesure. Le second est de rendre l'intégration et le maintien de réseaux distribués de capteurs moins onéreux et le dernier est de rendre facile l'interfaçage de plusieurs capteurs de types différents.

Dans le but de normaliser les implémentations, la norme IEEE 1451 précitée propose une architecture générique rappelée en Figure 2.4. Cette dernière introduit la notion de capteur virtuel qui est l'association d'un capteur, du conditionnement et du traitement de signal requis pour obtenir une estimation correcte de la grandeur mesurée (i.e. un voltage). La donnée ainsi produite est numérisée, souvent par le biais d'un microcontrôleur, afin d'être comprise par l'application principale. Celle-ci va ensuite exécuter un ou plusieurs algorithmes dans le but de réaliser un comportement « intelligent » en fonction de cette entrée (e.g. alarme, reconnaissance, surveillance). Pour finir, si elle en a besoin et si elle le peut, l'application stockera la donnée, mettre à jour

une interface utilisateur voire envoyer des informations sur un réseau.

Le concept et l'architecture d'un capteur intelligent clairement définis, il convient désormais d'étudier les évolutions du matériel qui se sont opérées depuis les débuts des habitats intelligents.

2.2.2 ÉVOLUTION DU MATÉRIEL : VERS DES CAPTEURS PLUS INTELLIGENTS

Les premiers habitats intelligents se sont créés dans la première décennie des années 2000 avec notamment Gator Tech en 2005 (King et Jansen, 2005) et les laboratoires DOMUS en 2002 et LIARA en 2009 (Bouchard *et al.*, 2014). Ces environnements, créés avec les technologies de l'époque, sont antérieurs à beaucoup d'innovations faites à cette période et n'ont donc pu en disposer. Parmi ces innovations, l'apparition et surtout la démocratisation à grande échelle des System on Chip (SoC) a rendu possible la création d'appareils de plus en plus intelligents avec de plus en plus de fonctionnalités. Les SoC sont des systèmes complets embarqués sur un unique substrat et contenant tous les éléments nécessaires à une application (e.g. processeur, mémoire, RF, etc.). Cette intégration a permis d'augmenter les performances pour des prix, tailles et consommations énergétiques toujours plus réduites (Martin *et al.*, 2006). Ils constituent désormais le fer-de-lance de l'électronique moderne et se retrouvent dans toutes les applications embarquées allant des capteurs intelligents (e.g. ESP32 (Espressif, 2016)) aux nano-ordinateurs (e.g. Raspberry Pi (Raspberry, 2016)).

Les performances et fonctionnalités du matériel n'ont cessé d'augmenter, et ce en parfaite adéquation avec la loi de Moore (Moore, 1995; Schaller, 1997). La Table

2.2 présente la comparaison de deux plateformes microcontrôleurs emblématiques des années 2005 et 2016 : l'Arduino USB (Arduino, 2016) et le SparkFun ESP32 Thing (SparkFun, 2016). Ce tableau utilise comme éléments de comparaison l'année de production, la fréquence processeur, la mémoire, la connectivité sans-fil, la taille relative et pour finir le prix des plateformes précitées. Il est facile de constater qu'en 2005, il était devenu simple et relativement peu onéreux (35 dollars) d'acquérir un microcontrôleur 8bits cadencé à 16 MHz avec 1 ko de mémoire. Néanmoins, il est désormais encore moins onéreux (16 dollars) d'obtenir et d'utiliser du matériel bien plus performant et deux fois plus petit tel que l'ESP32. Ce dernier est un microcontrôleur à deux cœurs cadencés à 240MHz avec 512 ko de mémoire et intégrant le Wifi et le Bluetooth. En plus des plateformes basées sur des microcontrôleurs comme l'Arduino USB et l'ESP32 Thing, l'évolution touche aussi celles basées sur micro-processeurs. La Table 2.3 présente une comparaison, basée sur les mêmes critères que précédemment, de différentes plateformes de la Raspberry Pi entre 2012, année de la création de l'organisme, et maintenant. Deux points majeurs ressortent de cette comparaison. Le premier est que pour le même prix, on a multiplié par deux la mémoire (en passant de 512 Mo à 1 Go), ajouté trois cœurs tout en multipliant par 1,7 la fréquence (en passant de 700 MHz à 1,2 GHz) et intégré le Wifi et le Bluetooth. Le second point majeur est que pour une fréquence seulement 1,4 fois supérieure sans ajouter de cœur ni de mémoire, mais en réduisant la taille par deux, le prix passe de 35 dollars à 5 dollars.

Les évolutions du matériel entre le début des années 2000 et maintenant ont été spectaculaires. La démocratisation des SoC a notamment permis une baisse des coûts et des tailles tout en augmentant considérablement la puissance. Ajouté à ce phéno-

Tableau 2.2 : Une comparaison de plateformes basée sur des microcontroleurs.

	Arduino USB	Lillypad Arduino	ESP32 Thing
Année	2005	2007	2016
Fréquence	16MHz	8MHz	2 cœurs à 240MHz
Mémoire	1kB	16kB	512kB
Connectivité sans-fil	Aucune	Aucune	Bluetooth + BLE et Wi-fi
Taille relative	1	0.7	0.5
Prix (USD)	35$	13$	16$

Tableau 2.3 : Une comparaison des Raspberry Pi en fonction du temps

	Raspberry Pi 1	Raspberry Pi Zero W	Raspberry Pi 3
Année	2012	2016	2016
Fréquence	700MHz	1GHz	4 cœurs à 1.2GHz
Mémoire	512MB	512MB	1GB
Connectivité sans-fil	Aucune	Wi-fi et BLE	Wi-fi et BLE
Taille relative	1	0.5	1
Prix (USD)	35$	5$	35$

mène, les systèmes deviennent de plus en plus intégrés avec l'apparition, sur les mêmes puces, de fonctionnalités bien plus avancées comme le Wifi et le Bluetooth. Il semble désormais possible, avec la puissance embarquée de telles entités, de réaliser des applications embarquées de plus en plus performantes. Et l'une de ces applications est la création de capteurs toujours plus intelligents et connectés tels que définis dans la norme IEEE 1451.

2.3 LES PROTOCOLES DE COMMUNICATIONS

Que ce soit dans la littérature scientifique ou l'industrie, il existe de nombreuses façons de communiquer par messages entre différentes entités embarquées. Pour autant que nous le sachions, les protocoles les plus populaires sont MQTT, RabbitMQ et ZeroMQ (Videla et Williams, 2012; Hunkeler et Truong, 2008; Hintjens, 2013).

Message Queuing Telemetry Transport (MQTT), est le protocole majeur de l'Internet des Objets (Internet of Things (IoT)). Cela est dû au fait qu'il est hautement portable et qu'il a été conçu pour consommer le moins de mémoire et de puissance possible. Il fonctionne selon le principe de Publisher/Subscriber (Pub/Sub). Cela signifie que les clients s'abonnent, auprès d'un « broker » centralisé, à un sujet afin de recevoir les messages propres à ce sujet et uniquement celui-ci. De plus, il supporte différents types de qualité de service (Quality of Service (QoS)) permettant de régler finement la fiabilité de la communication (le message est délivré une fois au maximum, au moins une fois ou exactement une fois) ce qui est un de ses atouts majeurs. Pour finir, il implémente un mécanisme de Last Will and Testament (LWT) qui permet d'émettre un message sur un sujet particulier quand une entité est retirée du réseau de manière

anormale (perte de courant ou de réseau). Le principale défaut de MQTT réside dans son « broker » centralisé qui constitue un SPoF pouvant impacter le fiabilité de toute l'architecture.

RabbitMQ lui est un des protocoles les plus utilisé dans les architectures distribuées en entreprise. Il implémente Advanced Message Queuing Protocol (AMQP), un standard ouvert de messagerie se situant sur la couche applicative du modèle OSI. Ce standard, définit une architecture comprenant un « broker » et un ensemble de messages de différents types facilitant, par le biais de métadonnées, l'interopérabilité des systèmes et le développement d'architectures distribuées communiquant par message. Malheureusement, le standard privilégie grandement la sécurité et la facilité de développement par rapport à la vitesse, la taille des messages et les capacités de mise à l'échelle.

Pour terminer cette étude des protocoles existants, il convient de se pencher sur ZeroMQ. Il s'agit d'un Framework permettant à n'importe quel développeur de définir, dans son code, son propre système de messagerie. Il permet ainsi de créer toutes sortes d'infrastructures y compris certaines sans « broker ». Bien que cette solution semble idéale dans notre cas, elle comporte un gros défaut de portabilité puisque ZeroMQ repose sur des sockets POSIX qui ne sont bien supportés que par les systèmes Unix et Windows.

Ainsi, aucun des protocoles de messagerie majeurs de l'industrie ou de la littérature ne semble convenir aux besoins complexes de notre application. En effet, nous désirons tout d'abord obtenir une infrastructure sans entité centralisée comme un « bro-

ker ». Car même si celui-ci peut implémenter une forme de redondance celle-ci peut devenir onéreuse. Ensuite, il faut que cette infrastructure soit supportée par la plus grande majorité des dispositifs embarqués qui ne supportent pas forcément un système POSIX.

2.4 CONCLUSION

Depuis la formulation du problème de reconnaissance d'activités et la définition d'environnement intelligent, de nombreux laboratoires se sont créés. Ceux-ci fournissent à la communauté scientifique des exemples d'architectures matérielles fonctionnelles, divisées en trois grandes familles, simplifiant l'intégration de l'intelligence dans l'environnement. Cependant, la plupart de ces habitats expérimentaux partagent le même défaut qui est l'hyper centralisation de leur architecture. Ce défaut amène la présence de nombreux points de rupture menaçant tout le processus d'assistance aux habitants. Il convient donc, une décennie après ces premiers habitats, d'essayer de définir, à bas coûts, une architecture sans points de rupture garantissant en tout temps et peu importe les circonstances la bonne marche de l'assistance. Les expériences Gator Tech et CASAS ont prouvé que l'un des moyens d'arriver à réduire les coûts est de remplacer les capteurs standards par des capteurs intelligents. Dans ce domaine, la dernière décennie a connu de profonds changements permettant aujourd'hui de réaliser des capteurs réellement intelligents pour des coûts extrêmement faibles. Il convient désormais d'étudier comment, à partir de ces capteurs intelligents, il est possible de reconnaître des activités.

CHAPITRE III

LA RECONNAISSANCE D'ACTIVITÉS ET LE FORAGE DE DONNÉES DISTRIBUÉ

Les architectures et avancées matérielles que nous avons présentées précédemment sont la pierre angulaire des environnements intelligents. Néanmoins, il n'y a pas d'environnements intelligents sans intelligence. Aussi, nous consacrerons une partie de ce chapitre à l'étude de la reconnaissance d'activités qui en est la principale source. Ensuite, nous définirons et passerons en revue des techniques de forage de données distribuées. Ces dernières étant des outils vitaux dans la réalisation de comportements intelligents pour des systèmes non centralisés.

3.1 LA RECONNAISSANCE D'ACTIVITÉS

Afin d'assister un habitant dans un environnement intelligent, il convient en premier lieu de reconnaître l'activité qu'il réalise. Cette reconnaissance des activités est une composante de l'intelligence artificielle et est directement issue de la reconnaissance de plans formulée par Kautz (Kautz, 1991). Cette étude se découpe en deux parties principales. La première concerne l'approche probabiliste où une présentation des réseaux bayésiens et des modèles de Markov cachés est réalisée. Dans un second temps, une étude des techniques basée sur le forage de données est présentée.

3.1.1 L'APPROCHE PROBABILISTE

Les modèles probabilistes et statistiques sont souvent utilisés dans la reconnaissance d'activités. Cela est dû au fait qu'ils permettent, en raison de leur nature, de prendre en compte la composante incertaine, inhérente à ce domaine (Charniak et Goldman, 1993). Dans la littérature, plusieurs modèles sont présentés (Nazerfard *et al.*, 2010; Salah *et al.*, 2015; Subramanya *et al.*, 2006). Dans cette partie, les deux plus courants sont expliqués de manière distincte avec dans un premier temps, les réseaux bayésiens puis, dans un second temps, la logique markovienne et plus précisément les modèles de Markov cachés.

LES RÉSEAUX BAYÉSIENS

Charniak et Goldman (1993) furent les premiers à utiliser les méthodes bayésiennes dans la reconnaissance d'activités. Leur travail est directement hérité de la reconnaissance de plans logiques présentée par Kautz (1991) et répond à ses défauts. Pour résumer cette méthode logique, les auteurs prennent un exemple. Soit trois plans P1, P2 et P3 (qui dans le cas de la reconnaissance d'activités sont des activités) et deux actions A1 et A2. Si A1 appartient à P1 et P2 et A2 à P2 et P3 alors le plan, ou l'activité, visé est P2 puisque c'est le seul incluant toutes les actions réalisées. Selon les auteurs, cette approche présente trois problèmes majeurs. Le premier est que tant qu'il existe au moins deux plans décrivant l'ensemble des actions, la méthode logique est incapable de conclure et ce peu importe l'impossibilité d'un plan ou, au contraire, la très forte probabilité d'un autre. Pour illustrer ce propos, Charniak et Goldman (1993)

prend l'exemple de Jack qui fait son sac et se dirige vers l'aéroport. La méthode logique est incapable de conclure si Jack prend l'avion (très probable) ou s'il va commettre un acte terroriste (presque impossible) puisque les deux actions de base (« faire son sac » et « se diriger vers l'aéroport ») font partie des deux plans. Le second défaut majeur est la différenciation entre les plans de haut niveau et les actions de base. Dans ce cas, l'exemple pris par les auteurs est la marche. En effet, si la marche dessert souvent un plan de plus haut niveau (e.g. marcher vers l'épicerie) il est possible que la marche soit un plan en lui-même (e.g. le sujet a eu envie de marcher). Or, dans la méthode de Kautz (1991), une action ne peut jamais être un plan. Pour finir, la méthode logique prône la minimisation comme principe de déduction. Cela signifie que si un plan explique toutes les actions il est meilleur que deux plans expliquant ces mêmes actions. Or cette assertion est fondamentalement fausse (Charniak et Goldman, 1993). C'est pour pallier à ces défauts que Charniak et Goldman (1993) présentent un nouveau modèle basé sur la logique bayésienne.

Un réseau bayésien est une représentation des connaissances sous la forme d'un graphe acyclique (Charniak et Goldman, 1993; Friedman *et al.*, 1997; Jensen, 1996). Dans ce graphe, chacun des nœuds représente une variable aléatoire et chacune des arêtes la dépendance conditionnelle existant entre un ou plusieurs nœuds (Heckerman, 2010). Il est possible d'appliquer ce principe à la reconnaissance d'activités en considérant que chaque nœud est une action ou une activité et que les arêtes sont les liens unissant plusieurs actions à une ou plusieurs activités. Si l'on connaît la probabilité de chacune des activités ainsi que les probabilités conditionnelles des nœuds, il est possible de recalculer la probabilité de chacune des activités en fonction des observations

faites dans l'environnement.

Afin d'expliquer au mieux ce principe, nous prenons ici un exemple simple. Soit le réseau bayésien présenté en Figure 3.1, trois activités sont réalisables : faire des pâtes, faire du riz ou faire du café (respectivement nommées A_1, A_2 et A_3). De même, quatre événements peuvent se produire dans l'environnement : prendre des pâtes, faire bouillir de l'eau, prendre du riz et prendre du café (respectivement nommées e_1 à e_4). On connaît les habitudes de l'habitant et par conséquent, on sait que $p(A_1) = 0.4$, $p(A_2) = 0.4$ et $p(A_3) = 0.2$. Il est aussi facile de déterminer les probabilités conditionnelles rappelées, dans la Figure 3.1, sous chacun des événements concernés. Si l'on applique la règle de Bayes présentée en Équation 3.1, il devient possible de calculer la probabilité de chaque activité en fonction de la liste des observations faites. Ainsi, si l'on observe l'événement e4 qui est « prendre du café », on peut faire les calculs suivants :

$$p(A_1|e_4) = \frac{0*0.4}{0*0.4+0*0.4+1*0.2} = 0$$

$$p(A_2|e_4) = \frac{0*0.4}{0*0.4+0*0.4+1*0.2} = 0$$

$$p(A_3|e_4) = \frac{1*0.2}{0*0.4+0*0.4+1*0.2} = 1$$

Avec ces probabilités mises à jour, l'activité A3 devient la seule activité possible avec une probabilité de 1 et il est donc facile d'en déduire que l'habitant veut se faire du café.

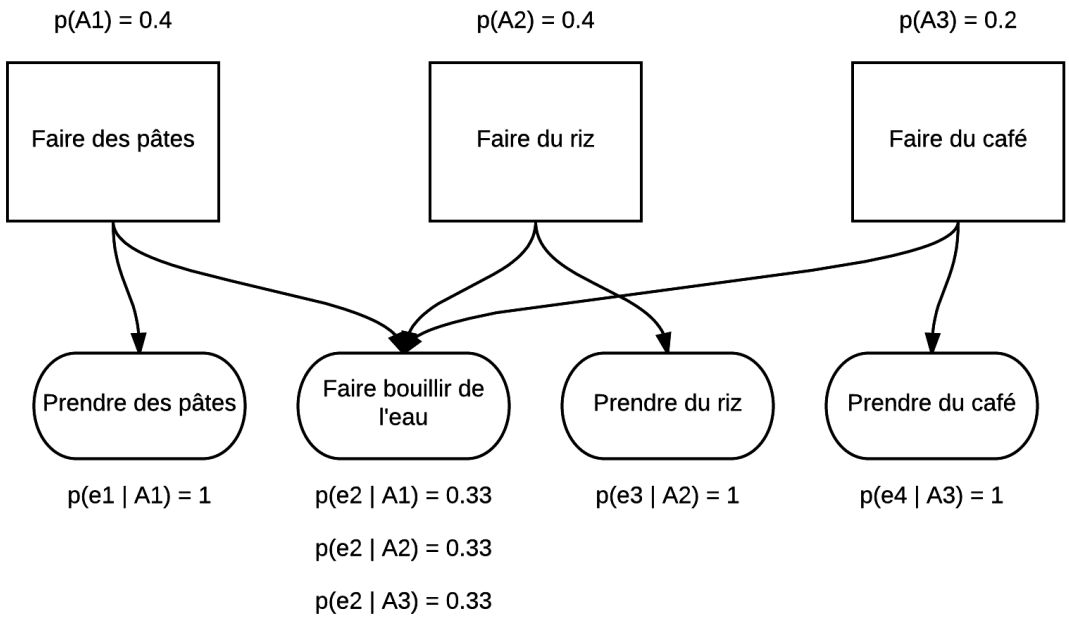

Figure 3.1 : Exemple de réseau bayésien naïf

$$p(h_i|e_1 \cap e_2 \cap ... \cap e_n) = \frac{p(e_1|h_i) * ... * p(e_n|h_i) * p(h_i)}{\sum_{j=1}^{m} p(e_1|h_j) * ... * p(e_n|h_j) * p(h_j)} \tag{3.1}$$

Les réseaux bayésiens présentent plusieurs avantages. Tout d'abord, leur implémentation est basée sur quelque chose d'extrêmement connu : les graphes. Ce point particulier permet de simplifier leur implémentation par l'existence d'un grand nombre de librairies. Ensuite, malgré cette facilité d'implémentation, ils fournissent de très bons résultats avec d'excellentes performances en termes de temps d'exécution et de mémoire (Friedman *et al.*, 1997). Malheureusement, il est nécessaire, pour le créer, de connaître toutes les possibilités d'activités et d'actions ainsi que les probabilités conditionnelles les liant ce qui n'est pas toujours possible. De plus, il est possible, pour des problèmes complexes comme la reconnaissance d'activités, de devoir construire un ré-

seau très complexe nécessitant des capacités de calcul non négligeables.

LES AUTOMATES DE MARKOV À ÉTATS CACHÉS

Les automates de Markov à états cachés (ou Hidden Markov Model (HMM)) ont souvent été utilisés pour reconnaître des activités au sein d'un environnement intelligent (Oliver *et al.*, 2004; Subramanya *et al.*, 2006; Van Kasteren *et al.*, 2011). Un HMM est défini par la combinaison de cinq ensembles différents : $\{S, K, \Pi, A, B\}$. Ainsi, $S_i \in S$ représente un état du système (c.-à-d. une activité), $k \in K$ représente une observation possible (e.g. l'état d'un capteur), $\pi_i \in \Pi$ la probabilité qu'un état S_i soit l'état initial, $a_{(i,j)} \in A$ la probabilité de passer d'un état S_i à un état S_j et $b_i(k) \in B$ la probabilité qu'un état S_i émette l'observation k. De plus, deux règles s'appliquent, et ce en tout temps afin de garantir l'intégrité du modèle statistique : $\sum_{i \in \Pi} \pi_i = 1$ et $\{\forall a_i \in A, \sum_{j \in A} a_{i,j} = 1\}$

Afin d'expliquer au mieux la reconnaissance d'activités utilisant les HMM, la Figure 3.2 présente un exemple simplifié d'utilisation. Dans ce modèle, l'ensemble des états S est constitué de « faire bouillir de l'eau », « faire une sauce » et « faire des pâtes ». L'unique observation, contenue dans l'ensemble K, est « déplacement de la casserole ». L'ensemble Π est constitué des trois probabilités π_s, π_e, π_p dont la somme est bien égale à 1. Les probabilités de changement d'état sont $a_{(e,s)}$ et $a_{(e,p)}$ et les probabilités d'émission sont $b_e(c)$, $b_s(c)$ et $b_p(c)$. Ainsi, en connaissant ce modèle et en appliquant l'algorithme de Viterbi (Forney, 1973), il est possible d'estimer au mieux la séquence d'événements à partir des observations produites par ces événements.

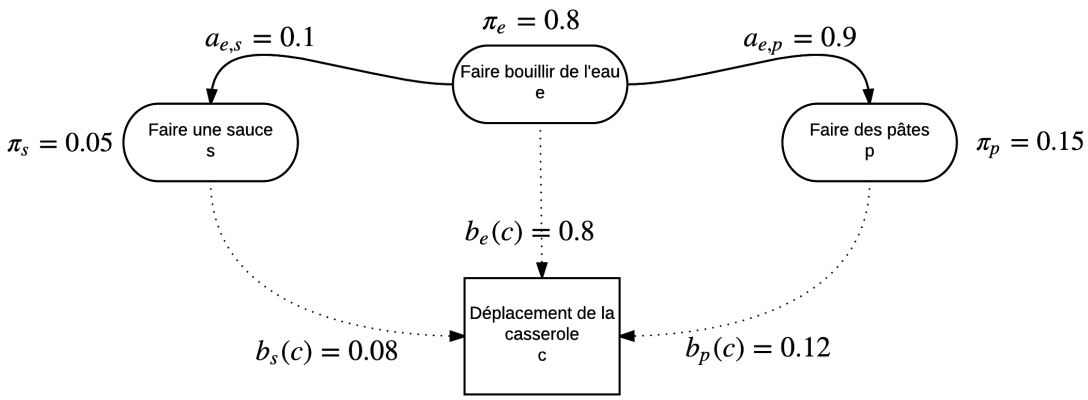

Figure 3.2 : Exemple d'un modèle de Markov caché appliqué à la reconnaissance d'activités

Les HMM sont couramment utilisés dans la reconnaissance d'activités (Oliver *et al.*, 2004; Patterson *et al.*, 2003; Subramanya *et al.*, 2006; Van Kasteren *et al.*, 2011). S'ils offrent d'excellentes performances de reconnaissance et d'exécution, ils sont malheureusement très complexes à mettre en place et à comprendre pour l'utilisateur standard. Ceci restreint leur utilisation à un nombre très limité d'activités. De plus, afin de réaliser un HMM, il faut énumérer et par conséquent connaître toutes les observations possibles. Ce dernier point en particulier oblige de contrôler totalement l'environnement ce qui n'est pas toujours réalisable.

3.1.2 L'APPROCHE FORAGE DE DONNÉES

Dans la littérature, beaucoup d'auteurs (Delachaux *et al.*, 2013; Hey *et al.*, 2009; Huynh et Schiele, 2005; Tapia *et al.*, 2007) ont proposé d'utiliser les techniques de forage de données couramment employées dans la reconnaissance de motifs à la reconnaissance d'activités. Ces méthodes se découpent en deux grandes familles suivant

que leur apprentissage se fait de manière supervisée ou non. Ainsi, les unes ont besoin d'ensembles d'apprentissage déjà entièrement étiquetés tandis que les autres peuvent s'en passer. Cette partie se porte sur l'étude de quatre des plus grandes techniques de forage de données : les arbres de décision, la segmentation (clustering), les machines à vecteurs de support et les réseaux de neurones artificiels (Witten *et al.*, 2016).

LES ARBRES DE DÉCISION

La famille des arbres de décision est constituée d'algorithmes représentant le processus de classification par une suite de choix logiques mis sous la forme d'un arbre. Dans ce dernier, les feuilles représentent les classes ou activités et les nœuds des choix logiques à réaliser sur les attributs de la donnée testée. Dans le domaine de la reconnaissance d'activités, les arbres couramment utilisés sont ID3 et sa version améliorée prenant en charge les attributs continus C4.5 (Bao et Intille, 2004; Ravi *et al.*, 2005; Tapia *et al.*, 2007). Ces deux algorithmes sont très ressemblants et il est possible de les résumer en quatre étapes très simples. Tout d'abord, chaque attribut se voit associer son entropie calculée sur l'ensemble d'apprentissage. Ensuite, l'attribut avec la plus faible entropie est sélectionné et une séparation des données est faite en fonction des valeurs de celui-ci. Ainsi, il est possible de construire l'arbre avec un nœud et des branches respectivement associés à l'attribut sélectionné et ses valeurs. Pour finir, on recommence l'algorithme de manière récursive pour chaque branche nouvellement créée.

Afin de faciliter la compréhension d'un tel ensemble de méthodes, nous présentons en Figure 3.3 un exemple d'arbre de décision appliqué à la reconnaissance d'acti-

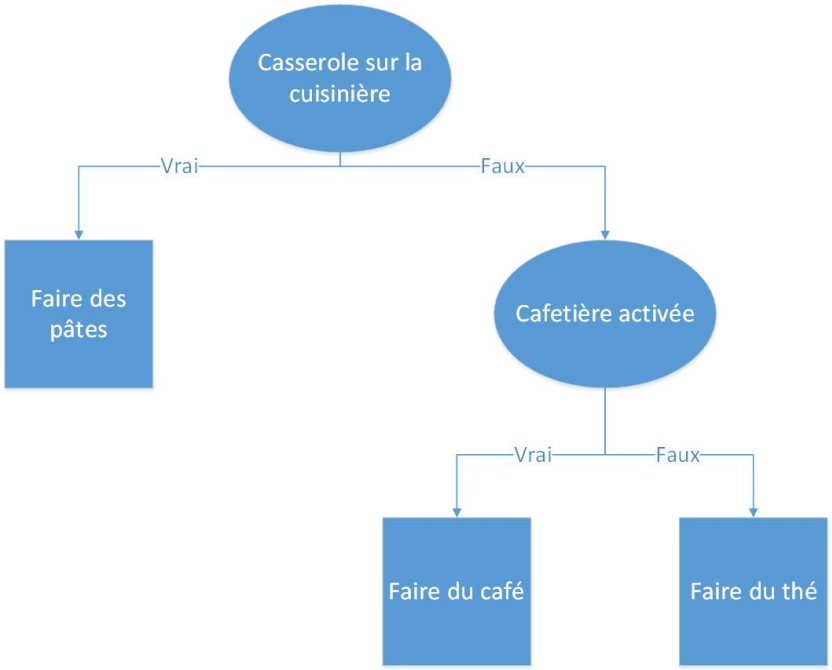

Figure 3.3 : Un exemple basique de reconnaissance d'activités basée sur un arbre de décision

vité. Dans ce dernier, les valeurs des capteurs définissent trois activités possibles : faire du café, préparer des pâtes et faire du thé. Ainsi, on peut constater que seules deux données issues de capteurs définissent l'entièreté de la reconnaissance d'activités : la localisation de la casserole et l'activation de la cafetière. Grâce à la méthode des arbres, la classification des activités devient très simple : à chaque fois qu'une nouvelle donnée arrive, l'algorithme vérifie la valeur des attributs « localisation de la casserole » et « activation de la cafetière » et prend une décision grâce à une suite de choix logiques définis par l'arbre appris.

Les avantages principaux de ces algorithmes sont la facilité de compréhension et la rapidité d'exécution pour des taux de reconnaissance très corrects. En contrepartie, les arbres de décision souffrent de trois défauts majeurs. Premièrement, l'apprentis-

sage supervisé d'un tel algorithme force l'utilisateur à avoir en sa possession un jeu de données important, dont chacune des activités est connue et étiquetée. Ensuite, il faut refaire l'entraînement chaque fois qu'une activité s'ajoute ou est modifiée. Pour finir, les arbres de décision sont, dans l'ensemble, assez peu efficaces quand il s'agit de différencier un grand nombre d'activités dans un même ensemble de données.

LA SEGMENTATION

Les méthodes de segmentation (clustering) constituent une part largement majoritaire des algorithmes à apprentissage non supervisé (Witten *et al.*, 2016). Le dénominateur commun entre toutes ces méthodes et qu'elles cherchent à regrouper les données en « cluster » qui sont des sous-ensembles de l'ensembles d'apprentissage. Ces derniers sont constitués en minimisant la distance intracluster (c.-à-d. la distance entre les éléments du cluster) tout en maximisant la distance intercluster (c.-à-d. la distance entre les clusters). L'algorithme va ensuite associer à chacun de ces ensembles une donnée considérée comme étant la plus représentative. Suivant les algorithmes, cette sélection peut se faire de plusieurs façons différentes. Certains vont créer de toute pièce cet exemple en calculant des barycentres de cluster, d'autres vont convenir d'une donnée parmi celles classifiées. Dans ce domaine, l'algorithme le plus connu et le plus utilisé en reconnaissance d'activités est K-means (Huynh et Schiele, 2005; Kovashka et Grauman, 2010; Messing *et al.*, 2009; Xia et Aggarwal, 2013). Celui-ci part du principe que l'on connaît déjà le nombre K de clusters que l'on désire et va commencer par choisir K barycentres (e.g. aléatoirement, parmi les données, etc.). Une fois ces points choisis,

chacune des données de l'ensemble d'entraînement va calculer sa distance par rapport à chacun des K barycentres et s'associer au plus proche. Une fois toutes les données évaluées, les barycentres sont mis à jour en fonction des clusters nouvellement formés et l'algorithme recommence jusqu'à se stabiliser.

Une fois le modèle appris, la reconnaissance d'activités peut être réalisée. Pour faciliter la compréhension de cette phase, un exemple de clustering est présenté en Figure 3.4. On peut y voir que l'ensemble d'apprentissage (représenté par des cercles) a été divisé en trois clusters que l'on nomme, pour faciliter l'explication, Activité 1 à 3. Chacun de ces ensembles de données est défini par son barycentre représenté ici par une étoile. Quand une nouvelle donnée arrive, par exemple le carré, il suffit de calculer la distance entre cette dernière et tous les barycentres. La plus petite d'entre elles définira l'appartenance de cette nouvelle donnée à un cluster et donc à une activité (ici, l'activité 3).

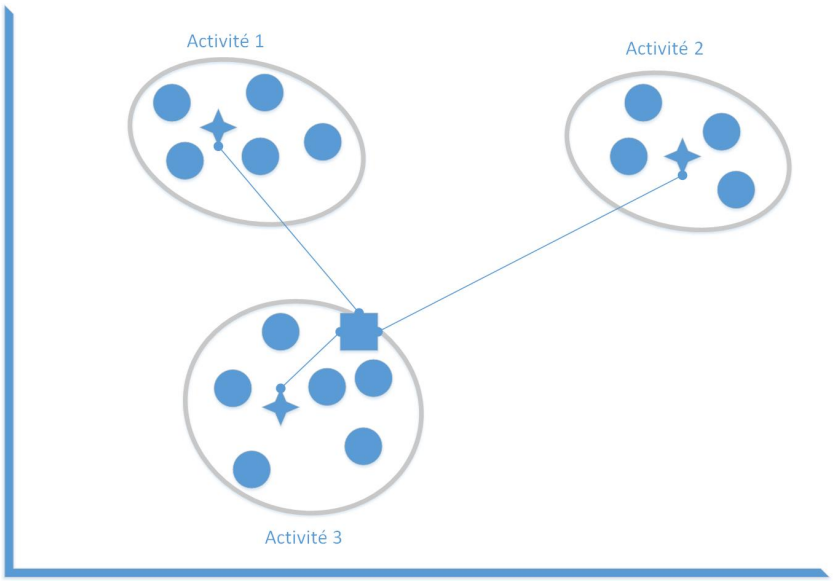

Figure 3.4 : Un exemple de reconnaissance d'activités utilisant le clustering

L'avantage principal d'une telle méthode et son côté non supervisé qui permet d'avoir des données non étiquetées en entraînement. Néanmoins, la plupart de ces algorithmes ont des phases d'entraînement lourdes en temps et en mémoire et pour beaucoup il faut connaître le nombre de cluster que l'on désire à l'avance.

LES MACHINES À VECTEURS DE SUPPORT

Les machines à vecteurs de support (Support Vector Machine (SVM)) sont des algorithmes à apprentissage supervisé (Witten *et al.*, 2016). L'objectif principal d'un SVM est de créer un hyperplan partageant au mieux les données d'entraînement. Pour ce faire, l'algorithme va chercher à maximiser la marge, c'est-à-dire la distance entre les échantillons d'apprentissage et l'hyperplan lui-même. Ce problème de maximisation peut ainsi s'exprimer sous la forme d'une optimisation quadratique pour laquelle de nombreuses méthodes de résolution existent depuis longtemps (Burges, 1998; Chih-Wei Hsu et Lin, 2008; Smola et Schölkopf, 1998). Dans le cas où il n'existerait aucun hyperplan linéaire de séparation, SVM utilise une fonction de noyau permettant de convertir les données d'un ensemble d'apprentissages vers un ensemble de dimension supérieure où il existe un tel hyperplan. Bien que cette technique ne s'applique historiquement qu'aux problèmes de classification binaires, de nombreuses adaptations, plus récentes, permettent son utilisation dans des cas où il y a plusieurs classes (Witten *et al.*, 2016).

Une fois l'hyperplan défini, la classification d'une activité peut être exécutée. Un exemple de celle-ci est présenté en Figure 3.5 où deux activités coexistent, une

représentée par des étoiles et l'autre par des cercles. L'hyperplan dans notre exemple est un cercle séparant parfaitement l'ensemble d'apprentissage en deux. Un tel plan, non linéaire, implique que l'algorithme a eu recours, durant l'apprentissage, à une fonction de noyau. Dans notre cas, le processus de reconnaissance cherche à découvrir la classe de la nouvelle donnée représentée ici par un carré. Pour ce faire, il va calculer de quel côté du plan cette dernière se situe. Si elle est à l'intérieur du cercle il s'agit d'une activité étoile sinon d'une activité cercle. Ici, il est clair que l'activité inconnue est à classer dans la partie des activités étoiles et c'est donc ainsi qu'elle sera labélisée par l'algorithme.

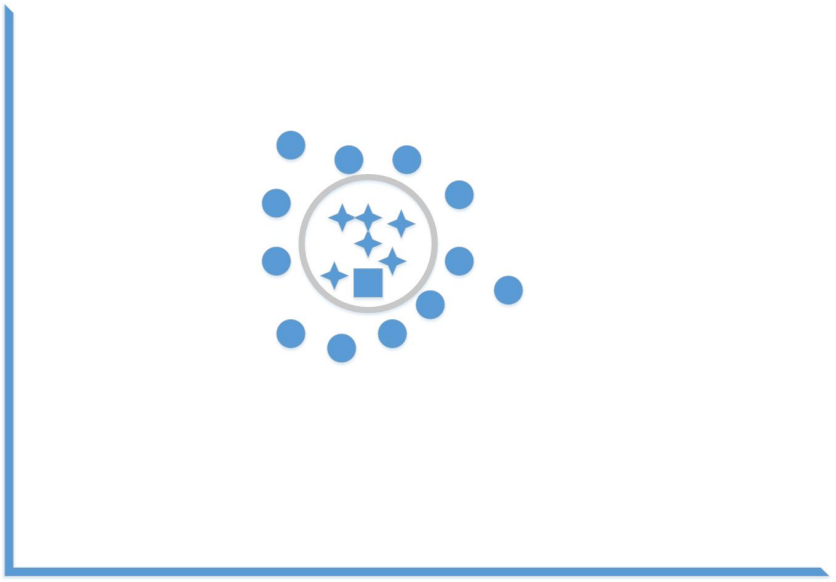

Figure 3.5 : Un exemple de reconnaissance d'activités basée sur SVM

Les SVM sont couramment utilisés dans la reconnaissance d'activités (Anguita *et al.*, 2012; He et Jin, 2008, 2009) pour deux raisons principales. Premièrement, ces méthodes ont des taux de reconnaissance souvent très élevés. Deuxièmement, une fois le modèle calculé, la reconnaissance se limite à l'application d'une formule mathéma-

tique ce qui, en termes de mémoire et de temps de calcul, est relativement léger. Ces avantages sont néanmoins à nuancer puisque l'apprentissage peut être lourd en calcul et très difficile à implémenter.

LES RÉSEAUX DE NEURONES ARTIFICIELS

Héritées de la biologie, ces méthodes visent à reproduire le comportement des neurones du monde animal. Un réseau de neurones artificiels (Artificial Neural Network (ANN)) est composé de plusieurs neurones connectés entre eux et s'échangeant des signaux (Witten *et al.*, 2016). Ces entités sont composées d'un nombre *n* d'entrées synaptiques chacune pondérées par un poids qui, par le biais d'un additionneur, vont être agglomérées en une seule donnée. Cette dernière est passée à une fonction d'activation qui suivant un certain biais (c.-à-d. seuil) va ou non s'activer produisant ainsi le signal de sortie du neurone. L'apprentissage, supervisé, de ces réseaux se réalise en modifiant la valeur des poids associés à chacune des synapses, et ce dans le but d'atténuer l'erreur entre les sorties du réseau et l'étiquette de la donnée testée.

Une fois l'intégralité des poids mis à jour, la reconnaissance peut s'effectuer. La Figure 3.6 présente un perceptron multicouche simplifié permettant de reconnaître deux activités : « faire du café » et « faire du thé ». Les neurones d'entrée dans ce type de réseaux représentent les valeurs des capteurs de l'environnement et sont représentés par des carrés. La reconnaissance d'activités se base sur les valeurs fournies par les neurones de sortie, ici il s'agit des hexagones. Dans le cas de l'exemple étudié, les valeurs sont déjà « Vrai » ou « Faux » mais il aurait aussi pu s'agir de probabilités et dans ce cas

un simple maximum aurait aussi réussi à sélectionner la bonne activité. Pour classifier une nouvelle donnée, il suffit de mettre les valeurs des capteurs de cette dernière dans les bons neurones d'entrée et les neurones de sortie détermineront automatiquement la bonne activité à laquelle appartient la donnée considérée.

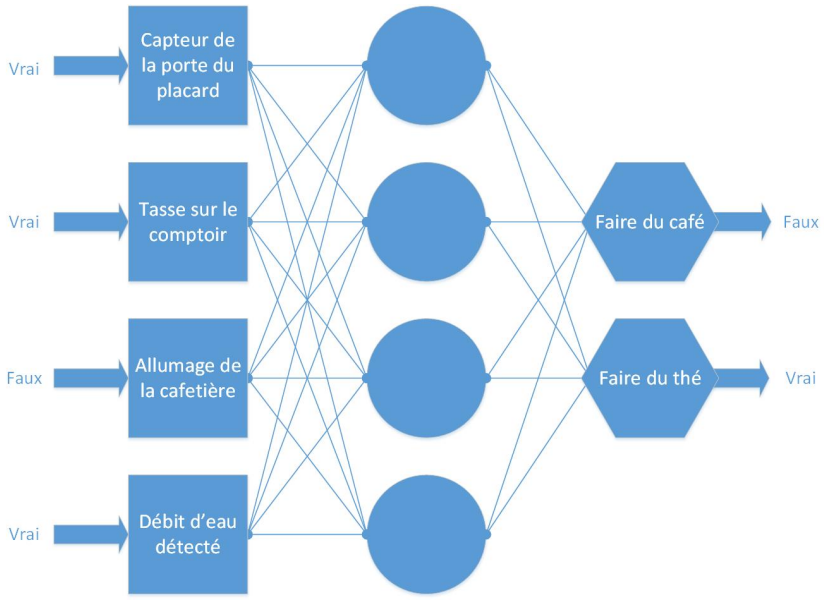

Figure 3.6 : Un exemple de reconnaissance d'activités bssée sur un réseau de neurones artificiels

Dans le domaine de la reconnaissance d'activités, ces techniques ont souvent été utilisées (Delachaux *et al.*, 2013; Pärkkä *et al.*, 2006) pour leur relativement bon taux de reconnaissance et leur robustesse face aux données bruitées. Cependant, leur temps d'apprentissage pouvant durer plusieurs jours, la nécessité de refaire ledit apprentissage à l'ajout d'une activité, la complexité de leur structure quand l'ensemble possède de nombreux attributs et la facilité de sur-apprendre un problème, constituent des défauts majeurs et freinent leur utilisation.

3.1.3 BILAN DE L'APPROCHE FORAGE DE DONNÉES

De nombreux travaux présentent différentes facettes de la reconnaissance d'activités. Chacune des approches présentées apporte ses avantages et ses inconvénients. Ainsi, l'approche probabiliste offre des taux de reconnaissance très intéressants pour des coûts en termes de mémoire et de calcul très faibles. Néanmoins, ces méthodes se basent sur une connaissance parfaite du problème et une maîtrise quasi totale de l'environnement. De son côté, le forage de données amène une grande quantité d'outils pour apprendre et reconnaître les différentes activités. Ces outils ont chacun leurs forces et faiblesses suivant la connaissance et la quantité de données que l'on possède sur le problème. Néanmoins, le point commun de ces méthodes est que, par nature, il est plus facile d'avoir toutes les données d'apprentissage et de test au même endroit. Cette façon de penser très centralisée peut être un problème si les données ou les puissances de calcul sont réparties. Ainsi, il convient d'étudier les variantes réparties de ces algorithmes pour répondre au problème de centralisation déjà étudié dans le second chapitre de ce document.

3.2 LE FORAGE DE DONNÉES DISTRIBUÉ

Le forage de données classique est, par définition, l'extraction d'une connaissance à partir d'une grande quantité de données (Chakrabarti *et al.*, 2012). Une des branches de ce domaine est le forage de données distribué (Distributed Data Mining (DDM)). L'objectif principal de ce dernier est de fournir des moyens d'extraire la même quantité de connaissance tout en prêtant attention au caractère distribué des données et des ressources de calcul exploitées sur chacun des sites (c.-à-d. les entités possédant les

données ou les capacités de calcul) (Dasilva *et al.*, 2005).

En DDM, deux grands types de distribution des données sont couramment admis. Le premier est une distribution homogène, parfois appelée distribution horizontale, dans laquelle chaque sous-ensemble de données est une partition de l'ensemble de données général possédant tous ses attributs. Le second type de répartition est qualifié d'hétérogène, ou distribution verticale. Dans ce dernier, les différents sites ne contiennent plus des groupes de lignes de l'ensemble général, mais des colonnes différentes de ce même ensemble. Néanmoins, on assume toujours la présence d'une clé, commune à tous les sites, permettant, s'il le faut, d'associer les différentes lignes entre elles. Un exemple d'une répartition homogène est présenté en Table 3.1 dans lequel, les sites 1 et 2 ont des données différentes, mais contenant exactement les mêmes attributs (humidité, température, pression et altitude). Un exemple de répartition hétérogène est, quant à lui, présenté en Table 3.2. Dans celui-ci, les sites ne contiennent plus les mêmes attributs (humidité et pression pour l'un contre température et altitude pour l'autre), mais partagent cependant une clé commune qui ici est le temps.

Tableau 3.1 : Exemple de distribution homogène des données

Site	Humidité	Température	Pression	Altitude
Site 1	75%	26.5°C	101.5kPA	235m
	65%	27.3°C	100.8 kPa	8 m
Site 2	85%	32.5°C	101.8 kPa	486 m
	56%	33.6°C	101.5 kPa	85 m

Que l'ensemble soit distribué de manière homogène ou pas, chaque site extrait

Tableau 3.2 : Exemple de distribution hétérogène des données

Site 1			Site 2		
Temps	Humidite	Pression	Temps	Temperature	Altitude
T1	75%	101.5 kPa	T1	26.5°C	235 m
T2	65%	100.8 kPa	T2	27.3°C	8 m
T3	85%	101.8 kPa	T3	32.5°C	486 m
T4	56%	101.5 kPa	T4	33.6°C	85 m

un modèle de la fraction du jeu dont il dispose. Ce modèle local va être partagé en communiquant, avec une entité centralisée ou avec les sites voisins, et ce dans le but de bâtir, ensemble, un modèle commun à tous les sites. Ce type de fonctionnement est particulièrement adapté si le système possède une des quatre caractéristiques suivantes (Dasilva *et al.*, 2005) :

1. Il est composé de multiples entités (de stockage ou de calcul) communiquant uniquement par messages
2. La communication entre les différents sites est onéreuse en temps ou en énergie donc il convient de limiter au maximum les messages
3. Les sites ont des contraintes de ressources (e.g. processeurs, batteries, mémoire, etc.)
4. La confidentialité des données doit être respectée. En effet puisque l'ensemble des données est distribué, il est difficile pour une entité malicieuse de prendre connaissance de l'entièreté des données.

Pour illustrer ce propos dans le cadre de la reconnaissance d'activités, on peut

prendre l'exemple d'un ensemble de capteurs intelligents récoltant chacun certaines données de l'environnement. Chaque capteur communique avec ses voisins par des messages tout en gardant à l'esprit qu'en ce qui concerne la batterie, chaque message a un coût. De plus, ces entités sont encore loin des ressources quasi illimitées présentes dans les serveurs de calculs modernes et l'algorithme doit donc prendre cette particularité en compte. Pour finir, si quelqu'un de mal intentionné venait à posséder l'ensemble des données, il pourrait avoir accès à des informations confidentielles sur les habitudes de vie de l'habitant. Ainsi, par application des quatre points de Dasilva *et al.* (2005), il est possible de dire que le forage de données distribué est particulièrement bien adapté à ce cas.

Le reste de cette partie présente des exemples des trois grands types d'algorithmes que l'on peut trouver dans la littérature du forage de données distribué. La première partie sera dédiée aux réseaux bayésiens distribués. En second, la distribution des arbres de décision sera passée en revue suivie pour finir des techniques de clustering.

3.2.1 LES RÉSEAUX BAYÉSIENS DISTRIBUÉS

Quand il s'agit de réseaux bayésiens, deux approches sont possibles. La première, étudiée précédemment, est statistique et demande la parfaite connaissance du problème et de ses probabilités. La seconde, basée sur le forage de données, prône l'apprentissage du modèle à partir d'un ensemble de données. Si l'on considère le sous-domaine du forage de données distribué, plusieurs auteurs ont apporté des solutions à l'apprentissage d'un réseau bayésien (Chen *et al.*, 2003; Shen *et al.*, 2003; Tedesco *et al.*, 2006; Wright

et Yang, 2004; Yamanishi et Kenji, 1997). Cette partie présente deux travaux suivant que les données sont réparties de façon homogène ou hétérogène.

Yamanishi et Kenji (1997) proposent une méthode d'apprentissage pour les ensembles de données distribués de manière homogène. Les auteurs présentent une architecture où chaque site possède un agent chargé d'apprendre un modèle local en appliquant, sur les observations qu'il possède, une version probabiliste de l'algorithme de Gibbs (Geman et Geman, 1984). Ce modèle local va ensuite être partagé à une entité centralisée nommée « population learner » chargée de construire le modèle global en combinant tous les modèles locaux des différents agents. Cette méthode comporte deux avantages majeurs. Le premier est qu'en aucun cas les données d'apprentissage ne sortent des sites où elles sont situées renforçant la confidentialité de celles-ci. En effet, seul le modèle local est transmis à travers le réseau jusqu'au « population learner ». Le deuxième avantage est la simplicité, car sur les sites, c'est un algorithme bien connu d'apprentissage qui est exécuté, la méthode de Gibbs, et le « population learner » n'effectue qu'une simple moyenne de toutes les probabilités venant des modèles locaux. Cependant, cette méthode souffre d'un défaut qui est l'obligation d'une entité centralisée qui peut être considérée comme un handicap dans certaines architectures à très haute fiabilité.

Plutôt que les données homogènes, Chen *et al.* (2003) étudient eux l'apprentissage dans le cas où les données sont réparties de manière hétérogène. Leur méthode se découpe en trois étapes principales. La première est l'apprentissage local où chacun des sites établit une structure bayésienne à partir de ses données en utilisant un algo-

rithme tel que K2 ou Gibbs (Cooper et Herskovits, 1992; Geman et Geman, 1984). La seconde phase consiste à apprendre un modèle croisé correspondant aux liens existants entre les variables des différents sites. Pour ce faire, les auteurs proposent que chacun des sites choisisse un sous-ensemble de valeurs à l'aide d'un seuil de correspondance au modèle local. Toutes les observations en dessous de ce seuil (c.-à-d. les moins ressemblantes au modèle appris) sont considérées comme de possibles relations croisées et sont transmises à une entité centralisée. Cette dernière, une fois toutes ces données reçues, construit un réseau bayésien représentant les relations existantes entre les différentes observations des sites. Pour finir, ce réseau est combiné aux réseaux locaux dans le but d'y ajouter les liens entre les sites et de supprimer les liens causés par le problème de la variable cachée appris à la première étape. Bien que cette méthode offre de bons résultats, elle connait deux problèmes majeurs. Le premier est que dans le cas où les données sont confidentielles, une partie d'entre elles transite sur le réseau permettant à une entité malicieuse de les récupérer. Le second est la présence de cette entité centralisée qui dans certaines architectures n'a pas sa place.

La littérature a démontré que peu importe si les données sont réparties de manière homogène ou pas il est possible d'apprendre la structure d'un réseau bayésien les représentant. Bien que les méthodes présentées souffrent de quelques défauts, elles permettent tout de même d'apprendre un réseau quasi identique à un appris de manière centralisée. De plus, elles ne font transiter que très peu d'informations sur le réseau ce qui dans le cas où les coûts de communication sont élevés (e.g. batterie ou faible bande passante) est bien meilleur que de transmettre toutes les données d'apprentissage.

3.2.2 LES ARBRES DE DÉCISION DISTRIBUÉS

Dans leur travail de 2004, Caragea et Silvescu (2004) proposent une méthode pour apprendre la structure d'un arbre de décision sur un ensemble de données réparti. L'originalité majeure de leur travail est que l'algorithme proposé fonctionne autant sur une répartition verticale qu'horizontale. Ici, le postulat de base est que l'élément le plus important pour la construction d'un arbre est le nombre d'instances respectant un ensemble de conditions sur un ensemble d'attributs (e.g. nombre d'instances dont l'attribut X est égal à une certaine valeur). Pour ce faire, Caragea et Silvescu (2004) proposent de centraliser ces statistiques calculées par chacun des sites de l'architecture distribuée. Ainsi, à chaque fois que l'algorithme veut ajouter un nœud à l'arbre, tous les sites calculent le nombre d'instances locales respectant certaines conditions sur les attributs considérés et renvoient le résultat. Une fois les instances correctement comptées, l'entité centralisée peut effectuer un calcul d'entropie pour chacun des attributs et, à partir de ces résultats, choisir l'attribut partageant le mieux les données.

Dans le cas où les données sont réparties de manière homogène, le problème est trivial et chaque site ne fait que compter ses exemples et renvoyer cette valeur. Mais dans le cas où les données sont réparties de manière hétérogène, tous les sites n'ont pas les mêmes attributs. Pour répondre à ce défi, les auteurs prennent pour acquis l'existence pour chaque exemple de l'ensemble de données d'une clé unique répartie sur tous les sites. Ceci permettant d'affirmer que chaque site possède toujours un et un seul sous-ensemble de l'exemple général. Par conséquent, l'opération de comptage peut se faire de la même manière qu'en cas de répartition homogène si ce n'est que les sites

comptent que les instances des attributs qu'ils possèdent. La méthode proposée par Caragea et Silvescu (2004) offre plusieurs avantages. Le premier d'entre tous est qu'ici le fonctionnement ne diffère pas de la construction d'un arbre de manière centralisée permettant d'exécuter n'importe quel algorithme de construction. Le second est que seules les statistiques transitent sur le réseau réduisant de manière considérable la consommation de bande passante par rapport à l'approche centralisée. Pour finir, cette méthode fonctionne avec des données homogènes et hétérogènes ce qui est assez rare dans la littérature.

Une autre approche est celle de Giannella *et al.* (2004) pour les ensembles de données répartis verticalement. Les auteurs proposent une méthode pour diminuer le nombre de messages échangés lors de la création de l'arbre. Pour ce faire, ils démontrent la possibilité du calcul de l'indice Gini de manière distribué. Cet indice, couramment utilisé comme mesure de dispersion, peut, selon les auteurs, être exprimé sous la forme du produit de vecteurs binaires (c.-à-d. des vecteurs ne contenant que des 0 et des 1) représentant chacun l'ensemble des valeurs d'un site. Ainsi, plutôt que d'échanger tous les ensembles de données pour calculer cette mesure, les sites n'échangent que des vecteurs binaires. Pour optimiser de manière plus radicale encore ces coûts de communication, Giannella *et al.* (2004) utilisent une projection aléatoire de chaque vecteur sur un espace de dimension réduite. De ce fait, ce n'est ni le jeu de données de base ni les vecteurs de même taille qui transitent sur le réseau, mais leurs projections. Cette approche permet, d'après les tests réalisés par les auteurs, de diviser par cinq les communications par rapport au partage de tout le jeu de données tout en conservant une précision de 80% par rapport à une méthode d'apprentissage centralisée. Cette mé-

thode offre l'avantage de réduire considérablement la quantité de messages nécessaire à la création de l'arbre final ce qui, dans le cas où les communications ont un coût important, est vital. Néanmoins, l'algorithme ne réalise qu'une approximation de l'arbre idéal et par conséquent il n'offre pas nécessairement la même précision.

La littérature existante sur les arbres de décision prouve que l'apprentissage de ces structures, couramment utilisé en reconnaissance d'activités, peut être réalisé de manière distribuée. Certaines méthodes se concentrent sur l'optimisation du nombre de messages échangés, d'autre sur l'obtention du meilleur arbre possible et certaines même sur l'aspect confidentiel d'un tel apprentissage (Emekçi *et al.*, 2007). Par conséquent, il faut faire attention à bien définir le cas dans lequel l'application désirée se situe de sorte à utiliser l'algorithme le plus adapté possible.

3.2.3 LE CLUSTERING DISTRIBUÉ

En tant que part importante des algorithmes d'apprentissage non supervisés, les méthodes de clustering ont toujours bénéficié d'un grand intérêt de la part de la communauté scientifique. En forage de données distribué, on retrouve le même type d'intérêt avec de nombreux travaux sur la manière de distribuer ces algorithmes. Selon Dasilva *et al.* (2005), les algorithmes de clustering distribués peuvent se diviser en quatre catégories principales. La première division peut se faire suivant le type de répartition des données (c.-à-d. homogène ou hétérogène), tandis que la seconde intervient sur l'objectif principal de l'algorithme proposé. En effet, certains travaux se concentrent sur la préservation de la confidentialité tandis que d'autres préfèrent optimiser l'efficacité de

l'algorithme.

De nombreux travaux existent sur la préservation de la confidentialité au sein des algorithmes de clustering distribués (Dasilva *et al.*, 2005; Klusch *et al.*, 2003; Merugu et Ghosh, 2003; Vaidya et Clifton, 2003). Quand on considère une répartition horizontale des données, un bon exemple est le travail de Klusch *et al.* (2003) amélioré plus tard par Dasilva *et al.* (2005). Les auteurs proposent l'algorithme Kernel Density Estimation based Clustering (KDEC) utilisant la densité des régions du jeu de données pour découvrir les clusters. Dans cet algorithme, les sites n'échangent aucune donnée de leur ensemble d'apprentissage, mais des estimations de densités locales d'une région. Ces dernières sont, par la suite, sommées au sein d'une entité centralisée permettant de recréer une estimation de la densité globale de la région qui est retournée aux sites. De la sorte, chacun d'entre eux est capable de regrouper les régions denses en cluster, et ce sans échanger d'exemples d'apprentissage.

Pour le cas où la répartition est verticale, un bon exemple est le travail de Vaidya et Clifton (2003). Les auteurs montrent tout d'abord qu'assigner un point à un cluster de manière confidentielle étant donné que les attributs sont répartis sur différents sites est un problème non trivial. Pour résoudre ce dernier, un K-means utilisant un chiffrement homomorphique (Benaloh, 1994) est proposé. L'intérêt d'un tel chiffrement est qu'il est possible de calculer la valeur chiffrée de la composition de deux éléments sans avoir à déchiffrer lesdits éléments (e.g. $H(x+y) = H(x) + H(y)$ avec H une fonction de chiffrement). Ainsi, il est facile de réaliser des sommes et des comparaisons qui sont les éléments de base de K-means sans jamais divulguer les données non chiffrées.

L'autre facette du clustering distribué est l'optimisation de l'efficacité des algorithmes en réduisant au maximum les communications entre les sites (Dhillon et Modha, 2002; Johnson et Kargupta, 2000; Merugu et Ghosh, 2005). Dans le cadre d'une répartition horizontale des données, un bon exemple de ce type de travail est l'article de Dhillon et Modha (2002). Dans ce dernier, les auteurs proposent une version distribuée de K-means où les sites n'échangent que les barycentres des clusters. Ainsi, à chaque itération de l'algorithme, plutôt que d'échanger leurs ensembles de données complets, les différents sites exécutent l'algorithme localement et émettent aux autres les barycentres découverts. Dès qu'un site a reçu tous les barycentres, il en fait la moyenne obtenant ainsi le barycentre moyen pour l'ensemble du jeu de données.

Quand les données sont reparties de manière hétérogène à travers les sites, le problème est un peu plus complexe. Dans ce domaine, Johnson et Kargupta (2000) apportent un très bon exemple avec un algorithme de clustering hiérarchique. Un tel algorithme cherche à construire un arbre appelé dendrogramme dont les feuilles sont les données d'apprentissage et les branches les distances séparant ces points ou clusters. Si cet arbre est parcouru depuis sa racine jusqu'aux feuilles, il représente une série de clusters hiérarchisés. La méthode employée par les auteurs est de réaliser tous les dendrogrammes locaux et d'envoyer ceux-ci à une entité centralisée chargée de réaliser le modèle global. De manière classique, envoyer un dendrogramme signifie aussi renvoyer tous les points puisque ceux-ci constituent les feuilles de cet arbre. Dans leur cas, les auteurs préfèrent stipuler l'existence d'une clé unique pour chaque donnée et utilisent cette clé comme feuille de leur arbre. Ainsi, les coûts en matière de communication sont drastiquement réduits.

Les travaux des auteurs présentés ci-dessus démontrent la possibilité de réaliser des algorithmes de clustering sur des données distribuées. Nous avons montré une série d'exemples prouvant que peu importe le cas il existe des travaux dans la littérature le concernant. Ainsi, il semble possible de réaliser du clustering de données confidentielles comme l'ont montré les travaux de Klusch *et al.* (2003) et Vaidya et Clifton (2003) respectivement sur des données homogènes et hétérogènes. De la même manière, les travaux de Dhillon et Modha (2002) et Johnson et Kargupta (2000) sur des données respectivement réparties de manière horizontale et verticale prouvent la possibilité d'effectuer un clustering distribué très efficace.

3.2.4 BILAN DU FORAGE DE DONNÉES DISTRIBUÉ

Tout au long de cette partie, nous nous sommes efforcés de présenter différents outils existants permettant la réalisation de forage de données distribué. Le travail de Dasilva *et al.* (2005) nous a tout d'abord permis de définir dans quel contexte il convenait d'appliquer ces techniques. Ainsi, si le système possède plusieurs entités limitées avec des coûts de communications élevés et des données confidentielles, il convient de réaliser du forage de données distribué. Ensuite, la première question à laquelle il faut toujours répondre concerne la façon dont les données sont réparties. En effet, nous avons montré que l'ensemble des algorithmes et méthodes se divise en deux grandes catégories en lien direct avec le type de répartition des données qui peut être homogène ou hétérogène. Pour finir, nous avons montré qu'il existait, dans la littérature, un grand nombre de travaux (Tedesco *et al.*, 2006; Caragea et Silvescu, 2004; Dasilva *et al.*, 2005) notamment en ce qui concerne les réseaux bayésiens, les arbres de décision et

les méthodes de clustering. Chacun de ces domaines du forage de données possède son équivalent distribué, et ce, que les données soient réparties verticalement ou horizontalement.

3.3 CONCLUSION

La littérature nous montre qu'il existe un bon nombre de méthodes différentes pour reconnaître des activités. On peut tout d'abord utiliser des méthodes probabilistes directement issues de la reconnaissance de plan. Celles-ci offrent d'excellentes performances, mais obligent leur utilisateur à maîtriser à la fois toutes les variables du problème et l'environnement où celui-ci s'exécute. Une autre possibilité est l'utilisation du forage de données. Dans ce domaine les techniques sont nombreuses et ont chacune leurs avantages et leurs inconvénients. Les arbres de décision offrent une bonne précision, mais ils forcent l'utilisation d'un vaste jeu de données d'apprentissage déjà étiqueté. Le clustering propose une alternative à cet étiquetage au détriment de la légèreté de la phase d'apprentissage tant en mémoire qu'en calcul. De plus, beaucoup des méthodes de clustering obligent la connaissance du nombre de clusters à l'avance. Les machines à vecteurs de support sont quant à elle excellentes en termes de performance et de taux de reconnaissance néanmoins leur apprentissage peut être complexe et lourd à implémenter. Pour finir, les réseaux de neurones sont souvent employés, car ils offrent de bons taux de reconnaissance et une excellente robustesse face aux données bruitées. Ces avantages sont cependant à nuancer puisque leur apprentissage est très long et leur architecture nécessite une certaine expertise dans le domaine.

La plupart des algorithmes utilisés dans la reconnaissance d'activités sont centra-

lisés. En effet, ceux-ci se basent sur les architectures, centralisées elles aussi, présentées au chapitre précédent. Le meilleur moyen pour concevoir des architectures robustes est de décentraliser ce qui signifie des données et des puissances de calcul réparties. Dans ce cas, le défi consiste à fournir la même intelligence tout en prenant en compte la caractéristique distribuée du système. Une des réponses à ce problème est le forage de données distribué. La littérature à ce sujet montre que peu importe le type de répartition des données (homogène ou hétérogène), il est toujours possible de réaliser la plupart des algorithmes de reconnaissance d'activités. Que ce soit les réseaux bayésiens, les arbres de décision ou le clustering, il semble toujours y avoir au moins une façon de réaliser ces algorithmes. Et ce, peu importe, que les données soient réparties horizontalement ou verticalement.

CHAPITRE IV

UNE NOUVELLE MANIÈRE DE COMMUNIQUER AU SEIN DE LA MAISON INTELLIGENTE

Nous avons vu, au cours des chapitres précédents, que les architectures existantes comportent certaines faiblesses quand il s'agit de se mettre à l'échelle ou de fournir suffisamment de fiabilité pour leur domaine d'application, l'assistance. Dans cette thèse, nous proposons une nouvelle architecture répondant à ces problématiques. Pour ce faire, nous allons utiliser les capteurs et effecteurs, déjà présents dans l'environnement, comme support de l'infrastructure. Malheureusement, avant de pouvoir mettre en place une telle avancée, il convient de se pencher en premier lieu sur la faisabilité de la communication entre ces entités. Cette partie a été le sujet principal d'un article (Plantevin *et al.*, 2017) publié dans le cadre de cette thèse dans le journal *Sensors*.

Le problème majeur quand il s'agit de communication au sein de ce type d'architecture hétéroclite est la différence entre les entités qui la compose. En effet, si nous voulons que notre solution supporte le plus de types de matériel différents, nous devons nous rendre compatibles avec le plus de plateformes. Ainsi, nous désirons rendre possible l'intégration de transducteurs embarquant différents systèmes (e.g. Linux, Windows, FreeRTOS) implémentés sur différents composants matériels, mais aussi communiquant de manière différente (e.g. ZigBee, Wi-Fi ou BLE). Pour répondre à ce problème, il nous faut penser à un nouveau moyen de communiquer. Ce chapitre présente Light Node Communication Framework (LNCF), un protocole de communication créé

durant cette thèse. Nous commençons par présenter le protocole ainsi que les autres existants puis nous démontrons ses capacités aux moyens de tests réalisés en laboratoire.

4.1 Light Node Communication Framework

La contribution explicitée dans ce chapitre est un protocole de communication ayant deux caractéristiques principales. La première est qu'il peut être embarqué sur n'importe quels dispositif, matériel ou système tant que celui-ci implémente une pile IP. Celle-ci est indépendante des protocoles matériels de communication qui peuvent aller de l'Ethernet au ZigBee IP en passant évidemment par le Wi-Fi ou le 6LowPan. La deuxième caractéristique majeure de notre protocole est qu'il n'a besoin d'aucune entité centralisée pour fonctionner et donc aucun « broker ». Ce point particulier était le problème dans la plupart des solutions déjà existantes (e.g. MQTT, NATS, RabbitMQ, etc.). Afin de répondre à cette problématique et implémenter notre solution, nous avons formulé deux hypothèses de base. Tout d'abord, nous pensons que les messages doivent rester au sein du réseau de la maison intelligente et donc ne peuvent pas traverser Internet. Ensuite, nous avons statué que le protocole UDP est le protocole minimal à implémenter pour assurer une bonne communication, puisque celui-ci est à la base de nombreux protocoles de contrôle et de configuration des réseaux (e.g. DHCP ou DNS).

L'un des premiers défis auquel nous avons dû répondre est la différence fondamentale entre deux grands types d'informations. En effet, au sein de l'environnement intelligent subsistent deux flux d'informations différents qui sont la configuration et les données. Le premier doit être basé sur un système de livraison fiable, permettant la

communication point à point et dont la vitesse importe peu puisque ce sont des données qui changent rarement. Le second flux est tout l'inverse du premier puisqu'une grande quantité de données doit transiter en un minimum de temps à un maximum de clients pour prendre des décisions d'assistance rapidement. Néanmoins, si un paquet de données se perd à un moment ce n'est pas critique, car il sera vite remplacé par un nouveau généré très peu de temps après. Dans le but de répondre à cette problématique particulière, nous proposons d'utiliser deux canaux différents comme c'est le cas dans le protocole File Transfer Protocol (FTP) et ce même si celui-ci est basé sur TCP (Postel, 1980). Le reste de cette section est consacré à l'explication complète du protocole présenté.

4.1.1 CANAL DE CONFIGURATION

Comme nous l'avons explicité plus haut, la configuration d'une entité intelligente au sein d'un environnement se doit d'être distribuée au travers d'un canal de communication fiable. Afin d'implémenter cette caractéristique, complexe avec le protocole UDP, nous avons décidé d'utiliser Constrained Application Protocol (CoAP), un protocole connu du monde de l'IoT et déjà implémenté sur de nombreuses plateformes (Shelby *et al.*, 2014).

CoAP est un protocole qui utilise UDP pour implémenter un système de requête/réponse, très ressemblant à HTTP, afin de récupérer et/ou modifier des valeurs représentées par des URI. De plus, il définit déjà une manière fiable d'échanger des informations en utilisant un système basé sur des accusés de réception pour les messages

qui en auraient besoin (i.e. les messages demandant une haute fiabilité). Ici, nous proposons d'utiliser cette représentation sous forme d'Unified Resource Identifier (URI) et les messages à haute fiabilité de CoAP pour la configuration des entités de notre système.

Dans le but de simplifier la compréhension de ce concept, la Table 4.1 présente un exemple d'une telle configuration implémentée en CoAP. Dans ce tableau, les lignes représentent des variables de configuration potentielles associées à leur URI, les méthodes autorisées sur ces URI pour récupérer ou modifier les variables de configuration et le type de donnée attendu. La première variable est assez évidente et correspond au taux de rafraîchissement d'un capteur. C'est un entier (e.g. 50Hz), représenté par l'URI */rate* sur laquelle les méthodes GET et POST sont autorisées permettant donc de récupérer ou modifier cette valeur. De la même manière, le fonctionnement semblable à HTTP nous permet de définir des variables en lecture seule comme la version de l'entité sous forme d'une chaîne de caractères, récupérable par une requête GET sur l'URI */version* mais non modifiable. De plus, CoAP implémente des types de données bien plus complexes comme le JSON que nous utilisons ici pour récupérer ou modifier la configuration matérielle de l'entité (i.e. les capteurs qui lui sont connectés). Une autre possibilité intéressante de CoAP, est sa capacité à transférer de gros fichier binaire par l'utilisation de l'extension *Block-Wise*. Ainsi, notre exemple utilise cette fonctionnalité pour implémenter un système de mise à jour du logiciel embarqué derrière l'URI */update*. Pour conclure cet exemple, nous présentons une capacité importante de CoAP qui est la sécurisation de l'échange en utilisant DTLS. Cela permet, par exemple, d'échanger des clés de chiffrements symétriques avec n'importe quel dispositif. Cette opération

	URI	Methodes autorisées	Type de données
Taux de rafraîchissement	/rate	GET/POST	Entier
Version	/version	GET	String
Materiel	/hardware	GET/POST	JSON
Mise à jour	/update	POST	Binaire
Clés de chiffrement	/keys	POST	Binaire

Tableau 4.1 : Un exemple de configuration utilisant CoAP

est critique et doit être accomplie avec une grande sécurité, car quiconque récupérerait la ou les clés de chiffrement pourrait écouter et parler sur le réseau comme s'il était une entité légitime. Avec notre méthode, nous utilisons juste CoAP avec son échange DTLS pour garantir la confidentialité, l'intégrité et l'authenticité de ces clés.

Maintenant que le canal de configuration de notre protocole est expliqué, nous pouvons nous attarder sur l'échange de données au travers du protocole de messagerie que nous avons développé pour LNCF.

4.1.2 CANAL DE DONNÉES

En plus d'un flux de communication fiable pour la configuration, nous nous devons de fournir un moyen d'échanger des messages entre chacune des entités de notre réseau. Les caractéristiques principales de ce canal sont, permettre l'envoi d'un maximum de données en un minimum de temps, découvrir les autres entités connectées au réseau, permettre de chiffrer les données sensibles et compresser certaines données

trop grandes. Afin d'atteindre ces objectifs, nous proposons ici un simple protocole de messagerie Pub/Sub basé sur UDP pour sa capacité à transférer des donnés à plusieurs entités en même temps (i.e. multicast) ou à une seule (i.e. unicast) le tout avec la plus faible latence possible.

Pour résumer le protocole, et ce pour aider la compréhension générale de celui-ci, les clients se connectent sur le réseau et rejoignent le groupe multicast défini par son adresse IP. Ensuite, ils s'abonnent à un ou plusieurs sujets représentés par une simple chaîne de caractères et ils recevront tous les messages labélisés avec ce sujet. Avant cela ils peuvent, s'ils le veulent, demander quels capteurs sont connectés aux réseaux et émettent sur quel sujet par l'envoi d'un simple paquet de découverte contenant une requête (e.g. tous les capteurs ayant le service « temperature »). En réponse à cette requête, les capteurs concernés vont répondre, à la manière du protocole mDNS, avec leurs adresses IP et les sujets qu'ils exposent. Nous allons maintenant expliquer dans le détail, le fonctionnement des quatre modes de fonctionnement de notre protocole qui sont l'envoi de données brutes, la découverte du réseau, l'envoi de données sécurisées et pour finir l'envoi de données compressées.

L'ENVOI DE DONNÉES

La Figure 4.1a représente un message de donnée non compressé ni chiffré. Chaque paquet commence par un octet d'option divisé en deux parties. Les trois bits de poids forts (Most Significant Bit (MSB)) forment le numéro de la version du protocole utilisé donc 0 ou 1 dans le cas de ce paquet. Les cinq bits restants constituent les différentes

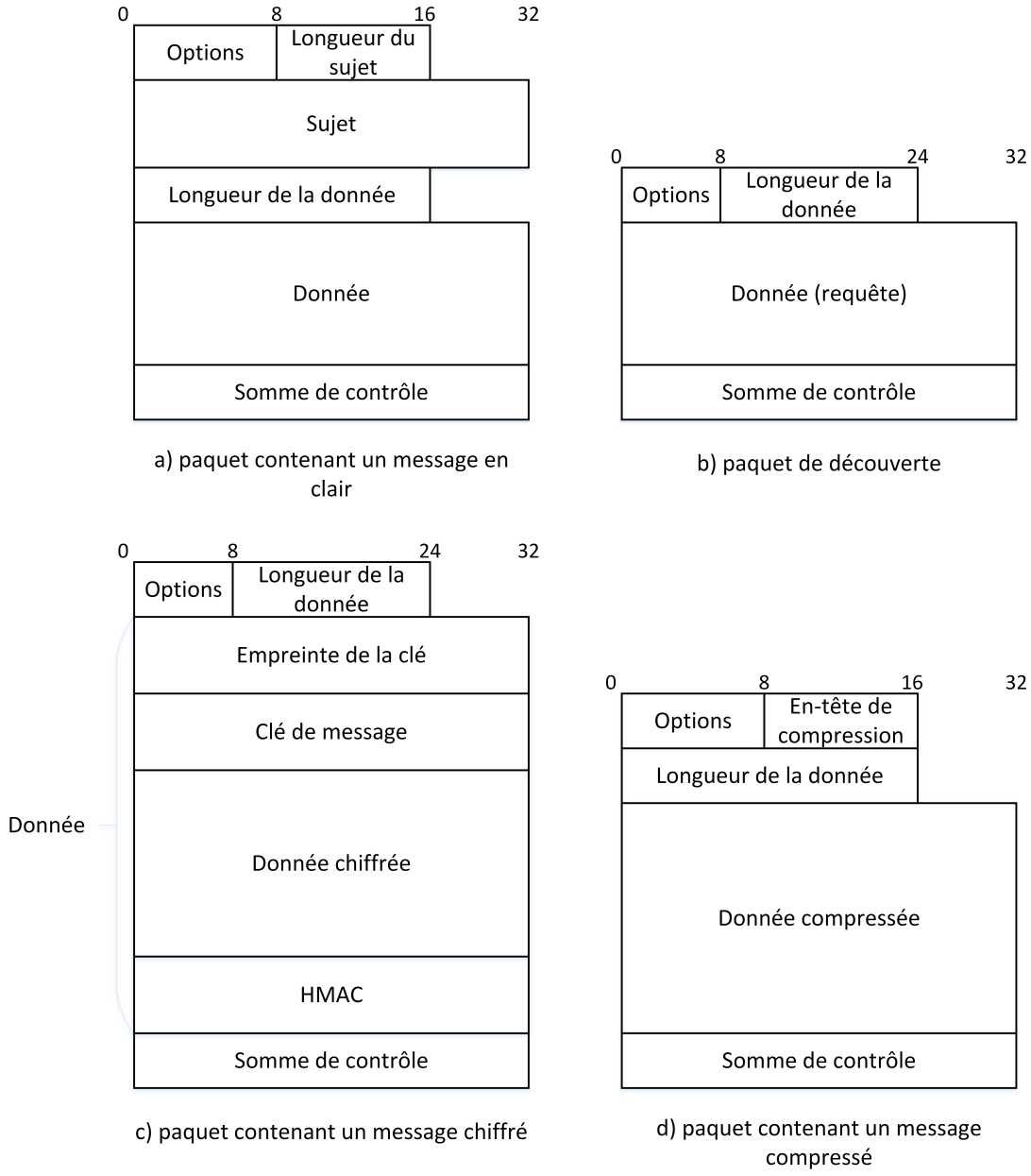

Figure 4.1 : Une représentation schématique des quatre paquets utilisés dans le protocole de messagerie.

options du paquet avec le bit de poids fort non utilisé et réservé, les quatre bits restants sont tous à 0 dans le cas d'un simple paquet de données sans chiffrement ni compression, mais sont utilisés dans les autres modes de communication. L'octet suivant représente la longueur du sujet pouvant par conséquent aller de 1 à 255 caractères encodés sur un seul octet. Tout message ne comprenant pas de sujet doit être considéré comme un paquet erroné et doit par conséquent être ignoré. Ensuite vient le sujet en lui-même d'une taille variable, mais défini précédemment, suivi de la longueur de la donnée encapsulée encodée sur 2 octets (le protocole tolère les paquets ne contenant aucune donnée) lui-même suivi de la donnée de taille variable. Pour clore ce paquet, une somme de contrôle est ajoutée à la fin calculée sur le paquet complet en utilisant CRC-32 un algorithme léger et rapide, déjà couramment utilisé par le protocole Ethernet (Mitra et Nayak, 2015). Il convient de noter que la taille du paquet total est limitée à 65,535 octets, qui est la limite théorique d'un paquet UDP. Quand un client reçoit le paquet, la première chose qu'il doit toujours faire est de vérifier la bonne réception de celui-ci en recalculant le CRC-32 et en le comparant avec celui reçu. Le protocole ne supportant aucun mécanisme de renvoi des paquets corrompus ceux-ci doivent être ignorés. Dans l'autre cas, le paquet peut être découpé facilement en connaissant les différentes tailles définies précédemment.

MESSAGE DE DÉCOUVERTE

Une des fonctionnalités que nous avons à fournir est la possibilité de découvrir les entités présentes dans le réseau. Pour atteindre cet objectif, ces dernières peuvent

enregistrer un ensemble de clé/valeur en plus de toutes variables de configuration qui peut être lue (i.e. qui autorise la méthode GET) et qui seront exposées en cas de requête de découverte.

La Figure 4.1b représente un de ces paquets de découverte et l'on peut apercevoir qu'il commence avec un octet d'option. Cet octet, comme pour le paquet de données, commence par le numéro de version suivi des différentes options. Ici, seule le LSB est mis à 1 signifiant que le paquet représente une requête de découverte. Suivant ce premier octet vient la taille de la donnée encapsulée, codée sur deux octets, puis la requête en elle même. Cette dernière est une simple structure de donnée représentant un ensemble de clés/valeurs au format JSON. Pour des raisons de sécurité et dans le but de limiter le nombre de réponses, le paquet de découverte doit obligatoirement comporter une requête. Par conséquent, tout paquet avec une donnée de taille zéro doit impérativement être ignoré.

Une fois le paquet reçu et vérifié, le receveur décide s'il doit y répondre ou pas. Pour ce faire, il commence par vérifier que toutes les clés de la requête sont enregistrées dans sa mémoire. Si une d'entre elles n'a pas été enregistrée au préalable, il ne doit en aucun cas répondre. Ensuite, il peut commencer à regarder les valeurs associées à ces clés. Ici, deux cas peuvent exister : soit la clé contient une valeur unique soit elle contient un tableau de valeur. Par conséquent, 4 cas peuvent arriver en fonction de la combinaison des types de valeur du receveur et de l'envoyeur. Des exemples de ces 4 cas sont présentés dans la Table 4.2 où la première colonne correspond à la clé de la valeur concernée, la deuxième la valeur reçue dans le paquet, la troisième la valeur

Clé	Valeur demandée	Valeur exposée	Conclusion
version	"1.0.1"	"1.0.1"	OK
nom	"temp2"	"temp32"	NOK
unité	"C"	["C", "F"]	OK
unité	"K"	["C", "F"]	NOK
unité	["C", "F"]	"C"	OK
unité	["C", "F"]	"K"	NOK
unité	["C", "F"]	["C", "K"]	OK
unité	["C", "F"]	["K", "R"]	NOK

Tableau 4.2 : Un exemple du processus de décision en cas de paquet de découverte.

exposée et pour finir la conclusion avec OK si le receveur doit répondre et NOK dans le cas contraire. Le premier cas est quand les deux clés contiennent des valeurs uniques. Dans ce cas, les deux valeurs doivent être strictement égales comme présenté dans les deux premières lignes du tableau. Ensuite, il est possible qu'une des clés contienne un tableau et l'autre une valeur. Ici, il faut que la valeur unique soit comprise dans le tableau comme représenté dans les lignes 3 à 6 du tableau. Pour finir, quand les deux clés encapsulent des tableaux il faut qu'une des valeurs du tableau reçu soit dans le tableau exposé comme le montrent les deux dernières lignes du tableau.

Si toutes les décisions sont d'envoyer une réponse, le receveur va envoyer, via une communication unicast, à l'expéditeur un paquet encapsulant un JSON contenant les valeurs exposées et les différents sujets que l'entité expose ainsi que ceux auxquels elle est abonnée. Ce paquet est constitué comme un paquet de découverte si ce n'est que

cette fois-ci ce n'est pas le LSB de l'octet d'option qui est mis à 1, mais le bit numéro 3 signifiant un paquet de réponse à une requête de découverte. Si lors du processus de décision, une des valeurs ne correspond pas, le receveur doit impérativement ignorer le paquet.

MESSAGE SÉCURISÉ

Dans les environnements intelligents, nous sommes souvent amenés à faire transiter des données confidentielles ou personnelles. Par conséquent, notre protocole se doit de fournir une méthode simple permettant d'échanger des données de manière sécurisée si l'émetteur de la donnée l'estime nécessaire. Pour arriver à cet objectif, nous avons adapté un protocole connu, et déjà implémenté dans l'application Telegram (Telegram, 2013), pour qu'il fonctionne avec nos limitations (UDP, entités légères).

Le processus de chiffrement, résumé en Figure 4.2, commence toujours par l'ajout d'octets aléatoires de remplissage devant le paquet d'octets composant le sujet et la donnée comme stipulé dans le paquet de données standard. Cette opération doit être faite, car Advanced Encryption Standard (AES), l'algorithme de chiffrement que nous utilisons utilise le chiffrement par bloc et donc, la donnée chiffrée doit être un multiple de la taille d'un bloc B_s. Le nombre d'octets aléatoire P_s que nous devons ajouter est défini par la formule 4.1. Ainsi, nous ajoutons toujours au minimum B_s octets pour insérer de l'aléatoire dans le paquet final ensuite nous ajoutons suffisamment d'octets pour devenir un multiple de la taille du bloc (16 dans le cas de AES 128 bits). Dans la formule 4.1 nous utilisons $Longueur\_donnee + 2$, car l'étape finale de la préparation

au chiffrement est l'ajout, devant les octets de rembourrage, de la taille de la donnée, encodée sur deux octets.

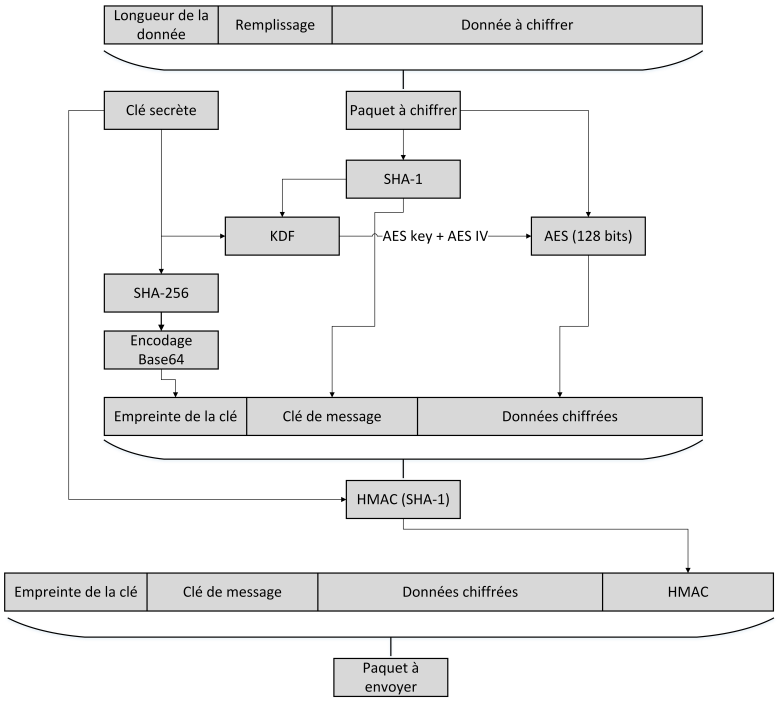

Figure 4.2 : Le processus de chiffrement d'un paquet.

$$P_s = B_s + (B_s - (Longueur\_donnee + 2)\%B_s) \tag{4.1}$$

Une fois le paquet prêt pour le chiffrement, nous créons ce que nous appelons une clé de message en calculant le SHA-1 de ce dernier. Ce hash va servir à générer la clé et l'Initialization Vector (IV) utilisé par l'algorithme AES responsable du chiffrement. Pour cela, nous passons la clé de message et la clé secrète, préconfigurée en utilisant le canal de configuration, à une fonction de dérivation de clé (Key Derivation Function (KDF)). Cette dernière, explicitée dans l'Algorithme 4.1, est une suite de hachages (utilisant l'algorithme SHA-1) donnant deux résultats, une clé AES de 128 bits et un

IV utilisés pour chiffrer le paquet. Ensuite, dans le but d'identifier la clé secrète utilisée pour calculer la clé de chiffrement, on ajoute, à l'avant du paquet chiffré, l'empreinte de celle-ci calculée en utilisant l'algorithme SHA-256 et en convertissant ce résultat en Base64. Pour finir, nous calculons le code d'authentification d'une empreinte cryptographique de message avec clé (Hash Message Authentication Code (HMAC)) en utilisant la clé secrète et l'algorithme de hachage SHA-1.

Données : secret_key[16], msg_key[20]

Résultat : aes_key[16], aes_iv[16]

```
1 sha1_a = sha1(msg_key + secret_key[0...3]);
2 sha1_b = sha1(secret_key[4...5] + msg_key + secret_key[6...7]);
3 sha1_c = sha1(secret_key[8...11] + msg_key);
4 sha1_d = sha1(msg_key + secret_key[12...15]);
5 aes_key = sha1_a[0...3] + sha1_b[0...7] + sha1_c[4...7];
6 aes_iv = sha1_a[12...15] + sha1_b[12...19] + sha1_d[0...3];
```

Algorithme 4.1 : La fonction de dérivation de clé

Pour déchiffrer un paquet préalablement chiffré (identifié par un 1 sur le bit 1 de l'en-tête d'option) le receveur doit exécuter une suite d'opérations, résumées en Figure 4.3. En premier lieu, il doit récupérer la clé secrète utilisée pour réaliser toutes les opérations de chiffrement. Cette tâche est facilitée par la présence de l'empreinte de celle-ci en début du paquet. Ensuite, il doit utiliser cette clé pour vérifier le HMAC permettant de valider l'intégrité et l'authenticité du message. Cette opération se réalise

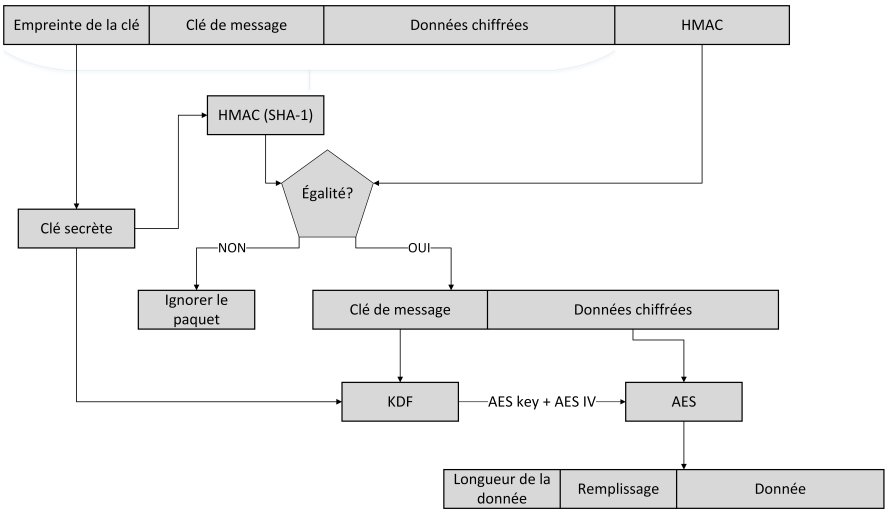

Figure 4.3 : Le processus de déchiffrement.

en recalculant le HMAC du paquet reçu (sans le HMAC) puis en comparant le résultat avec la signature reçue. Si l'opération échoue, le paquet doit être ignoré sinon, le processus de déchiffrement peut commencer. Tout d'abord, il faut recalculer la clé de chiffrement et l'IV en utilisant la fonction de KDF, la clé de message reçue et la clé secrète récupérée précédemment. Ensuite, on utilise l'algorithme AES-128 bits en mode déchiffrement pour récupérer le contenu clair du message. Pour finir, on utilise la taille du message encodée sur les deux premiers octets de celui-ci pour retirer les octets de remplissage. Ainsi, le receveur a pu déchiffrer le message et il peut désormais récupérer le sujet et la donnée en utilisant les étapes d'un message non chiffré présentées plus haut.

L'ENVOI DE DONNÉES COMPRESSÉES

Suivant la version du protocole utilisé (0 ou 1), deux modes d'envoi des données sont tolérés. Le premier, déjà présent en version 0 et déjà présenté dans un article publié

(Plantevin *et al.*, 2017) et plus haut, permet l'envoi de données brutes à travers le réseau. Le second, présent uniquement en version 1 du protocole, permet l'envoi de données compressées à travers le réseau. Nous allons maintenant expliquer le compression de ces paquets.

La Figure 4.1 d représente un message contenant une donnée compressée. De la même manière qu'une donnée standard, le premier octet représente les options du paquet. Les trois premiers bits doivent représenter un 1 puisque la compression n'est supportée que dans la version 1 du protocole. Ensuite, viennent les options avec les deux premiers bits toujours réservés et non-utilisés suivis du bit de compression, mis à la valeur 1, du bit de chiffrement mis à 1 ou 0 si la donnée en plus d'être compressée est chiffrée et pour finir le bit de découverte qui doit toujours être à 0 car le message de découverte ne supporte aucune compression pour des raisons de rapidité de réponse. Après les options vient l'en-tête de compression encodé sur un seul octet. Seuls les deux LSB sont utilisés dans cette version du protocole et représentent l'algorithme de compression utilisé. Si le LSB est mis à 1 l'algorithme est GZIP (Deutsch, 1996) et si c'est le bit précédent il s'agit de l'algorithme LZ4 (Collet, 2013). Le premier est un algorithme extrêmement connu déjà couramment utilisé dans le domaine du Web. Le second est un algorithme entièrement basé sur la vitesse de compression/décompression couramment utilisé dans les architecture distribuées comme Hadoop (Shvachko *et al.*, 2010) ou les bases de données comme MySQL (MySQL, 2001). Après cet en-tête de compression vient la longueur de la donnée compressée sur deux octets puis la donnée en elle-même. Pour finir, comme le paquet de donnée standard, une somme de contrôle vérifie l'intégrité du message. Quand un paquet compressé est reçu, la somme de contrôle doit être

vérifiée en premier. Si elle est bonne, le paquet peut être décompressé en utilisant le bon algorithme. Une fois décompressé, le receveur retire l'en-tête de compression et met le bit de compression des options à 0. Une fois cette tâche effectuée, le paquet peut être découpé comme un paquet normal.

FLUX DE TRAVAIL DE LNCF

Afin de résumer l'intégralité des différents modes et actions à réaliser pour interpréter un paquet reçu, nous présentons en Figure 4.4 l'algorigramme complet de notre protocole. Ainsi, après la réception du paquet, la somme de contrôle doit toujours être vérifiée en premier et il faut ignorer le paquet si celle-ci est incorrecte. Ensuite, il faut regarder les options et répondre en priorité aux requêtes de découverte puisque celles-ci ne peuvent être ni compressées ni chiffrées. Si l'on continue dans la branche principale, on peut voir qu'il faut ensuite vérifier l'état de compression du paquet et cela, car un paquet chiffré peut aussi être compressé. Suivant la décompression (ou non si le paquet n'était pas compressé) viennent la vérification du chiffrement et le déchiffrement du paquet s'il le faut. Pour finir, il est possible d'interpréter le sujet et sa donnée en utilisant les tailles fournies dans le paquet reçu.

4.2 TESTS ET DISCUSSION

Afin de valider le protocole proposé dans ce chapitre, nous avons conduit une série de trois tests. Le premier d'entre eux, vérifie les capacités en termes de bande passante et va maximiser le nombre de messages échangés pour démontrer la vitesse

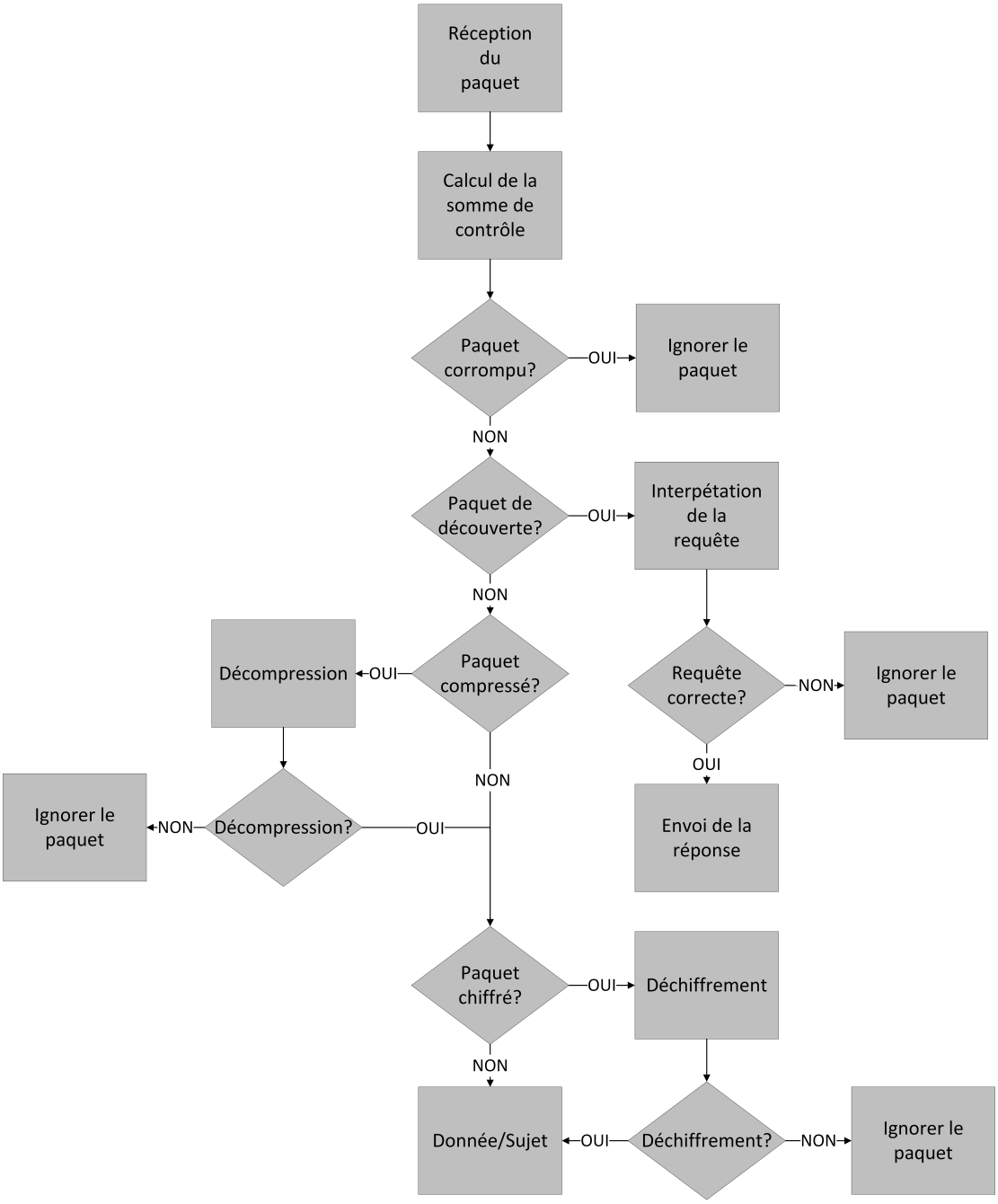

Figure 4.4 : L'algorigramme du protocole de messagerie de LNCF.

du protocole. Le second répond à une incertitude posée par l'utilisation du protocole UDP. En effet, ce dernier ne fournit aucun mécanisme de sécurité pour empêcher la perte ou corruption d'un paquet de données. Par conséquent, nous avons décidé de montrer dans les tests le nombre de paquets perdus ou mal ordonnancés à cause de ce manque. Pour finir, nous avons voulu prouver que même si nous utilisons la même clé de sécurité avec le même message, la donnée chiffrée reste totalement différente afin de prévenir les attaques sémantiques qui auraient pu survenir puisque nous ne générons pas aléatoirement l'IV pour le chiffrement AES. Pour valider cela, nous avons calculé la similarité entre plusieurs paquets chiffrés avec la même clé et contenant la même donnée. Le matériel utilisé pour ce test est un ordinateur portable (MSI GT62VR), une Rapsberry Pi 3 et une Raspberry Pi Zero W. Le premier est connecté à un routeur Gigabit (LinkSys WRT1900AC) par son port Ethernet et sa carte Wi-Fi (Wi-Fi en norme AC) alors que les deux autres uniquement par Wi-Fi. Concernant l'implémentation, celle-ci a été réalisée en C++ en utilisant les librairies Boost ASIO et Crypto++ qui ont respectivement permis d'uniformiser et faciliter le développement des couches réseau et cryptographique.

Dans le but de tester la bande passante de notre solution, nous avons généré 9 messages totalement aléatoires avec des tailles allant de 16 octets à 60 kibioctets (kio). Ensuite, nous avons attaché un processus écoutant le réseau sur la carte filaire de l'ordinateur portable afin de pouvoir contrôler les paquets transitant sur le réseau pendant que nous envoyions 20 000 paquets (10 000 chiffrés et 10 000 non) pour chaque taille différente depuis la carte Wi-Fi. Les résultats complets de ces tests sont présentés dans les Tables 4.3 à 4.5 qui représentent la durée, le nombre de messages par seconde et

enfin le débit en mébioctets (Mio) par seconde pour les 10 000 paquets chiffrés et non chiffrés en fonction de la taille de la donnée encapsulée et cela pour l'ordinateur, la Raspberry Pi Zero W et la Raspberry Pi 3. Chacun de ces tableaux est associé à un graphique (Figure 4.5 à 4.7) résumant les résultats en message par seconde pour le sous-graphe du haut et mébioctets par seconde pour le sous-graphe du bas.

La première observation qu'il est possible de réaliser est une perte importante de bande passante quand le message atteint les 2 kio et ce qu'il soit chiffré ou non. Avec respectivement 10.9, 13.1, 3.8, 4.6, 2.2 et 2.78 moins de paquets envoyés suivant la plateforme et le chiffrement, notre protocole semble avoir une limite en ce qui concerne l'envoi à haute fréquence (plus de 1000 par seconde) de longs messages. Nous avons cherché la cause de cette importante perte et après une étude des trames réseaux, la fragmentation des paquets Ethernet semble responsable de ce phénomène ce qui explique aussi pourquoi la vitesse reprend quand les messages deviennent significativement plus longs. En effet, la fragmentation intervient quand le paquet final fait plus de 1522 octets or, pour 2 kio (2048 octets) ce mécanisme n'est absolument pas performant, car le temps d'envoi du paquet est grandement inférieur aux temps de fragmentation et défragmentation de celui-ci. Cependant, pour des paquets de taille supérieure, le temps d'envoi du paquet dépasse le temps de reconstruction de celui-ci et par conséquent, la fragmentation redevient performante. La seconde observation que nous pouvons réaliser est assez triviale. Le chiffrement impacte la bande passante et ceci est dû au fait que des algorithmes de cryptographie (AES et SHA) viennent s'ajouter à l'envoi normal. Néanmoins cela n'impacte que très peu les paquets dont la taille ne dépasse pas 2 kio. Pour finir, une observation intéressante peut être tiré du graphique des résultats

de la Raspberry Pi Zero W, la plus légère des plateformes du test. Le fait que les deux courbes (messages chiffrés ou non) soient asymptotiques autour de la même valeur tend à prouver que notre protocole ne surcharge pas le processeur d'une telle entité, mais uniquement ses capacités réseau. Cette observation venant confirmer que notre protocole peut être utilisé sur de petites plateformes sans trop impacter leurs capacités de calcul.

Taille msg (o)	Non chiffrés			Chiffrés		
	Durée	Msg/s	Mio/s	Durée	Msg/s	Mio/s
16	0.086	116,279	1.86	0.149	67,114	1.07
32	0.088	113,636	3.64	0.151	66,225	2.12
128	0.084	119,048	15.24	0.161	62,112	7.95
512	0.090	111,111	56.89	0.194	51,546	26.39
1ki	0.094	106,382	106.38	0.226	44,248	45.31
2ki	0.187	53,475	106.95	0.311	32,154	65.85
10ki	2.041	4,899	48.99	4.060	2,463	25.22
30ki	3.529	2,833	84.99	8.823	1,133	34.82
60ki	6.989	1,430	85.80	15.188	658	40.45

Tableau 4.3 : Les résultats complets du test de bande passante sur l'ordinateur portable

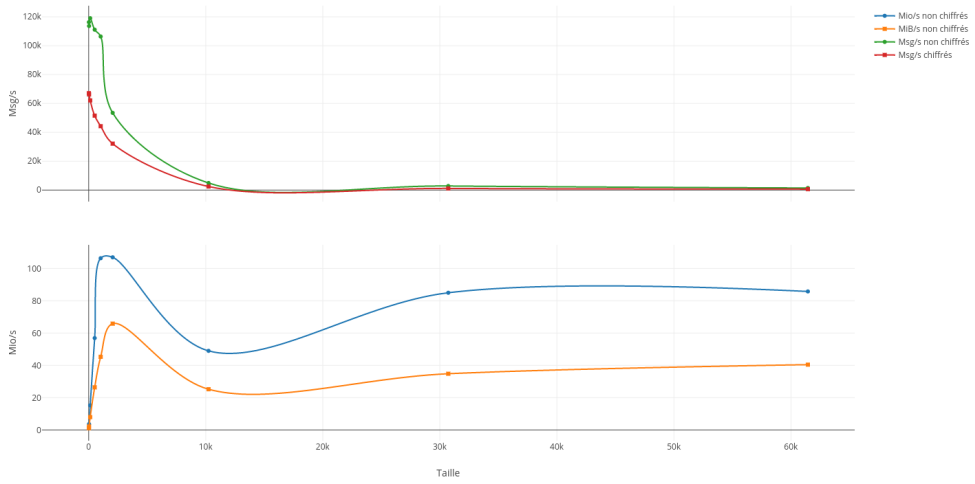

Figure 4.5 : Résultat de bande passante en msg/s et Mio/s pour 10,000 envois depuis l'ordinateur d'un paquet de taille variable en mode clair et chiffré

Taille msg (o)	Non chiffrés			Chiffrés		
	Durée	Msg/s	Mio/s	Durée	Msg/s	Mio/s
16	3.100	3226	0.05	3.661	2731	0.04
32	2.535	3945	0.13	2.930	3413	0.11
128	3.633	2753	0.35	3.307	3024	0.39
512	3.637	2750	1.41	4.985	2006	1.03
1ki	5.948	1681	1.72	6.075	1646	1.69
2ki	13.216	757	1.55	12.135	824	1.69
10ki	51.070	196	2.01	56.375	177	1.82
30ki	149.167	67	2.06	148.496	67	2.07
60ki	303.746	33	2.02	302.036	33	2.03

Tableau 4.4 : Les résultats complets du test de bande passante sur la Raspberry Pi Zero W

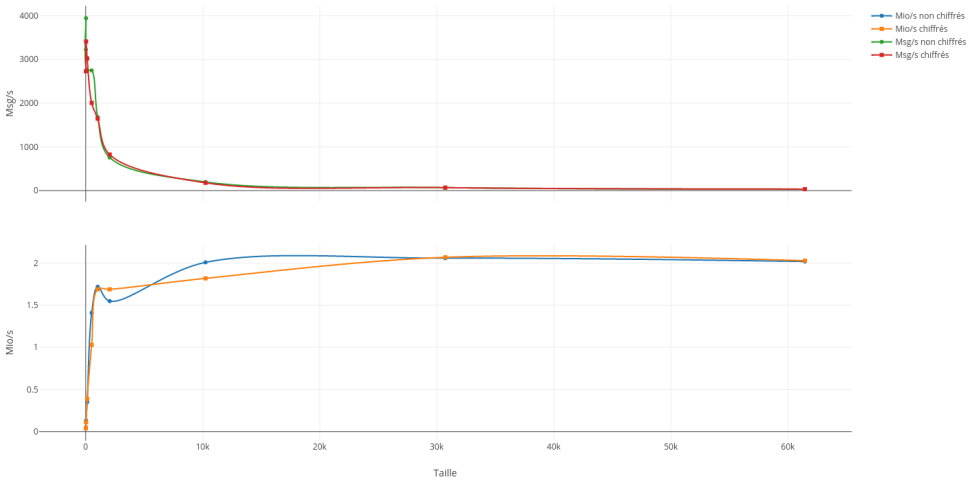

Figure 4.6 : Résultat de bande passante en msg/s et Mio/s pour 10,000 envois depuis la Raspberry Pi Zero W d'un paquet de taille variable en mode clair et chiffré

Taille msg (o)	Non chiffrés			Chiffrés		
	Durée	Msg/s	Mio/s	Durée	Msg/s	Mio/s
16	9.257	1080	0.02	12.042	830	0.01
32	4.950	2020	0.06	9.162	1091	0.03
128	6.819	1466	0.19	23.649	423	0.05
512	7.772	1287	0.66	13.443	744	0.38
1ki	5.990	1669	1.71	22.748	440	0.45
2ki	12.643	791	1.62	23.988	417	0.85
10ki	28.008	357	3.66	66.709	150	1.54
30ki	73.351	136	4.19	176.107	57	1.74
60ki	160.556	62	3.83	346.138	29	1.78

Tableau 4.5 : Les résultats complets du test de bande passante sur la Raspberry Pi 3

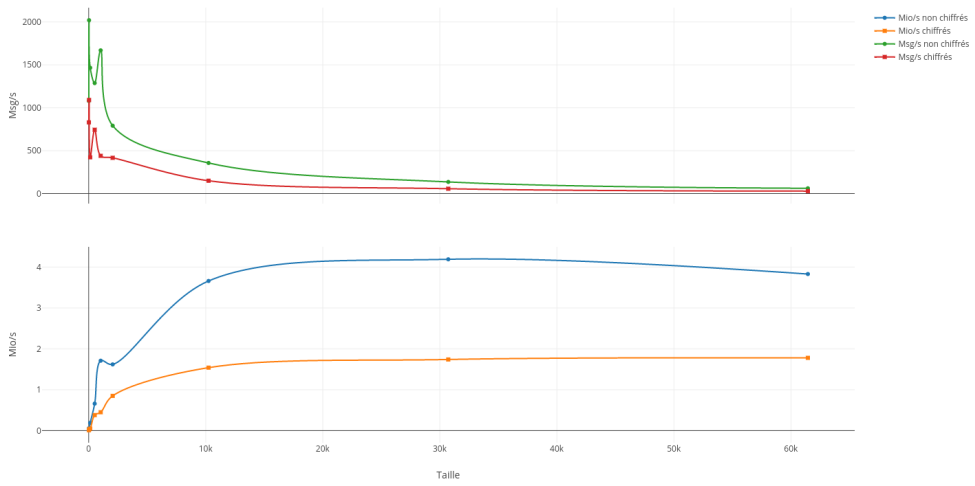

Figure 4.7 : Résultat de bande passante en msg/s et Mio/s pour 10,000 envois depuis la Raspberry Pi 3 d'un paquet de taille variable en mode clair et chiffré

UDP est un protocole ne garantissant aucune fiabilité de l'envoi. Cela signifie que des paquets peuvent être perdus ou corrompus et que le protocole ne fournit, de base, aucun mécanisme permettant de renvoyer ces paquets contrairement à TCP. Notre travail utilisant UDP, et ne fournissant pas ces mécanismes, nous voulions, au travers de ce test, montrer et quantifier le risque de perte de données. Pour cela, nous avons crée 10,000 paquets de 1 kio composés d'un numéro de séquence stocké sur 2 octets et d'octets de remplissage aléatoires. Ces paquets ont été envoyés depuis l'interface sans-fil des plateformes et récupérés par l'interface filaire de l'ordinateur qui contrôle l'arrivée des paquets (l'ordre, la corruption ou la perte). Les résultats de ces tests sont présentés dans la Table 4.6 où chaque ligne représente une plateforme différente (ordinateur, Raspberry Pi 3 ou Zero W) et où les trois colonnes représentent le nombre de paquets perdus, corrompus et finalement la quantité de problèmes d'ordonnancement des paquets. Comme nous pouvons le constater, sur le réseau de la maison nous

n'avons ni perdu ni reçu de paquets corrompus même si nous avons eu quelques problèmes de séquence (2 paquets sur 10 000 pour la Raspberry Pi 3 et l'ordinateur et 3 pour la Raspberry Pi Zero W). Cela signifie que sur 10,000 messages seuls 2 ou 3 arrivent dans le mauvais ordre et qu'aucun ne s'est perdu ce qui démontre clairement les capacités de notre protocole en termes de fiabilité au sein d'un réseau unique d'une maison intelligente.

Plateforme émetrice	Paquets perdus	Paquets corrompus	Problème d'ordonnancement
Ordinateur	0	0	2
Pi 3	0	0	2
Pi Zero W	0	0	3

Tableau 4.6 : Nombre de paquets perdus, corrompus ou mal ordonnancés pour 10 000 envois sur des plateformes différentes

Le dernier test que nous avons réalisé est un test de similarité. En effet, bien que nous assurons la confidentialité des messages par l'utilisation d'AES en mode chiffré, nous ne générons pas aléatoirement l'IV de celui-ci ce qui constitue pourtant une pratique sécuritaire. Au lieu de cela, nous utilisons un algorithme (KDF) présenté précédemment pour calculer l'IV à partir d'informations que les parties possèdent (hash du message et clé secrète). Or, le caractère aléatoire de l'IV et la seule chose garantissant la sécurité sémantique d'AES. Par conséquent, nous voulions vérifier que notre algorithme, utilisant des octets aléatoires de remplissage, offrait suffisamment d'aléatoire pour assurer la non-ressemblance des paquets chiffrés avec la même clé secrète et contenant la même donnée. Pour valider cette hypothèse, nous avons calculé la distance

euclidienne entre 1000 paquets sécurisés contenant le même message de 1kio de données aléatoires. Les résultats de ce test sont résumés dans la Table 4.7. Dans ce tableau, la première ligne représente la distance maximum théorique entre deux paquets. Cette dernière, utilisée comme référence pour les prochains calculs, est obtenue en calculant la distance entre un paquet ne contenant que des octets à 0 et un autre ne contenant que des octets à 255. Ensuite, en deuxième ligne, vient la distance moyenne entre tous les paquets. Avec une valeur de 3458.79 et un maximum et minimum (présentés en ligne 3 et 4) respectivement de 3775.54 et 3175.23 on peut avancer que notre algorithme garantit la non-similarité entre les paquets. Finalement, nous avons calculé le pourcentage de différence moyen entre tous les paquets et avec un résultat de 42.39 % pour des paquets contenant tous la même donnée, nous pouvons confirmer ce que nous disions précédemment, notre algorithme fournit suffisamment d'aléatoire dans les paquets chiffrés pour garantir la sécurité sémantique de ceux-ci. Par conséquent, les données chiffrées avec notre protocole semblent immunisées contre les attaques sémantiques ainsi que celles de similarité.

Variable	Valeur
Distance maximale théorique	8160
Distance moyenne	3458.79
Distance maximale	3775.54
Distance minimale	3175.23
Pourcentage de différence moyen	42.39%

Tableau 4.7 : Distance entre 1000 paquets contenant la même donnée et le même sujet et chiffrés avec la même clé secrète

4.3 CONCLUSION

Nous avons présenté au long de ce chapitre, une nouvelle façon de communiquer au sein d'un environnement intelligent. Notre solution répond d'abord à la grande différence qui existe entre configuration et flux de donnée en utilisant deux canaux différents. Le premier se base sur CoAP, un protocole connu du monde de l'IoT, pour garantir la fiabilité des informations de configuration. Le deuxième canal utilise un protocole que nous avons défini et tester ici. Ce dernier utilise UDP multicast pour fournir plusieurs points importants au sein d'un habitat intelligent. Ces derniers sont la capacité à découvrir les entités sur le réseau, la capacité d'envoyer des données, chiffrés ou non, à hautes fréquences et à tous les clients connectés au réseau et pour finir, la capacité de compresser certains messages, trop lourds pour l'infrastructure. LNCF est, au mieux de nos connaissances, un travail original qui apporte une innovation importante dans le domaine de la communication au sein des environnements intelligents. Et c'est à ce titre qu'il a fait l'objet d'un article scientifique paru dans un journal spécialisé (Plantevin

et al., 2017).

Pour finir, les trois tests présentés nous ont permis de valider plusieurs points importants de notre protocole. Le premier a permis de valider les capacités en termes de bande passante de notre solution. Les résultats démontrent la capacité d'envoyer un grand nombre de messages en un minimum de temps (environs 1000 par seconde dans le pire cas) même si on peut observer une grosse perte de performance pour les messages faisant 2 kio. Une autre conclusion tirée directement de ces résultats est que le chiffrement impacte très peu les débits finaux de notre solution. Le second test que nous avons réalisé nous a servi à assurer la fiabilité de notre protocole même si celui-ci se base sur UDP. Avec aucun message perdu ou corrompu et seulement 2 ou 3 problèmes d'ordonnancement sur 10,000 envois, le manque de mécanisme de fiabilité d'UDP ne semble pas impacter la fiabilité générale de notre solution. Par conséquent nous pouvons affirmer que notre avancée est suffisamment fiable pour envoyer des données au sein d'une maison intelligente. Pour finir, le dernier test que nous avons réalisé est la comparaison de la similarité entre 1,000 paquets chiffrés avec la même clé secrète et encapsulant la même donnée. Avec une différence moyenne de 42.39 % des paquets, nous pouvons affirmer que malgré le fait que nous ne générons pas l'IV d'AES totalement aléatoirement, notre solution reste immunisée contre les attaques de sémantiques et de similarité qui auraient pu être exécutées si notre algorithme ne fournissait pas assez d'aléatoire dans l'IV.

CHAPITRE V

VERS UNE NOUVELLE ARCHITECTURE DE MAISON INTELLIGENTE

Les premiers chapitres de cette thèse nous ont montré que beaucoup d'infrastructures existantes comportaient certaines faiblesses architecturales, notamment concernant la fiabilité et les capacités de mise à l'échelle. Dans le chapitre précédent, nous avons introduit rapidement le besoin d'un nouveau type d'architecture pour les environnements et avons répondu au premier défi rencontré dans cette réalisation avec l'introduction d'un nouveau protocole de communication, créé uniquement pour cela. Il convient désormais de présenter la nouvelle architecture qui constitue un pan majeur de cette thèse et, au mieux de nos connaissances, une nouveauté dans le domaine de la maison intelligente qui s'était, jusqu'à présent, uniquement concentré sur la reconnaissance d'activités. Cette partie est le principal sujet discuté dans un article soumis au « Journal of Ambient Intelligence and Humanized Computing » (Plantevin *et al.*, 2018b).

5.1 ARCHITECTURE PROPOSÉE

La contribution majeure présentée dans ce chapitre est un nouveau type d'architecture pour les environnements intelligents. Celle-ci est résumée en Figure 5.1 qui nous servira de guide pendant toutes les explications.

Une des façons d'améliorer la fiabilité d'une infrastructure est d'en retirer tous les points chauds (Single Point of Failure (SPoF)). Afin de faciliter la compréhension

d'une telle notion, nous allons prendre l'exemple du LIARA. Dans cette infrastructure, les SPoF étaient les îlots, l'automate et le serveur, car quand ces derniers cessaient de fonctionner, c'était une grande partie, si ce n'est l'entièreté, de l'infrastructure qui arrêtait de fonctionner par la même occasion. Afin de supprimer ces entités particulières, il convient d'amener une nouvelle façon de récolter, envoyer et traiter les informations venant des capteurs. Pour cela, nous avons décidé de mettre à jour les capteurs en les faisant évoluer de simples transducteurs passifs à des transducteurs intelligents, dotés de capacités de communication et représentés en partie **A** et **D** de la Figure 5.1.

La définition d'un transducteur intelligent a déjà été clairement définie dans la norme IEEE 1451.4 (Mark, 2004). Pour résumer cette notion, il est possible de dire qu'il s'agit d'une entité qui fournit plus de fonctionnalités que celles nécessaires pour générer une bonne représentation de la grandeur mesurée. Ces caractéristiques supplémentaires peuvent être l'identification des transducteurs, un processus simplifiant l'installation et la maintenance, des interfaces de communication réseau ou des mécanismes de coordination et de synchronisation entre les entités (Lewis, 2005).

Afin de simplifier la compréhension, les tests, développements et adaptations de notre solution, nous avons divisé notre transducteur intelligent en deux parties distinctes. La première d'entre elles est l'unité chargée de ressentir et d'agir sur l'environnement que nous appelons unité passive. Elle permet de faire l'interface entre le monde extérieur (e.g. température, contrôle des lumières, API Twitter, etc.) et le monde des données de haut-niveau pour l'intelligence (e.g. température de 25.3°C, lumière allumée ou il y a eu 26 retweet de votre statut). La seconde moitié des transducteurs

intelligents présentés ici est l'unité intelligente, responsable des communications au sein de l'environnement, la programmation des unités passives qui lui sont associées ainsi que des décisions prisent à partir des valeurs envoyées par celles-ci et des messages des autres transducteurs. La communication entre ces entités est assurée de deux façons différentes suivant que l'unité passive soit matérielle (i.e. connectée à des capteurs matériels comme un thermomètre) ou logicielle (i.e. un programme qui génère les données depuis une API). Dans le premier cas, un lien série entre le matériel et l'unité intelligente assure la communication alors que dans le second cas c'est un mécanisme de communication utilisant ZeroMQ (ZMQ) (ZeroMQ, 2016). Tous ces éléments sont répertoriés dans la Figure 5.1. Nous allons maintenant expliquer en détail chaque partie de cette architecture en commençant par l'unité intelligente suivie de l'unité passive et pour finir nous expliquerons comment les transducteurs peuvent communiquer entre eux avec une haute fiabilité.

5.1.1 UNITÉ INTELLIGENTE

Représentée en partie **C** de la Figure 5.1, l'unité intelligente est l'entité de calcul principale de notre transducteur le rendant intelligent. Ses deux tâches principales sont de communiquer avec le reste de l'environnement et de réaliser les calculs de l'intelligence artificielle à partir des valeurs données par les unités passives. De plus, elle agit comme agent facilitateur de l'installation et de la maintenance des différents types de matériels que l'on retrouve dans une maison intelligente en permettant de programmer et mettre à jour les pilotes embarqués dans les unités passives matérielles et les logiciels

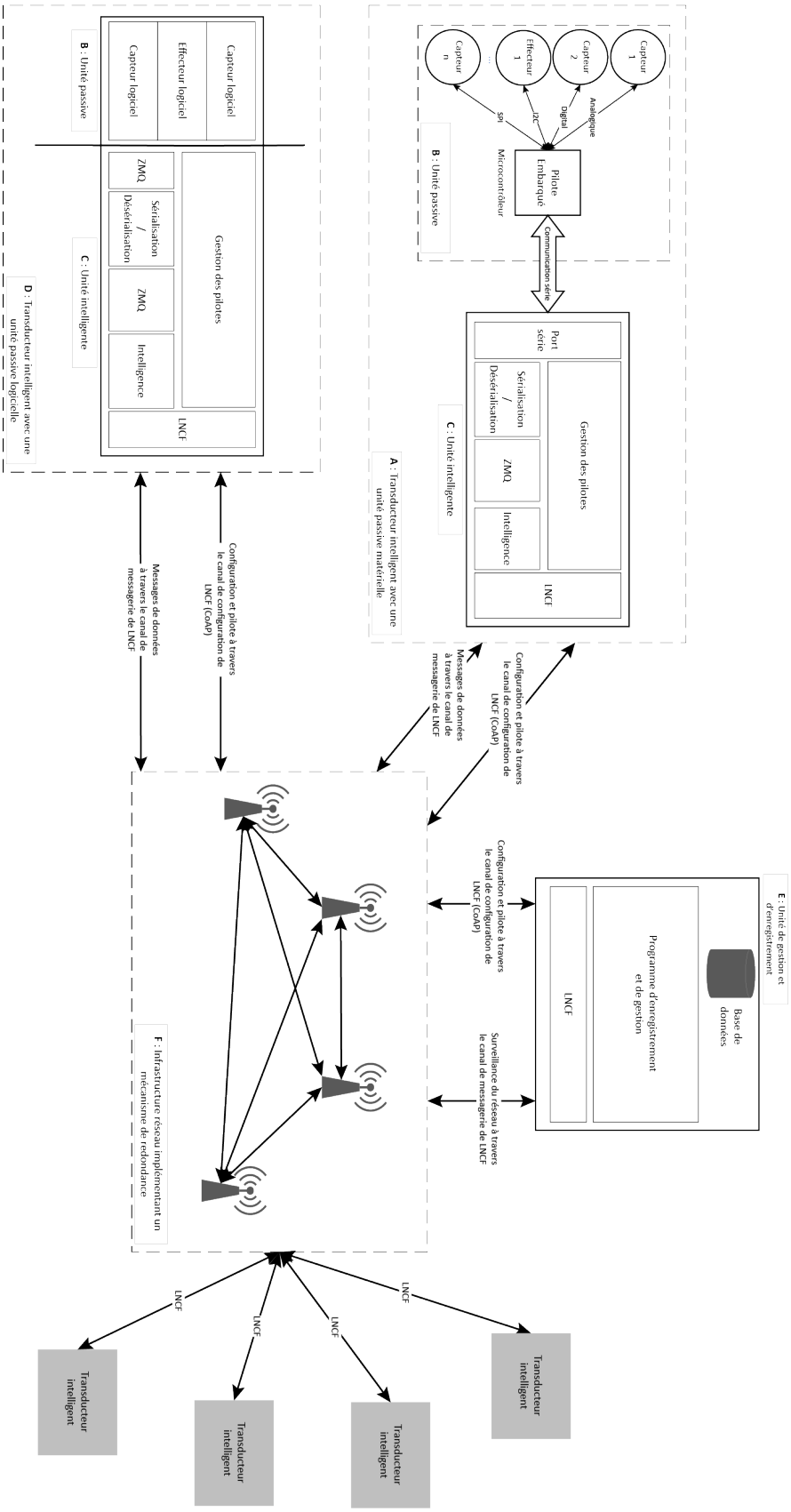

Figure 5.1 : L'ensemble de l'architecture présentée dans ce chapitre.

représentant les transducteurs virtuels.

Les pilotes des entités sont représentés par une archive Zip contenant un fichier de description au format JSON, appelé *driver.json*, et les fichiers nécessaires au fonctionnement des pilotes (e.g. fichiers sources, configuration, etc.). Le schéma du fichier de description est représenté en Figure 5.2 et contient un tableau, appelé *drivers*, et un objet appelé *intelligence*. Le tableau contient l'ensemble des pilotes spécifiques aux unités passives connectées à l'unité intelligente. Chaque pilote d'unité passive est composé de quatre éléments principaux. Le premier est le type d'unité passive soit matérielle (HARDWARE dans la description) ou logicielle (SOFTWARE dans la description). Le second, dénommé *installation*, est un tableau contenant une suite de commandes permettant d'installer le pilote en question. Ces commandes sont exécutées dans l'ordre par le système de l'entité intelligente et doivent permettre, au final, de programmer le matériel ou d'installer les logiciels nécessaires au fonctionnement du pilote. Le troisième élément présent dans la description d'un pilote représente la communication entre l'unité intelligente et l'unité passive. Dans le cas où cette dernière est matérielle, il s'agit de la description complète du lien série (port, baudrate, bits de donnée, bits d'arrêt, parité et contrôle du flux). Dans le cas où il s'agit d'une unité passive logicielle, cet élément représente la configuration ZeroMQ nécessaire pour communiquer soit le transport utilisé (inter-processus, flux TCP ou multicast) et l'extrémité réseau correspondant. Pour finir, le dernier élément d'un pilote est la description des données qu'il expose. Il s'agit d'un tableau composé d'un objet par donnée envoyée contenant un identifiant unique (ici *id*) représentant la place de la donnée dans le tableau envoyé par l'unité passive, un nom pour faciliter la lecture humaine des ensembles de données

et déserialiser la donnée vers une représentation JSON de plus haut niveau et un type suivant que la valeur est continue (e.g. une température) ou nominale (e.g. la lumière est à ON ou OFF). En plus de ces trois champs de base, viennent s'ajouter deux champs optionnels, le premier est présent uniquement si la valeur est nominale et contient l'ensemble des valeurs possibles. Le second est un booléen permettant de savoir si cette donnée est associée à un effecteur ou pas (par défaut cette valeur est à faux).

Pour finir la description complète du fichier de pilote, l'objet *intelligence* contient les paramètres permettant d'installer et de communiquer avec l'intelligence artificielle présente dans le programme principal. Cette communication est assurée par un pont ZMQ et va servir à transférer les données déserialisées afin que celles-ci soient traitées. En effet, le problème qui se pose avec la gestion de plusieurs unités passives est la synchronisation temporelle des différentes données en provenance de celles-ci. Pour répondre à ce défi, nous avons introduit un bloc logiciel avant l'intelligence artificielle chargé de déserialiser et synchroniser les différentes données avant de les transmettre pour traitement ultérieur. Ainsi, à l'arrivée de données en provenance des unités passives, sous la forme d'un tableau utilisant le séparateur point-virgule (SSV), ce logiciel déserialise la donnée grâce au fichier de description et attend que toutes les données soient reçues de toutes les entités. Une fois cela fait, il transmet toutes les valeurs au format JSON au logiciel principal (responsable de l'intelligence) via une connexion ZMQ.

5.1.2 UNITÉ PASSIVE

Représentée en partie **B** de la Figure 5.1, une unité passive n'est rien d'autre qu'un matériel ou un logiciel permettant de générer des données de haut-niveau à partir d'évènements de bas-niveau et ce pour faciliter la prise de décision faite par l'intelligence artificielle localisée dans l'unité intelligente.

Dans le cas où cette unité est matérielle, il s'agit d'un microcontrôleur connecté à des capteurs et effecteurs matériels et embarquant un logiciel défini et installer par l'unité intelligente et appelé pilote. Celui-ci lit et interprète les valeurs des différents capteurs matériels (e.g 523 pour le thermomètre) et envoie une représentation de haut-niveau (e.g. 23.5°C) au format SSV à travers une communication série à l'unité intelligente. En plus d'un microcontrôleur, l'unité passive doit embarquer un moyen de reprogrammer celui-ci afin que l'unité intelligente puisse facilement changer le pilote.

En ce qui concerne les unités passives logicielles, il s'agit de services s'exécutant dans l'entité intelligente et générant des données de haut niveau à partir d'API existantes (e.g. Twitter, état des composants, etc.). Surtout utilisées dans notre cas pour des fins de tests, elles permettent néanmoins de consommer des données venant d'Internet si le transducteur est connecté à celui-ci. Ceci permettant d'obtenir des données extérieures à la maison comme, par exemple, la localisation des habitants de celle-ci depuis leur téléphone intelligent et de les intégrer dans le processus de décision. Ces unités passives communiquent avec les composants de plus hauts nivaux au travers d'un pont ZMQ.

5.1.3 ENTITÉ DE GESTION

Lors du premier démarrage d'un transducteur intelligent, celui-ci génère un identifiant unique au format UUID représenté par une chaîne de caractères. Cet identifiant unique va lui permettre de s'enregistrer auprès d'une entité centralisée représentée dans la partie **E** de la Figure 5.1.

Cette entité doit être la plus légère possible (e.g. une Raspberry Pi ou un téléphone) et son unique tâche est de garder une liste des transducteurs présents dans l'environnement. Il est important de noter que cette entité n'est en aucun cas requise pour le fonctionnement général de l'environnement. Elle permet simplement d'émettre des alertes si un des transducteurs arrête de communiquer, de garder, dans une base de données, l'association transducteur/pilote et peu offrir une visualisation aidant le développement (nombre de messages échangés, localisation des autres entités, envoi automatique des pilotes, etc.). Par conséquent, cette entité est nécessaire uniquement lors de l'installation de nouveaux transducteurs ou quand il faut changer les pilotes contenus dans une ou plusieurs autres entités. Théoriquement parlant, ce serveur peut être hébergé dans le cloud permettant de garantir une haute fiabilité. Néanmoins, comme la plupart des maisons intelligentes fonctionnent sur un réseau fermé pour des raisons de sécurité, il est conseillé de l'héberger localement.

5.1.4 RÉSEAU ET COMMUNICATION

Tous nos transducteurs implémentent au moins une pile IP et nous permettent donc de les connecter à un réseau implémentant une forme de redondance, représenté

en partie **F** de la Figure 5.1. Malgré cela, la communication est loin d'être triviale, car la plupart des mécanismes existants introduisent des SPoF dans les infrastructures. Pour répondre à cette problématique, nous avons décidé d'utiliser le protocole LNCF présenté en Chapitre 4.

À l'instar de FTP, et comme montré précédemment, LNCF utilise deux canaux principaux pour faire transiter les informations sur le réseau. Le premier est un canal de configuration et implémente une grande fiabilité des données en utilisant le protocole CoAP déjà largement utilisé dans l'industrie et plus spécifiquement dans le domaine de l'IoT. Ici, nous utilisons cette caractéristique pour implémenter l'échange des pilotes entre l'entité de gestion et les différentes entités intelligentes. Ainsi, ces dernières définissent trois terminaisons CoAP particulières. Les deux premières sont en lecture seule et permettent à n'importe qui de récupérer la version et l'identifiant du transducteur (nommées respectivement *version* et *id*). La dernière est le point terminal permettant de téléverser le fichier Zip représentant le pilote. Par conséquent, celui-ci est en lecture seul et attend un fichier binaire. Pour finir, le pilote pouvant être assez volumineux, il doit implémenter le *Block-Wise transfer*, permettant de transférer de manière fiable des fichiers volumineux en utilisant CoAP.

Le deuxième canal de LNCF est un protocole de messagerie utilisant le multicast UDP comme vecteur de propagation. Cette caractéristique permet de réaliser un protocole de messagerie Pub/Sub sans avoir aucun serveur centralisé (appelé broker) qui pourrait constituer un SPoF réduisant la fiabilité de toute l'infrastructure à la fiabilité de ce point unique. De plus, le canal de données de LNCF, met à disposition quatre

modes principaux permettant de découvrir les instances présentes sur le réseau, échanger des messages à très haute fréquence, chiffrer les messages sensibles et compresser ceux qui sont trop larges pour être échangés. Le premier mode nous permet de facilement implémenter la découverte de l'entité de gestion puisque le nouveau transducteur n'a qu'à faire une requête pour savoir à quelle IP est implémenté le service de gestion (appelé *MANAGEMENT_SERVICE* dans notre implémentation). Le deuxième mode est une obligation dans le domaine du Big Data, domaine auquel appartient la maison intelligente. Bien qu'il n'implémente aucun mécanisme garantissant la fiabilité de l'information, il nous permet d'échanger des messages avec une latence très faible, une très haute fréquence et surtout à tout l'environnement en même temps. Par conséquent, nous utilisons ce mode de transmission pour la plupart de nos messages et nous avons choisi d'implémenter trois sujets en particulier qui sont : *ACTIVITY*, *ACTUATORS* et *DATA*. Le premier fait transiter les activités reconnues par les différents transducteurs. Le deuxième permet de demander l'activation d'un effecteur en utilisant le dénomination : $ID_{transducteur}\_ID_{effecteur} : STATE$ ou l'état(STATE) est une chaîne de caractères comprise par le pilote de l'effecteur (e.g. ON pour une lampe). Enfin, le troisième, implémenté en laboratoire, permet de créer des jeux de données de test en centralisant tous les évènements qui apparaissent dans l'environnement. Les deux derniers modes de communication de LNCF ne sont, pour le moment, pas utilisés dans notre implémentation, mais pourraient parfaitement être activés à volonté pour garantir plus de sécurité dans les échanges ou moins de charge réseau.

5.2 TESTS ET DISCUSSION

Pour valider notre architecture, nous avons conduit une série de tests dessus. Tout d'abord, nous avons vérifier la latence de celle-ci afin de nous assurer qu'elle est au moins aussi bonne que notre ancienne architecture qui possédait une latence cumulée se situant entre 250 ms et 1 s. De plus, pendant la conduite de ce test, nous avons augmenté graduellement le nombre d'entités pour confirmer les capacités de mise à l'échelle de notre solution. Ensuite, nous avons exécuter une série de manipulations pour s'assurer de la fiabilité, qualité mise en avant durant ce chapitre. Pour finir, nous avons noté la somme de tous les coûts permettant de construire une telle infrastructure et ce dans le but de nous comparer à CASAS et à notre ancienne solution. Mais avant tout, il convient d'expliciter le materiel que nous avons utilisé pour réaliser ces tests.

5.2.1 MATÉRIEL ET INFRASTRUCTURE UTILISÉS

Afin de réaliser les tests précités, nous avons dû, en premier lieu, implémenter les transducteurs intelligents tel qu'expliquer dans ce chapitre. Ensuite, nous avons dû fournir une infrastructure réseau fiable leur permettant de communiquer. Nous allons tout d'abord expliquer comment nous avons implémenté notre solution puis nous décrirons l'infrastructure réseau utilisée pour réaliser nos tests.

Nous avons choisi un matériel Open Source pour implémenter notre solution, et ce pour que nos tests soient le plus reproductibles possible. Ainsi, les unités intelligentes ont été implémentées sur des Raspberry Pi Zero W possédant suffisamment de General Purpose Input Output (GPIO) pour communiquer avec plusieurs unités pas-

sives matérielles tout en nous fournissant suffisamment de puissance de calcul pour implémenter des comportements intelligents par la suite. Pour ce qui est des unités passives matérielles, nous avons utilisé un ESP32 Thing de l'entreprise Sparkfun. Nous l'avons choisi, car il embarque suffisamment d'entrées/sorties pour être intéressant en termes de quantité de capteurs, suffisamment de puissance de calcul pour traiter ces informations et qu'il est totalement compatible avec la plateforme Arduino, facilitant nos développements. Il a été connecté à la Raspberry Pi Zero W à travers le port USB de celle-ci et a été programmé, grâce au pilote, pour générer aléatoirement entre 5 et 20 capteurs générant des données aléatoires à exactement 2 kHz, ceci nous permettant de simplifier les développements puisque, pour nos tests, les vraies valeurs de capteurs ne sont pas intéressantes. Le problème qui s'est posé et que nous n'avions pas assez d'ESP32 pour chaque Raspberry Pi. Par conséquent nous avons testé les unités passives matérielles sur 5 d'entre elles et nous en avons implémenté 25 logicielles pour un total de 30 transducteurs intelligents. Les unité passives logicielles s'échelonnant de la génération de simples valeurs aléatoires à la consommation de flux en provenance de l'API Twitter pour les plus demandant d'entre elles. Nous pensons que ces 30 transducteurs intelligents constituent un premier test intéressant des capacités de notre architecture même si seulement 5 d'entre eux implémentent des unités passives matérielles. L'implémentation générale a été réalisée en C++ avec l'aide des librairies Boost.

Suivant l'implémentation vient l'architecture. Celle-ci est composée de deux routeurs Wi-Fi (LinkSys WRT1900AC) reliés entre eux par un câble Ethernet et configurés en service de distribution sans-fil (Wireless Distributed Services). Ce mode particulier

consiste à créer deux points d'accès sans-fil sur le même réseau permettant aux différents transducteurs intelligents de se connecter sur celui ayant le plus fort signal. De plus, un ordinateur portable (MSI GT62VR) et une Raspberry Pi 3 sont connectés au réseau par le Wi-Fi. La Raspberry sert d'entité de gestion à laquelle les différents transducteurs viennent s'enregistrer durant la phase d'installation et l'ordinateur représente un scientifique désirant accéder aux données de l'infrastructure et mesurant la latence entre l'envoi de la donnée et la réception de celle-ci. Cette infrastructure est représentée en Figure 5.3. Bien entendu et puisque nous partageons la même infrastructure de base, toutes les applications qui communiquent sur notre réseau ont temporairement été coupées afin que les communications de notre ancienne architecture ne viennent pas impacter nos résultats.

5.2.2 TESTS DE LATENCE ET DE MISE À L'ECHELLE

Normalement, les transducteurs intelligents ne font pas transiter les données brutes des capteurs sur le réseau. En effet, ils disposent tous de suffisamment de puissance de calcul pour les traiter et même prendre des décisions intelligentes à partir de celles-ci. Néanmoins, pour les besoins de ce test, et parce qu'il est possible qu'un laboratoire veuille générer des ensembles de données brutes grâce à notre infrastructure, les unités intelligentes sont configurables afin de leur faire relayer les données brutes en les associant avec le temps précis de l'envoi, le tout au format JSON, sur le canal de messagerie du protocole LNCF sous le sujet *DATA*. L'ordinateur portable récupère toutes ces données et utilise le temps de chacune pour les réordonner en un seul ensemble

de données tout en calculant la latence de chacun des paquets afin de voir quel taux d'échantillonnage notre solution peut atteindre et quel est le délai moyen entre la génération d'une donnée et sa réception. De plus, durant toute le durée de ce test, nous avons ajouté à intervalle régulier (toutes les minutes) de nouveaux transducteurs intelligents pour voir comment notre solution se met à l'échelle. Nous avons commencé avec les 5 transducteurs ayant des unités passives matérielles puis nous les ajoutions 5 par 5 jusqu'à arriver à 30 transducteurs intelligents que nous avons laissés fonctionner 1 minute. Parallèlement à ces ajouts, nous augmentions à chacune des étapes la fréquence d'échantillonnage la faisant passer graduellement de 1 Hz à 1 kHz. Ces résultats, combinés avec la latence moyenne de notre solution, nous donnant une bonne idée des capacités de mise à l'échelle de l'infrastructure proposée. Pour finir, et ce pour ajouter des métriques à notre test, nous avons décidé de rapporter ici les pourcentages d'utilisation des deux points d'accès et de la carte Wi-Fi de l'ordinateur dans le meilleur (5 transducteurs à 1 Hz) et le pire cas (30 transducteurs à 1 kHz).

Nombre de transducteurs	1 Hz	10 Hz	100 Hz	500 Hz	1 kHz
5	3.0 ms	2.7 ms	3.1 ms	3.1 ms	2.9 ms
10	2.8 ms	2.6 ms	3.4 ms	2.6 ms	3.2 ms
15	3.1 ms	2.4 ms	2.6 ms	2.8 ms	2.6 ms
20	3.1 ms	3.3 ms	3.6 ms	3.5 ms	2.4 ms
25	3.2 ms	5.1 ms	2.3 ms	2.2 ms	2.8 ms
30	2.9 ms	2.6 ms	2.2 ms	3.4 ms	2.9 ms

Tableau 5.1 : Résultats des tests de latence et de mise à l'échelle de notre solution.

Les résultats de notre test sont présentés dans la Table 5.1. Dans celle-ci, chaque ligne représente le nombre de transducteurs intelligents présents dans notre architecture pendant le test. Les colonnes, quant à elle, représentent les fréquences d'envoi de données telles que configurées dans les transducteurs via le canal de configuration de LNCF. La latence moyenne de notre infrastructure, définie comme étant la différence entre le temps de réception et le temps d'émission, est reportée dans chacune des cases du tableau suivant le nombre d'entités et leur fréquence d'envoi, le tout sur 1 minute d'enregistrement. Comme il est possible de le constater dans le tableau, la plupart des valeurs sont proches, et ce même quand notre solution doit supporter 30 entités à 1kHz. Cette première conclusion est confirmée quand on calcule la moyenne et l'écart-type de toutes ces latences puisque la première est de 2.95 ms pour seulement 0.55 ms d'écart-type. Nous avons investigué afin d'expliquer cette faible variation et nous avons trouvé la solution dans les taux d'utilisation des infrastructures réseau. Avec un pourcentage d'utilisation s'échelonnant de 0.5 % à 40.1 % pour la carte réseau du portable et de 0.2 % à 35.6 % pour les points d'accès, nous sommes loin des capacités maximales du réseau et ces charges ne constituent aucun défi pour lui.

Ces premiers résultats nous permettent de faire trois conclusions différentes. La première est que notre solution présentée ici nous permet de récupérer les valeurs de tous les transducteurs à une fréquence de 1 kHz, et ce avec une simple latence de 2.95 ms en moyenne. Ce premier point constitue une excellente amélioration de notre ancienne architecture qui, au meilleur cas, nous permettait de récupérer ces valeurs toutes les 250 ms (soit 4 Hz). Ensuite, la relativement faible latence réseau et les taux d'utilisation des équipements réseau nous permettent de conclure que même avec

30 transducteurs à 1 kHz, nous sommes encore loin de surpasser les capacités réseau de notre infrastructure, et ce même si celle-ci est relativement simple avec seulement deux routeurs que l'on peut déjà trouver dans des maisons de nos jours. Pour finir, il convient de nuancer ces résultats en précisant qu'il convient d'exécuter plus de tests avec plus d'entités pour voir comment notre solution se comporte quand on approche les 100 % d'utilisation du réseau.

5.2.3 TESTS DE FIABILITÉ

Nous avons écrit que notre architecture est plus fiable et nous nous devons de confirmer cette affirmation. Par conséquent nous avons exécuté différents scénarii sur celle-ci pour voir ses réactions. Le premier est l'extinction d'un des deux routeurs (nous lui avons coupé le courant). Le second scénario est la suppression de l'entité de gestion implémentée ici sur la Raspberry Pi 3 et chargée de surveiller l'infrastructure. Pour finir, nous avons jouer un scénario catastrophe en coupant tous les disjoncteurs de la maison les uns après les autres avec une minute de délai entre chaque le tout en finissant par l'un des routeurs afin de pouvoir ensuite tous les rallumer dans le sens inverse. Tous ces tests ont été réalisés avec 30 entités envoyant leurs données à 1 kHz.

Durant le premier test, où nous éteignions un des routeurs, nous avons immédiatement perdu 17 transducteurs intelligents. Après 19 secondes, 10 d'entre eux étaient de nouveau en ligne et capable d'envoyer leurs données et il a fallu un total de 32 secondes pour retrouver un réseau complet. Après ce temps, le routeur restant était à 72.6 % d'utilisation. Une fois le deuxième routeur redémarré, il a fallu 5 minutes et

35 secondes pour que le réseau se stabilise (c.-à-d. plus aucun transducteur, ne changeaient de point d'accès). Le second test réalisé est plutôt simple, après avoir éteint l'entité de gestion nous n'avons vu aucune modification dans le réseau, mais nous avions perdu la possibilité de changer les pilotes des entités ainsi que celle d'ajouter des entités. Ces deux capacités sont revenues dès le retour de l'entité. Pour finir, quand nous avons coupé tous les disjoncteurs de la maison, nous avons graduellement perdu toute l'infrastructure puisqu'aucun transducteur n'est autonome en énergie. Néanmoins, lors de la remise en route, nous avons très vite retrouvé l'entièreté du réseau entièrement fonctionnel.

Ces différents tests ont forcé notre solution à montrer ses capacités en termes de fiabilité et d'autorétablissement. Le premier, où nous avons éteint un des routeurs, ce qui peut facilement arrivé au sein d'une maison habitée, nous a montré que l'infrastructure était capable, en une trentaine de secondes, de redevenir entièrement opérationnelle. Ce résultat est assez impressionnant si l'on considère que la moitié de son réseau était éteint. Le deuxième test a confirmé que l'entité de gestion n'était pas nécessaire pour que notre solution fonctionne, et ce bien qu'elle aurait pu être le seul SPoF. Néanmoins, cela nous a montré aussi que nous perdions toute possibilité d'ajouter ou modifier un transducteur intelligent. Afin d'être totalement fiable, il convient d'apporter une solution à ce problème avec peut-être l'utilisation de plusieurs entités créant ainsi une forme de redondance. Pour finir, le dernier test a montré que même après une extinction totale, notre architecture est totalement capable de redémarrer d'elle-même sans aucune intervention humaine et comme une perte totale de courant est possible dans une maison il s'agit là d'une fonctionnalité obligatoire.

5.2.4 PRIX DE NOTRE SOLUTION

Pour finir ces résultats, nous avons estimé le coût total, en dollars américains, de notre solution. Le détail complet de ce chiffrage est présenté en Table 5.2. Dans ce dernier, chaque ligne représente un élément de notre solution associé à la quantité nécessaire pour réaliser nos tests ainsi que son prix unitaire et le prix total qui correspond au prix unitaire multiplié par la quantité. La dernière ligne représente le prix total correspondant à la somme de tous les totaux de chacun des éléments. Comme il est possible de voir, notre solution coûte un total de 1948.5 USD. Ce chiffre constitue une excellente amélioration par rapport à notre ancienne solution dont les coûts estimés s'élevaient à 13,500 USD, et ce sans infrastructure réseau. De plus, il convient de préciser que les coûts des Raspberry Pi Zero W rapportés ici sont ceux des ensembles de développement fournit par Adafruit et non pas ceux des Raspberry Pi en elles-mêmes qui sont à 5 USD pièce. Nous avons choisi ce prix plutôt que le prix réel, car nous nous sommes servi de beaucoup d'éléments du kit pour réaliser nos tests, mais ce prix peut être grandement et facilement réduit si on ne compte que les éléments vitaux (la Raspberry, la source d'énergie et la carte SD).

Les coûts à fournir par an pour une personne de plus de 65 ans atteinte d'Alzheimer ou d'autre forme de démence sont estimés à 46,786 USD aux États-Unis dont une moyenne de 10,315 USD doit être fournie directement par le malade si celui-ci bénéficie d'assurances et d'aides gouvernementales (Alzheimer's Association, 2017). Parallèlement, le bureau du recensement américain estime que le revenu médian pour les personnes de plus de 65 ans en 2016 est de 47,432 USD (Bureau, 2016). Par conséquent,

Élément	Quantité	Prix unitaire	Total
Routeur sans-fil	2	140 $	280 $
Pi Zero W	30	35.50 $	1035 $
ESP32 Thing	30	19.95 $	598.5 $
Pi 3	1	35 $	35 $
TOTAL			1948.5 $

Tableau 5.2 : Le prix total de notre infrastructure en dollars américains.

il est important que le prix d'une solution de maison intelligente pour ces personnes ne soit pas trop élevé, car ils ont déjà énormément d'autres coûts médicaux, pas toujours remboursés, pour un revenu amoindri. Avec un prototype coûtant moins de 2000 USD, nous pensons que cet objectif est partiellement atteint. En effet, ce prix final est moins élevé que celui de CASAS (2,765 USD) pour plus de fiabilité, de capacité de calcul et de capteurs, mais il ne contient pas le prix des capteurs et effecteurs matériels (e.g. un thermomètre, un contacteur de porte). Bien que ces éléments sont maintenant relativement peu coûteux et ne devraient par conséquent pas trop augmenter notre estimation, nous nous devons de les prendre en compte dans une analyse complète. Mais il convient de rappeler que nos coûts sont aussi ceux d'une version prototype dont le matériel n'a pas été acheté en grande quantité et nous pensons donc que notre prix annoncé peut aussi être grandement réduit si une version industrielle devait être faite.

5.3 CONCLUSIONS

Tout au long de ce chapitre, nous avons présenté un nouveau type d'architecture pour les maisons intelligentes. Contrairement aux solutions existantes, nous avons mis l'accent sur la fiabilité avec notamment la suppression de tous les points uniques de rupture ou SPoF. Pour ce faire, nous avons ajouté de l'intelligence et des capacités de communication dans tous les transducteurs de l'environnement. Afin de faciliter le développement, nous avons introduit une façon générique de réaliser les transducteurs intelligents. Ainsi, ils sont divisés en deux parties distinctes, une chargée de récupérer les valeurs venant de capteurs matériels ou logiciels et une autre chargée de l'intelligence et de la communication au sein de l'environnement. Ces unités communiquent entre elles par un lien série ou interprocessus suivant leur type qui permet de récupérer des valeurs de haut niveau à partir de capteurs de bas niveau ainsi que de modifier le comportement des unités passives en modifiant le pilote qui les constitue et ce sans avoir à modifier tout le transducteur. Pour finir, l'utilisation de LNCF un protocole de messagerie présenté au Chapitre 4 a permis d'augmenter la fiabilité de la communication en retirant tous les points chauds en matière de communication.

Nous avons exécuté une série de tests sur notre architecture afin de démontrer sa fiabilité, ses capacités de mise à l'échelle et ses performances. Concernant ce dernier point en particulier, les tests ont prouvé que même dans le pire cas possible (i.e. 30 transducteurs intelligents envoyant des valeurs à 1 kHz) notre solution ne souffre d'aucune latence excessive exceptée celle provenant de l'infrastructure réseau mesurée à environs 3 ms. De plus, cette latence ne varie que très peu quand on passe de 5 à

30 transducteurs, confirmant les capacités de mise à l'échelle de notre architecture. En ce qui concerne la fiabilité, nous avons conduit plusieurs tests consistant à priver notre infrastructure des éléments la composant. Ces tests ont confirmé trois éléments importants. Le premier est la non-nécessité de l'entité de gestion qui aurait pu constituer un SPoF dans notre architecture. Le second est que même si l'on prive notre solution de la moitié de sa réseautique, il ne lui faut qu'une trentaine de secondes pour redevenir totalement opérationnelle. Le dernier est que même après une perte totale, notre solution est capable de redémarrer et redevenir entièrement opérationnelle sans aucune intervention humaine. Pour finir, nous avons rapporté les coûts nécessaires pour réaliser ce prototype. Avec un prix en dessous de 2,000 USD, nous pouvons affirmer que notre solution est relativement peu coûteuse ce qui est un point important quand on considère les frais que payent déjà les malades atteints d'Alzheimer.

Figure 5.2 : Le schéma JSON de la description d'un ensemble de pilotes.

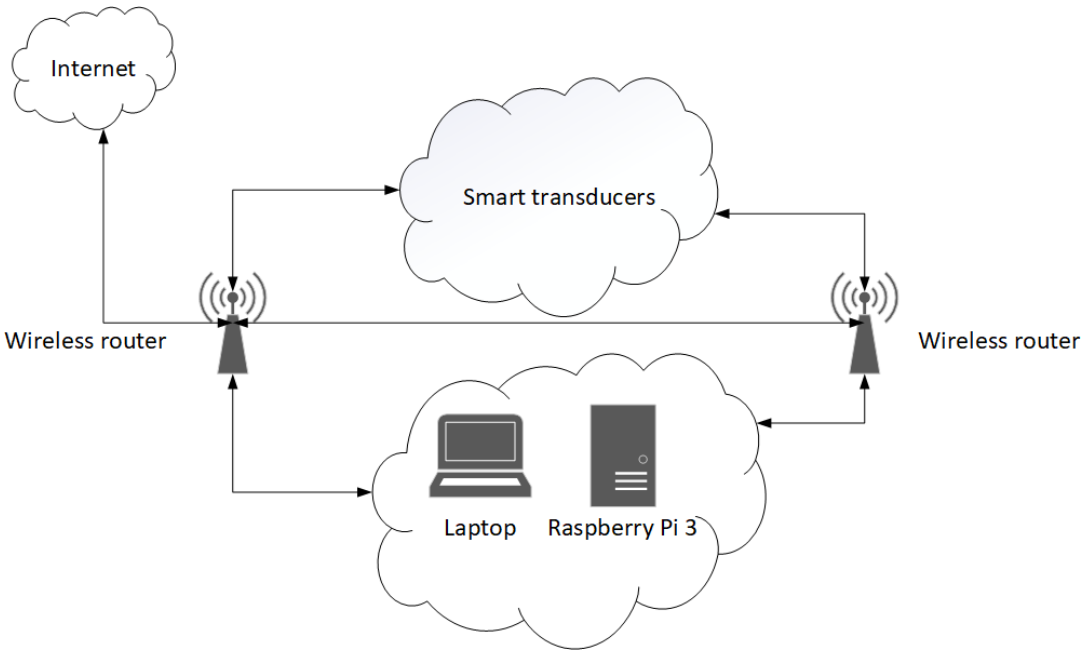

Figure 5.3 : L'infrastructure de test

CHAPITRE VI

EMBARQUER LA RECONNAISSANCE D'ACTIVITÉS SUR LES TRANSDUCTEURS

Nous avons, au cours des chapitres précédents, proposé une nouvelle façon de communiquer et de concevoir les maisons intelligentes. Nous avons établi qu'une architecture distribuée sans points chauds était de facto plus fiable. De plus, nos tests ont prouvé que l'ajout de capacités de calcul et de communication dans les transducteurs de l'environnement aidait à la mise à l'échelle de celui-ci. Le problème qui se pose ici et que la majorité de la littérature dans la reconnaissance d'activités se concentre sur des versions centralisées de celle-ci alors que notre architecture est distribuée. Il convient donc de fournir une preuve tangible que malgré cette différence de conception, la reconnaissance d'activités est toujours faisable sur notre solution. Ce point particulier est l'axe principal de ce chapitre avec, pour commencer la description d'une nouvelle façon d'exécuter cette intelligence de manière distribuée suivie de tests sur des ensembles de données connus permettant de vérifier la faisabilité et les résultats d'une telle tâche. Ce Chapitre fait l'objet d'un article soumis dans une conférence (« The 15th IEEE International Conference on Ubiquitous Intelligence and Computing ») (Plantevin *et al.*, 2018a).

6.1 UNE RECONNAISSANCE D'ACTIVITÉS DISTRIBUÉE

Notre façon de concevoir l'architecture et ses communications implique que la reconnaissance d'activités proposée ici comporte deux caractéristiques la différenciant

de la plupart des autres travaux du domaine. Premièrement, ses données d'apprentissage sont distribuées de manière hétérogène dans l'infrastructure. En effet, aucun site ne possède les mêmes attributs puisque les données enregistrées dans ces derniers dépendent directement des unités passives qui sont associées aux unités intelligentes. La deuxième caractéristique majeure est que notre algorithme s'exécute au plus proche des capteurs et peut même, s'il le faut, directement agir avec sans aucune forme de communication réseau. Cette particularité peut constituer un grand avantage notamment si l'intelligence applique le principe suggéré par Kane *et al.* (2015) qui consiste à implémenter des réactions réflexes à certains stimuli. Afin d'expliquer au mieux notre nouveauté dans ce domaine, nous allons commencer par décrire son fonctionnement pour un site puis nous nous intéresserons à la collaboration entre les sites pour arriver à la décision finale.

6.1.1 UNE INTELLIGENCE À DEUX NIVEAUX

Embarquée au cœur de chacun des transducteurs intelligents, se trouve une intelligence comme l'a suggéré Kane *et al.* (2015). Cette dernière implémente un mécanisme de secours basé sur une série de réflexes, à l'image de l'humain retirant sa main d'une source de chaleur après une brûlure. Afin de créer une intelligence prenant en compte une certaine forme de réflexe, nous sommes arrivés à la conclusion que cette dernière devait être constituée de plusieurs couches différentes. Par conséquent, notre solution implémente deux couches majeures qui sont les réflexes et l'intelligence et une couche hybride que nous appelons les réflexes appris. Ces différentes couches sont décrites en Figure 6.1.

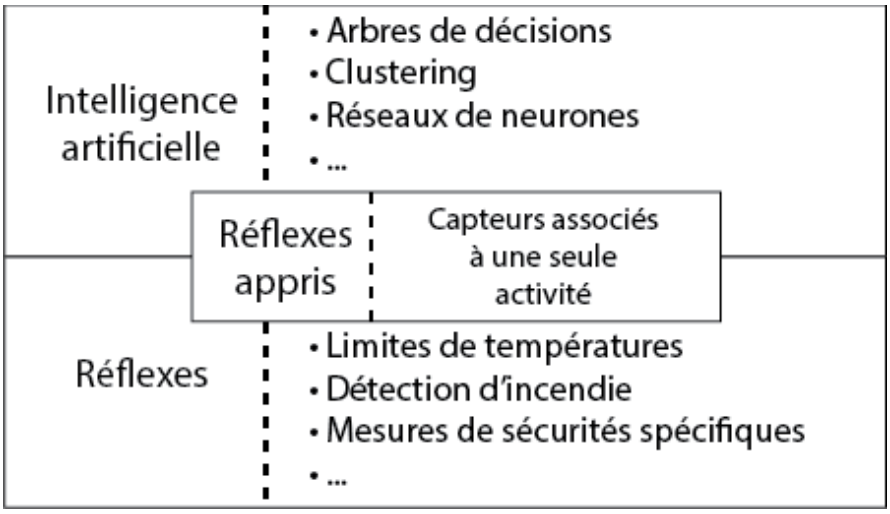

Figure 6.1 : Une représentation des différentes couches de notre intelligence embarquée.

La première couche que va rencontrer toute donnée arrivant pour une décision est la couche des réflexes. Celle-ci est constituée d'une suite de seuils et règles garantissant la sécurité de l'habitant en tout temps, et ce même si la reconnaissance d'activités échoue. Ces règles peuvent être très simples comme assez complexes, mais se différencient de l'intelligence par le fait qu'aucun calcul (préalable ou pas) ni communication n'est nécessaire pour arriver à une décision. Afin d'expliquer au mieux cette différence cruciale, nous proposons de prendre deux exemples concrets. Le premier est une mesure de sécurité bien connue qui consiste en un simple seuil sur un capteur permettant de détecter la fumée. Si la valeur de ce capteur dépasse une certaine valeur, l'action associée à ce seuil est immédiatement exécutée, et ce sans plus de réflexion. Dans ce cas particulier, le transducteur demandera la mise en marche d'une alarme et éteindra tous les feux d'une gazinière. Le second exemple n'est pas un réflexe, il s'agit d'éteindre un four si l'habitant n'est pas allé vérifier la cuisson depuis plus d'une heure. En effet, dans ce cas particulier certains calculs, ou des communications, sont nécessaires pour inférer le fait que la personne n'a pas été vérifier la cuisson depuis plus d'une heure.

Par conséquent ce type de règles se retrouvera dans la couche intelligence artificielle et pas dans la couche réflexes. Pour résumer, il est important de retenir que la couche des réflexes n'appartient aucunement au domaine de l'assistance ou de la guidance. Il s'agit uniquement d'une série de règles garantissant la sécurité des habitants, qu'ils soient atteints de démence ou pas, en tout temps et contre des phénomènes graves (e.g. incendie, intoxication au monoxyde de carbone, etc.).

La seconde couche majeure de notre solution, l'intelligence artificielle, concerne uniquement les décisions réfléchies. C'est ici que résident les modèles appris par forage de données et la reconnaissance d'activités de manière générale. L'apprentissage de ces modèles se réalise en local sur le transducteur même ou de manière centralisée si l'algorithme distribué le demande. Néanmoins, les conclusions, tirées de ce modèle, sont toutes calculées localement uniquement. Cela signifie que même si l'apprentissage peut se faire de manière centralisée, la reconnaissance ne doit se faire qu'à partir des capteurs et effecteurs présents sur le transducteur intelligent, il ne faut pas que ce dernier ait besoin des valeurs des capteurs d'autres transducteurs pour réussir à prendre une décision. Nous forçons cette règle pour s'assurer que l'intelligence de la maison est bien totalement distribuée dans sa prise de décision. Pour finir, cette couche est aussi la seule à posséder des capacités de communication qui peuvent notamment servir à connaître les décisions des autres transducteurs pour mettre en place un système de vote, mécanisme à la base de nombreux algorithmes de forage de données distribué comme nous l'avons vu au Chapitre 3. Il est à noter que nous ne recommandons ni ne décrivons aucun algorithme en particulier. En effet, il s'agit ici uniquement de l'architecture générale de la prise de décision, basée réflexe, de notre transducteur intelligent.

Pour finir, nous avons ajouté, en plus des deux couches majeures, une couche hybride entre les deux. Celle-ci représente ce que nous appelons les réflexes appris. Il s'agit d'un comportement similaire à celui des réflexes, mais qui implémente un apprentissage très léger en terme de puissance de calcul. En effet, ce dernier crée des relations associant un capteur unique à une activité unique si cela peut-être fait. Ainsi, si un capteur est activé lors de la réalisation d'une et une seule activité, la décision de reconnaître cette dernière sera faite de manière réflexe sans même passer par la couche intelligence artificielle. La phase d'apprentissage, extrêmement simple, est rappelée dans l'Algorithme 6.1. Tout d'abord, on parcourt l'ensemble des données d'apprentissage en associant à chaque capteur les activités dans lesquelles il apparaît. Ensuite, nous parcourons l'ensemble des capteurs et retirons des données ceux qui ne sont associés qu'à une et une seule activité. Pour finir, nous supprimons de l'ensemble d'apprentissage les activités trouvées en réflexe. Nous avons ajouté cette couche suite à plusieurs tests réalisés avec des ensembles de données dont certains capteurs n'étaient présents que dans une et une seule activité ce qui nous a permis de faire baisser le nombre d'activités que nos modèles de forage de données devaient reconnaître, facilitant ainsi leur travail.

Nous avons présenté ici l'architecture générale de l'intelligence embarquée dans un transducteur intelligent. Il convient maintenant de présenter une manière simple de prendre des décisions au sein de l'environnement intelligent en combinant les décisions de chacun des différents sites le composant.

Entrée : X l'ensemble des entrées d'apprentissage

Y l'ensemble des activités attendues

Sortie : Z l'ensemble des réflexes appris

1 **début**

2 $tempMap \Leftarrow Activite[][]$ *avec* $taille(tempMap) = largeur(X)$

3 **pour** $x_{i,j} \in X$ **faire**

4 **si** $x_{i,j} \neq 0 \wedge !ContenirValeur(tempMap_j, Y_i)$ **alors**

5 $AjouterValeur(tempMap_j, Y_i)$

6 **fin**

7 **fin**

8 **pour** $temp_i \in tempMap$ **faire**

9 **si** $longeur(temp_i) = 1$ **alors**

10 $AjouterValeur(Z, \{i; temp_{i,0}\})$

11 $EnleverColonne(X, i)$

12 $EnleverActivite(X, Y, temp_{i,0})$

13 **fin**

14 **fin**

15 **fin**

Algorithme 6.1 : L'algorithme d'apprentissage des réflexes appris.

6.1.2 UNE INTELLIGENCE COLLABORATIVE

Il nous faut, dans ce chapitre, montrer que, malgré le caractère distribué de notre solution, il est toujours possible de reconnaître des activités. Pour cela, nous proposons ici un algorithme de reconnaissance des activités très simple, mais dans lequel la décision finale est prise grâce à une collaboration entre les différents transducteurs intelligents de l'environnement. La méthode que nous proposons ici permet d'exécuter l'apprentissage et la reconnaissance des activités de manière distribuée sur les transducteurs de l'habitat. Sa force réside dans sa simplicité puisque la décision globale consiste en un simple système de vote pondéré entre les différentes décisions des site locaux. Notre solution ne préconise aucun algorithme de forage de données en particulier et permet, de manière simple, d'appliquer le modèle qui correspond le mieux aux données présentes localement. La seule limitation que nous imposons est la définition, pour le modèle, d'au moins une mesure de performance permettant de quantifier la qualité de sa décision, et ce pour chacune des activités considérées. Cette mesure n'est pas imposée, il peut s'agir de la Kappa, la F-Score, voire même une mesure spécifique à un algorithme en particulier. Les seules restrictions s'appliquant sont qu'elle doit être commune à tout le réseau, puisque celle-ci sert à définir la pondération du vote lors de la décision finale, et bornée puisqu'elle devra être ramenée à un intervalle d'entiers fini.

L'apprentissage se déroule en trois étapes principales. La première est le partage des activités à reconnaître. En effet, afin de compresser les messages, et donc de minimiser l'impact sur l'infrastructure réseau, notre solution n'échange pas les noms complets des activités reconnues, mais uniquement une représentation numérique de

ceux-ci. Ainsi, tous les sites envoient les activités présentes dans leurs ensembles d'apprentissage local à une entité centralisée désignée (il peut s'agir d'un transducteur intelligent ou de l'entité de gestion que nous avons définie au Chapitre 5). Une fois que cette entité à reçu toutes les activités, elle supprime les potentiels doublons avant de les trier. Cette opération effectuée, un tableau ordonné contenant toutes les activités est renvoyé aux différents sites locaux. Cette étape complétée la seconde peut être exécutée. Chaque site apprend son modèle local en se servant uniquement de son ensemble de données d'apprentissage local contenant donc ses caractéristiques propres puisque nos données sont distribuées de manière hétérogène (cf. Chapitre 3). Une fois le modèle local appris, la troisième et dernière étape qui constitue à attribuer les pondérations peut commencer. Pour cela, la métrique choisie est calculée pour chacune des classes de l'ensemble d'apprentissage au moyen d'une validation croisée (« 10-folds cross validation »). Celle-ci est ensuite mise à l'échelle pour être comprise entre 0 et 100 inclus puis arrondie pour être transformée en entier. Cette étape est importante pour éviter les possibles problèmes de précisions qui pourraient arriver avec de trop petits nombres à virgules. Pour finir, le nombre obtenu est associé à la classe en question dans un tableau associatif et constitue la pondération du vote.

Une fois l'apprentissage réalisé, la reconnaissance est assez triviale. À chaque évènement sur un transducteur, celui-ci prend une décision en se servant de son modèle local puis envoie sur le réseau un message LNCF avec le sujet *ACTIVITY* contenant le numéro (ou identifiant) de l'activité, la pondération du vote pour cette dernière et l'identifiant du transducteur le tout séparé par le caractère « : ». Ainsi, tous les transducteurs intelligents peuvent mettre à jour un tableau ordonné contenant les activités et

le nombre de votes qu'elles ont reçus. De plus, si un transducteur change de décision, il est très facile de supprimer son vote de l'ancienne activité pour ajouter celui de la nouvelle. Ainsi, il est garanti qu'au fur et à mesure de l'utilisation du réseau, les votes des anciennes activités sont bien supprimés. Pour finir, la décision prise par le réseau complet est l'activité ayant le plus de votes.

6.2 TESTS ET DISCUSSIONS

Afin de valider la méthode proposée dans ce chapitre, et par conséquent valider la faisabilité de la reconnaissance d'activités sur notre architecture distribuée, nous nous devons d'exécuter des tests en utilisant des ensembles de données connus. Nous allons, dans cette partie, commencer par décrire les données et le matériel utilisé pour ces tests ainsi que le déroulement de ceux-ci et pour finir, nous présenterons et discuterons nos résultats.

Dans le but de tester notre solution, il nous fallait des ensembles de données reliés au monde de la reconnaissance d'activités. Pour ce faire, nous en avons choisi deux parmi ceux mis en ligne par le projet CASAS (Cook et Schmitter-Edgecombe, 2009). Le premier dénommé « adlnormal » est constitué de 5 activités de la vie quotidienne qui sont *téléphoner*, *se laver les mains*, *cuisiner*, *manger* et *nettoyer* toutes exécutées par 23 participants différents. Le nombre de capteurs matériels utilisés pour créer ce dernier est de 39, répartis en plusieurs types : 26 capteurs de mouvement, 8 capteurs de présence d'objets, 1 contacteur de porte, 1 capteur de température pour la gazinière, 2 capteurs d'écoulement d'eau dans l'évier et 1 capteur d'utilisation du téléphone. Le second ensemble de données utilisé dans ce test se nomme *adlinterweave* et est, quant à

lui, composé de 8 activités différentes exécutées par 21 participants. Dans ce deuxième cas, les activités sont : *remplir l'armoire à pharmacie*, *regarder un DVD*, *arroser les plantes*, *répondre au téléphone*, *remplir une carte d'anniversaire*, *préparer une soupe*, *nettoyer* et *choisir une tenue*. Les capteurs présents sont ici plus nombreux avec un total de 78 capteurs composés de 51 capteurs de mouvement, 8 capteurs de présence d'objets, 12 contacteurs de portes, 2 capteurs d'écoulement d'eau, 1 capteur de température de la gazinière, 1 capteur sur le téléphone et 3 capteurs de température.

Pour réaliser ce test nous avons utilisé 6 Raspberry Pi représentant les transducteurs intelligents. Contrairement aux tests du Chapitre 5 il s'agit ici de Raspberry Pi 3 et non de Raspberry Pi Zero W, car, pour faciliter les développements, nous avons utilisé les technologies .NET Core 2.0 et Accord.NET (une librairie d'apprentissage automatique), deux technologies incompatibles avec la plus petite des plateformes de Raspberry. Ces 6 transducteurs intelligents étaient connectés sur un même réseau sans fil utilisant un simple routeur (TP-Link Archer C5) et donc n'implémentant aucune forme de redondance, car la démonstration de la fiabilité n'était pas l'objectif principal. Chacun des transducteurs embarquait une intelligence comme présentée plus tôt basée sur l'algorithme C4.5 (Quinlan, 1986) et utilisant la Kappa comme mesure d'exactitude du modèle apprit pour le calcul de la pondération du vote final. Pour finir, chacune de ces entités intelligentes s'est vu associer des capteurs en fonction de la localisation de ceux-ci dans l'appartement de test créant ainsi 6 zones différentes qui sont : la cuisine, le salon, la chambre 1, la chambre 2, la salle de bain et le couloir. Ainsi, les ensembles de données autrefois centralisés se sont retrouvés distribués.

Avant de discuter des résultats de ces tests, il convient d'en expliquer le déroulement. Nous avons tout d'abord calculé des caractéristiques très simples sur les ensembles de données. Ces dernières sont, le nombre d'activations des capteurs et, uniquement pour les capteurs analogiques, la moyenne, la médiane et l'écart type des valeurs. Ces calculs ont été effectués sur chaque activité complète créant, au final, une ligne de donnée par activité et par participant. Ensuite, nous avons pris 33 % des participants (participants 1 à 8 pour *adlnormal* et 4, 13, 14, 15, 17 et 18 pour *adlinterweave*) et avons séparé leurs activités dans une autre ensemble de données destiné aux tests de performance de notre solution. Pour finir, nous avons séparé les colonnes des deux ensembles de données finaux en 6, suivant la localisation des capteurs correspondants aux caractéristiques calculées, et nous les avons envoyés aux 6 transducteurs intelligents. Il est à noter ici que seuls deux transducteurs intelligents ont reçu des données puisque les activités des ensembles de données ne se déroulaient que dans la cuisine et le salon et par conséquent seules ces deux entités ont effectivement appris et testé un modèle. Ces dernières ont utilisé le premier ensemble pour apprendre l'arbre lui correspondant et extraire la Kappa du modèle pour chacune des activités en utilisant une validation croisée à 10 plis. Cette Kappa a été ensuite multipliée par 100 puis arrondie afin de créer la pondération du vote. Pour finir, chacune des entités a utilisé l'ensemble de test pour valider le modèle appris, envoyant sur le réseau les décisions correspondant à chaque ligne au format JSON contenant les informations, activités attendues, participant, décision et pondération. Un ordinateur portable s'est chargé de centraliser les votes et calculer les mesures de précision et de Kappa sur l'ensemble du réseau. C'est le même ordinateur qui a parallèlement aux entités appris un modèle de C4.5 sur l'ensemble de

données centralisé et calculé ses métriques en validation croisée sur l'ensemble d'apprentissage puis en utilisant l'ensemble de test. Tous ces tests ont été réalisés deux fois, le premier sans le comportement de réflexe appris présenté plus haut et le deuxième avec dans le but de comparer les performances de reconnaissance.

Tous les résultats détaillés des tests décrits précédemment sont rappelés dans les Tables 6.1 à 6.4. Dans celles-ci, la première colonne correspond toujours à l'ensemble de données utilisé pour réaliser le test il y en a 4 au total : *adlnormal*, *adlinterweave*, *adlnormal sans réflexes* et *adlinterweave sans réflexes*, les deux derniers correspondant aux tests où les réflexes n'ont pas été appris. Ainsi, l'ensemble de données *adlnormal* comprend une activité de moins (*téléphoner*) et *adlinterweave* deux de moins (*répondre au téléphone* et *préparer une soupe*) qui ont été apprise de manière réflexe. Les colonnes qui suivent correspondent aux différentes métriques calculées sur les modèles. Les Table 6.1 et 6.2 présentent les Kappa et précisions internes des transducteurs de la salle à manger et de la cuisine suivies de la Kappa générale du réseau, les deux premières étant calculées avec la validation croisée sur l'ensemble d'apprentissage et la dernière avec l'ensemble de test. Les Tables 6.3 et 6.4 présentent le même type de résultats, mais pour la version centralisée. Ainsi, la deuxième colonne présente les Kappa ou précision calculées avec une validation croisée sur l'ensemble d'apprentissage puis avec l'ensemble de test. Pour finir, la Table 6.5 présente quant à elle la comparaison entre les façons distribuées et centralisées ne rappelant que les Kappa et précisions du test final utilisant l'ensemble de test.

Deux conclusions évidentes peuvent être tirées de ces tests. La première est que

Ensemble de données	Salle à manger	Cuisine	Globale
adlnormal sans réflexes	0.58	0.84	0.97
adlnormal	1	0.9	0.98
adlinterweave sans réflexes	0.85	0.86	0.81
adlinterweave	0.95	0.85	0.89

Tableau 6.1 : Ensembles des Kappa détaillées pour la reconnaissance distribuée

Ensemble de données	Salle à manger	Cuisine	Globale
adlnormal sans réflexes	0.68	0.88	0.98
adlnormal	1	0.92	0.99
adlinterweave sans réflexes	0.87	0.88	0.83
adlinterweave	0.92	0.88	0.90

Tableau 6.2 : Ensembles des précisions détaillées pour la reconnaissance distribuée

Ensembles de données	Kappa (10-folds)	Kappa (33%)
adlnormal sans reflexes	0.86	0.91
adlnormal	0.89	0.96
adlinterweave sans reflexes	0.95	0.81
adlinterweave	0.97	0.87

Tableau 6.3 : Ensembles des Kappa détaillées pour la reconnaissance centralisées

quand on regarde en détail la Table 6.5 on s'aperçoit que les Kappa et précisions de la méthode distribuée sont soient meilleures ou au pire égales à la version centralisée. Cette première particularité vient non seulement confirmer que la reconnaissance d'ac-

Ensembles de données	Accuracy (10-folds)	Accuracy (33%)
adlnormal sans reflexes	0.89	0.93
adlnormal	0.92	0.97
adlinterweave sans reflexes	0.97	0.83
adlinterweave	0.97	0.88

Tableau 6.4 : Ensembles des précision détaillées pour la reconnaissance centralisées

	Kappa		Précision	
Ensembles de données	Distribuée	Centralisée	Distribuée	Centralisée
adlnormal sans réflexes	0.97	0.91	0.98	0.93
adlnormal	0.98	0.96	0.99	0.97
adlinterweave sans réflexes	0.81	0.81	0.83	0.83
adlinterweave	0.89	0.87	0.9	0.88

Tableau 6.5 : Comparaison des Kappa et précisions des reconnaissances d'activités distribuée et centralisée

tivités est toujours faisable sur notre nouvelle architecture, mais en plus, elle est plus performante que l'ancienne version, centralisée. La deuxième conclusion intéressante concerne le mécanisme des réflexes appris que nous avons introduit dans ce chapitre. En effet, si l'on regarde uniquement les variations de Kappa et précision entre les mêmes ensembles de données, mais avec ou sans réflexes, on s'aperçoit que la version avec les réflexes est systématiquement meilleure que la version sans. Cela est dû au fait que les réflexes suppriment des activités et des colonnes des ensembles d'apprentissage, facilitant le problème. Malgré ces très bons résultats, une petite nuance doit être pré-

cisée dans le sens où il serait intéressant de trouver un autre ensemble de données qui comporterait des activités nécessitant les 6 transducteurs intelligents pour confirmer définitivement que la version distribuée est plus performante.

6.3 CONCLUSION

Tout au long de ce chapitre, nous avons présenté une nouvelle façon d'exécuter la reconnaissance d'activités sur notre architecture distribuée. Celle-ci est composée de deux couches principales comme l'avait suggéré Kane *et al.* (2015). La première est purs réflexes et constitue un système de la dernière chance au cas où la reconnaissance d'activités échouerait, fournissant tout de même un minimum de sécurité dans l'habitat. La deuxième est pure réflexion, ici réside la reconnaissance d'activités qui va, plus tard, permettre l'assistance. Cette dernière utilise un système de vote pondéré permettant de prendre une décision de manière distribuée au sein de l'environnement intelligent. Entre ces deux couches, nous avons introduit une couche hybride constituée de ce que nous avons appelé les réflexes appris qui sont l'association d'un capteur avec une activité quand cela est possible.

Pour finir, nous avons réalisé quelques tests afin de valider trois points importants. Le premier est qu'il est toujours possible de réaliser la reconnaissance d'activités sur notre architecture distribuée ce qui a été démontré et même surpassé par le fait que nos résultats sont au pire aussi bons que la manière centralisée sinon meilleurs. Le deuxième point important est que notre intelligence basée sur un vote pondéré est fonctionnelle, cela a été démontré par le fait que nous avons, avec notre méthode d'excellents résultats de reconnaissance. Pour finir, le dernier point important était de confir-

mer que les réflexes appris amélioreraient l'intelligence. Les résultats étant meilleurs avec la couche hybride présentée, nous pouvons ici aussi conclure que cette partie est une réussite. Enfin, nous pensons qu'il convient de nuancer ces résultats en précisant qu'il serait intéressant de réaliser des tests avec des ensembles de données utilisant tous les transducteurs intelligents de l'environnement et pas seulement un sous-ensemble d'entre eux.

CHAPITRE VII

CONCLUSION GÉNÉRALE

Nous avons présenté ici un travail original permettant d'augmenter la fiabilité des habitats intelligents tout en baissant le coût de ceux-ci. Nous avons commencé par décrire au Chapitre 2 où en était la littérature dans ce domaine. Nous avons remarqué que les infrastructures existantes créées au cours des dix dernières années implémentaient toutes le même type d'architecture centralisée facilitant grandement la récupération des données et l'intelligence artificielle (Bouchard *et al.*, 2014; Hu *et al.*, 2016). Malheureusement, ce type d'architecture introduit également de nombreux points chauds qui sont un réel problème pour la fiabilité de toute l'infrastructure. Parallèlement à ces développements dans le monde de la maison intelligente, cette dernière décennie a vu de grandes innovations en termes d'électronique embarquée qui rendent possible la création de ce que les normes appellent des transducteurs intelligents. Ces derniers permettant, selon la littérature (Mark, 2004), d'augmenter la facilité d'installation, les capacités de mise à l'échelle et, en délocalisant l'intelligence du serveur centralisé pour la distribuer directement sur les transducteurs, la fiabilité.

Le Chapitre 3 a présenté quant à lui les différentes techniques existantes pour reconnaître des activités. Nous y avons vu que la majeure partie d'entre elles sont de simples applications d'algorithmes centralisés de forage de données. Dans ce grand domaine de l'informatique qu'est le forage de données, nous avons vu qu'un sous-ensemble de méthodes, distribuées, tente de répondre à certaines problématiques concer-

nant les systèmes composés de multiples entités, contraintes en ressources, communiquant par messages des données dont la confidentialité doit être respectée, et ce en prenant en compte que cette communication à un coût (e.g. batterie). Ce type de systèmes étant très proche de notre idée d'architecture « idéale » que nous avons présentée, nous nous sommes intéressés aux algorithmes de forage de données distribués, montrant ainsi que toutes les méthodes utilisées pour reconnaître des activités de manière centralisée, avaient leur équivalent en version distribué.

La problématique de fiabilité de l'environnement intelligent est importante et bien qu'une multitude de travaux récents se concentrent sur la reconnaissance d'activités, très peu se focalisent dessus. Dans cette thèse, nous avons apporté des réponses à cette problématique principale en la décomposant en deux problèmes spécifiques, auxquels nous avons répondu. Le premier est de trouver une façon de décentraliser, à bas coûts, l'infrastructure de la maison intelligente. Le deuxième quant à lui est de démontrer qu'il est possible de transformer la reconnaissance d'activités pour qu'elle fonctionne sur cet environnement distribué. Afin de répondre aux différentes problématiques posées, nous avons mis en place, à travers trois contributions, une architecture de maison intelligente fiable et distribuée permettant, autant que les précédentes, de reconnaître les activités. Cette contribution principale permet notamment de fusionner définitivement deux grands domaines de l'informatique moderne qui sont l'intelligence ambiante et l'Internet des objets permettant d'appliquer les méthodes découvertes chez l'un sur l'autre.

Dans les prochaines sections, nous allons nous intéresser à la réalisation des ob-

jectifs définis plus tôt puis nous décrirons les apports personnels que cette thèse a permis.

De la même façon que nous avions défini deux problèmes majeurs auxquels il nous fallait apporter des réponses, nous avions défini deux objectifs dans cette thèse. Le premier était la réalisation de l'architecture distribuée et peu coûteuse définissant ainsi, pour la première fois, une façon de créer ce type d'infrastructure. Le second objectif était d'implémenter une intelligence à deux niveaux permettant de reconnaître des activités tout en conservant le plus de sécurité possible si jamais la reconnaissance échouait.

7.1 OBJECTIF 1 : RÉALISATION DE L'ARCHITECTURE

Nous pensions que la seule façon de démontrer que la reconnaissance d'activités peut être effectuée sur une infrastructure distribuée est, en premier lieu, de prouver la viabilité de cette dernière. Ici, nous voulions défendre l'idée que la multitude de transducteurs déjà présents dans l'environnement peut servir de support à la distribution. Pour autant que nous le sachions, ce type d'architecture n'a jamais été implémentée au sein de la maison intelligente même si des auteurs (Angulo et Téllez, 2004) ont simulé son exécution montrant, de manière théorique, sa possibilité. De notre côté, nous pensions que la meilleure façon de démontrer la faisabilité d'une telle infrastructure est de l'élaborer et de l'implémenter en situation réelle. Néanmoins, il existait deux obstacles

majeurs à cette réalisation que nous avons résolus au sein de deux contributions.

Les deux obstacles qui se sont posés sont liés à l'hétérogénéité des transducteurs, inhérente aux environnements intelligents (Ghayvat *et al.*, 2015). Ce phénomène fait que le code enregistré dans chaque capteur intelligent doit soit contenir toutes les façons de communiquer avec tous les capteurs, soit être spécialisé dans un type de capteur. Le premier cas est, en raison du caractère limité de ces plateformes, impossible, car cela occuperait trop de stockage. Le second quant à lui, force à reprogrammer le capteur en cas de changement matériel, ce qui rend difficile la mise à l'échelle et quasiment impossible la configuration automatique. Pour finir, la grande différence entre les entités se retrouve aussi dans la façon qu'elles ont de communiquer entre elles. Afin de répondre à ces défis, nous avons d'abord introduit un nouveau protocole de communication, « The Light Node Communication Framework », permettant d'unifier les communications. Ensuite, nous avons proposé une architecture distribuée et peu coûteuse, une première dans le domaine, où il est facile d'ajouter de nouveaux transducteurs.

THE LIGHT NODE COMMUNICATION FRAMEWORK

La première contribution faite dans le cadre de cette thèse, présentée au Chapitre 4, est une nouvelle façon de communiquer au sein d'un environnement intelligent appelée « Light Node Communication Framework ». En effet, nous voulions un protocole de messagerie indépendant de tout matériel et sans aucun point de rupture unique (SPoF). Or, nous avons montré que la littérature n'en contenait aucun. En effet, les grands protocoles sont soit non portables sur toutes les entités (e.g. ZeroMQ (ZeroMQ,

2016)), soit constitués autour d'une architecture avec « broker » qui est un point de rupture unique. En plus de correspondre à ces caractéristiques, notre solution répond à la grande différence qui existe entre configuration et flux de donnée, en utilisant deux canaux différents. Le premier se base sur CoAP, un protocole connut du monde de l'IoT, pour garantir la fiabilité des informations de configuration. Le deuxième canal utilise un protocole que nous avons défini et testé dans cette thèse. Ce dernier utilise UDP « multicast » pour fournir plusieurs points importants au sein d'un habitat intelligent. Ces derniers sont la capacité à découvrir les entités sur le réseau, la capacité d'envoyer des données, chiffrés ou non, à hautes fréquences et à tous les clients connectés au réseau et pour finir, la capacité de compresser certains messages, trop lourds pour l'infrastructure.

Cette première contribution importante a été testée en laboratoire permettant de valider différents points importants de notre protocole. Tout d'abord, nous avons confirmé les capacités en termes de bande passante de notre solution. Les résultats démontrent la capacité d'envoyer un grand nombre de messages en un minimum de temps (environs 1000 par seconde dans le pire cas) même si l'on peut observer une grosse perte de performance pour les messages faisant 2 kio. Une autre conclusion tirée directement de ces résultats est que le chiffrement impacte très peu les débits finaux de notre solution. Le second test que nous avons réalisé nous a servi à assurer la fiabilité de notre protocole même si celui-ci se base sur UDP. Avec aucun message perdu ou corrompu et seulement 2 ou 3 problèmes d'ordonnancement sur 10,000 envois, le manque de mécanisme de fiabilité d'UDP ne semble pas impacter la fiabilité générale de notre solution. Par conséquent, nous pouvons affirmer que notre avancée est suffi-

samment fiable pour envoyer des données au sein d'une maison intelligente. Pour finir, le dernier test que nous avons réalisé est la comparaison de la similarité entre 1,000 paquets chiffrés avec la même clé secrète et encapsulant la même donnée. Avec une différence moyenne de 42.39 % des paquets, nous pouvons affirmer que malgré le fait que nous ne générons pas l'IV deAES totalement aléatoirement, notre solution reste immunisée contre les attaques de sémantiques et de similarité qui auraient pu être exécutées si notre algorithme ne fournissait pas assez d'aléatoire dans l'IV. Ces tests nous ont aussi permis de confirmer le point le plus important qui est que LNCF est, au mieux de nos connaissances, un travail original qui apporte une innovation importante dans le domaine de la communication au sein des environnements intelligents. Et c'est à ce titre qu'il a fait l'objet d'un article scientifique paru dans le journal *Sensors* (Plantevin *et al.*, 2017).

L'ARCHITECTURE DISTRIBUÉE

La deuxième contribution importante de cette thèse, présentée au Chapitre 5 et faisant l'objet d'un article de journal soumis dans « Journal of Ambient Intelligence and Humanized Computing » (Plantevin *et al.*, 2018b), est une architecture générique permettant de concevoir n'importe quel habitat intelligent en garantissant la fiabilité, un faible coût et une grande facilité de mise à l'échelle. Pour cela, nous avons pris le parti d'utiliser les transducteurs déjà présents dans l'environnement. Contrairement aux solutions existantes, nous avons mis l'accent sur la fiabilité avec notamment la suppression de tous les points uniques de rupture ou SPoF. Pour ce faire, nous avons

ajouté de l'intelligence et des capacités de communication dans tous les transducteurs de l'environnement. Afin de faciliter le développement, nous avons introduit une façon générique de réaliser les transducteurs intelligents. Ainsi, ils sont divisés en deux parties distinctes, une chargée de récupérer les valeurs venant de capteurs matériels ou logiciels et une autre chargée de l'intelligence et de la communication au sein de l'environnement. Ces unités communiquent entre elles par un lien série ou interprocessus suivant leur type qui permet de récupérer des valeurs de haut niveau à partir de capteurs de bas niveau ainsi que de modifier le comportement des unités passives en modifiant le pilote qui les constitue, et ce sans avoir à modifier tout le transducteur. Pour finir, l'utilisation de LNCF, un protocole de messagerie présenté lui aussi dans cette thèse, nous a permis d'augmenter la fiabilité de la communication en retirant tous les points centraux de communication.

Afin de confirmer notre avancée et montrer sa faisabilité, nous avons exécuté une série de tests. Ces tests ont permis de valider sa fiabilité, ses capacités de mise à l'échelle et ses performances. Concernant ce dernier point en particulier, les tests ont démontré que même dans le pire cas possible (i.e. 30 transducteurs intelligents envoyant des valeurs à 1 kHz) notre solution ne souffre d'aucune latence excessive excepté celle provenant de l'infrastructure réseau mesurée à environ 3 ms. De plus, cette latence ne varie que très peu quand on passe de 5 à 30 transducteurs, confirmant les capacités de mise à l'échelle de notre architecture. En ce qui concerne la fiabilité, nous avons conduit plusieurs tests consistant à priver notre infrastructure des éléments la composant. Ces tests ont confirmé trois éléments importants. Le premier est la non-nécessité de ce que nous avons appelé l'entité de gestion qui aurait pu constituer le seul et unique

SPoF dans notre architecture. Le second est que même si l'on prive notre solution de la moitié de sa réseautique, il ne lui faut qu'une trentaine de secondes pour redevenir totalement opérationnelle. Le dernier est que même après une perte totale (e.g. un arrêt complet du système suite à une coupure de courant), notre solution est capable de redémarrer et redevenir entièrement opérationnelle sans aucune intervention humaine. Pour finir, nous avons rapporté les coûts nécessaires pour réaliser ce prototype. Avec un prix en dessous de 2,000 USD, nous pouvons affirmer que notre solution est relativement peu coûteuse (Bouchard *et al.*, 2014; Cook *et al.*, 2012) ce qui est un point important quand on considère les frais que payent déjà les malades atteints d'Alzheimer. Tous ces résultats nous confirment que notre innovation fonctionne et qu'elle répond à une question que nous nous sommes posée dans cette thèse et qui est de trouver une façon de décentraliser, à bas coûts, l'infrastructure de la maison intelligente.

7.2 OBJECTIF 2 : UNE RECONNAISSANCE D'ACTIVITÉS DISTRIBUÉE

Pour finir nos contributions, nous avons répondu à la deuxième sous-question de notre problématique principale qui est de fournir une méthode pour reconnaître les activités et amener une sécurité logicielle au sein d'un environnement intelligent distribué. Cette contribution, présentée au Chapitre 6 et décrite dans un article de conférence soumis à UIC (« Ubiquitous Intelligence and Computing ») « PlantevinIA2018 », a pris la forme d'une nouvelle façon d'exécuter la reconnaissance d'activités sur une architecture non plus centralisée, mais distribuée. Celle-ci est composée de deux couches principales comme l'avait suggéré Kane *et al.* (2015). La première est purs réflexes et constitue un système de la dernière chance au cas où la reconnaissance d'activi-

tés échouerait, fournissant tout de même un minimum de sécurité dans l'habitat. La deuxième, purement réfléchie, est le lieu de résidence de la reconnaissance d'activités qui va, plus tard, permettre l'assistance. Cette dernière utilise un système de vote pondéré, présenté dans cette thèse, mais déjà couramment utilisé dans la littérature du forage de données distribué, permettant de prendre une décision de manière répartie au sein de l'environnement intelligent. Entre ces deux couches, nous avons introduit une couche hybride constituée de ce que nous avons appelé les « réflexes appris » qui sont l'association d'un capteur avec une activité quand cela est possible. Cette dernière permettant de faciliter le problème de la reconnaissance d'activités en diminuant, si possible, le nombre d'activités à reconnaître.

De la même manière que les autres contributions, nous avons confirmé celle-ci par une série de tests. Ces derniers ont permis de valider trois points importants. Le premier est qu'il est toujours possible de réaliser la reconnaissance d'activités sur notre architecture distribuée ce qui a été démontré et même surpassé par le fait que nos résultats sont, en pire cas, aussi bons que la manière centralisée sinon meilleurs. Le deuxième point important est que notre intelligence basée sur un vote pondéré est fonctionnelle, cela a été démontré par le fait que nous avons, avec notre méthode d'excellents résultats de reconnaissance. Pour finir, le dernier point important était de confirmer que les réflexes appris amélioreraient l'intelligence. Les résultats étant meilleurs avec la couche hybride présentée, nous pouvons ici aussi conclure que cette partie est une réussite. Enfin, nous pensons qu'il convient de nuancer ces résultats en précisant qu'il serait intéressant de réaliser des tests avec des ensembles de données utilisant tous les transducteurs intelligents de l'environnement et pas seulement un sous-ensemble d'entre eux.

7.3 RÉPONSE AU PROBLÈME GÉNÉRAL

La problématique qui a guidé cette thèse de ses débuts jusqu'à sa fin était d'augmenter la fiabilité de l'habitat intelligent. Pour répondre à ce problème qui constitue, selon nous, un frein majeur à l'adoption des habitats intelligents, nous avons pris le parti de changer complètement la vision de l'environnement intelligent passant d'un monde centralisé à distribué. Le premier obstacle que nous avons rencontré était la grande hétérogénéité des transducteurs et des moyens de communication. Pour surpasser cet obstacle, nous avons introduit un tout nouveau protocole de messagerie, LNCF, et une nouvelle architecture totalement distribuée de l'habitat intelligent. Le deuxième problème majeur qui s'est posé était que la reconnaissance d'activités à souvent, si ce n'est toujours, été effectuée de manière centralisée avec toutes les données concentrées en un seul endroit qui constituait un point de rupture de toutes les architectures présentées. Pour répondre à ce problème particulier, nous avons introduit une nouvelle façon de reconnaître les activités totalement distribuée et fournissant même un mécanisme de sécurité au cas ou aucune activité ne peut être reconnue, ce qui est totalement plausible si l'on considère l'infinité que constitue l'ensemble des activités réalisables par un être humain. Pour conclure, ces trois contributions au domaine permettent, en les combinant, de répondre à la question originelle, fournissant en plus à la communauté une nouvelle façon de communiquer qui peut être utilisée sur d'autres problèmes, une nouvelle architecture standard permettant de concevoir des habitats intelligents et pour finir, une nouvelle façon simple et distribuée de reconnaître des activités.

7.4 LIMITATIONS ET POSSIBILITÉS D'AMÉLIORATION

Bien que la solution que nous avons présentée tout au long de cette thèse apporte des réponses aux problèmes posés, nos différentes contributions peuvent encore être améliorées et surtout plus testées. Ainsi, notre protocole pourrait inclure un mode lui permettant de faire transiter les messages à travers Internet puisque certaines architectures pourraient vouloir envoyer des messages vers un « cloud » gérant différentes habitations. Bien que cela puisse causer des problèmes de sécurité, il n'est plus concevable à l'heure de l'IoT qu'un protocole de messagerie ne puisse relayer ses messages sur Internet. Concernant l'architecture, de nombreux tests supplémentaires peuvent être réalisés avec, pour commencer, l'implémentation du transducteur intelligent sur d'autres plateformes plus légères et n'embarquant, par conséquent, pas de Linux. De la même manière, un test de charge à grande échelle incluant beaucoup plus de ces transducteurs serait le bienvenu pour conclure sur la résistance à la charge de notre solution. Ensuite sur le sujet de notre intelligence, il conviendrait de tester notre méthode d'intelligence avec plus d'algorithmes de forage de données, confirmant ses capacités, et ce peu importe l'ensemble de données et l'algorithme utilisé. Pour finir, une piste de travail extrêmement intéressante serait d'amener une manière simple pour que l'architecture que nous avons présentée ici puisse interagir avec les autres infrastructures de l'IoT, extérieures à l'habitation. Ceci permettrait de prendre des décisions dans l'environnement intelligent à partir de capteurs extérieurs à l'habitat. Nous pensons que cela serait facilement réalisable par l'utilisation des transducteurs virtuels que nous avons présentés plus tôt, car ceux-ci peuvent déjà, si une connexion à Internet existe, aller chercher de l'information en dehors de l'habitation.

7.5 APPORTS PERSONNELS

Ce doctorat a eu de nombreux apports personnels tant sur le plan technique et intellectuel que sur ma propre personne. Il m'a permis d'améliorer grandement mon autonomie dans le travail ainsi que ma rigueur, deux qualités qui ont été grandement mises à mal tout au long de ces trois ans. D'un point de vue technique, cette thèse a été un grand défi. Il m'a fallu maîtriser de nombreux domaines différents allant de l'électronique avancée à l'intelligence artificielle en passant par la théorie des communications et des architectures distribuées. Pour finir, tous les travaux mis en œuvre pour cette thèse ou pour perfectionner mes compétences afin de réussir cette thèse ont permis d'écrire plusieurs publications scientifiques améliorant grandement mes compétences en anglais.

BIBLIOGRAPHIE

(2014). Recognizing blind spot check activity with car drivers based on decision tree classifier approach. *AAAI Workshop - Technical Report*, *WS*, 22–26.

Advantech (2016). Automation Controllers & I/Os. Récupéré de http://www.advantech.com/products/automation-controllers-i-os/sub_1-2mlf31

Al-Shaqi, R., Mourshed, M. et Rezgui, Y. (2016). Progress in ambient assisted systems for independent living by the elderly. *SpringerPlus*, *5*(1), 624.

Alliance, O. (2016). OSGi™ Alliance – The Dynamic Module System for Java. Récupéré de https://www.osgi.org/

Alzheimer's Association (2016). 2016 Alzheimer's disease facts and figures. *Alzheimer's & Dementia*, *12*(4), 459–509. http://dx.doi.org/http://dx.doi.org/10.1016/j.jalz.2016.03.001. Récupéré de http://www.sciencedirect.com/science/article/pii/S1552526016000856

Alzheimer's Association (2017). 2017 Alzheimer's Disease Facts and Figures. *Alzheimers Dement*, *13*, 325–373. http://dx.doi.org/10.1016/j.jalz.2017.02.001. Récupéré de https://www.alz.org/documents{_}custom/2017-facts-and-figures.pdf

Anderson, J. W., Meling, H., Rasmussen, A., Vahdat, A. et Marzullo, K. (2017). Local recovery for high availability in strongly consistent cloud services. *IEEE Transactions on Dependable and Secure Computing*, *14*(2), 172–184.

Anguita, D., Ghio, A., Oneto, L., Parra, X. et Reyes-Ortiz, J. L. (2012). Human Activity Recognition on Smartphones using a Multiclass Hardware-Friendly Support Vector Machine.

Angulo, C. et Téllez, R. (2004). Distributed Intelligence for smart home appliances. *Tendencias de la minería de datos en España.* Récupéré de http://ouroboros.org/papers/distributed.pdf

Arduino (2016). Arduino - Boards. Récupéré de https://www.arduino.cc/en/Main/Boards

Atzori, L., Iera, A. et Morabito, G. (2010). The internet of things : A survey. *Computer networks*. Récupéré de http://www.sciencedirect.com/science/article/pii/S1389128610001568

Bao, L. et Intille, S. S. (2004). Activity Recognition from User-Annotated Acceleration Data. *International Conference on Pervasive Computing*.

Barker, S., Mishra, A., Irwin, D., Cecchet, E., Shenoy, P. et Albrecht, J. (2012). Smart* : An Open Data Set and Tools for Enabling Research in Sustainable Homes. *ACM SustKDD*.

Benaloh, J. (1994). Dense probabilistic encryption. *Proceedings of the workshop on selected areas of*. Récupéré de http://sacworkshop.org/proc/SAC_94_006.pdf

Bouchard, K., Bouchard, B. et Bouzouane, A. (2012). Guidelines to Efficient Smart Home Design for Rapid AI Prototyping : A Case Study. Dans *International Conference on PErvasive Technologies Related to Assistive Environments*, p. 1. http://dx.doi.org/10.1145/2413097.2413134

Bouchard, K., Bouchard, B. et Bouzouanea, A. (2014). Practical guidelines to build smart homes : lessons learned. *Opportunistic Networking, Smart Home, Smart City, Smart Systems (Book Chapter)*, 1–37.

Bureau, U. C. (2016). Household Income. Récupéré le 2018-02-19 de https://www.census.gov/data/tables/time-series/demo/income-poverty/cps-hinc/hinc-02.html

Burges, C. J. C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, *2*(2), 121–167. http://dx.doi.org/10.1023/A:1009715923555. Récupéré de http://www.springerlink.com/index/Q878561731267 71Q.pdf

Callaway, E., Gorday, P. et . . ., L. H. (2002). Home networking with IEEE 802.15.4 : a developing standard for low-rate wireless personal area networks. *ieeexplore.ieee.org*.

Caragea, D. et Silvescu, A. (2004). A framework for learning from distributed data using sufficient statistics and its application to learning decision trees. *International Journal of*. Récupéré de http://content.iospress.com/articles/

international-journal-of-hybrid-intelligent-systems/his010

Chakrabarti, S., Ester, M., Fayyad, U. et Gehrke, J. (2012). Data mining curriculum : a proposal, Version 1.0 (2006). *www.Kdd.Org/Curriculum/*, 1–10. Récupéré de http://pdf.aminer.org/000/303/279/decision_tree_construction_from_multidimensional_structured_data.pdf%5Cnhttp://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Data+mining+curriculum:+A+proposal+(Version+1.0)#4%5Cnhttp://scholar.google.com/scholar

Charniak, E. et Goldman, R. (1993). A Bayesian model of plan recognition. *Artificial Intelligence*. Récupéré de http://www.sciencedirect.com/science/article/pii/0004370293900600

Chen, M., Mao, S. et Liu, Y. (2014). Big data : A survey. *Mobile Networks and Applications*, *19*(2), 171–209.

Chen, R., Sivakumar, K. et Khargupta, H. (2003). Learning Bayesian Network Structure from Distributed Data. *SDM*. Récupéré de http://epubs.siam.org/doi/abs/10.1137/1.9781611972733.31

Chen, W., Augusto, J. C. et Seoane, F. (2015). *Recent Advances in Ambient Assisted Living-Bridging Assistive Technologies, e-Health and Personalized Health Care*, volume 20. IOS press.

Chih-Wei Hsu, C.-C. C. et Lin, C.-J. (2008). A Practical Guide to Support Vector Classification. *BJU international*, *101*(1), 1396–400. Récupéré de http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

Collet, Y. (2013). Lz4 : Extremely fast compression algorithm. *code. google. com*.

Cook, D. et Krishnan, N. (2014). Activity learning from sensor data.

Cook, D. J., Augusto, J. C. et Jakkula, V. R. (2009). Ambient intelligence : Technologies, applications, and opportunities. *Pervasive and Mobile Computing*, *5*(4), 277–298. http://dx.doi.org/10.1016/j.pmcj.2009.04.001

Cook, D. J., Crandall, A. S., Thomas, B. L. et C., K. N. (2012). CASAS : A Smart Home in a Box. *100*(2), 130–134. http://dx.doi.org/10.1016/j.pestbp.

2011.02.012.Investigations

Cook, D. J. et Schmitter-Edgecombe, M. (2009). Assessing the Quality of Activities in a Smart Environment. *Methods of Information in Medicine*, *48*(5), 480–485. http://dx.doi.org/10.3414/ME0592. Récupéré de http://www.ncbi.nlm.nih.gov/pubmed/19448886

Cooper, G. et Herskovits, E. (1992). A Bayesian method for the induction of probabilistic networks from data. *Machine learning*. Récupéré de http://link.springer.com/article/10.1007/BF00994110

Dasilva, J., Giannella, C., Bhargava, R., Kargupta, H. et Klusch, M. (2005). Distributed data mining and agents. *Engineering Applications of Artificial Intelligence*, *18*(7), 791–807. http://dx.doi.org/10.1016/j.engappai.2005.06.004. Récupéré de http://dx.doi.org/http://dx.doi.org/10.1016/j.engappai.2005.06.004%5Cnhttp://linkinghub.elsevier.com/retrieve/pii/S095219760500076X

Delachaux, B., Rebetez, J. et Perez-Uribe, A. (2013). Indoor activity recognition by combining one-vs.-all neural network classifiers exploiting wearable and depth sensors. *Work-Conference on ...*. Récupéré de http://link.springer.com/chapter/10.1007/978-3-642-38682-4_25

Dell (2016). Dell PowerEdge Rack Servers. Récupéré de http://www.dell.com/ca/business/p/poweredge-rack-servers

Deutsch, L. P. (1996). Gzip file format specification version 4.3.

Dhillon, I. et Modha, D. (2002). A data-clustering algorithm on distributed memory multiprocessors. *Large-Scale Parallel Data Mining*. Récupéré de http://link.springer.com/chapter/10.1007/3-540-46502-2_13

Drumea, A., Popescu, C. et Svasta, P. (2005). GSM solutions for low cost embedded systems for industrial control. Dans *28th International Spring Seminar on Electronics Technology : Meeting the Challenges of Electronics Technology Progress, 2005.*, 240–244. IEEE. http://dx.doi.org/10.1109/ISSE.2005.1491034. Récupéré de http://ieeexplore.ieee.org/document/1491034/

Emekçi, F., Sahin, O., Agrawal, D. et Abbadi, A. E. (2007). Privacy preserving decision tree learning over multiple parties. *Data & Knowledge*. Récupéré de http:

//www.sciencedirect.com/science/article/pii/S0169023X07000365

Espressif (2016). ESP32 Overview. Récupéré de https://espressif.com/en/products/hardware/esp32/overview

Forney, G.D., J. (1973). The viterbi algorithm. *Proceedings of the IEEE*, *61*(3), 268–278. http://dx.doi.org/10.1109/PROC.1973.9030. Récupéré de http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1450960

Friedman, N., Geiger, D. et Goldszmidt, M. (1997). Bayesian Network Classifiers. *Machine learning*, *29*, 131–163. http://dx.doi.org/10.1023/A:1007465528199. Récupéré de http://dx.doi.org/10.1023/A:1007465528199%5Cnpapers2://publication/doi/10.1023/A:1007465528199

Geman, S. et Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on pattern analysis*. Récupéré de http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4767596

Ghayvat, H., Mukhopadhyay, S., Gui, X. et Suryadevara, N. (2015). WSN- and IOT-based smart homes and their extension to smart buildings. *Sensors (Switzerland)*, *15*(5), 10350–10379. http://dx.doi.org/10.3390/s150510350

Giannella, C., Liu, K. et Olsen, T. (2004). Communication efficient construction of decision trees over heterogeneously distributed data. *Data Mining, 2004. ICDM'*. Récupéré de http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1410268

Giroux, S., Leblanc, T., Bouzouane, A., Bouchard, B., Pigot, H. et Bauchet, J. (2009). The Praxis of Cognitive Assistance in Smart Homes. *BMI Book*, 183–211. http://dx.doi.org/10.3233/978-1-60750-048-3-183. Récupéré de http://liara.uqac.ca/giroux.leblanc.bouzouane.bouchard.pigot.bauchet.praxis.cognitive.assistance.pdf

Guo, Z. et Yang, Y. (2015). Exploring server redundancy in nonblocking multicast data center networks. *IEEE Transactions on Computers*, *64*(7), 1912–1926.

Gutierrez, J., Naeve, M., Callaway, E. et ..., M. B. (2001). IEEE 802.15. 4 : a developing standard for low-power low-cost wireless personal area networks. *ieeexplore.ieee.org*.

He, Z. et Jin, L. (2009). Activity Recognition from acceleration data Based on Discrete Consine Transform and SVM.

He, Z.-Y. et Jin, L.-W. (2008). ACTIVITY RECOGNITION FROM ACCELERATION DATA USING AR MODEL REPRESENTATION AND SVM. 12–15.

Heckerman, D. (2010). LEARNING WITH BAYESIAN NETWORKS. *Learning in graphical models*, (September 2002), 1–5. Récupéré de http://link.springer.com/chapter/10.1007/978-94-011-5014-9_11

Hey, A., Tansley, S. et Tolle, K. (2009). The fourth paradigm : data-intensive scientific discovery. Récupéré de http://202.120.81.220:81/inter/uploads/readings/four-paradigm.pdf

Hintjens, P. (2013). *ZeroMQ : messaging for many applications*. Récupéré de https://books.google.ca/books?hl=en&lr=&id=KWtT5CJc6FYC&oi=fnd&pg=PR2&dq=ZeroMQ&ots=Z1AcI6FU4X&sig=q--tmuE_tuFLQk6ZGif-8DjOFeM

Hu, Y., Tilke, D., Adams, T., Crandall, A. S., Cook, D. J. et Schmitter-Edgecombe, M. (2016). Smart home in a box : usability study for a large scale self-installation of smart home technologies. *Journal of reliable intelligent environments*, *2*(2), 93–106.

Hunkeler, U. et Truong, H. (2008). MQTT-S—A publish/subscribe protocol for Wireless Sensor Networks. *systems software and ...*. Récupéré de http://ieeexplore.ieee.org/abstract/document/4554519/

Huynh, T. et Schiele, B. (2005). Analyzing Features for Activity Recognition.

Intille, S. S., Larson, K., Tapia, E. M., Beaudin, J. S., Kaushik, P., Nawyn, J. et Rockinson, R. (2006). Using a Live-In Laboratory for Ubiquitous Computing Research.

Jensen, F. V. (1996). Bayesian networks basics. *AISB quarterly*, 9–22.

Johnson, E. et Kargupta, H. (2000). Collective, hierarchical clustering from distributed, heterogeneous data. *Large-Scale Parallel Data Mining*. Récupéré de http://link.springer.com/chapter/10.1007/3-540-46502-2_12

Kane, J., Tang, B., Chen, Z., Yan, J., Wei, T., He, H. et Yang, Q. (2015). Reflex-Tree : A Biologically Inspired Parallel Architecture for Future Smart Cities. Dans *2015 44th International Conference on Parallel Processing*, 360–369. IEEE. http://dx.doi.org/10.1109/ICPP.2015.45. Récupéré de http://ieeexplore.ieee.org/document/7349591/

Katz, S., Ford, A. B., Moskowitz, R. W., Jackson, B. A. et Jaffe, M. W. (1963). Studies of Illness in the Aged. *JAMA*, *185*(12), 914. http://dx.doi.org/10.1001/jama.1963.03060120024016. Récupéré de http://jama.jamanetwork.com/article.aspx?doi=10.1001/jama.1963.03060120024016

Kautz, H. A. (1991). A Formal Theory of Plan Recognition and its Implementation. (2), 69–126.

King, J. et Jansen, E. (2005). The Gator Tech Smart House. *Computer*, *38*(3), 50–60.

Klusch, M., Lodi, S. et Moro, G. (2003). Agent-based distributed data mining : The KDEC scheme. *Intelligent information agents*. Récupéré de http://link.springer.com/chapter/10.1007/3-540-36561-3_5

Kovashka, A. et Grauman, K. (2010). Learning a Hierarchy of Discriminative Space-Time Neighborhood Features for Human Action Recognition.

Lawton, M. P. et Brody, E. M. (1969). Assessment of Older People : Self-Maintaining and Instrumental Activities of Daily Living. *The Gerontologist*, *9*(3 Part 1), 179–186. http://dx.doi.org/10.1093/geront/9.3{_}Part{_}1.179. Récupéré de http://gerontologist.oxfordjournals.org/cgi/doi/10.1093/geront/9.3_Part_1.179

Lewis, F. (2005). Wireless sensor networks. In *Smart Environments : Technology , Protocols, and Applications* chapitre 2

Li, S., Xu, L. D. et Zhao, S. (2015). The internet of things : a survey. *Information Systems Frontiers*, *17*(2). http://dx.doi.org/10.1007/s10796-014-9492-7

Lin, R.-T., Chin-Shun Hsu, Tee Yuen Chun et Sheng-Tzong Cheng (2008). OSGi-Based Smart Home Architecture for heterogeneous network. Dans *2008 3rd International Conference on Sensing Technology*, 527–532. IEEE. http://dx.doi.org/10.1109/ICSENST.2008.4757162. Récupéré de http://ieeexplore.ieee.org/document/4757162/

Mark, J. (2004). The IEEE 1451.4 Standard for Smart Transducers.

Martin, G., Zurawski, R. et Philips, C. (2006). Trends in embedded systems Opportunities and challenges for System-on-Chip and Networked Embedded Systems technologies in industrial automation. *ABB Review*, *2*.

Merugu, S. et Ghosh, J. (2003). Privacy-preserving distributed clustering using generative models. *Data Mining, 2003. ICDM 2003. Third*. Récupéré de `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1250922`

Merugu, S. et Ghosh, J. (2005). A distributed learning framework for heterogeneous data sources. *Proceedings of the eleventh ACM SIGKDD*. Récupéré de `http://dl.acm.org/citation.cfm?id=1081896`

Messing, R., Paí, C. et Kautz, H. (2009). Activity recognition using the velocity histories of tracked keypoints.

Mitra, J. et Nayak, T. K. (2015). Reconfigurable Concurrent VLSI (FPGA) Design Architecture of CRC-32 for High-Speed Data Communication. Dans *2015 IEEE International Symposium on Nanoelectronic and Information Systems*, 112–117. IEEE. `http://dx.doi.org/10.1109/iNIS.2015.66`. Récupéré de `http://ieeexplore.ieee.org/document/7434408/`

Montenegro, G., Kushalnagar, N., Hui, J. et Culler, D. (2007). Transmission of IPv6 packets over IEEE 802.15. 4 networks.

Moore, G. (1995). Lithography and the future of Moore's law. *SPIE's 1995 Symposium on*. Récupéré de `http://proceedings.spiedigitallibrary.org/data/Conferences/SPIEP/54397/2_1.pdf`

MySQL, A. (2001). Mysql.

Nazerfard, E., Das, B., Holder, L. B. et Cook, D. J. (2010). Conditional random fields for activity recognition in smart environments. Dans *Proceedings of the 1st ACM International Health Informatics Symposium*, 282–286. ACM.

Novák, M. et Binas, M. (2011). An architecture overview of the smart-home system based on OSGi. *SCYR 2011 : 11th Scientific Conference of Young Researchers of Faculty of Electrical Engineering and Informatics Technical University of Ko{š}ice*,

221–224.

Oliver, N., Garg, A. et Horvitz, E. (2004). Layered representations for learning and inferring office activity from multiple sensory channels. *Computer Vision and Image Understanding*, *96*, 163–180. http://dx.doi.org/10.1016/j.cviu.2004.02.004. Récupéré de http://research.microsoft.com/~nuriawww.elsevier.com/locate/cviu

Pärkkä, J., Ermes, M., Korpipää, P., Mäntyjärvi, J., Peltola, J. et Korhonen, I. (2006). Activity Classification Using Realistic Data From Wearable Sensors. *IEEE TRANSACTIONS ON INFORMATION TECHNOLOGY IN BIOMEDICINE*, *10*(1). http://dx.doi.org/10.1109/TITB.2005.856863

Patterson, D. J., Liao, L., Fox, D. et Kautz, H. (2003). Inferring High-Level Behavior from Low-Level Sensors.

Plantevin, V., Bouzouane, A., Bouchard, B. et Gaboury, S. (2018a). A Novel, Distributed and Biological Like Approach for Activity Recognition. Dans *The 15th IEEE International Conference on Ubiquitous Intelligence and Computing*. IEEE.

Plantevin, V., Bouzouane, A., Bouchard, B. et Gaboury, S. (2018b). Towards a More Reliable and Scalable Architecture for Smart Home Environments. *Journal of Ambient Intelligence and Humanized Computing*.

Plantevin, V., Bouzouane, A. et Gaboury, S. (2017). The Light Node Communication Framework : A New Way to Communicate Inside Smart Homes. *Sensors 2017, Vol. 17, Page 2397*, *17*(10), 2397. http://dx.doi.org/10.3390/S17102397. Récupéré de http://www.mdpi.com/231478

Postel, J. (1980). User Datagram Protocol. *RFC*.

Prince, M., Comas-Herrera, M. A., Knapp, M., Guerchet, M. et Karagiannidou, M. M. (2016). World Alzheimer Report 2016 Improving healthcare for people living with dementia coverage, QualIty and costs now and In the future.

Quinlan, J. (1986). Induction of Decision Trees. *Machine Learning*, *1*(1), 81–106. http://dx.doi.org/10.1023/A:1022643204877. Récupéré de http://link.springer.com/10.1023/A:1022643204877

Raspberry (2016). Raspberry Pi. Récupéré de https://www.raspberrypi.org/

Ravi, N., Dandekar, N., Mysore, P. et Littman, M. (2005). Activity recognition from accelerometer data. *Aaai*. Récupéré de http://www.aaai.org/Papers/IAAI/2005/IAAI05-013

Rogers, W. A., Meyer, B., Walker, N. et Fisk, A. D. (1998). Functional Limitations to Daily Living Tasks in the Aged : A Focus Group Analysis. *Human Factors : The Journal of the Human Factors and Ergonomics Society*, *40*(1), 111–125. http://dx.doi.org/10.1518/001872098779480613. Récupéré de http://hfs.sagepub.com/cgi/doi/10.1518/001872098779480613

Roy, P. C., Abidi, S. R. et Abidi, S. S. (2017). Possibilistic activity recognition with uncertain observations to support medication adherence in an assisted ambient living setting. *Knowledge-Based Systems*, *133*, 156–173. http://dx.doi.org/10.1016/J.KNOSYS.2017.07.008. Récupéré de https://www.sciencedirect.com/science/article/pii/S0950705117303246

Roy, P. C., Bouchard, B., Bouzouane, A. et Giroux, S. (2010). Combining Pervasive Computing with Activity Recognition and Learning. *Web Intelligence and Intelligent Agents*, 447–462. http://dx.doi.org/10.5772/8382. Récupéré de http://www.intechopen.com/books/web-intelligence-and-intelligent-agents/combining-pervasive-computing-with-activity-recognition-and-learning%5Cnhttp://cdn.intechopen.com/pdfs-wm/9599.pdf

Roy, P. C., Bouchard, B., Bouzouane, A. et Giroux, S. (2013). Ambient Activity Recognition in Smart Environments for Cognitive Assistance. *International Journal of Robotics Applications and Technologies*, *1*(1), 29–56. http://dx.doi.org/10.4018/ijrat.2013010103. Récupéré de http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/ijrat.2013010103http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/ijrat.2013010103%5Cnhttp://d.scholar.cnki.net/detail/SJCR_U/SJCR14022800394542

Salah, A. A., Kröse, B. J. et Cook, D. J. (2015). Behavior analysis for elderly. In *Human Behavior Understanding* 1–10. Springer.

Schaller, R. (1997). Moore's law : past, present and future. *IEEE spectrum*. Récupéré de http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=591665

Schmidt, C., Sridharan, N. et Goodson, J. (1978). The plan recognition problem : An intersection of psychology and artificial intelligence. *Artificial Intelligence*, *11*(1), 45–83. http://dx.doi.org/10.1016/0004-3702(78)90012-7

Shelby, Z., Hartke, K. et Bormann, C. (2014). The constrained application protocol (CoAP). Récupéré de https://www.rfc-editor.org/info/rfc7252

Shen, J., Lesser, V. et Carver, N. (2003). Minimizing communication cost in a distributed Bayesian network using a decentralized MDP. *of the second international joint conference ...*. Récupéré de http://dl.acm.org/citation.cfm?id=860684

Shvachko, K., Kuang, H., Radia, S. et Chansler, R. (2010). The hadoop distributed file system. Dans *Mass storage systems and technologies (MSST), 2010 IEEE 26th symposium on*, 1–10. Ieee.

Smola, A. J. et Schölkopf, B. (1998). A Tutorial on Support Vector Regression. *Statistics and Computing*, (NC-TR-98-030). Récupéré de http://link.springer.com/article/10.1023/B:STCO.0000035301.49549.88

SparkFun (2016). SparkFun ESP32 Thing. Récupéré de https://www.sparkfun.com/products/13907

Subramanya, A., Raj, A., Bilmes, J. et Fox, D. (2006). Hierarchical Models for Activity Recognition.

Tan, P., He, W.-t., Lin, J., Zhao, H.-m. et Chu, J. (2018). Design and reliability, availability, maintainability, and safety analysis of a high availability quadruple vital computer system. In *China's High-Speed Rail Technology* 481–496. Springer.

Tapia, E. M., Intille, S. S., Haskell, W., Larson, K., Wright, J., King, A. et Friedman, R. (2007). Real-Time Recognition of Physical Activities and Their Intensities Using Wireless Accelerometers and a Heart Rate Monitor.

Tedesco, R., Dolog, P., Nejdl, W., Allert, H. et Austria, I. (2006). Distributed Bayesian networks for user modeling. *IN : ELearn*. Récupéré de http://people.cs.aau.dk/~dolog/pub/elearn2006.pdf

Telegram (2013). Telegram Protocol. Récupéré de https://core.telegram.org/mtproto

United Nations, Department of Economic and Social Affairs et Population Division (2015). *World Population Ageing 2015*. Rapport technique

Vaidya, J. et Clifton, C. (2003). Privacy-preserving k-means clustering over vertically partitioned data. *Proceedings of the ninth ACM SIGKDD international*. Récupéré de http://dl.acm.org/citation.cfm?id=956776

Van Kasteren, T. L. M., Englebienne, G. et Kröse, B. J. A. (2011). Hierarchical Activity Recognition Using Automatically Clustered Actions. *Springer*. Récupéré de https://sites.google.com/site/tim0306/

Videla, A. et Williams, J. (2012). RabbitMQ in action : distributed messaging for everyone. Récupéré de http://cds.cern.ch/record/1528001

Weiser, M. (1991). The Computer for the 21st Century. *Scientific American*, *265*(3), 94–104. http://dx.doi.org/10.1038/scientificamerican0991-94. Récupéré de http://www.nature.com/doifinder/10.1038/scientificamerican0991-94

Witten, I. H., Frank, E., Hall, M. A. et Pal, C. J. (2016). *Data Mining : Practical machine learning tools and techniques*. Morgan Kaufmann.

Wright, R. et Yang, Z. (2004). Privacy-preserving Bayesian network structure computation on distributed heterogeneous data. *Proceedings of the tenth ACM SIGKDD international*. Récupéré de http://dl.acm.org/citation.cfm?id=1014145

Xia, L. et Aggarwal, J. K. (2013). Spatio-Temporal Depth Cuboid Similarity Feature for Activity Recognition Using Depth Camera. http://dx.doi.org/10.1109/CVPR.2013.365

XMPP (2016). XMPP. Récupéré de https://xmpp.org/

Yamanishi, K. et Kenji (1997). Distributed cooperative Bayesian learning strategies. Dans *Proceedings of the tenth annual conference on Computational learning theory - COLT '97*, 250–262., New York, New York, USA. ACM Press. http://dx.doi.org/10.1145/267460.267507. Récupéré de http://portal.acm.org/citation.cfm?doid=267460.267507

ZeroMQ (2016). Zeromq. Récupéré de http://zeromq.org/

Zhihua, S. (2016). Design of Smart Home System Based on ZigBee. *2016 International Conference on Robots & Intelligent System (ICRIS)*, 167–170.

Ziaeefard, M. et Bergevin, R. (2015). Semantic human activity recognition : A literature review. *Pattern Recognition*, *48*(8), 2329–2345. http://dx.doi.org/10.1016/J.PATCOG.2015.03.006. Récupéré de https://www.sciencedirect.com/science/article/pii/S0031320315000953

Zou, Z., Li, K.-J., Li, R. et Wu, S. (2011). Smart Home System Based on IPV6 and ZIGBEE Technology. *Procedia Engineering*, *15*, 1529–1533.