De l'IoT à l'IoT-a : une approche pour des communications dynamiques : interactions autour d'un mur d'écrans connectés

Alexandre Schmitt

UNIVERSITE BRETAGNE LOIRE / EDUCATION, LANGAGES, INTERACTIONS, COGNITION, CLINIQUE

Le Mans Université

THÈSE DE DOCTORAT

LE MANS UNIVERSITÉ

ÉCOLE DOCTORALE N° 603
Éducation, Langages, Interaction, Cognition, Clinique
Spécialité : Informatique

Par
Alexandre SCHMITT

De l'IoT à l'IoT-a : une approche pour des communications dynamiques

Interactions autour d'un mur d'écrans connectés

Thèse présentée et soutenue à Le Mans, le 19 octobre 2020
Unité de recherche : CREN – Centre de Recherche en Éducation de Nantes
Thèse N° : 2020LEMA1034

Rapporteurs avant soutenance :

Marie-Pierre Gleizes	Professeure, Université Toulouse III - Paul Sabatier
Fabienne Nouvel	Maître de conférences HDR, INSA de Rennes

Composition du Jury :

Président :	Olivier Simonin	Professeur, INSA de Lyon
Examinateurs :	Marie-Pierre Gleizes	Professeure, Université Toulouse III - Paul Sabatier
	Fabienne Nouvel	Maître de conférences HDR, INSA de Rennes
Dir. de thèse :	Pascal Leroux	Professeur, Le Mans Université
Co-encadrants de thèse :	Florent Carlier	Maître de conférences, Le Mans Université
	Valérie Renault	Maître de conférences, Le Mans Université

Invité :

Frédéric Le Bouguenec, Directeur de la Direction Technique et Innovation, SARP-VEOLIA Nanterre

À mes parents, Fabienne et Jean-François
et mes sœurs, Nadège et Virginie.

Table des matières

Liste des tableaux

Liste des définitions

Résumé

L'essor des objets connectés (Internet of Things, IoT) depuis ces dernières années change profondément nos liens avec la technologie. Les industries connaissent de nouveaux modèles technologiques et économiques pendant que les foyers s'ouvrent à des usages inédits. L'IoT est vu comme une nouvelle opportunité de marché et l'approche adoptée pour la gestion de ces objets repose principalement sur la centralisation des traitements et des données dans des fermes de serveurs. Pourtant de nombreux travaux montrent les atouts offerts par la décentralisation dans ce domaine : meilleure tolérance aux pannes, mobilité accrue, gain en évolutivité, ou encore déploiement facilité. Nos travaux portent sur l'analyse des choix d'organisations afin que les objets connectés puissent porter leurs données et leurs traitements de façon décentralisée. Le paradigme multi-agents offre de multiples leviers facilitant la réponse à cette approche. Il décrit le comportement d'entités autonomes agissant dans un environnement commun. Pour cela, nous proposons une approche permettant d'adapter des agents à la complexité des architectures matérielles. L'objectif est de considérer les objets comme des « Internet of Things - agent », IoT-a. Chaque élément progresse de façon autonome et dispose des moyens suffisants pour interagir avec d'autres afin de créer de nouveaux comportements. Nous nous concentrons sur les problématiques de communication, avec des systèmes embarqués hétérogènes, en présentant un modèle de communication dynamique. Il permet le partage actif d'informations à différents niveaux applicatifs et matériels entre les agents. Nous définissons les notions de centralisation locale et décentralisation collective de constellations d'IoT-a, permettant l'évolution autonome des objets agents dans un environnement proactif. Nous expérimentons notre proposition au travers de la mise en œuvre réelle d'un ensemble d'IoT-a, formant un mur d'écrans, capable de répondre à plusieurs types d'applications de façon décentralisée. Pour sa validation et son évaluation, un scénario se focalise sur l'interopérabilité du système. Le retrait et l'ajout d'écrans, à la volée, durant la diffusion de visuels permet d'adapter les flux vidéos à la surface d'écrans disponible. Plusieurs prototypes d'outils de visualisation dynamique ont été proposés dans le cadre d'un partenariat de thèse CIFRE avec l'entreprise SARP-VEOLIA.

Abstract

The rise of Internet of Things (IoT) in recent years has profoundly changed our relationship with technology. Industries are experiencing new technological and economic models while households are opening up to new uses for them. IoT is seen as a new market opportunity and the approach adopted for the management of these things is mainly based on the centralization of processing and data in datacenter. However, numerous studies have shown the advantages offered by decentralization in this area : better fault tolerance, increased mobility, greater scalability, and easier deployment. Our work focuses on the analysis of organizational choices so that connected things can carry their data and processing in a decentralized way. The multi-agent paradigm offers multiple vehicles to facilitate the response to this approach. It describes the behavior of autonomous entities acting in a common environment. For this, we propose an approach that allows agents to be adapted to the complexity of different hardware architectures. The goal is to consider objects as "Internet of Things - agent", IoT-a. Each element progresses autonomously and has sufficient means to interact with others in order to create new kinds of behavior. We have focused on communication issues, with heterogeneous embedded systems, by presenting a dynamic communication model. This model allows the active sharing of information at different levels of application and hardware between agents. We have defined the notions of local centralization and collective decentralization of IoT-a constellations, allowing the autonomous evolution of things agent in a proactive environment. We have experimented our proposal through the real implementation of a set of IoT-a, forming a wall of screens, capable of responding to several types of applications in a decentralized way. For its validation and evaluation, a scenario focuses on the interoperability of the system. The removal and addition of plug and play screens during the diffusion of visuals allows to adapt the video flows to the screen surface available. Several prototypes of dynamic visualization tools have been proposed within the context of a CIFRE thesis partnership with the company SARP-VEOLIA.

Remerciements

Je remercie respectueusement mon directeur de thèse, le professeur Pascal Leroux, pour ses conseils, ses encouragements et sa confiance qu'il m'a accordé depuis bientôt sept ans, bien avant même ce doctorat.

Je tiens à remercier particulièrement mes encadrants de thèse, Florent Carlier et Valérie Renault, sans qui cette thèse n'aurait pas vu le jour. Ils m'ont, entre autre, guidé et épaulé mais ils m'ont surtout accordé une pleine confiance, me laissant une grande liberté dans le déroulement de cette thèse.

Mes remerciements vont aussi aux membres de mon jury de thèse dont mes rapporteurs Marie-Pierre Gleizes et Fabienne Nouvel ainsi que mon président de jury Olivier Simonin. Merci à eux pour leur disponibilité et pour l'intérêt qu'ils ont porté à mon travail.

Cette thèse CIFRE m'a offert l'opportunité de collaborer et apprendre énormément des équipes que j'ai pu rencontrer dans l'entreprise SARP-Véolia qui m'a accueilli durant ces trois années. Je remercie chaleureusement Jacques Jouanique, directeur d'agence du Mans, et Dominique Lust, directeur opérationnel de secteur grand ouest, pour avoir cru en ce projet de thèse et grâce à qui j'ai toujours bénéficié d'un soutien total.
Je tiens aussi à remercier vivement Frédéric Le Bouguénec, directeur technique et innovation du groupe SARP pour son encadrement toujours bienveillant, son engouement inflexible face à mon travail et aux nombreuses responsabilités qu'il m'a confiées durant cette aventure.
Merci également à Claire Dechelotte, ma responsable industrielle durant ma dernière année de thèse. Je garde un souvenir plaisant de nos heures à monter ce premier mûr d'écrans au sein de l'agence SARP Le Mans.
Je pense aussi à mes collègues de la DTI, Pascal, Fabien, Mathieu et tous les autres, grâce à qui les réunions d'équipe étaient toujours ponctuées de bonne humeur et de convivialité.

Mener un doctorat passe aussi par la vie de laboratoire de recherche et une émulation avec d'autres chercheurs, doctorants et BIATSS. Merci d'ailleurs à Émilie et Séverine, nos supers secrétaires, sur qui j'ai pu compter pour chacune de mes démarches. Merci à Juliette, Moises, Élodie, Pauline, Tetiana, Clément, Jean, Adrien, Loredana et Lorenzo, mes collègues et maintenant amis doctorants, qui m'ont permis de garder la motivation grâce à nos nombreuses blagues toujours plus perfectionnées après chacun de nos déjeuners. Je n'oublie pas non plus Arnaud et nos conversations techniques sans fin, mais ô combien passionnantes. Je remercie tout ceux qui m'ont apporté une aide, un soutien ou même simplement accordé un peu de leur temps pour que je puisse mener à bien ce doctorat.

Merci à ma famille et mes amis pour vous être intéressés à mon travail et m'avoir soutenu tout au long de ce projet.

Merci infiniment à mes parents, toujours présents pour moi, à me féliciter quand tout allait bien et me soutenir quand c'était nécessaire. Je n'en serais pas là sans eux.

Enfin, merci à Catharina, partageant cette aventure du doctorat avec moi, et avec qui la vie est plus belle encore.

Chapitre 1

Introduction

1.1 | Contexte

L'accroissement du nombre des objets connectés (Internet of Things, IoT) dans la société fait naître de nouvelles questions à la croisée de différents domaines. En électronique et microélectronique, les chercheurs doivent réfléchir à de nouveaux modèles de gestion de l'énergie, de puissance de calcul et miniaturisation des fonctions. En chimie, la communauté se concentre activement sur de nouvelles solutions de batteries permettant une meilleure autonomie et des temps de charge plus courts. En radiocommunication, de nombreuses solutions émergent pour communiquer plus vite, plus loin, de façon plus économique en énergie et avec des déploiements simplifiés. Enfin, en informatique de nombreux sujets restent à développer de façon à déployer et utiliser tous ces objets de sorte qu'ils puissent être interopérables, quelles que soient leurs caractéristiques.

Le sociologue et philosophe Edgar Morin a introduit le concept de la pensée complexe dans les années 1980. Selon lui, l'imbrication entre les domaines scientifiques, la transdisciplinarité, permet l'émergence de nouvelles connaissances. Le terme de complexité est pris au sens de son étymologie "complexus" qui signifie "ce qui est tissé ensemble", dans un enchevêtrement d'entrelacements (plexus) [Morin, 1995]. Le sociologue explique que la science sépare les domaines, elle a tendance à délimiter les études pour mieux les analyser. Il estime que cette façon de penser est simplifiante, qu'elle disjoint et réduit les dimensions. La pensée complexe est un processus qui prend en compte la multidimensionnalité d'une situation. Elle s'intéresse à l'hétérogénéité des éléments pour mieux les analyser.

Parallèlement, Joël de De Rosnay, scientifique, propose un nouvel outil, le macroscope, pour analyser nos sociétés qui se complexifient [De Rosnay, 1975]. Il adopte une vision systémique pour analyser les domaines qui nous entourent quotidiennement. Ainsi, il parvient à montrer les liens complexes qu'il peut y avoir entre des sujets très éloignés comme : l'écologie, le système économique, l'entreprise, la ville, l'organisme et la cellule.

Sans prétention de vouloir inscrire cette thèse au même échelon que les études de Morin et Rosnay, nous souhaitons développer dans nos travaux cette perspective qu'ouvrent les sujets de la systémique et de la pensée complexe. En effet, le domaine des objets connectés est particulièrement vaste, tant par les technologies qu'ils implémentent, les secteurs dans lesquels ils interviennent, que par les questions scientifiques, éthiques, sociologiques et politiques qu'ils génèrent. Les études scientifiques sont d'ailleurs en pleine effervescence depuis les dix dernières années [Nauman et al., 2020].

Nous nous intéressons dans cette thèse au triptyque : matériel, communication et application. Nous pensons que ces trois composants sont indissociables, car ils sont intrinsèquement liés les uns aux autres. Une application ne peut être réalisée que par un objet disposant du matériel suffisant. À l'inverse, un objet est développé avec du matériel ne pouvant répondre qu'à un certain nombre d'applications. Nous retrouvons d'ailleurs au travers de ce lien entre matériel et application, le principe récursif énoncé dans le livre [Morin, 1986], analysant les liens de causalité entre les fonctions d'un système, au travers de l'exemple des êtres humains qui forment la société et la société qui façonne les individus.

Nous observons qu'il est nécessaire de proposer une solution assurant le développement de nouvelles applications respectant à la fois : les contraintes matérielles fortes du système embarqué, les contraintes de communication, mais surtout, le respect des fonctions essentielles à l'objet lui-même. Néanmoins, nous avons conscience des enjeux économiques actuels du domaine des objets connectés. Un très grand nombre d'entreprises proposent des IoT ayant chacun des caractéristiques pouvant satisfaire un certain nombre de fonctions. Le marché propose aujourd'hui un panel important d'objets et le développement de nouveaux éléments ne semble pas être la solution idéale. Pourquoi ne pas utiliser les objets déjà existants ? Pourquoi ne pas adapter le logiciel aux matériels à la place de la conception de nouveaux matériels ?

Nous souhaitons proposer des solutions se satisfaisant du matériel préexistant, prenant en compte les contraintes de celui-ci et celles des autres éléments de notre triptyque.

L'IoT forme donc une société numérique constituée d'éléments capables d'interagir entre eux s'ils sont correctement organisés. Ces éléments ont la capacité de capter et d'agir sur leur environnement grâce à leurs périphériques. Lorsque nous prenons du recul, cette société s'apparente à celles que nous rencontrons dans le domaine du vivant. Les organisations sociales comme les ruches, les fourmilières, les meutes ou encore les sociétés humaines reposent sur des principes semblables, mais qui paraissent plus évolués. Les éléments sont doués de comportements plus ou moins complexes comme butiner, couver et protéger. Ils interagissent, marchandent ou négocient pour établir des compromis. Ils captent et agissent sur le monde qui les entoure, en pollinisant, en chassant, en construisant, etc. Pourquoi les IoT ne pourraient-ils pas s'organiser pour former une société numérique douée d'une intelligence collective ?

1.2 | Problématique

S'il est un domaine de l'informatique qui s'intéresse à cette question d'intelligence collective, c'est bien le paradigme multi-agents. Nous le détaillons dans notre troisième chapitre, mais il est important de définir brièvement le concept d'agent et de système multi-agents pour mieux illustrer notre propos dans la suite de cette problématique. Un agent est un élément matériel ou immatériel capable de réaliser des fonctions simples en ayant une vision limitée de son environnement. Il évolue avec d'autres agents et a la faculté d'échanger des intentions ou des informations avec eux. Un système multi-agents (SMA) englobe l'ensemble des agents qui évoluent dans un environnement commun.

Le paradigme multi-agents est un sujet qui gagne en importance depuis quelques années dans le domaine des objets connectés et s'insère dans différents travaux [Mzahm et al., 2013] [Mihailescu et al., 2017] [Yu et al., 2013]. De nombreux leviers sont encore à actionner pour permettre aux objets connectés de former des sociétés douées d'intelligence. Nous pensons que le développement du paradigme multi-agents au sein du domaine des objets connectés ouvre la possibilité de nouveaux champs d'applications. Nous pouvons notamment citer les travaux conjoints de [Carlier and Renault, 2016] et [Renault et al., 2017] concernant l'approche *Internet of Things - agent*, qu'ils notent *IoT-a*. Dans ces travaux, chaque objet porte un ou plusieurs agents. L'objet est responsable d'une "partie" de l'intelligence globale portée par l'ensemble des agents. Les agents, quant à eux, sont formalisés de façon à pouvoir respecter les contraintes qu'impose le matériel de l'objet connecté. Notons que les auteurs, entendent par le terme "un IoT-a", la désignation d'un objet connecté pouvant intégrer un ou plusieurs agents.

L'approche IoT-a s'appuie sur le domaine des systèmes embarqués pour formaliser certaines solutions en matière de communication et d'interaction entre les objets-agents. Cette approche nous intéresse tout particulièrement, car elle aborde les trois domaines de notre triptyque. Le concept IoT-a prend en compte l'aspect matériel par la mise en place d'outils bas niveau, respectueux des caractéristiques limitées des objets. En outre, la communication repose sur des échanges simples, robustes et légers permettant l'interopérabilité des objets et l'émergence de comportements sociaux. Néanmoins, l'approche IoT-a n'est, pour le moment, qu'une étude préliminaire qui ébauche les grandes lignes d'un sujet qui mérite d'être approfondi.

La diversité des architectures matérielles rencontrées dans les objets connectés montre l'hétérogénéité à laquelle les agents de l'approche IoT-a doivent répondre. La modélisation et l'intégration de SMA dans de tels systèmes doivent prendre en compte les contraintes liées aux systèmes embarqués, tout en étant suffisamment ouvertes pour s'adapter à différents environnements matériels.

Nous pouvons désormais introduire notre problématique issue des éléments que nous venons de présenter :

Problématique

Comment faire évoluer et adapter des constellations d'IoT-a à la complexité des architectures matérielles des objets connectés, pour satisfaire un cadre applicatif ?

Nous évoquons dans cette problématique la notion de constellation d'IoT-a. Bien des définitions se cachent derrière le mot constellation. Il est bien évident que la première à laquelle nous pensons est la définition astronomique, la caractérisant comme étant un *"groupe d'étoiles formant une figure plus ou moins précise"*[1]. La notion de constellation d'IoT-a nous semble intéressante, car elle permet d'introduire les concepts identifiés dans les définitions suivantes, issues des mêmes sources : *"éparpillement d'objets prosaïques et disparates"*, ainsi que : *"ensemble de choses abstraites ayant entre elles une certaine relation"*. Nous pouvons donc expliquer cette idée par la définition suivante :

Définition 1 : Une constellation d'IoT-a

Une constellation d'IoT-a est un ensemble d'IoT-a répartis dans différents espaces, matériels ou virtuels, ayant entre eux une certaine relation.

Notre problématique évoque la *satisfaction d'un cadre applicatif* et il nous semble nécessaire de préciser cette idée. En effet, nous entendons par le terme *cadre applicatif*, un ensemble d'éléments ou fonctions définies dans un cahier des charges. Nous proposons alors la définition suivante :

Définition 2 : Satisfaction d'un cadre applicatif

La satisfaction d'un cadre applicatif est la prise en compte fonctionnelle de l'ensemble des éléments ou fonctions définies dans un cahier des charges.

1.3 | Contribution visée

Nous ciblons, au travers de notre problématique, un travail autour des trois éléments précédemment présentés : le matériel, la communication et les applications. Dans un premier temps, nous allons analyser les architectures et la composition des objets connectés existants, afin de déterminer quels sont les éléments qui caractérisent leur hétérogénéité. Nous nous intéressons aussi, aux technologies de communication permettant actuellement aux objets connectés d'échanger des données tout en respectant les contraintes des systèmes embarqués. Nous nous concentrons alors dans un second temps sur le paradigme multi-agents et les leviers qu'ils apportent pour faire émerger de ces objets connectés, de nouvelles applications issues de leurs interactions.

1.3.1 | *Hypothèses*

Hypothèse 1

L'intégration d'un ou plusieurs agents au sein d'un objet connecté, répartis à différents niveaux matériels ou logiciels, permet la réponse à des besoins applicatifs spécifiques.

La première hypothèse que nous avançons se concentre sur l'association du paradigme multi-agents avec le domaine des objets connectés. Elle guide notre proposition en précisant les éléments permettant la migration d'un IoT vers IoT-a. Nous notons la question de différenciation de niveaux matériel et logiciel. Elle sous-entend l'hétérogénéité des objets connectés et les difficultés rencontrées pour organiser et faire communiquer des agents à l'intérieur de chaque objet, mais aussi entre les objets eux-mêmes.

1. Centre Nationale de Ressources Textuelles et Lexicales, `https://www.cnrtl.fr/definition/constellation`, consultation 2019

Hypothèse 2

Une organisation en constellations des IoT-a permet leur adaptation au contexte matériel ou logiciel des objets connectés et ce quelque soit le cadre applicatif.
La centralisation locale d'agents donne lieu à des interactions intra-objet alors que la décentralisation collective permet des échanges inter-objets.

Notre seconde hypothèse se focalise sur notre proposition d'organisation en constellation. Nous souhaitons, par cette proposition, homogénéiser les interactions entre les agents répartis dans les objets. Nous proposons pour cela, de distinguer les interactions intra et inter-objets par les notions de centralisation locale et décentralisation collective.

1.3.2 | *Contexte industriel*

Cette thèse s'est effectuée dans un environnement industriel par l'intermédiaire d'une convention CIFRE avec l'entreprise SARP-VEOLIA. Elle est un acteur majeur des travaux d'assainissement et de l'entretien de canalisations sur tout le territoire français. L'entreprise intervient chez les particuliers, les professionnels, l'industrie et les collectivités. Elle souhaite bénéficier d'un outil de visualisation dynamique pour son activité.

Nous expérimentons donc nos propositions au travers de la mise en œuvre d'un ensemble d'IoT-a, formant un mur d'écrans, capable de répondre à plusieurs types d'applications de façon décentralisée. L'objectif pour l'entreprise est de bénéficier d'un outil adapté à ses besoins de supervision et de contrôle d'activités. Le nôtre, complémentaire, permet d'implémenter, intégrer et tester nos hypothèses sur un système réel. Trois preuves de concept se focalisent alors sur l'interopérabilité et la résilience du système.

1.4 | Structure du mémoire

Ce manuscrit se compose ensuite de trois parties principales. Lors de cette introduction, nous avons présenté les problématiques liées aux deux domaines principaux de cette thèse ainsi que la problématique de recherche globale.

Dans la partie I, nous analysons les deux domaines propres à cette thèse. Le premier chapitre se concentre sur les objets connectés. Nous présentons tout d'abord, une revue des secteurs dans lesquels l'IoT change les mœurs et modèles de fonctionnement. Nous nous attardons sur certaines questions en matière de gestion de ces objets, puis nous présentons le principe de résilience et ses mécanismes. Nous détaillons ensuite les technologies internes rencontrées dans les objets connectés. Les moyens de communication sont aussi abordés afin de déterminer quelles solutions sont envisageables pour la communication intra et inter-systèmes. Nous développons ensuite les spécificités des systèmes multi-agents en revenant sur les différentes formalisations théoriques et pratiques de la littérature. Nous dressons un aperçu de quelques-uns des projets de recherche, ayant pour objectif de relier les domaines des objets connectés et des systèmes multi-agents. Plusieurs organisations de système multi-agents sont ensuite présentées. Elle permet d'aboutir à certaines conclusions validant notre hypothèse d'une organisation en constellation d'IoT-a. Nous terminons ce chapitre par la présentation de plusieurs normes de la *Foundation for Intelligent Physical Agent*, nous permettant de détailler nos choix relatifs à nos propositions de centralisation locale et décentralisation collective.

Dans la partie II, nous analysons les différents points de vues de l'IoT-a. Le premier chapitre se concentre sur nos propositions. La première section détaille nos travaux concernant le passage de l'IoT vers l'IoT-a, pour finir par dresser le modèle de l'IoT-a. Dans la seconde section, nous nous focalisons sur la notion de constellation permettant l'organisation des agents dont nous présentons le modèle. Nous nous attardons sur la communication des agents et nous présentons notamment nos propositions permettant la mise en place d'un système connecté, dynamique et résilient. Dans le second chapitre, nous présentons un cas d'usage des IoT-a dans le cadre de l'entreprise. Nous détaillons le contexte industriel, les besoins et les éléments mis en œuvre.

La partie III reprend ces contributions et propose les éléments nécessaires à nos expérimentations. Nous présentons, dans un premier temps, la plate-forme SMA Triskell3S nous permettant de mettre en œuvre nos modèles de l'IoT-a et de la constellation d'IoT-a. Puis, nous exposons trois expérimentations. La première a pour objectif de montrer la pertinence de notre proposition de communication dynamique. La seconde et la troisième mettent en pratique des agents à différents niveaux matériel et logiciel pour valider nos hypothèses.

À travers la conclusion, nous récapitulons nos contributions et analysons nos résultats vis-à-vis de nos hypothèses et de notre problématique. Nous ouvrons ensuite les perspectives des recherches menées tant au niveau de l'approfondissement de nos travaux qu'au niveau de leur extension à d'autres domaines d'application que nous avons abordés durant cette thèse.

Bibliographie

[Carlier and Renault, 2016] Carlier, F. and Renault, V. (2016). Iot-a, embedded agents for smart internet of things : Application on a display wall. In *2016 IEEE/WIC/ACM International Conference on Web Intelligence, The First International Workshop on the Internet of Agents (IoA)*, pages 80–83, Omaha, Nebraska, USA. IEEE Computer Society.

[De Rosnay, 1975] De Rosnay, J. (1975). *Le macroscope, vers une vision globale*. Editions du Seuil.

[Mihailescu et al., 2017] Mihailescu, R.-C., Spalazzese, R., Davidsson, P., and Heyer, C. (2017). A role-based approach for orchestrating emergent configurations in the internet of things. *Internet of Agents Workshop 2017 (IoA@AAMAS 2017)*.

[Morin, 1986] Morin, E. (1986). *La méthode : La connaissance de la connaissance 3*. Éditions du Seuil, Paris.

[Morin, 1995] Morin, E. (1995). La stratégie de reliance pour l'intelligence de la complexité. In *Revue Internationale de Sytémique*, volume 9 of *2*, pages 105–112.

[Mzahm et al., 2013] Mzahm, A. M., Ahmad, M. S., and Tang, A. Y. C. (2013). Agents of things (aot) : An intelligent operational concept of the internet of things (iot). In *2013 13th International Conference on Intellient Systems Design and Applications*, pages 159–164.

[Nauman et al., 2020] Nauman, A., Qadri, Y. A., Amjad, M., Zikria, Y. B., Afzal, M. K., and Kim, S. W. (2020). Multimedia internet of things : A comprehensive survey. *IEEE Access*, 8 :8202–8250.

[Renault et al., 2017] Renault, V., Carlier, F., and Schmitt, A. (2017). Framework sma pour visualisation multi-écrans. In *Journée Interaction Homme-Machine et Intelligence Artificielle*, Université Pierre et Marie-Curie, Paris, France.

[Yu et al., 2013] Yu, H., Shen, Z., and Leung, C. (2013). From internet of things to internet of agents. In *Proceedings of the 2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*, GREENCOM-ITHINGS-CPSCOM '13, pages 1054–1057, Washington, DC, USA. IEEE Computer Society.

Partie I

État de l'art

Introduction

L'objectif de cette partie est de dresser un tableau détaillé de ce qui compose les domaines des objets connectés et des systèmes multi-agents. Notre problématique soulève la question de l'adaptation et de l'évolution de l'approche IoT-a par rapport aux facettes techniques des IoT. Nous détaillerons l'approche IoT-a dans les prochains chapitres de cette thèse, mais pour mieux saisir notre démarche dans les sections qui vont suivre nous nous devons d'en dresser un aperçu.

L'IoT-a est l'association du domaine des objets connectés avec celui des systèmes multi-agents. Cette approche, proposée dans des travaux préliminaires à cette thèse [Renault and Carlier, 2016], a pour but de mener des réflexions sur des architectures pouvant conduire à de l'intelligence collective. Nous souhaitons poursuivre cette idée et proposer des leviers concernant certains aspects, notamment sur les questions de l'adaptation, de l'organisation et de l'évolution des agents dans les objets.

Nous nous devons, pour cela, de détailler les caractéristiques techniques des objets connectés, savoir de quoi ils sont faits, comment ils s'organisent et comment ils communiquent. Nous présentons donc, dans le premier chapitre, les différentes technologies permettant l'émergence de nouveaux objets, dits connectés. Nous détaillons leurs applications au travers de plusieurs exemples. Nous nous intéressons à leurs constituants internes et leurs organisations. Nous terminons en analysant les méthodes de communication mises en œuvre.

Dans le deuxième chapitre, nous abordons le second domaine des IoT-a : les Systèmes Multi-Agents (SMA). Comme pour les objets connectés, les SMA s'organisent, communiquent et disposent de mécanismes leur permettant de s'adapter à certaines situations. Nous analysons les travaux dans la littérature afin de mieux appréhender ce domaine et examinons les nombreux projets alliant les systèmes connectés ou réels et les systèmes multi-agents. Nous précisons les normes établies pour la communication et l'organisation des agents, afin de comprendre comment ces derniers peuvent s'adapter à l'hétérogénéité des technologies des objets connectés.

Chapitre 2

Les objets connectés

Le terme d'objet connecté est de plus en plus rencontré dans de nombreux secteurs comme l'énergie, les transports, la logistique, l'agriculture, l'alimentation, la finance, les soins de santé, la construction, les villes et bâtiments intelligents, pour n'en citer que quelques-uns. Derrière ce terme se cache un ensemble d'éléments connectés par une liaison filaire ou des signaux hertziens, échangeant des informations de natures diverses et pouvant mesurer et/ou agir sur le monde extérieur.

2.1 | Omniprésence des objets connectés

Un grand nombre d'entreprises évoquent une évolution exponentielle du nombre d'objets dans les 30 prochaines années étant données les applications compatibles avec ce domaine.

Le domaine des bâtiments intelligents ou de la ville intelligente (smart buildings ou smart city) a pour but de rendre plus efficientes les ressources nécessaires à la vie d'une population importante dans un espace restreint [Sebastian et al., 2018] [Vlacheas et al., 2013] [Cao and Liu, 2016]. La collecte d'informations pour l'énergie, l'eau, la santé, le transport, le commerce ou encore l'éducation et les services publics sont des indicateurs qui interviennent dans ce domaine. Toutes ces informations sont recoupées puis modélisées afin d'extraire des utiles permettant de réduire certains effets néfastes de la ville sur l'environnement. Elles donnent lieu, entre autres, à la réduction du trafic routier, l'amélioration des échanges de coursiers, ou encore l'augmentation des rendements des distributions d'énergies. Le concept de *smart city* est de plus en plus plébiscité par les grandes métropoles à travers le monde malgré des contraintes techniques souvent coûteuses, étant donné la complexité des mises en place dans les réseaux existants et le nombre d'équipements nécessaires.

La mesure d'activités concernant la santé touche les particuliers plus précisément et devient un secteur de vente de plus en plus en vogue dans le domaine du sport et du bien-être. De nombreux appareils permettent la mesure de différents paramètres corporels [Coyle et al., 2010] : la fréquence cardiaque, la tension, la température, la position, etc. Le suivi d'activités comme le sommeil, le sport ou la marche effectuée dans une journée est proposé par des fabricants de montres ou de bracelets connectés. Tous ces objets sont généralement reliés à une ou plusieurs stations de base servant de concentrateur de données : un téléphone, une tablette ou un ordinateur personnel. Le regroupement de ces données donne la possibilité aux personnes d'accéder à des statistiques sur leur "bien-être" quotidien. Il va de soi que ce domaine pose de nombreux problèmes d'éthique et de sécurité, car il concerne au plus près les informations personnelles de chaque utilisateur. Nous pouvons notamment citer l'article paru en juillet 2018 [Tan, 2018] relatant le problème de géolocalisation de troupes militaires révélées publiquement sur le site du fabricant de montres connectées utilisées durant leur entraînement physique quotidien dans un pays en guerre. Quand bien même les noms des militaires n'étant pas révélés, les traces mises à disposition étaient suffisamment caractéristiques pour comprendre qu'il s'agissait des chemins de ronde de certaines bases militaires disposées dans des endroits normalement tenus secrets. Ces problèmes sont analysés dans certaines études [Mendoza et al., 2018], montrant qu'aujourd'hui de nombreux leviers sont encore à actionner pour permettre une utilisation sécurisée des données de chacun avec les objets connectés.

Le secteur des transports se tourne de plus en plus vers l'IoT lui aussi. Les développements du véhicule autonome prennent en compte les données de la route, de la ville et des autres conducteurs [D'Angelo et al., 2017]. Les véhicules communiquent en temps réel afin de trouver des compromis sur les itinéraires à suivre en fonction du trafic, de la météo ou de travaux. Ils se préviennent par des envois de messages localisé avant même de se détecter physiquement, afin d'anticiper leurs vitesses avant une intersection [Wu et al., 2018].

Enfin, les objets connectés touchent tous les appareils du quotidien utilisés dans une habitation : électroménager, télévision, téléphone, meuble, jardin, volets, pièces, etc. L'hyper connexion de tous ces éléments permet d'en faire émerger une utilisation plus efficace et efficiente [Kodali et al., 2016]. Ainsi, un réfrigérateur connecté est en mesure de déterminer les aliments manquant pour le quotidien de ses utilisateurs et propose automatiquement la liste de courses au drive préféré du foyer. La sécurité des logements en bénéficie aussi avec l'utilisation de capteurs plus ou moins évolués pour la détection d'intrusion, capteurs de franchissement, capteur de contact, capteur de mouvement, analyse d'images depuis des caméras, capteur acoustique, etc.

2.2 | Challenges du domaine des objets connectés

Le principal problème rencontré aujourd'hui est la méthode d'interconnexion de l'ensemble de ces objets. En effet, chaque constructeur développe des solutions technologiques propriétaires souvent incompatibles avec d'autres constructeurs. Les consommateurs se voient alors dans l'obligation d'investir toujours chez le même fabricant pour bénéficier d'une solution globale d'objets connectés dans leur vie quotidienne. Les modèles économiques [Leminen et al., 2012] de ces constructeurs tendent à se généraliser vers une centralisation du service. Ils mettent sur le marché des objets connectés à faible ou très faible coût avec un contrat commercial proposant un paiement mensuel pour la mise à disposition des données aux utilisateurs et une assurance de la sécurité de leurs données. Tous les objets renvoient leurs résultats via internet à un service central et fermé, appartenant au constructeur. L'intelligence intrinsèque des objets est faible et tous les traitements sont déportés dans une ferme de serveurs distante. Les utilisateurs ne maîtrisent pas les informations collectées à leurs sujets et ne savent pas où, comment et par qui sont traitées leurs données.

Sur un plan structurel, une gestion centralisée d'un grand nombre de données des utilisateurs d'un pays par une nation étrangère, expose ce pays à une souveraineté politique de la part de cette nation étrangère [Liaropoulos, 2017]. Toutes les informations d'entreprises ou de panels d'habitants sont collectées, traitées et analysées par une force étrangère pouvant mener à des adaptations politiques, économiques ou militaires à la défaveur des pays souches dépossédés de leurs informations. Aussi, ces modèles centralisés soulèvent de nombreuses réflexions politiques et économiques qui dépassent le cadre de la recherche scientifique. Néanmoins, nous pouvons nous poser certaines questions, d'un point de vue scientifique, afin d'aborder ce sujet sous l'axe de la décentralisation plutôt que la centralisation. Comment proposer une gestion plus décentralisée des objets connectés ? Comment rendre un écosystème d'objets connectés plus intelligent localement sans un traitement centralisé complexe ? La collaboration entre objets peut-elle faire émerger de nouveaux modèles de communication plus ouverts et plus proches des utilisateurs ?

2.3 | Résilience dans les objets connectés

Nous avons soulevé quelques-uns des risques politiques, éthiques et économiques qu'entraînent la mise en œuvre des objets connectés, mais nous devons aussi nous concentrer sur les risques techniques qu'impliquent de tels systèmes. Cette section s'intéresse à un sujet encore peu abordé pour le domaine de l'IoT dans la littérature scientifique [Delic, 2016]. Il est pourtant en pleine expansion et nous nous devons de le détailler, afin de le prendre en compte dans notre approche IoT-a.

La résilience est un terme utilisé dans de nombreux domaines. Les premiers rencontrés sont la psychologie et la psychiatrie dans lesquelles la médecine s'intéresse aux mécanismes de l'humain à se remettre d'un traumatisme, d'un choc ou de la perte d'un proche [Cyrulnik et al., 2012]. En écologie et en étude de la biodiversité, les travaux se concentrent sur la capacité d'un écosystème à se remettre de perturbations naturelles ou humaines. À titre d'exemple, nous pouvons évoquer le cas d'une espèce de plantes, le *papaver californicum* [Joachim et al., 2011], capable de repousser dans les forêts après un feu. Elle est dite pyrophyte. La chaleur du feu fait éclater les graines de la plante qui finit par pousser et végétaliser de nouveau des lieux arides, ayant été ravagés par les flammes.

La résilience se retrouve aussi dans le domaine de l'économie et de la finance, notamment dans le cas de systèmes financiers capables de résister à des crises politiques, économiques ou des changements technologiques profonds [Reggiani et al., 2002]. Enfin, nous retrouvons la résilience dans le secteur de l'ingénierie avec l'étude des systèmes et de leurs limites [Hollnagel et al., 2006]. Les systèmes sensibles comme le transport de passagers, les applications militaires, spatiales, énergétiques et informatiques doivent résister à des perturbations importantes et réagir convenablement à des scénarios de fonctionnement inhabituels ou critiques.

C'est donc sur ce dernier domaine de l'ingénierie des systèmes que nous nous concentrons et nous faisons cas plus particulièrement de la résilience dans les objets connectés [Delic, 2016]. Malgré la dimension technique de ce domaine, il est possible d'établir notre définition de la résilience.

Définition 3 : La résilience

La résilience est la capacité d'un système à reprendre une activité satisfaisant les utilisateurs après un changement anormal ou profond de son environnement habituel.

L'une des meilleures illustrations de la résilience dans le domaine de l'ingénierie et plus particulièrement dans les systèmes embarqués interconnectés est l'amerrissage sans encombre de l'Airbus A310 du vol US Airways 1549, le 15 janvier 2009. Dans cet événement, l'avion a rencontré à quelques centaines de mètres après le décollage, un groupe de bernaches du Canada qui a fini sa course dans les deux moteurs de l'appareil. Les moteurs ont pris feu et se sont arrêtés successivement. Après quelques minutes de vol et à faible altitude, l'avion s'est retrouvé sans aucune poussée au-dessus de la ville de New York. Le pilote a alors pris la décision d'amerrir dans le fleuve Hudson river, ce qu'il réussit avec succès, sauvant tous les passagers et les membres d'équipage.

Selon Jean Pariès [Pariès, 2013], cet événement souvent annoncé comme un miracle, n'est que la somme de la résilience de tous les systèmes mécaniques et électroniques à bord de l'avion, ainsi que le suivi scrupuleux des procédures. En effet, sans ses moteurs un avion n'a plus d'électricité, c'est donc le générateur de secours présent dans la queue de l'appareil qui a pris le relais automatiquement. Le feu qui s'est déclenché dans les moteurs a été arrêté par les extincteurs automatiques présents dans les ailes. Avant l'amerrissage, le contrôleur de vol a prévenu les autorités maritimes qu'un avion allait se poser sur l'Hudson. Durant l'amerrissage, l'avion ne s'est pas disloqué, car la structure mécanique du fuselage est prévue pour ces situations (jusqu'à une certaine limite). Enfin, l'avion n'a pas coulé une fois amerri, car l'avionneur Airbus propose un système fermant toutes les voies d'eau d'un avion favorisant une flottaison durant plusieurs minutes, suffisante pour l'évacuation des passagers. C'est donc selon Pariès, la somme des éléments techniques résilients, de procédures résilientes suivies par le pilote, le contrôleur et l'équipage de l'avion qui ont contribué au succès de cet événement.

Nous pouvons alors nous poser la question suivante : quel est l'intérêt de la résilience dans le domaine des objets connectés ? D'un point de vue systémique, l'évolution du nombre d'objets connectés dans les prochaines années nous laisse envisager l'émergence de systèmes à très grande échelle. Plusieurs millions d'objets constamment en fonctionnement et interdépendants finissent par avoir des comportements sociaux proches de certains systèmes vivants [Delic, 2016]. De tels systèmes sont difficiles à superviser et à contrôler et lors de perturbations des effets de seuil sont susceptibles d'apparaître avant une chute globale du système. Par conséquent, la résilience des objets connectés est une question de première importance.

Avec la résilience, les IoT seraient plus résistants aux stress issus de leur environnement. La littérature explique que l'une des méthodes pour rendre un système résilient est qu'il puisse s'adapter à son environnement [Hollnagel et al., 2006]. Cette adaptation résulte d'une évolution dynamique des compétences [Chouhan et al., 2009]. En l'occurrence, la communication est un des points sur lesquels les objets connectés peuvent aujourd'hui gagner en dynamisme. Pour rendre les objets plus résilients, il est nécessaire de créer des solutions dynamiques de communication mettant en œuvre certaines redondances. Nous pouvons notamment citer les travaux de [Sándor et al., 2015] qui s'intéressent à l'augmentation de la résilience des objets connectés en multipliant les canaux de communication et leur permettant de les adapter en fonction de leur environnement et de la qualité de communication existante.

Plus concrètement, la résilience des communications pour les objets connectés passe, avant tout, par la gestion des erreurs. Dans les communications sans fil, les erreurs sont fréquentes et peuvent être le résultat de nombreux facteurs. La littérature propose certaines méthodes pour le traitement d'une erreur de communication, notamment [Pujolle, 2006a] qui le décompose en quatre étapes :

— **la détection** est le premier élément déclencheur de la gestion d'une erreur. Elle peut être issue de différents vecteurs :

- Les traces d'erreurs ou messages d'erreurs émis par le système qui est encore capable de communiquer, mais qui stipule certains éléments en défauts.
- Les tests effectués sur un ou plusieurs éléments. Tout ou partie d'un système peut être responsable de la mise en place de tests pour s'assurer de l'intégrité des éléments. Les tests ou diagnostics sont cycliques et nécessitent une base de connaissances pour déterminer la gravité d'une erreur.
- Les seuils permettent, au travers d'une limite ou d'un curseur, de détecter des comportements anormaux afin d'émettre des alarmes. Une communication peut disposer de seuils lui permettant de déterminer si un hôte a bien reçu un message. Si le temps de réponse de la part de l'hôte est trop important au-delà d'un seuil, il est alors le reflet d'une communication erronée.

— **la localisation** permet d'obtenir le plus souvent des informations plus précises concernant l'erreur,

— **la réparation** prend en compte les informations de détection et de localisation pour déterminer la meilleure stratégie,

— **la confirmation** rend compte de l'état du système après réparation et donne, le cas échéant, les nouveaux éléments dus à la réparation aux autres systèmes alentour.

Nos propositions pour l'IoT-a souhaitent donc répondre à ce critère de résilience. Comme nous venons de le voir, ce domaine est encore nouveau pour les applications qui concernent les objets connectés. Nous nous concentrons donc principalement sur les questions concernant les communications dynamiques. Les objets connectés forment par nature un système décentralisé. En effet, chaque objet est autonome et répond localement à un objectif. Cette gestion est en opposition à une gestion centralisée où une unité centrale traite l'ensemble des objectifs. Intuitivement, nous pourrions penser que la décentralisation augmente naturellement la résilience d'un système, car un élément isolé est moins impacté par la panne d'un des autres éléments. Néanmoins, la communication entre les objets fait qu'ils ne sont plus isolés, mais interconnectés. Certaines topologies d'interconnexion peuvent les exposer à des risques face à la perte de quelques éléments [Takabatake, 2011] et contraindre le fonctionnement de toute une organisation.

Afin de répondre à ces critères, nous devons nous intéresser aux technologies mises en place dans les objets connectés. Une fois que nous aurons détaillé les fonctionnements des objets et comment ils communiquent, alors nous pourrons tirer des conclusions sur les moyens à mettre en œuvre pour rendre nos IoT-a plus résilients.

2.4 | Technologies des objets

Derrière toutes les applications énoncées ci-dessus se cache une très grande diversité technologique en ce qui concerne les objets connectés. Cette section est consacrée à l'étude de ce qui compose le matériel des objets connectés, de quoi sont-ils faits ? Quelles sont leurs architectures ? Comment communiquent-ils ?

À la fin des années 90, plusieurs dizaines de millions d'ordinateurs personnels ont été vendus par différents fabricants comme IBM ou Apple. C'est le point de départ d'une informatisation à grande échelle de notre société. L'informatique entre dans les foyers, mais aussi dans le quotidien avec son utilisation dans les entreprises et les industries. C'est à cette époque que Mark Weiser propose le terme novateur d' "*informatique omniprésente*"[1] [Weiser, 1999] en rupture avec la vision du développement des ordinateurs à ce moment. Pour Weiser, l'ordinateur tel qu'il existe en 1999, n'est qu'une transition vers un ensemble de technologies plus vaste, ayant une portée plus globale sur le monde qui l'entoure. Son approche amorce alors un tournant dans l'informatique pour donner naissance à ce que Weiser qualifie de "*systèmes invisibles*" ou à l'"*informatique généralisée*". Il entend par là, que l'informatique doit avoir un but à long terme et proposer de l'information partout et tout le temps sans que l'on puisse déterminer précisément sa provenance.

Au fur et à mesure des années, la miniaturisation des systèmes permet d'approcher les théories de Weiser et un nouveau domaine apparaît donnant naissance aux Systèmes Embarqués (SE)[Marwedel, 2018]. Les systèmes embarqués se distinguent des systèmes conventionnels proposés jusqu'alors par une intégration totale de leurs composants de façon à ne plus dépendre d'autres éléments matériels. Ils sont intégrés, autonomes, et répondent

1. De l'anglais "*ubiquitous computing*"

à une fonction précise. Les dispositifs suivants intègrent, par exemple, des systèmes embarqués : les voitures, les trains, les avions, les appareils de télécommunication ou encore les équipements industriels ou électroménager.

Les systèmes embarqués disposent de nombreuses définitions [White, 2011] [Marwedel, 2018] plus ou moins précises. Nous retiendrons celle de Marwedel, reflétant au mieux l'aspect singulier d'un système embarqué comme base de définition des objets connectés.

"*Embedded systems are information processing systems embedded into enclosing products.*"

Il est intéressant de noter que cette définition n'intègre pas la notion d'interaction avec l'environnement extérieur. Lee propose dans ses travaux sur les systèmes cyberphysiques [2] [Lee, 2008] l'introduction d'un lien fort avec l'environnement physique et les systèmes embarqués par la définition suivante :

"*Cyber-Physical Systems (CPS) are integrations of computation and physical processes.*"

Cette définition propose un lien entre le traitement de l'information et l'environnement physique qui entoure le système. Le système a la capacité de capter ou agir physiquement sur le monde extérieur. Les systèmes cyberphysiques sont à l'opposé des ordinateurs personnels ou des serveurs de calcul qui ne prennent pas en compte l'environnement physique dans lequel ils évoluent pour pouvoir fonctionner.

La mise en relation des deux précédents concepts permet à Marwedel de proposer le modèle du système cyberphysique présent en figure 2.1 dans lequel un CPS est la somme d'un système embarqué auquel sont ajoutés les composants électroniques capables d'interagir avec l'environnement physique.

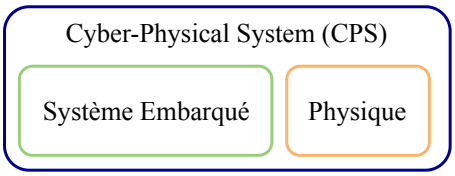


FIGURE 2.1 – Composition d'un système cyberphysique

Les définitions de l'objet connecté sont diverses et peuvent avoir des facettes différentes selon les acteurs. Il est intéressant de reprendre quelques-unes de ces définitions pour les mettre en perspective avec les concepts de systèmes embarqués et systèmes cyberphysiques. Ainsi, il est possible de retrouver dans la littérature les propositions suivantes [Serpanos and Wolf, 2018] :

- "*appareils physiques compatibles avec Internet, bien que de nombreux appareils n'utilisent pas le protocole Internet,*
- *réseaux de capteurs temps réel,*
- *réseaux dynamiques et évolutifs de dispositifs informatiques embarqués.*"

Il est possible de retrouver au travers de ces propositions les éléments composant les systèmes cyberphysiques : *dispositifs informatiques embarquées* pour les systèmes embarqués et *capteurs temps réel* pour la partie physique du modèle de Marwedel. Ces propositions évoquent, de plus, l'idée de réseau et c'est en effet l'élément principal qui vient s'ajouter aux objets connectés en comparaison des systèmes cyberphysiques. L'échange d'informations entre les objets pour former un réseau est une fonction supplémentaire de chacun des systèmes cyberphysiques.

Nous pouvons alors proposer une définition des objets connectés issue de la synthèse des trois concepts initiaux, à savoir les systèmes embarqués, les éléments d'interactions physiques avec le monde réel et la possibilité d'échanges d'informations entre plusieurs objets :

Définition 4 : LES OBJETS CONNECTÉS

Les objets connectés sont des entités matérielles capables de traiter de l'information, de capter ou d'agir sur leur environnement physique et de communiquer au travers d'un réseau.

L'infrastructure des objets connectés est alors la somme de systèmes embarqués, de capteurs ou d'actionneurs et d'éléments de communication. La figure 2.2 détaille cette architecture par l'échange d'informations entre les différents objets.

2. De l'anglais "Cyber-Physical Systems (CPS)"

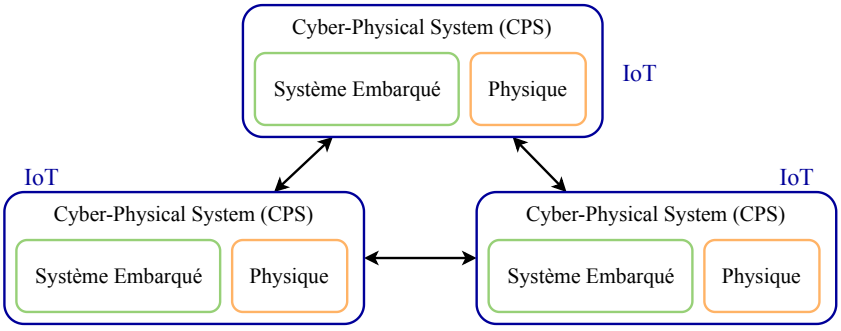


FIGURE 2.2 – Architecture d'objets connectés échangeant de l'information

2.4.1 | *Architectures*

Maintenant que nous avons précisé le cadre de l'objet connecté, intéressons-nous aux différentes facettes des architectures rencontrées dans ce domaine.

L'aspect clé de l'objet connecté est l'échantillonnage d'informations. Il peut être mis en place pour la prise d'informations provenant de capteurs ou l'attente d'événements depuis un réseau. Son inconvénient est qu'il occasionne une consommation importante d'énergie et de bande passante. La priorité est alors de disposer de matériels capables de réagir à des événements puis de se mettre en sommeil afin d'économiser un maximum d'énergie et de bande passante. Une autre approche avance l'idée de distribuer les fonctions dans de multiples objets [Serpanos and Wolf, 2018]. Ainsi, ces travaux se concentrant sur une pluralité de systèmes disposant chacun d'un nombre limité de capteurs et d'actionneurs. Cela évite la centralisation de nombreux éléments d'interaction avec le mode réel sur un seul système. La répartition de ces fonctions forme alors un ensemble d'objets échangeant leurs informations selon trois méthodes [Chiang and Zhang, 2016]. La première, appelée *Cloud computing*, centralise les informations vers un serveur central de traitement. La seconde, le *Fog computing*, se concentre sur la répartition des données vers différents serveurs ayant certaines spécialités de traitement Enfin, le *Decentralised computing* fait émerger un traitement décentralisé entre serveurs et objets connectés.

Comme évoqué en introduction de ce chapitre, les stratégies de Cloud computing et Fog computing ne cadrent pas au sujet de cette thèse, car elles rendent les objets connectés dépendant d'une ou plusieurs unités de traitement centralisant l'information. C'est sur la dernière méthode décentralisée que nous concentrons notre étude.

Les principaux fabricants de puces électroniques et architectes de processeurs ont toujours investi du temps et de l'argent dans l'amélioration des micro-architectures. Bon nombre de ces efforts, comme le pipeline profond, l'augmentation de la taille de la mémoire cache et l'amélioration d'instructions permettant le parallélisme, présentent des rendements décroissants en raison de l'augmentation de la superficie des puces et de la consommation électrique. La séparation en différents cœurs de traitement a donc été l'étape naturelle permettant de rester sur une courbe de performance toujours croissante. Cette importante décision de quitter l'approche à noyau unique a été prise à partir de 2004 par les différents leaders mondiaux des systèmes embarqués. Plusieurs cœurs communiquent entre eux par un cache matériel partagé apportant de meilleurs résultats tout en diminuant les fréquences de cadencement de chaque cœur.

Les systèmes intégrés sur puce (System on Chip, SoC) s'intègrent dans cette philosophie de découpage fonctionnel [Ben Abdallah, 2017a]. Ils sont de plus en plus communs dans la mise en place de solutions connectées. En effet, les SoC sont des systèmes embarqués composés de plusieurs modules (processeurs, mémoires, périphériques d'entrées/sorties, etc.) sur une seule puce électronique. Avec eux il est possible de traiter de l'information et d'exécuter des tâches précises à une vitesse importante pour une consommation faible et maîtrisée dans un espace restreint. L'évolution des SoC permet désormais de disposer de systèmes intégrant plusieurs processeurs sur une même puce (Multi-Core SoC, MCSoC ou Muti-Processor SoC, MPSoC), décuplant les performances et réduisant la consommation [Ben Abdallah, 2017b]. La figure 2.3 illustre un exemple d'architecture interne de ce type de système. Les MPSoC permettent d'effectuer le traitement de tâches nécessitant l'utilisation de processeurs spécifiques ou de scinder les tâches pour les répartir sur différents cœurs de traitement. À performance de traitement équivalente, la répartition des calculs sur différents cœurs permet de réduire la fréquence globale ainsi que les tensions d'alimentation et donc de réduire la consommation du système. De plus, la mise en sommeil de certains cœurs ou processeurs spécialisés, lorsqu'ils ne sont pas utilisés, permet de réduire d'autant

plus la consommation. C'est dans cette démarche de très faible consommation que les SoC ont été développés. Ils sont d'ailleurs qualifiés de *Low Power* ou *Ultra Low Power* étant donné les consommations de l'ordre du milliampère ou de la centaine de nanoampères dans certains cas.

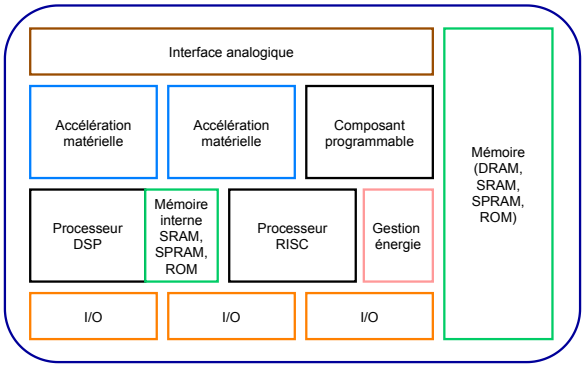


FIGURE 2.3 – Vue simplifiée de l'architecture d'un MPSoC, [Ben Abdallah, 2017c]

Les MPSoC intègrent aujourd'hui différents types d'unités de traitement (CPU, GPU, DSP, FPGA, ASIC, etc.) mais aussi des unités d'entrées/sorties et de communication filaire ou par ondes radio. Plus les MPSoc intègrent d'unités différentes, plus le nombre de tâches exécutables en parallèle est important. Aussi, le paradigme de programmation séquentielle a évolué pour disposer de solutions de programmation permettant l'exécution de tâches parallèles. Cette évolution a donné naissance à de nombreuses problématiques comme : la synchronisation de tâches, le partage de données, la priorité et la concurrence des tâches ou le phénomène de *load balancing* pour l'équilibre d'exécution des tâches entre les cœurs. Toutes ces problématiques sont d'autant plus complexes dans des systèmes avec des processeurs hétérogènes. Par exemple, les travaux de [Inguere, 2018] s'intéressent à la délégation de tâches dans un environnement comportant plusieurs MPSoC.

Le principal enjeu des MPSoC est la communication entre les unités. L'ensemble des éléments doit-être relié sur un bus partagé bidirectionnel pour qu'ils puissent envoyer et recevoir de l'information. Ce bus doit permettre la mise en place de nouvelles unités pour le développement des systèmes. Cependant, si un nombre trop important d'unités est présent, la bande passante, la latence et des biais d'horloge peuvent devenir des goulots d'étranglement. C'est pour cela que de nombreuses recherches ont été effectuées pour le développement de nouvelles approches en matière d'échanges de données. L'un des développements les plus importants a été celui du système de communication par paquets (similaire aux paquets utilisés en réseau IP) permettant des transferts de données asynchrones à haute vitesse en distribuant l'information au travers d'échangeurs appelés : "switches". Grâce à cette solution, les unités sont reliées les unes aux autres par des switches formant un réseau. La mise en place d'un réseau d'unités a donné naissance à une nouvelle appellation : Network on Chip, NoC [Takabatake, 2011].

Les NoC sont des systèmes embarqués disposant d'un grand nombre d'unités de calcul réparties selon une topologie en deux ou trois dimensions. C'est un domaine encore récent, il est implémenté dans certaines puces électroniques. Les NoC sont basés sur des approches multicœurs avec une technologie à une échelle nanométrique composée de centaines d'éléments de traitement et de stockage. La mise en réseau d'un grand nombre d'unités de traitement au travers d'un grand nombre de switches permet de relayer l'ensemble des informations entre les éléments et de répartir plus finement les tâches à exécuter. Les NoC prennent tout leur sens dans le traitement d'informations vectorielles comme celles d'un LIDAR par exemple. Chaque partie d'une matrice peut-être traitée par une unité spécifique. L'architecture NoC est principalement caractérisée par sa topologie de routage. Elle définit la façon dont les switches et les liens sont interconnectés. La topologie est un choix de conception important, car elle définit la distance de communication et sa régularité. Certaines des topologies les plus utilisées sont décrites dans la figure 2.4. Habituellement, les topologies régulières sont préférées aux topologies irrégulières, en raison de leur évolutivité et de leur structure facile à reproduire à large échelle. Il apparaît que la topologie en maille régulière est le choix principal retenu dans le déploiement des NoC [Salminen et al., 2007].

Nous nous sommes intéressés dans cette section aux composants électroniques, à leurs compositions et leurs architectures. En l'occurrence, les SoC, MPSoC et NoC intègrent plusieurs catégories de microcontrôleurs et microprocesseurs. Ils sont mis en place dans les objets connectés selon leurs caractéristiques et leurs performances. Cette présentation a pour but de mieux comprendre l'hétérogénéité des systèmes actuels, afin de mettre en place notre concept d'IoT-a sous la forme d'un modèle suffisamment polyvalent pour convenir aux nombreuses architectures rencontrées dans les objets connectés.

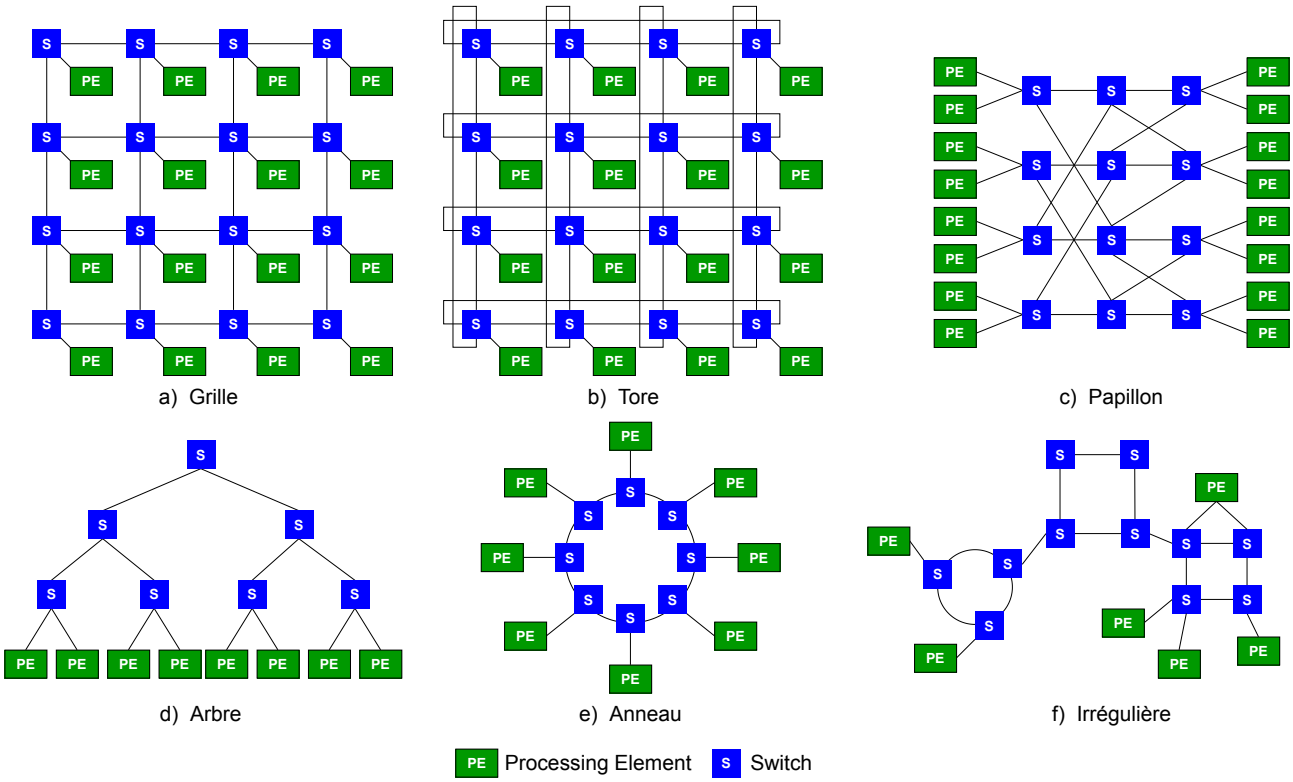


FIGURE 2.4 – Topologies dans les Network on Chip

2.4.2 | *Single-Board Computer*

Maintenant que nous avons détaillé l'architecture interne des composants, nous pouvons nous intéresser à leur intégration dans certains systèmes. Le terme de système monocarte ou Single-Board Computer (SBC) [Isikdag, 2015] [Johnston et al., 2018], est utilisé pour définir les systèmes informatiques qui se composent d'une seule carte de circuit imprimé. Ce type de système intègre tous les composants nécessaires à la réalisation d'une application complète. Ils intègrent notamment des alimentations, un SOC ou un NOC, des mémoires, des bus de communication, des interfaces d'entrées-sorties appelées GPIO et parfois des capteurs. La taille de ces systèmes varie de quelques centimètres à une dizaine de centimètres maximum. Certains SBC ont la capacité d'exécuter un système d'exploitation de type Linux ou Windows, et d'autres intègrent des microcontrôleurs programmables. Des développements récents ont montré que les SBC peuvent être utilisés comme serveurs Web, ou même comme nœuds dans des clusters de cloud.

De nombreux constructeurs de SBC ont vu le jour depuis plusieurs années et un large choix s'offre aux développeurs aujourd'hui. Certains projets issus de communautés (Beagle Board, Raspberry Pi, Arduino) connaissent une popularité importante auprès du grand public au point de modifier les stratégies commerciales de certains fondeurs les plus renommés. Intel, NXP, Asus, STMicroelectronics, Altera et Microship proposent désormais aussi des produits équivalents ou compatibles avec ces projets communautaires.

Les pratiques en matière de SBC évoluant très rapidement, la dernière stratégie technique proposée par les industriels et les communautés est la mise en place de System On Module (SOM). Un SOM est une carte fille connectable sur une carte mère de type SBC. Le SBC n'est alors plus qu'une carte intégrant tous les périphériques sans fonction logique. Un SOM vient s'ajouter, via des connecteurs, pour compléter la partie logique du SBC. Cette solution permet de proposer un environnement de périphériques communs (GPIO, interfaces de communication, capteurs, etc.) et plusieurs possibilités de traitement avec de plus ou moins grandes capacités techniques.

Nous avions évoqué en introduction notre intérêt face à la mise en place de notre approche dans des systèmes préexistants. Les SBC et les SOM sont des environnements particulièrement adaptés pour mener des expérimentations. Ils sont suffisamment ouverts et documentés pour que nous puissions modéliser, expérimenter et mesurer notre concept d'IoT-a à plusieurs niveaux logiciels et matériels.

2.5 | Communications

La communication est le point central du paradigme de l'IoT, ce pour quoi nous les qualifions "d'objets connectés". Les connexions sont diverses et font intervenir différents niveaux de liens. Ils se traduisent par de nombreuses technologies de communication standardisées pour correspondre aux types d'applications. Leur dimensionnement afin de répondre à une application s'effectue en fonction des paramètres suivants :

— du nombre et de la taille des données à transférer,
— de la sécurité nécessaire à appliquer à la transmission,
— de l'énergie disponible dans le système et de la durée d'utilisation,
— de l'existence du déploiement du réseau partout où les objets doivent être mis en œuvre,
— du coût de la solution par rapport à l'échelle du déploiement.

Une étude des technologies de communication abordées dans le domaine des objets connectés nous permet d'analyser leurs forces et leurs faiblesses face à certaines applications. Nous nous intéressons ici, à comprendre ce qui est à l'origine de l'hétérogénéité des objets connectés. La compréhension du fonctionnement de leurs communications peut nous aider à appréhender le rôle des interfaces dans les systèmes embarqués et nous fournir des techniques pratiques afin de bâtir notre modèle de l'IoT-a. Comment adapter notre modèle aux technologies existantes ? Et comment améliorer l'interopérabilité d'éléments hétérogènes ?

2.5.1 | *Interopérabilité*

Nous avons vu dans la section 2.1 que le grand défi de l'objet connecté à l'heure actuelle est l'interopérabilité des systèmes entre constructeurs. Les objets doivent pouvoir échanger leurs informations et permettre l'émergence de nouvelles données ou comportements. Le terme d'interopérabilité [Pimentel et al., 2008], est alors utilisé pour qualifier les mécanismes mis en œuvre pour la communication entre des systèmes hétérogènes. Nous définissons ce terme de la façon suivante :

Définition 5 : L'INTEROPÉRABILITÉ

L'interopérabilité est l'habilité des systèmes à fonctionner ensemble et échanger des informations ou réaliser des actions sans restriction d'accès ou de mise en œuvre. Un système informatique est interopérable s'il est en capacité d'évoluer avec d'autres systèmes informatiques de nature ou de structure différentes de lui.

La prise de conscience de l'importance de l'interopérabilité naît dans les années 1970 lorsque les premiers réseaux informatiques apparaissent. Les systèmes sont, à cette époque, incompatibles entre eux et ne peuvent communiquer. De la même façon que pour les objets connectés actuels, de nombreuses sociétés développent leurs propres solutions de communication pour l'échange d'informations entre différents terminaux. C'est à la fin des années 1970 que les premiers travaux communs entre plusieurs entreprises permettent de poser les bases de ce qui deviendra le modèle de référence pour les communications et encore utilisé pour internet de nos jours : le modèle Open Systems Interconnection (OSI) [Zimmermann, 1980] [Pujolle, 2006b] [Du and Swamy, 2010].

2.5.2 | Modèle OSI

Le modèle de référence OSI comporte sept couches, plus un médium physique. Le médium physique, que l'on appelle parfois couche 0, correspond au support physique de communication chargé d'acheminer les éléments binaires d'un point à un autre jusqu'au récepteur final. Ce médium physique peut prendre diverses formes, allant du câble métallique aux signaux hertziens, en passant par la fibre optique et l'infrarouge. Au-delà du transport, le découpage, la gestion de qualité de service et l'interopérabilité des informations doivent être assurés du début à la fin de la communication. Le modèle provenant de la norme International Standardization Organization (ISO), propose un ensemble de couches permettant des échanges d'informations variées que nous retrouvons illustrées sur la figure 2.5. OSI peut s'appliquer à de très nombreuses méthodes de communication, la téléphonie, le fax, les réseaux industriels, les réseaux de terrains, et bien entendu, les objets connectés.

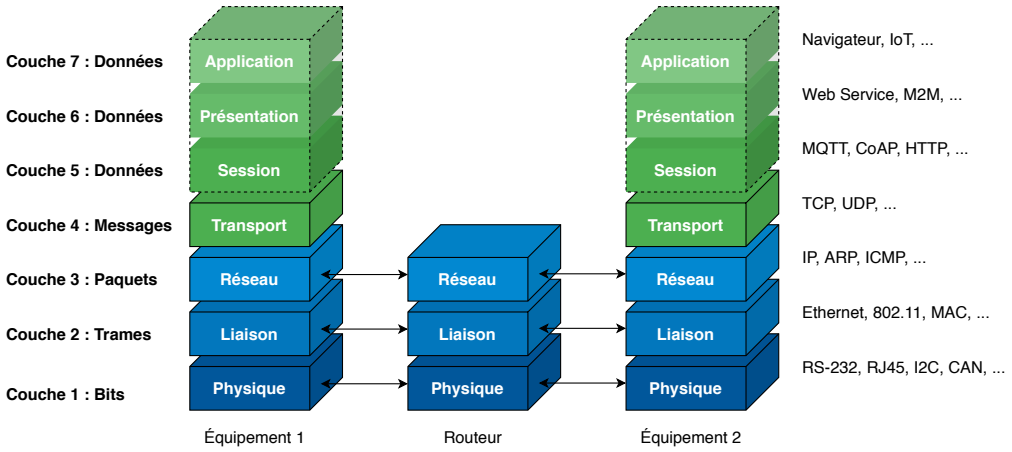


FIGURE 2.5 – Modèle OSI

Selon la norme, une couche est responsable de fonctions définies et sert à normaliser les différents protocoles existants. Les couches basses sont responsables de l'acheminement des informations entre les extrémités concernées alors que les couches hautes sont responsables du traitement de l'information, de leurs mises en forme et de leurs intégrités.

La communication entre les différentes couches s'effectue sous la forme de services. Chaque couche N fournit des services à la couche du dessus (N+1) et utilise les services de la couche du dessous (N-1). Une couche peut communiquer indépendamment vers une couche directement supérieure ou inférieure, mais pas avec les autres. Chaque couche ajoute à la donnée dont elle est responsable (Service Data Unit, SDU), des informations de contrôle (Protocol Control Information, PCI) pour générer une unité de données du protocole (Protocol Data Unit, PDU). Pour transmettre une unité de données vers un niveau supérieur ou inférieur, une couche doit rajouter des informations de contrôles d'interface (Interface Control Information, ICI). Cet ajout donne naissance à une unité de données d'interface (Interface Data Unit, IDU) transférable vers un niveau adjacent. Chaque couche dispose de tous ces éléments et permet un échange standard entre toutes les couches du modèle.

Les couches détaillées dans le modèle OSI sont définies de la façon suivante :

Couche 1 : Couche physique

La couche physique définit les règles et procédures pour l'acheminement des éléments binaires d'un composant à un autre au travers du médium physique (couche 0). À ce niveau un PDU est un bit.

On trouve à ce niveau les équipements matériels manipulant les éléments physiques comme les modems, modules de communication sans fil, ou ports Ethernet.

Couche 2 : Couche Liaison

La couche liaison ou niveau trame, est la couche où est générée une trame d'un ou plusieurs octets. Une trame est un ensemble successif de bits. Le rôle de cette couche est aussi d'être capable de détecter le début et la fin d'une trame. Différentes normes existent pour déterminer les marqueurs de début et fin de trame. Le PDU à ce niveau est donc la trame.

Couche 3 : Couche Réseau
La couche réseau est responsable de la gestion de paquets générant un échange d'informations de bout en bout donnant naissance au principe de réseau. Ce niveau doit permettre d'acheminer correctement les paquets d'informations d'un émetteur à un récepteur final. Au même titre qu'un paquet délivré par un facteur du service des postes, un paquet du modèle OSI comporte au moins une adresse source et une adresse de destinataire. La normalisation de ces adresses a donné naissance à un très grand nombre de normes et notamment la norme IP. Le PDU à ce niveau est un paquet.

Couche 4 : Couche Transport
La couche transport est responsable de l'acheminement des paquets d'un bout du réseau à un autre en optimisant l'infrastructure. Autrement dit, elle doit trouver les meilleurs chemins du réseau pour une communication. Cette couche est l'ultime niveau qui s'occupe de l'acheminement de l'information. C'est grâce à elle que l'utilisateur obtient la qualité de service susceptible de le satisfaire. C'est à ce niveau qu'on retrouve les protocoles TCP ou UDP du protocole IP que nous détaillerons dans une prochaine section. Le PDU à ce niveau est donc le message ou aussi le datagramme.

Couche 5 : Couche Session
Cette couche a pour rôle de fournir aux éléments du réseau les moyens nécessaires à l'organisation et à la synchronisation de leurs communications. Elle permet l'ouverture, la fermeture et le contrôle de session de connexion. Certains protocoles proposent des solutions pour la reconnexion ou la resynchronisation des communications. Les PDU pour les couches 5, 6 et 7 sont à ce niveau des données définies par l'utilisateur ou le protocole. Ils ne sont plus définit spécifiquement par le modèle OSI.

Couche 6 : Couche Présentation
La couche Présentation est chargé du codage des donnée applicatives. Aussi, il est nécessaire de définir dans quelle norme transite l'information entre les éléments du réseau pour qu'ils puissent décoder l'information. par exemple, la norme ASN.1 (Abstract Syntax Notation 1) normalisée par l'ISO, sert de langage de base de la couche présentation et normalise la syntaxe des éléments de langage.

Couche 7 : Couche Application
La couche application est la dernière du modèle et donne un point d'accès au service réseau pour les processus applicatifs. C'est par cette dernière qu'une application pourra récupérer ou proposer des données sur un réseau. Les systèmes d'exploitation ou systèmes purement matériels proposent des points d'accès au réseau au moyen d'API ou d'espaces mémoire dédiés.

Cet ensemble de couches normalise les interactions et les protocoles pour les réseaux informatiques. C'est la base de l'interopérabilité permettant la définition d'un grand nombre de normes et de protocoles ayant chacun leurs caractéristiques.

En plus de détailler les différentes couches proposées par le modèle OSI, la figure 2.5 illustre la différence entre deux catégories d'entre elles. La première, à la base du modèle et symbolisée par la couleur bleue, rassemble les trois premières couches responsables des médias [Wolf, 2019]. C'est dans cette catégorie que sont définis les bus de communication et technologies sans fil. Ils permettent des échanges de données internes, c'est-à-dire entre les composants électroniques d'un objet, et externes, entre les objets et les stations de base. Nous allons les détailler dans les deux prochaines sections afin d'appréhender les communications intra et inter-objets.

La seconde catégorie de couches, en vert sur la figure 2.5, met en jeu les protocoles de communication. Là encore, nous faisons un inventaire des normes et standards développés dans le domaine des objets connectés.

2.5.3 | *Technologies de communication intra et inter-systèmes*

Les objets connectés prennent leur sens grâce aux nombreux capteurs et actionneurs dont ils disposent.

Les bus de communication permettent de connecter les microcontrôleurs et microprocesseurs aux périphériques pouvant capter ou agir sur le monde physique. Ils sont constitués de peu de conducteurs afin de minimiser les entrées/sorties nécessaires à leur définition. C'est dans ce cadre que différents standards ont émergé, basés sur une communication série avec un minimum de signaux nécessaires [Wolf, 2019].

L'annexe A détaille les principaux bus de communication rencontrés pour la connexion de périphériques dans le domaine des objets connectés. Le tableau A.1 dresse un aperçu des principales caractéristiques de chacun d'entre eux. Nous pouvons noter la portée limitée de certains comme l'I2C ou le SPI étant développés pour des communications intra-système, pour des échanges de données entre composants électroniques. Ces bus ne sont pas immunisés au bruit et ne disposent pas de sommes de contrôle ou de code correcteur d'erreurs permettant la détection et la correction d'anomalies durant une transmission. D'autres, comme les bus Ethernet ou CAN, possèdent les technologies nécessaires pour résister aux parasites électromagnétiques et permettent des communications inter-systèmes. Le CAN est notamment largement utilisé dans le domaine des transports pour l'échange de données entre les équipements d'un véhicule et l'Ethernet dans les télécommunications entre les équipements réseau. La communication inter-systèmes se voit d'ailleurs largement transformée, depuis quelques décennies, par la démocratisation de technologies de communication sans fil.

Le domaine des communications sans fil pour les objets connectés connaît de très régulières innovations. Les consortiums font évoluer les normes et standards au quotidien. En parallèle, les constructeurs de circuits intégrés font preuve d'une très grande réactivité et sont d'ailleurs souvent à la genèse de nombreuses innovations.

Nous pouvons retrouver en annexe B une description de quelques-unes des technologies les plus communément employées actuellement. Le tableau B.1 propose une comparaison des portées, débits et fréquences de chacune. Nous pouvons relever, encore une fois, l'hétérogénéité des paramètres et les applications auxquelles ces technologies peuvent répondre.

Capteurs, actionneurs et objets connectés implémentant plusieurs de ces technologies ne peuvent communiquer sans avoir recours à des interfaces. Des fabricants de semi-conducteurs en proposent des versions matérielles [NXP, 2019] et plusieurs travaux dans la littérature cherchent à interfacer certains bus par des méthodes logicielles [Trivedi et al., 2018]. Certains travaux soulèvent aussi la difficulté de l'interopérabilité entre de multiples liens de communication [Pimentel et al., 2008].

Notre objectif est ici de permettre à nos IoT-a de pouvoir échanger des données intra et inter-systèmes quelque soit leurs technologies de communication. Les agents répartis dans les systèmes, échangent entre eux et avec des périphériques. Nous en déduisons, en raison de la diversité des technologies existantes, que des interfaces sont nécessaires. Elles doivent permettre la communication entre des technologies différentes, mais aussi des échanges entre des matériels distants hétérogènes, sur lesquels les agents évoluent. Mais au-delà des technologies que nous avons présentées, nous devons aussi nous intéresser aux informations qui transitent sur certains de ces supports.

2.5.4 | *Protocoles*

Après les technologies mises en œuvre dans les deux premières couches du modèle OSI, que nous avons présentées dans la section précédente, nous abordons ici les protocoles de communication. La solution la plus utilisée pour l'échange de données dans le domaine des objets connectés est le standard Internet Protocol (IP) [Pujolle, 2006a].

TCP/IP

Le standard IP s'intègre à la couche 3 - réseau. Il définit la mise en paquets de l'information, la définition des adresses des paquets, leurs tailles et leurs découpages. Il n'a pour seul rôle que le transport d'un ensemble de données mis dans un paquet sans avoir conscience de l'information contenue dans le paquet. Il est dit aveugle des données qui transitent. Chaque paquet IP est indépendant des autres et est acheminé individuellement dans le réseau par le biais de routeurs. Le standard IP ne propose pas de qualité de service, mais l'architecture TCP/IP offre des solutions au niveau 4.

Nous quittons maintenant les couches média du modèle OSI pour nous concentrer sur les protocoles. Les premiers d'entre eux, au niveau 4 - transport, à la base de tous les autres que nous allons présenter ensuite, sont les protocoles TCP et UDP.

Le TCP propose des communications en mode connecté alors que UDP est dit, non connecté. Le protocole TCP rassemble de nombreuses options pour le transport de données avec notamment la résolution de perte de paquets assurant alors une qualité de service non négligeable. L'UDP dispose d'un débit de transfert plus important étant donné qu'il n'assure aucune qualité de service, aucun renvoi de paquet ou d'acquittement susceptible de ralentir les transferts.

La force du protocole TCP/IP repose sur le fait qu'il n'est pas dépendant des couches matérielles inférieures et peut être mis en place, quelle que soit la topologie de connexion du réseau. C'est le protocole TCP qui se charge du découpage de l'information en paquets puis les mets à disposition de la couche IP. C'est encore TCP qui se charge d'envoyer plus ou moins de paquets en fonction de la charge réseau et permet la ré-émission des paquets n'ayant pas reçu d'acquittement de la part de l'utilisateur final.

Il est important de comprendre les mécanismes de qualité de service présents au sein du protocole TCP pour mettre en perspective les autres protocoles étudiés dans cette thèse. Un transfert ne peut avoir lieu qu'après l'établissement d'une connexion entre les machines concernées. Les données sont échangées sous la forme d'un flot de bits divisé en octets. Les octets doivent être reçu dans l'ordre dans lequel ils sont envoyés. Pour chaque paquet envoyé par un émetteur, le récepteur lui renvoie au moins un acquittement. Plusieurs acquittements peuvent être envoyés pour un même ensemble d'octets, dans ce cas, seul le dernier acquittement est considéré et les autres peuvent être perdus. Si jamais un acquittement met trop de temps à parvenir à l'émetteur du paquet, alors le paquet est ré-émis.

Protocoles M2M

Les protocoles TCP et UDP sont donc deux méthodes d'échange de messages entre des éléments capables d'implémenter un réseau IP. À partir du niveau 5 - données, il existe une multitude de protocoles disposant chacun de caractéristiques plus ou moins adaptées au domaine des objets connectés. Ces protocoles sont définis comme étant compatibles de machine à machine (Machine-to-Machine, M2M) [Thirunavukkarasu et al., 2005], car ils n'interviennent que lors de communications entre des machines.

Le nombre de protocoles M2M dédié aux échanges entre IoT est assez important et de nombreux protocoles fermés apparaissent au fur et à mesure des années et des plate-formes privées qui progressent sur le marché. Néanmoins, certaines fondations et organisations proposent des protocoles ouverts, standardisés et adaptés aux applications des objets connectés [Al-Fuqaha et al., 2015] [Vasileios et al., 2015]. Nous nous intéressons ici uniquement aux protocoles standardisés et ouverts, car l'étude des protocoles fermés repose sur des brevets ou des documents confidentiels, et ils sont bien souvent spécifiques à certaines applications.

Deux types d'interactions entre les périphériques sont possibles dans les protocoles M2M. Le premier est le type requête/réponse (request/response) [Vasileios et al., 2015]. Ce type fait appel à deux éléments, le premier, le client, souhaite bénéficier d'informations ou en proposer. Le second, le serveur, propose des services et répond aux demandes des clients. Avec le type requête/réponse, chaque client envoie une requête à un serveur qui lui répondra et effectuera le service demandé. Si un client souhaite communiquer avec un autre, il devra effectuer une requête de transfert de son message au serveur sur lequel l'autre client est connecté. La figure 2.6 illustre cette procédure.

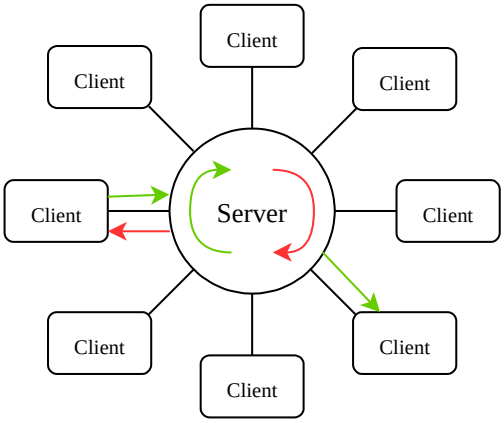


FIGURE 2.6 – Interaction de type request/response entre des clients et un serveur

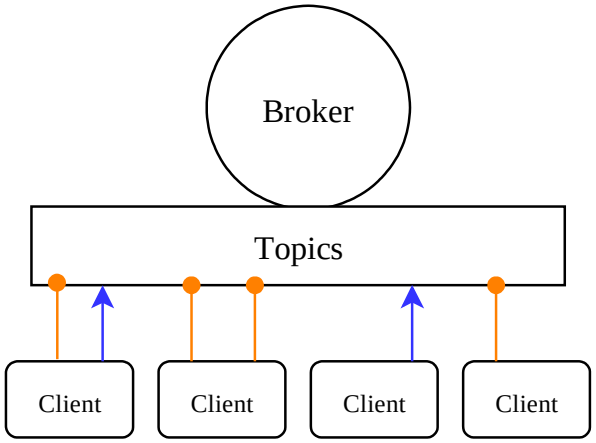


FIGURE 2.7 – Interaction de type publish/subscribe entre des clients et un broker

Le second type est la méthode de publication/abonnement (publish/subscribe) [Thirunavukkarasu et al., 2005]. Il s'appuie lui aussi sur deux éléments, le client qui peut publier un message et/ou s'abonner à des rubriques (topics) et le broker responsable de l'aiguillage des messages échangés entre les clients. Les rubriques se comportent à la manière d'une arborescence de dossiers où il existe un nom racine et des branches. Un client peut alors s'abonner à un certain niveau de branche de l'arborescence afin de recevoir toutes les informations d'un nœud ou de toutes les branches inférieures. Un client peut, en même temps qu'il souscrit à des branches, publier des messages, il est alors à la fois abonné et publicateur (subscriber and publisher). Le rôle du broker est d'enregistrer les clients, d'assurer la sécurité et la fiabilité des transactions et de faire le relais entre les messages échangés par les clients. Aussi, ce second type propose une méthode centrée sur les interactions entre les clients, faisant appel au broker uniquement pour ré-émettre les messages aux clients concernés. La figure 2.7 illustre plusieurs clients échangeant des messages au travers d'une architecture de type publish/subscribe.

Nous nous intéressons aux différents protocoles utilisés dans le domaine des objets connectés afin de déterminer lequel d'entre eux est le plus adapté pour notre approche de l'IoT-a. En l'occurrence, nous cherchons un protocole répondant ces critères :

— léger, capable d'être implémenté dans un maximum de type d'objets connectés et réseaux,
— stable, permettant une harmonie de communication malgré des objets hétérogènes,
— robuste, offrant des solutions pour continuer à communiquer malgré des pertes de connexions, des latences élevées ou une bande passante faible, de façon à répondre aux problèmes de résilience,
— sécurisé, apte à transporter de l'information sensible,
— centré sur la communication entre les objets.

Critères	HTTP	CoAP	XMPP	AMQP	MQTT
Mode de communication	req/resp	req/resp pub/sub	req/resp	req/resp pub/sub	pub/sub
Taille de l'entête	Non défini	4 octets	Variable	8 octets	2 octets
Taille du contenu max.	Non défini	64 Ko	Non défini	Non défini	256 Mo
Protocole de transport	TCP	UDP	TCP	TCP	TCP
Qualité de service	Aucune	2	Aucune	2	3
Sécurité	TLS/SSL	DTLS	TLS/SSL	TLS/SSL	TLS/SSL
Type de message	Text	Binaire	Binaire	Binaire	Binaire

Tableau 2.1 – Comparaison des protocoles de communication utilisés dans le domaine de l'objet connecté

L'annexe C détaille cinq protocoles dont la synthèse est illustrée dans le tableau 2.1.

Chacun des protocoles présentés dispose d'avantages et d'inconvénients qui leur confèrent des particularités en fonction des applications dans lesquelles ils sont les plus adaptés. Les travaux abordés dans cette thèse se concentrent sur la communication entre les objets connectés et non celles centrées sur des traitements globaux dans des fermes de serveurs. Nous avons vu que le mode de communication par publication/abonnement est centré sur les interactions entre les clients et c'est l'objectif de notre approche IoT-a. De plus, la nature des objets connectés et de leurs applications avance l'idée de mobilité et de connectivité pouvant être aléatoire [Luzuriaga et al., 2015]. Différentes situations sont susceptibles de rendre les communications instables. Des interférences dues à d'autres communications, à des rayonnements électromagnétiques ou à un phénomène de réflexion d'ondes peuvent être la raison de perturbations. Certains objets peuvent se retrouver dans des zones blanches, sans couverture de réseau, durant un certain temps. Des pannes d'infrastructure du réseau ou électriques sont susceptibles d'arriver. Enfin, les objets risquent de nécessiter de nouveaux liens en fonction de leurs applications, et ce de façon dynamique pour s'adapter à leur environnement.

Le protocole MQTT répond au mieux à l'ensemble des critères que nous avons annoncés ci-dessus. En effet, il propose un mécanisme favorisant les communications entre les objets. Il permet des échanges fiables via ses différents QoS en mode connecté grâce à l'utilisation du protocole TCP. Il permet la mobilité des objets et sa norme est suffisamment ouverte pour proposer des solutions de réagencement des liens de communication [Schmitt et al., 2019]. De plus, le MQTT respecte les critères de faible consommation, car il a été standardisé dans ce sens. Nous concentrons donc notre étude sur ce protocole et examinons comment il peut à la fois répondre aux exigences des objets connectés, de leur mobilité et de leurs connectivités dynamiques ainsi que celles des communications au sein d'un ou plusieurs systèmes multi-agents pour notre approche IoT-a.

2.5.5 | *MQTT, étude avancée*

Le MQTT est un protocole de messages binaires, simples et légers. Il repose sur l'architecture publication/abonnement. Sa standardisation est ouverte et il a été développé dans l'objectif d'être simple à implémenter. Le MQTT est conçu pour permettre des échanges de données entre des milliers de clients orchestrés par un seul broker léger [MQTT, 2014].

Ses caractéristiques le rendent adapté pour des environnements contraints, la où la bande passante est faible et les latences élevées. Les clients peuvent se reposer sur des systèmes comportant des ressources très limitées, en terme de mémoire ou de calcul [Espinosa-Aranda et al., 2015]. Analysons désormais plus en détail certains concepts développés dans la norme MQTT.

Principes de communication

Dans ce modèle de publication/abonnement, un publicateur peut envoyer des messages à un certain nombre d'abonnés en une seule opération de publication au broker. Le broker est chargé de la "diffusion" ou de l'envoi de messages à tous les clients abonnés au topic du message. Il existe un certain nombre de mises en œuvre du protocole, notamment Mosquitto [Light, 2017], [eMQTT, 2019], [HiveMQ, 2019], [Moquette, 2019], parmi beaucoup d'autres. La figure 2.8 présente le format de message du protocole MQTT. La figure 2.9 illustre un exemple d'une instance de flux de messages dans le protocole MQTT.

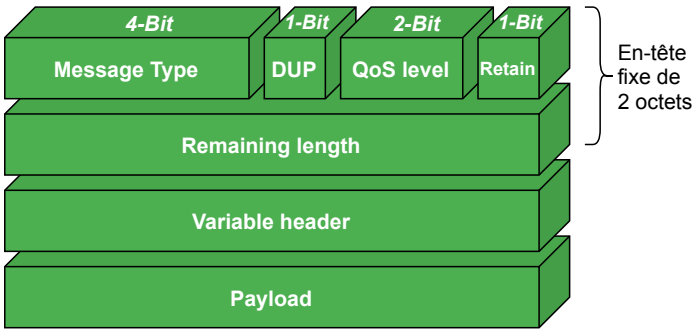


FIGURE 2.8 – Structure d'un message en MQTT

Dans le flux de contrôle des messages sans notion de qualité de service (QoS 0), illustré dans la figure 2.9, le client 1 souhaite se connecter au broker (CONNECT) qui accepte la demande (CONNACK). Le client 1 s'abonne alors au topic `/temperature` (SUBSCRIBE) et le broker répond avec un accusé de réception (SUBACK). Le client 2 publie au broker un message sur le topic `/temperature` (PUBLISH). Le broker transfert le message au client 1 abonné (PUBLISH). Le message, appelé *payload*, contient les données brutes. Le client 1 accuse ensuite réception du message publié (PUBCREC).

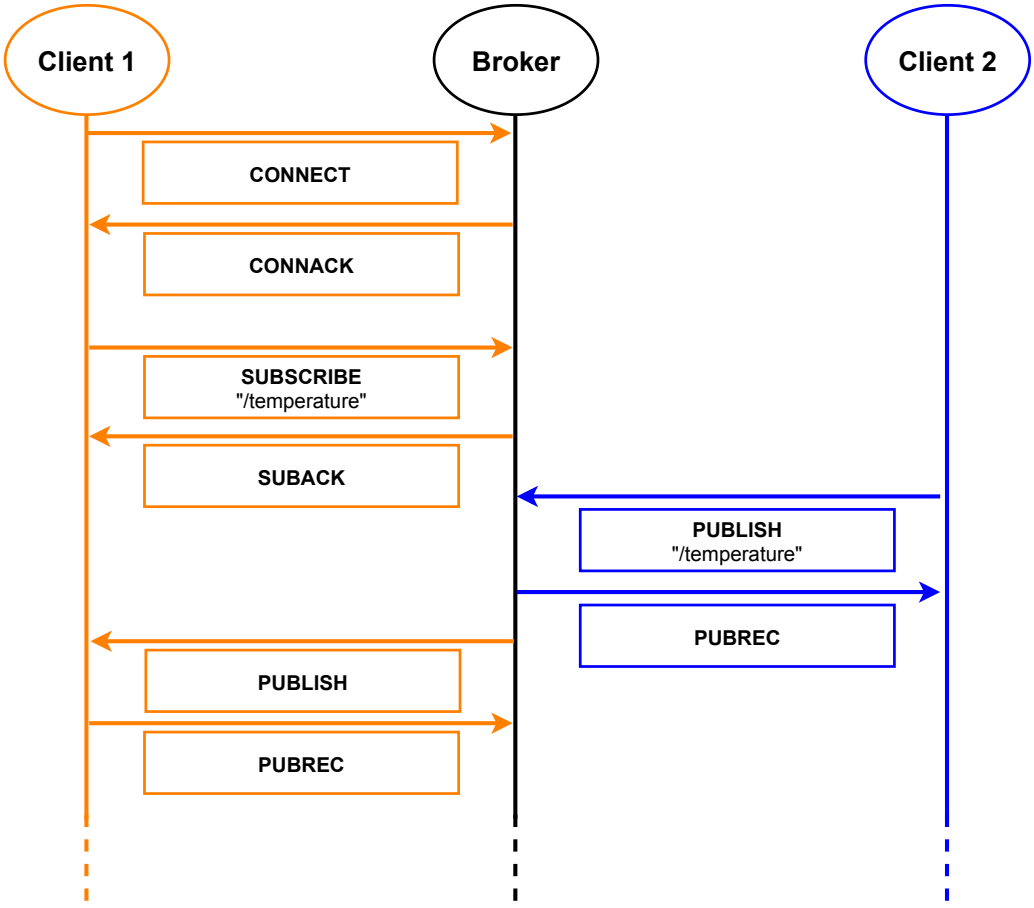


FIGURE 2.9 – Exemple de flux de contrôle de messages MQTT

Topics et abonnement

Avec MQTT, les messages sont publiés sur des *Topics*, qui peuvent être considérés comme des domaines thématiques. Les clients s'inscrivent sur ces topics par une procédure d'abonnement afin d'être informés des messages reçus sur les thèmes souscrits. Les topics sont organisés hiérarchiquement par des arbres de noms. Le caractère "/" marque la jointure avec un sous-topic. Les abonnements peuvent être stricts, dans ce cas seuls les messages correspondant aux topics spécifiés sont transmis. Cependant, ils peuvent aussi avoir un caractère générique par le biais de l'utilisation de signes réservés comme "`#`" et "`+`". La norme définit ces caractères comme étant des : "*wildcards*". Ils permettent aux clients de recevoir des messages sur une variété de sujets connexes.

Le wildcard "`#`" est dit multi-niveaux. Il est utilisé pour percevoir tous les topics aux niveaux inférieurs. Par exemple, l'abonnement au topic `voiture/#` permet de recevoir toutes les informations émises sur tous les topics inférieurs ou égaux à `voiture`, dont `voiture/vitesse` ou encore `voiture/moteur/temperature` par exemple.

Le wildcard "`+`" est lui utilisé pour un niveau unique. Il permet de relayer les informations du niveau directement inférieur au topic initial. Par exemple, l'abonnement au topic `voiture/+` permet de recevoir les informations sur `voiture/position` mais pas `voiture/pneus/pression`.

La figure 2.10 illustre l'utilisation des deux types de *wildcards* pour le calcul de la température moyenne d'une maison. Dans la figure 2.10 b, un client est abonné au topic `maison/#` et reçoit les températures de toutes les pièces pour en calculer la température moyenne dans la maison. Dans la figure 2.10 c, un premier client est abonné au topic `maison/+/alice` et un second au topic `maison/+/bob`. Le premier client à la possibilité de calculer la température moyenne de toutes les pièces dans lesquelles Alice vit. Le second se concentre uniquement sur les pièces concernées par Bob.

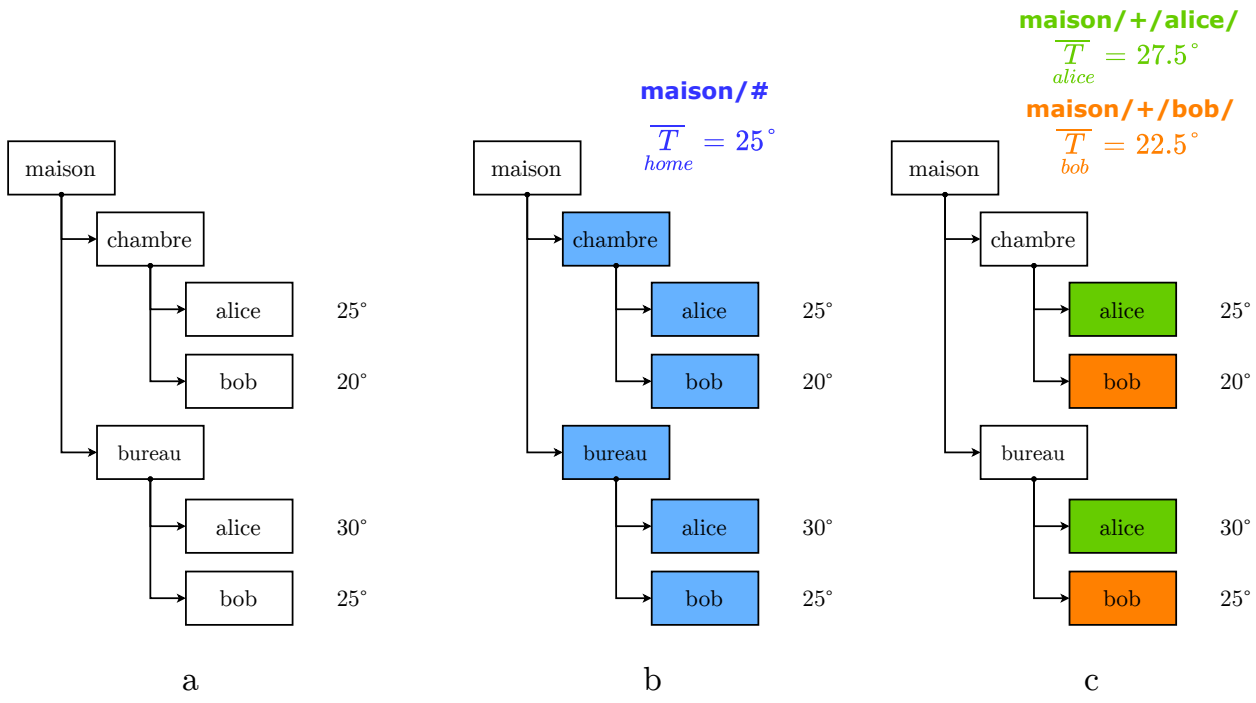


FIGURE 2.10 – Exemple d'utilisation de wildcards pour le protocole MQTT

Qualité de service

La norme MQTT définit trois niveaux de qualité de service (Quality of Service - QoS) pour la délivrance des messages. Chaque niveau désigne un effort plus important de la part du broker pour l'acheminement du message aux clients. Des niveaux de QoS plus élevés assurent une livraison plus fiable des messages, mais peuvent consommer plus de bande passante du réseau ou soumettre le message à des retards dus à des problèmes tels que la latence. Le tableau 2.2 présente les trois niveaux de QoS proposé par le protocole MQTT.

Niveau de QoS	**Niveau de fiabilité**	**Description**
QoS 0	Bas	**Délivré au maximum une fois** : - aucune réponse n'est envoyée par le destinataire, aucune nouvelle tentative n'est effectuée par l'expéditeur - le message arrive au destinataire qu'une seule fois ou jamais
QoS 1	Moyen	**Délivré au moins une fois** : - le message arrive au moins une fois chez le destinataire - le destinataire peut envoyer d'autres paquets PUBLISH en attendant de recevoir les accusés de réception
QoS 2	Élevé	**Délivré exactement une fois** : - garantit qu'un message arrive au destinataire exactement une fois sans duplication - augmente les coûts de communication

Tableau 2.2 – Niveau de qualité de service au sein du protocole MQTT

2.6 | Synthèse

Nous avons présenté dans ce chapitre une introduction au domaine des objets connectés. Nous nous sommes intéressés aux applications dans lesquelles ils occupent une place prépondérante. Nous avons étudié les limites de certaines architectures centralisées et avons abordé la question de résilience encore peu représentée dans ce domaine.

Dans un second temps, nous nous sommes concentrés sur les technologies mises en œuvre dans les objets connectés. Cet aperçu nous permet de mieux appréhender les mécanismes établis par les normes et modèles actuels afin d'adapter nos propositions pour notre approche IoT-a. En effet, notre objectif est de proposer un modèle de l'IoT-a satisfaisant l'hétérogénéité des structures matérielles rencontrées dans les objets connectés.

L'évolution des architectures matérielles, que nous avons présentées, montre comment les contraintes des systèmes embarqués ont été vectrices de nombreuses innovations techniques au travers des quarante dernières années. La mise en place des architectures à noyaux multiples comme les MPSoC, puis leurs évolutions vers des structures organisées comme les NoC, permettent d'envisager des systèmes capables d'interagir et raisonner comme des organisations sociales, en interne, entre leurs différents composants.

La communication entre les éléments prend alors tout son sens, car c'est elle qui permet les échanges de données entre les entités. Nous avons présenté certains bus de données permettant des communications internes entre les composants. Puis, nous avons évoqué quelques-unes des technologies de communication sans fil les plus rencontrées, permettant des interactions inter-objets. Enfin, nous avons introduit les protocoles de communication adaptés aux besoins des objets connectés et avons déterminé que le MQTT correspondait au mieux à l'approche IoT-a que nous souhaitons développer.

Nous avons présenté les objets physiques et nous nous sommes intéressés à leurs méthodes d'interconnexion. Nous devons désormais nous concentrer sur les méthodes nous permettant de faire émerger des comportements collectifs des objets connectés.

Bibliographie

[Al-Fuqaha et al., 2015] Al-Fuqaha, A., Aledhari, K., Mohsen, G., Ammar, R., and Mehdi, M. (2015). Toward better horizontal integration among iot services. *IEEE Communications Magazine*, 53(9) :72–79.

[Ben Abdallah, 2017a] Ben Abdallah, A. (2017a). Introduction to multicore systems on-chip. In *Advanced Multicore Systems-On-Chip : Architecture, On-Chip Network, Design*, pages 1–18. Springer Singapore, Singapore.

[Ben Abdallah, 2017b] Ben Abdallah, A. (2017b). Multicore soc on-chip interconnection networks. In *Advanced Multicore Systems-On-Chip : Architecture, On-Chip Network, Design*, pages 67–106. Springer Singapore, Singapore.

[Ben Abdallah, 2017c] Ben Abdallah, A. (2017c). Multicore soc organization. In *Advanced Multicore Systems-On-Chip : Architecture, On-Chip Network, Design*, pages 39–66. Springer Singapore, Singapore.

[Bendel et al., 2013] Bendel, S., Springer, T., Schuster, D., Schill, A., Ackermann, R., and Ameling, M. (2013). A service infrastructure for the internet of things based on xmpp. In *2013 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pages 385–388.

[Cao and Liu, 2016] Cao, J. and Liu, X. (2016). Introduction. In *Wireless Sensor Networks for Structural Health Monitoring*, chapter 1, pages 1–6. Springer International Publishing, Cham.

[Chang, 2014] Chang, K. (2014). Bluetooth : a viable solution for iot ? [industry perspectives]. *IEEE Wireless Communications*, 21(6) :6–7.

[Chiang and Zhang, 2016] Chiang, M. and Zhang, T. (2016). Fog and iot : An overview of research opportunities. *IEEE Internet of Things Journal*, 3(6) :854–864.

[Chouhan et al., 2009] Chouhan, S., Wan, H., Lai, H. J., Feliachi, A., and Choudhry, M. A. (2009). Intelligent reconfiguration of smart distribution network using multi-agent technology. In *2009 IEEE Power Energy Society General Meeting*, pages 1–6.

[Collotta et al., 2018] Collotta, M., Pau, G., Talty, T., and Tonguz, O. K. (2018). Bluetooth 5 : A concrete step forward toward the iot. *IEEE Communications Magazine*, 56(7) :125–131.

[Coskun et al., 2013] Coskun, V., Ozdenizci, B., and Ok, K. (2013). A survey on near field communication (nfc) technology. *Wireless Personal Communications*, 71.

[Coyle et al., 2010] Coyle, S., Lau, K., Moyna, N., O'Gorman, D., Diamond, D., Di Francesco, F., Costanzo, D., Salvo, P., Trivella, M. G., De Rossi, D. E., Taccini, N., Paradiso, R., Porchet, J., Ridolfi, A., Luprano, J., Chuzel, C., Lanier, T., Revol-Cavalier, F., Schoumacker, S., Mourier, V., Chartier, I., Convert, R., De-Moncuit, H., and Bini, C. (2010). Biotex—biosensing textiles for personalised healthcare management. *IEEE Transactions on Information Technology in Biomedicine*, 14(2) :364–370.

[Cyrulnik et al., 2012] Cyrulnik, B., Jorland, G., et al. (2012). *Résilience : Connaissances de bases*. Odile Jacob.

[D'Angelo et al., 2017] D'Angelo, G., Ferretti, S., and Ghini, V. (2017). Modeling the internet of things : a simulation perspective. In *2017 International Conference on High Performance Computing Simulation (HPCS)*, pages 18–27.

[Delic, 2016] Delic, K. A. (2016). On resilience of iot systems : The internet of things (ubiquity symposium). *Ubiquity*, 2016(February).

[Du and Swamy, 2010] Du, K.-L. and Swamy, M. N. S. (2010). *Wireless Communication Systems : From RF Subsystems to 4G Enabling Technologies*. Cambridge University Press.

[eMQTT, 2019] eMQTT (2019). The massively scalable mqtt broker for iot and mobile applications. `https://emqtt.io/`. Accès le : 10/11/2019.

[Espinosa-Aranda et al., 2015] Espinosa-Aranda, J. L., Vallez, N., Sanchez-Bueno, C., Aguado-Araujo, D., Bueno, G., and Deniz, O. (2015). Pulga, a tiny open-source mqtt broker for flexible and secure iot deployments. In *2015 IEEE Conference on Communications and Network Security (CNS)*, pages 690–694.

[Grigorik, 2013] Grigorik, I. (2013). Making the web faster with http 2.0. *Commun. ACM*, 56(12) :42–49.

[HiveMQ, 2019] HiveMQ (2019). Reliable data movement for connected devices. `https://www.hivemq.com/`. Accès le : 10/11/2019.

[Hollnagel et al., 2006] Hollnagel, E., Wood, D. D., and Leveson, N. (2006). *Resilience Engineering : Concepts and Precepts*. ASHGATE.

[Inguere, 2018] Inguere, T. (2018). *Multi-agents systems integration within embedded systems for tasks delegation*. Theses, Université du Maine.

[Isikdag, 2015] Isikdag, U. (2015). Internet of things : Single-board computers. In *Enhanced Building Information Models : Using IoT Services and Integration Patterns*, chapter 4, pages 43–53. Springer International Publishing, Cham.

[Joachim et al., 2011] Joachim, W., Kadereit, B., and Baldwin, G. (2011). Systematics, phylogeny, and evolution of papaver californicum and stylomecon heterophylla (papaveraceae). *Madroño*, 58(2) :92–100.

[Johnston et al., 2018] Johnston, S. J., Basford, P. J., Perkins, C. S., Herry, H., Po Tso, F., Pezaros, D., Mullins, R. D., Yoneki, E., Cox, S. J., and Singer, J. (2018). Commodity single board computer clusters and their applications. *Future Generation Computer Systems*, 89 :201–212.

[Kennedy and Hunt, 2008] Kennedy, T. and Hunt, R. (2008). A review of wpan security : Attacks and prevention. In *Proceedings of the International Conference on Mobile Technology, Applications, and Systems*, Mobility '08, pages 56 :1–56 :8, New York, NY, USA. ACM.

[Khanh Tuan, 2006] Khanh Tuan, L. (2006). Zigbee system-on-chip (soc) design. *High Frequency Electronics*, pages 16–25.

[Kodali et al., 2016] Kodali, R. K., Jain, V., Bose, S., and Boppana, L. (2016). Iot based smart security and home automation system. In *2016 International Conference on Computing, Communication and Automation (ICCCA)*, pages 1286–1289.

[Lee, 2008] Lee, E. A. (2008). Cyber physical systems : Design challenges. In *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, pages 363–369.

[Lee et al., 2007] Lee, J., Su, Y., and Shen, C. (2007). A comparative study of wireless protocols : Bluetooth, uwb, zigbee, and wi-fi. In *IECON 2007 - 33rd Annual Conference of the IEEE Industrial Electronics Society*, pages 46–51.

[Leminen et al., 2012] Leminen, S., Westerlund, M., Rajahonka, M., and Siuruainen, R. (2012). Towards iot ecosystems and business models. In Andreev, S., Balandin, S., and Koucheryavy, Y., editors, *Internet of Things, Smart Spaces, and Next Generation Networking*, pages 15–26, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Liaropoulos, 2017] Liaropoulos, A. N. (2017). Cyberspace governance and state sovereignty. In Bitros, George C.and Kyriazis, N. C., editor, *Democracy and an Open-Economy World Order*, pages 25–35. Springer International Publishing, Cham.

[Light, 2017] Light, R. (2017). Mosquitto : server and client implementation of the MQTT protocol. *The Journal of Open Source Software*, 2(13).

[Luzuriaga et al., 2015] Luzuriaga, J. E., Cano, J. C., Calafate, C., Manzoni, P., Perez, M., and Boronat, P. (2015). Handling mobility in iot applications using the mqtt protocol. In *2015 Internet Technologies and Applications (ITA)*, pages 245–250.

[Marwedel, 2018] Marwedel, P. (2018). *Embedded System Design : Embedded Systems Foundations of Cyber-Physical Systems and the Internet of Things*. Springer Publishing Company, Incorporated, 3rd edition.

[Mekki et al., 2019] Mekki, K., Bajic, E., Chaxel, F., and Meyer, F. (2019). A comparative study of lpwan technologies for large-scale iot deployment. *ICT Express*, 5(1) :1–7.

[Mendoza et al., 2018] Mendoza, F., Alonso, L., López, A., Patricia Arias Cabarcos, D., et al. (2018). Assessment of fitness tracker security : A case of study. In *Multidisciplinary Digital Publishing Institute Proceedings*, volume 2, pages 1235–1254.

[Moquette, 2019] Moquette (2019). Moquette mqtt broker. `https://moquette-io.github.io/moquette/`. Accès le : 10/11/2019.

[MQTT, 2014] MQTT (2014). Version 3.1.1. Edited by Andrew Banks and Rahul Gupta. OASIS Standard. Latest version : `http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html`.

[Naik, 2017] Naik, N. (2017). Choice of effective messaging protocols for iot systems : Mqtt, coap, amqp and http. In *2017 IEEE International Systems Engineering Symposium (ISSE)*, pages 1–7.

[NXP, 2019] NXP (2019). Sc18is602b i2c-bus to spi bridge. Consulté le 09/11/2019.

[Pariès, 2013] Pariès, J. (2013). Resilience engineering in practice : A guidebook. chapter Lessons from the Hudson. ASHGATE.

[Pimentel et al., 2008] Pimentel, A. D., Stefanov, T., Nikolov, H., Thompson, M., Polstra, S., and Deprettere, E. F. (2008). Tool integration and interoperability challenges of a system-level design flow : A case study. In Bereković, M., Dimopoulos, N., and Wong, S., editors, *Embedded Computer Systems : Architectures, Modeling, and Simulation*, pages 167–176, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Pujolle, 2006a] Pujolle, G. (2006a). Les réseaux. chapter Partie X, Le contrôle et la gestion, pages 737–866. Eyrolles, cinquième edition.

[Pujolle, 2006b] Pujolle, G. (2006b). Les réseaux. chapter Partie II, L'architecture en couche, pages 37–246. Eyrolles, cinquième edition.

[Quy Tran et al., 2018] Quy Tran, H., Van Phan, C., and Vien, Q.-T. (2018). An overview of 5g technologies. In Arya, K. V., Bhadoria, R. S., and Chaudhari, N. S., editors, *Emerging Wireless Communication and Network Technologies : Principle, Paradigm and Performance*, pages 59–80. Springer Singapore, Singapore.

[Reggiani et al., 2002] Reggiani, A., De Graaff, T., and Nijkamp, P. (2002). Resilience : An evolutionary approach to spatial economic systems. *Networks and Spatial Economics*, 2(2) :211–229.

[Renault and Carlier, 2016] Renault, V. and Carlier, F. (2016). Triskell3S, une plate-forme embarquée multi-agents pour les IoT-a. In *Journées Francophones sur les Systèmes Multi-Agents (JFSMA 2016)*, pages 181–190, Rouen, Saint-Martin-du-Vivier, France. Fabien Michel and Julien Saunier.

[Salminen et al., 2007] Salminen, E., Kulmala, A., and Hamalainen, T. D. (2007). On network-on-chip comparison. In *10th Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD 2007)*, pages 503–510.

[Schmitt et al., 2019] Schmitt, A., Carlier, F., and Renault, V. (2019). Data exchange with the mqtt protocol : Dynamic bridge approach. In *2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*, pages 1–5, Kuala Lumpur, Malaisie.

[Sebastian et al., 2018] Sebastian, A., Sivagurunathan, S., and Muthu Ganeshan, V. (2018). Iot challenges in data and citizen-centric smart city governance. In Mahmood, Z., editor, *Smart Cities : Development and Governance Frameworks*, chapter 6, pages 127–151. Springer International Publishing, Cham.

[Serpanos and Wolf, 2018] Serpanos, D. and Wolf, M. (2018). The iot landscape. In *Internet-of-Things (IoT) Systems : Architectures, Algorithms, Methodologies*, pages 1–6. Springer International Publishing, Cham.

[Sándor et al., 2015] Sándor, H., Genge, B., and Sebestyén-Pál, G. (2015). Resilience in the internet of things : The software defined networking approach. In *2015 IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)*, pages 545–552.

[Takabatake, 2011] Takabatake, T. (2011). Simulations of noc topologies for generalized hierarchical completely-connected networks. In *6th International Workshop on Reconfigurable Communication-Centric Systems-on-Chip (ReCoSoC)*, pages 1–5.

[Tan, 2018] Tan, R. (2018). Fitness app polar revealed not only where u.s. military personnel worked, but where they lived. *The Washinton Post*.

[Thirunavukkarasu et al., 2005] Thirunavukkarasu, S., Blair, G., and Coulson, G. (2005). Green : A configurable and re-configurable publish-subscribe middleware for pervasive computing. In Meersman, Tari, R., and Zahir, editors, *On the Move to Meaningful Internet Systems 2005 : CoopIS, DOA, and ODBASE : OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2005, Agia Napa, Cyprus, October 31 - November 4, 2005, Proceedings, Part I*, pages 732–749. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Trivedi et al., 2018] Trivedi, D., Khade, A., Jain, K., and Ruchira, J. (2018). Spi to i2c protocol conversion using verilog. In *Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, pages 1–4.

[Vasileios et al., 2015] Vasileios, K., Periklis, C., Francisco, V.-G., and Alonso-Zarate, J. (2015). A Survey on Application Layer Protocols for the Internet of Things. *Transaction on IoT and Cloud Computing (TICC), 2015*, 1(1).

[Vinoski, 2006] Vinoski, S. (2006). Advanced message queuing protocol. *IEEE Internet Computing*, 10(06) :87–89.

[Vlacheas et al., 2013] Vlacheas, P., Giaffreda, R., Stavroulaki, V., Kelaidonis, D., Foteinos, V., Poulios, G., Demestichas, P., Somov, A., Biswas, A. R., and Moessner, K. (2013). Enabling smart cities through a cognitive management framework for the internet of things. *IEEE Communications Magazine*, 51(6) :102–111.

[Weiser, 1999] Weiser, M. (1999). The computer for the 21st century. *SIGMOBILE Mob. Comput. Commun. Rev.*, 3(3) :3–11.

[White, 2011] White, E. (2011). *Making Embedded Systems : Design Patterns for Great Software*. O'Reilly Media, Inc.

[Wolf, 2019] Wolf, M. (2019). Chapter 2 - standard interfaces. In Wolf, M., editor, *Embedded System Interfacing*, pages 23–40. Morgan Kaufmann.

[Wu et al., 2018] Wu, C., Liu, Z., Zhang, D., Yoshinaga, T., and Ji, Y. (2018). Spatial intelligence toward trustworthy vehicular iot. *IEEE Communications Magazine*, 56(10) :22–27.

[Zimmermann, 1980] Zimmermann, H. (1980). Osi reference model - the iso model of architecture for open systems interconnection. *IEEE Transactions on Communications*, 28(4) :425–432.

Chapitre 3

Les systèmes multi-agents

Les programmes informatiques présentent généralement une flexibilité limitée dans leur capacité à gérer des événements et des conditions environnementales imprévus. En d'autres termes, ils ne peuvent agir que sur les événements et les conditions pour lesquels ils ont été conçus, et ils le font uniquement de la manière dont ils ont été programmés pour réagir à ces événements. Dans de nombreux domaines, cette rigidité des programmes répond bien aux exigences des applications. Cependant, dans certains domaines d'application, comme ceux que nous souhaitons développer dans cette thèse, une telle rigidité peut avoir un effet négatif sur le comportement d'un système et même le rendre difficile à utiliser. Par exemple, l'arrivée d'une nouvelle génération d'objets connectés sur un réseau comportant déjà des objets. Ces conditions peuvent nécessiter de nouveaux types d'échanges ou de nouvelles prises en charge pour lesquels les équipements existants n'ont pas été préprogrammés.

Des systèmes basés sur le paradigme multi-agents, semblent répondre aux besoins croissants des objets connectés à faire face à des environnements dynamiques, imprévisibles et interconnectés. Dans ce chapitre, nous présentons ce paradigme et les systèmes basés sur les agents.

3.1 | Définitions et spécifications

Le domaine des systèmes multi-agents (SMA) se concentre sur la résolution de problèmes décentralisés, dynamiques et ouverts. C'est un domaine qui comprend, entre autre, des travaux concernant : l'apprentissage [Wu et al., 2018], la négociation [Picard, 2018], les comportements stratégiques [Nguyen-Duc et al., 2003], les comportements sociaux [Huraux et al., 2015] [Othman et al., 2016] et les activités cognitives (raisonnement, croyance, désirs, intentions et émotions) [Belkaid and Sabouret, 2014].

Les SMA, combinent les recherches dans plusieurs domaines, en particulier l'intelligence artificielle et le génie logiciel. Ils émergent d'une remise en question de l'informatique séquentielle en considérant que les activités sont résolues par interaction et coopération d'entités autonomes appelées agents.

L'exploration de la littérature dans le domaine fait apparaître de nombreuses définitions du concept d'agent et de système multi-agents. Analysons quelques-unes d'entre elles.

Nous retiendrons les travaux de Kubera, qui énonce dans sa thèse une revue de la bibliographie afin de proposer un ensemble d'attributs concernant l'agent [Kubera, 2010]. Selon lui, *"un agent est une entité physique ou virtuelle autonome"*, il explique qu'un agent choisit lui-même comment il agit selon les comportements qui lui ont été programmés. Toujours selon lui, *"un agent est capable d'agir sur son environnement"*, il peut modifier son environnement au travers des actions qu'il effectue. Et enfin, il précise qu'*"un agent est situé dans un environnement"*. Il n'est pas omniscient et n'agit que sur les parties de l'environnement qu'il perçoit.

Fabien Michel, schématise d'ailleurs dans sa thèse [Michel, 2004], la relation entre un agent et son environnement par le modèle illustré figure 3.1. Il est ainsi possible de distinguer les trois éléments évoqués par Kubera dans sa définition, auxquelles s'ajoutent les notions d'action et de perception. Notons d'ailleurs, que ces deux notions sont à l'origine de nombreuses études [Ferber et al., 2005] [Stratulat et al., 2009], afin de définir comment un agent interagit avec son environnement, mais aussi avec d'autres agents. Certains estimeront que les agents interagissent uniquement avec leur environnement, et ce sont ces interactions qui permettent les échanges entre agents [Michel, 2004]. Du point de vue d'un agent, tout ce qui ne concerne pas son architecture interne fait partie de l'environnement, notamment les autres agents.

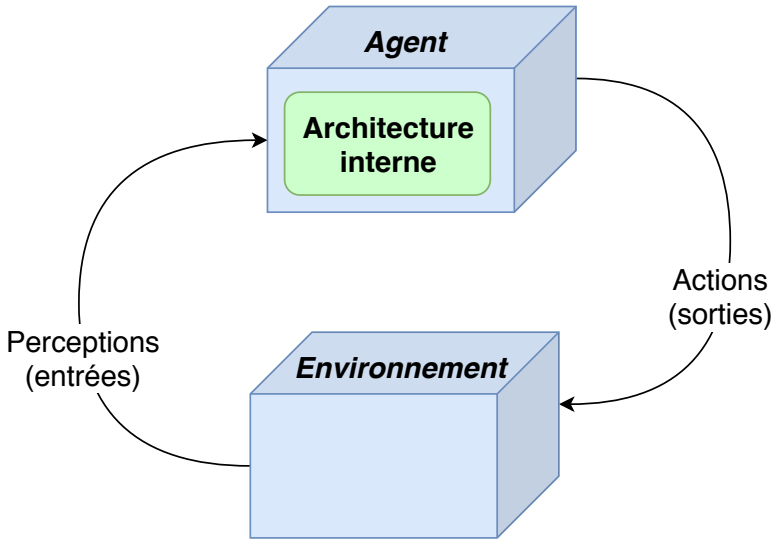


FIGURE 3.1 – Représentation d'un agent dans son environnement, [Michel, 2004]

Nous pouvons retrouver une seconde définition de l'agent au travers des travaux de [Leitão, 2009]. L'auteur concentre son analyse sur des agents mis en situation dans des objets physiques. Pour lui, un agent est *"un composant autonome qui représente des objets physiques ou logiques dans un système, en mesure d'agir pour atteindre ses buts et capable d'interagir avec d'autres agents quand il ne possède pas les connaissances ou les compétences pour atteindre seul ses objectifs"*. Ici, le terme de composant est proposé. Il soumet l'idée qu'un agent peut incarner un objet matériel et faire partie d'un système physique.

Comme nous l'avons évoqué dans le chapitre 2, les objets connectés disposent de nombreuses contraintes matérielles qu'il est nécessaire de prendre en compte pour répondre à un contexte applicatif spécifique. Les objets sont hétérogènes, ils captent et agissent sur l'environnement physique. Le paradigme multi-agents répond à des problématiques de décentralisation des traitements. Aussi, nous proposons une définition d'un agent alors empreinte des notions évoquées ci-dessus, mais aussi de celles intrinsèquement liées aux objets connectés et leurs interactions :

Définition 6 : UN AGENT

Un agent est un composant logiciel ou matériel, adapté à l'environnement matériel dans lequel il évolue. Il est apte à appréhender des grandeurs physiques réelles et logiques et en mesure de répondre à ses propres buts en autonomie. Un agent est capable d'interagir avec d'autres agents et d'autres composants matériels ou logiciels au travers de différents réseaux de communication.

Jamont évoque dans ses travaux [Jamont, 2016], le terme de *système multi-agents embarqués* et le rapproche de la notion d'agents capables de communiquer en réseau. Selon lui, *"la dimension physique, d'un système multi-agents, impacte notamment la nature des agents, des interactions et de l'environnement"*. Cette particularité fait que les agents embarqués évoluent dans un ou plusieurs systèmes et en perçoivent les réalités physiques.

L'organisation des agents joue un rôle majeur dans la façon de répondre à un cadre applicatif. Les agents sont-ils plusieurs dans un objet connecté ou bien chaque objet est-il un agent ? La question ouvre l'idée qu'un objet serait dans la capacité d'héberger plusieurs agents, qu'ils interagiraient au sein de l'objet et répondraient à un but local. Nous pensons qu'il est nécessaire de prendre ces paramètres en compte dans la définition d'un système multi-agents compatible avec le domaine des objets connectés et nous proposons donc la suivante :

Définition 7 : UN SYSTÈME MULTI-AGENTS

Un système multi-agents est une composition matérielle ou logicielle de plusieurs agents. Il propose une certaine organisation des agents. Un système multi-agents, offre la possibilité aux agents d'interagir par des vecteurs, absents ou disjoints des agents eux-mêmes.

3.2 | Système multi-agents et objets connectés

Le paradigme multi-agents se scinde principalement en trois champs de recherche. Le premier est la simulation, le second est la résolution (distribuée ou non) de problèmes et enfin, le dernier concerne les usages réels (robotique, objets connectés, industrie, etc.). La résolution de problèmes [Picard, 2018] est transversale et concerne aussi bien les problématiques de simulation que celles en usages réels. Lorsque nous évoquons le terme de systèmes multi-agents dans les systèmes réels, nous faisons allusion à des systèmes multi-agents directement confrontés à des contraintes physiques [Breivold and Sandström, 2015] : le temps, les biais techniques, les interférences, les débits, la synchronisation, etc.

Les travaux concernant la mise en place d'agents, et systèmes multi-agents, dans les systèmes réels sont de plus en plus nombreux depuis la dernière décennie. Le paradigme se développe dans de nombreux secteurs industriels [Mařík and Lažanský, 2007]. Le terme d'*agents industriels* trouve d'ailleurs son nom dans l'utilisation du paradigme multi-agents pour la gestion d'équipements industriels (machines-outils). Le secteur de l'industrie introduit cette vision dans les années 1990 aux moments où les machines industrielles se numérisent et permettent de communiquer les unes avec les autres. La notion de décentralisation de la gestion des machines-outils est alors particulièrement appréciée pour pallier les problèmes de maintenance et de tolérances aux pannes. Par exemple, une chaîne de production adapte automatiquement sa cadence en fonction des machines-outils fonctionnelles. Cette vision s'étend alors, au fur et à mesure des années, à d'autres secteurs comme l'énergie [Chouhan et al., 2013] [Abras et al., 2009] [Rocha et al., 2014], le contrôle de flux [Ioniţă and Ioniţă, 2014] [Madsen et al., 2015], l'automobile [Nam Bui and Jung, 2017] [Wu et al., 2018], puis plus récemment, les objets connectés que nous allons développer [Pico-Valencia and Holgado-Terriza, 2018] [Yu et al., 2013]. Le lecteur souhaitant avoir un aperçu bibliographique plus exhaustif de l'ensemble des secteurs dans lesquels le paradigme multi-agents est mis en application, peut se référer aux travaux suivants [Leitão et al., 2016].

Concentrons-nous plus particulièrement sur les travaux concernant les objets connectés intégrant le paradigme multi-agents. C'est-à-dire l'intégration d'un agent logiciel dans le dispositif lui-même afin qu'il acquiert les capacités inhérentes aux agents. Ainsi, l'objet peut incorporer un comportement plus dynamique, plus flexible et plus cognitif que ceux des objets connectés génériques actuellement disponibles sur le marché. Nous présentons trois projets ayant chacun une approche différente.

3.2.1 | *Agent of Things - AoT*

Le premier d'entre eux, est le projet Agent of Things (AoT) [Mzahm et al., 2013]. Il cherche à améliorer le domaine de l'IoT. Il est conçu pour atténuer les lacunes et les limites des objets connectés. L'idée centrale d'AoT est que chaque élément doit avoir une capacité de raisonnement interne. Cette intelligence permet aux objets d'interagir directement avec d'autres dans le même type de système ou dans des systèmes différents. De plus, l'AoT permet aux objets d'interagir avec d'autres systèmes directement ou par des intermédiaires. Cette caractéristique permet aux objets connectés d'être indépendants et mobiles si nécessaire.

L'hétérogénéité des architectures matérielles des objets connectés est un des points que le projet AoT soulève. Il est proposé pour cela, une architecture en six couches permettant une abstraction successive des objets et de leurs applications.

Une expérimentation dans le contexte de l'organisation du trafic permet une gestion automatisée d'un accident de la route. Plusieurs dispositifs placés le long d'une route observent les véhicules. Lorsque des véhicules rentrent en collision, ils émettent un signal au dispositif le plus proche. Ce dernier prévient alors les secours ainsi que les services de police et avertit tous les dispositifs voisins afin qu'ils émettent une alerte de sécurité aux véhicules approchant de l'accident.

3.2.2 | *Iota*

Le projet Iota [Singh and Chopra, 2017] est une introduction et une discussion face aux challenges qu'offre la mise en place d'agents dans les objets connectés. Les auteurs énoncent qu'un Iota se compose d'une interface utilisateur et d'une brique de raisonnement permettant des prises de décisions. Les Iotas n'interagissent pas directement avec les objets connectés, mais par l'intermédiaire d'un *middleware* permettant l'abstraction du matériel. Plusieurs types de middlewares sont d'ailleurs définis comme ceux responsables des capteurs, des actionneurs, du stockage de données, etc.

Un Iota est défini comme une machine unitaire, c'est-à-dire qu'un seul agent est présent sur un objet. L'ensemble des objets forme alors une organisation fonctionnant de façon similaire à un système multi-agents selon les auteurs.

Ce projet ne propose pas encore de réalisation pratique. Il ouvre de nombreuses questions concernant la synchronisation, la décentralisation, et l'organisation des objets et agents de façon à bénéficier de systèmes plus intelligents et autonomes.

3.2.3 | *Agent-based Internet of Things - AIoT*

Nous terminons par le projet Agent-based Internet of Things (AIoT) [Kato et al., 2017]. Ce projet propose une architecture appelée AIoT permettant la réalisation d'applications portées par des objets connectés. Elle est composée d'objets AIoT et répartie sur trois couches. La première, est la couche physique avec les objets connectés, la seconde est la couche middleware où chaque objet est représenté par un agent, et enfin la dernière est la couche application avec laquelle les utilisateurs peuvent interagir. Les objets de l'architecture sont des IoT *agentifiés*. Chacun d'entre eux intègre un agent midlleware (Mid-Ag). Avec cette approche, les objets connectés hétérogènes peuvent être contrôlés de manière homogène.

L'approche AIoT repose sur une organisation capable de composer les applications proposées par les objets connectés. Chaque agent middleware est relié à un agent applicatif (AP-Ag) faisant partie de la couche application. Aussi, lorsqu'un utilisateur émet une demande au système, seuls les objets permettant de répondre à la demande sont sollicités au travers de l'agent applicatif concerné. Cette organisation est aussi modulable afin de prendre en compte une reconfiguration lorsque des objets sont ajoutés ou enlevés du système. Néanmoins, cette approche est contrainte par le nombre d'agents applicatifs présents. Les utilisateurs effectuant une demande ne correspondant pas aux agents applicatifs existants ne pourront pas obtenir de résultat, alors que les objets connectés du système en ont peut-être les capacités.

L'architecture AIoT est expérimentée avec une application de logistique autonome. Différents robots agentifiés, disposent de capteurs connectés pouvant agir et capter l'environnement. Leur but est d'acheminer des paquets depuis un point de départ jusqu'à une destination. La distance à parcourir étant trop importante par rapport à leur autonomie, ils doivent se répartir le travail. Le trajet est donc découpé en plusieurs tronçons. Un ordinateur supervise les interactions entre tous les éléments. L'expérimentation montre que les robots arrivent à se répartir le travail à effectuer. Ils réussissent, de plus, à s'adapter en fonction d'obstacles en travers de la route. Dans ce cas, les robots capables de surmonter les obstacles prennent le relais des tâches à effectuer.

Ce projet soulève plusieurs points dont le fait d'être le seul à proposer une architecture, un framework, une méthode de communication et une expérimentation. L'architecture est mise en place dans des objets et ils sont capables de répondre à une application de façon décentralisée et résiliente. Néanmoins, bien que les expérimentations fassent intervenir plusieurs types d'objets connectés, l'administration des agents est déportée dans une machine distante, ici un ordinateur portable. Les auteurs de l'étude expliquent d'ailleurs que leur objectif est de proposer une architecture entièrement déportée dans un cloud dans de prochains travaux. Nous souhaitons développer une gestion entièrement décentralisée dans notre approche IoT-a et une organisation comme le propose l'architecture AIoT ne remplit pas cet objectif.

Nous observons au travers de ces travaux que l'association du paradigme multi-agents avec le domaine des objets connectés n'en est qu'à ses débuts. Les travaux [Savaglio et al., 2020] [Pico-Valencia and Holgado-Terriza, 2018] proposent d'ailleurs des méta-analyses formant une revue des travaux effectués dans ce domaine.

3.3 | Organisation des SMA

Les SMA ont été initialement développés pour le domaine de la simulation. En médecine par exemple, la simulation multi-agents permet d'approcher des modèles de développement pathologique. Les agents sont considérés comme des entités qui peuvent agir indépendamment dans un environnement en fonction de leurs comportements. Scheutz s'intéresse, dans ses travaux, à la question de hiérarchie biologique [Scheutz et al., 2005]. Il modélise les interactions entre les molécules, les membranes et les cellules afin de mieux comprendre comment ces éléments se comportent dans le corps humain. Il adopte dans sa modélisation une hiérarchie entre ses agents afin de modéliser les liens entre chaque élément qui constitue une cellule et un groupe de cellulaire. Aussi, il est capable de simuler la naissance d'un cancer et son cheminement dans le corps humain. Il s'avère que ces résultats sont très proches de certaines observations réelles, relevées sur des patients.

Nous pouvons observer l'approche systémique menée dans l'étude de Scheutz. Les agents représentent ici des entités à plusieurs niveaux d'interactions prenant en compte de nombreux paramètres de leur environnement. À un niveau microscopique, les agents représentent des molécules, évoluant en autonomie. Les agents disposent de leurs propres objectifs et interagissent les uns avec les autres sans prise en compte de l'échelle. En prenant de la hauteur, à un niveau macroscopique, les agents s'organisent en groupes et interagissent d'un groupe à un autre pour former des masses cancéreuses ou des réactions immunitaires. Cette étude permet d'introduire les différentes organisations qui existent dans la littérature. Nous résumons ci-dessous, les propriétés des modèles d'organisation les plus fréquemment rencontrés [Dorri et al., 2018] [Horling and Lesser, 2004] [Yichuan and Zhaofeng, 2011].

3.3.1 | *Organisation hiérarchique*

Les SMA hiérarchiques sont organisés de telle sorte que les agents ne peuvent interagir que s'ils sont soumis à une structure hiérarchique [Ghijsen et al., 2010]. Un avantage important de ce type de structure est la réduction significative de la complexité, et donc de la communication dans le système. L'inconvénient de l'organisation hiérarchique est la structure rigide, qui ne permet pas aux agents de s'organiser de manière dynamique pour répondre au mieux aux différents besoins et aux tâches spécifiques. En outre, la hiérarchie implique généralement que les agents de niveau inférieur dépendent des agents de niveau supérieur, et les agents de niveau supérieur peuvent même contrôler partiellement ou totalement les agents de niveau inférieur. Cela peut contraster avec les exigences d'autonomie des agents.

Une organisation hiérarchique peut également impliquer, dans une certaine mesure, un contrôle centralisé. Cette caractéristique n'est pas toujours adaptée dans des systèmes composés d'éléments appartenant à différentes organisations, et qui peuvent également être répartis géographiquement.

3.3.2 | *Organisation horizontale*

L'organisation horizontale d'un SMA implique que chaque agent peut contacter directement n'importe quel autre agent. Aucune structure fixe n'est imposée au système [Dorri et al., 2018]. Cependant, les agents peuvent former dynamiquement des structures pour effectuer des tâches spécifiques. En outre, aucun contrôle d'un agent par un autre agent n'est assumé. Une telle organisation exige que le système soit fermé, en ce sens que chaque agent connaisse tous les autres à l'avance, ou (lorsque le système est ouvert) qu'un mécanisme de localisation des agents puisse être présent dans l'infrastructure. Une organisation horizontale est avantageuse, car elle soutient pleinement l'autonomie et l'intérêt des agents ainsi que la distribution et l'ouverture du SMA. Elle permet également des ajustements dynamiques de l'organisation en fonction des changements de tâches et de l'environnement. Toutefois, l'ouverture et le dynamisme ont un coût : ils imposent des frais de communication accrus, la nécessité de mécanismes de localisation des agents et de mécanismes de réorganisation dynamique.

En outre, la quantité de raisonnement qu'un agent effectue pour interagir avec d'autres agents, augmente dans une organisation horizontale. À titre d'exemple, les SMA basés sur la plate-forme Jade [Bellifemine et al., 2001] sont organisés horizontalement, bien que d'autres organisations puissent être mises en place à l'aide de cette plate-forme.

3.3.3 | *Organisation de subsomption*

Le terme de subsomption signifie une relation d'inclusion mutuelle entre plusieurs éléments. Dans notre cas, un système multi-agents organisé par un modèle de subsomption, implique que certains agents sont des composants d'autres agents [Shehory and Sturm, 2014]. Ces agents sont subsumés par des agents conteneurs, qui à leur tour peuvent être des composants d'agents conteneurs plus importants. Le modèle de subsomption s'applique aux systèmes de prise de décisions distribués et aux SMA en général. Il peut être imagé par le fonctionnement des matriochkas [1]. Il présente une certaine similitude avec le modèle hiérarchique, néanmoins, il va plus loin en imposant que les agents subsumés cèdent le contrôle à l'agent conteneur.

Du point de vue de l'architecture logicielle, une telle organisation ressemble à une inclusion d'objets dans un objet plus grand, à l'exception de la différence dans les méthodes de contrôle. En d'autres termes, alors que les objets sont généralement contrôlés et activés par un appel de procédure ou par l'invocation d'un événement, les agents sont activés par une transmission de messages. Les relations de contrôle dans une organisation de subsomption se traduisent par une exécution efficace des tâches et une faible surcharge de communication. Toutefois, un modèle de ce type limite le système à un ensemble de tâches définies, avec une flexibilité et une adaptabilité limitées. Un SMA organisé de la sorte est difficile à faire évoluer face aux changements à long terme des tâches et de l'environnement du système. Il est, par exemple, difficile de rajouter un nouveau composant logiciel.

3.3.4 | *Organisation modulaire*

Un SMA présente une organisation modulaire lorsqu'il est composé de plusieurs modules, chacun de ces modules pouvant être perçu comme un SMA pratiquement autonome [Barbosa et al., 2018]. Généralement, la partition du système en modules se fait selon des dimensions telles que la proximité géographique ou la nécessité d'une interaction intense entre les agents et les services au sein d'un même module. Souvent, le système est composé de telles parties en raison de son processus de développement, au cours duquel de nouveaux modules ont été progressivement ajoutés à un système déjà existant.

La modularité augmente l'efficacité de l'exécution des tâches du SMA et réduit les frais généraux de communication. De plus, comme dans une organisation horizontale, chaque module présente une grande flexibilité interne. Néanmoins, la réorganisation inter-modules est assez complexe, d'où une flexibilité limitée dans cette dimension. En outre, la modularité implique des contraintes sur la communication inter-modules. Par exemple, des redondances d'éléments entre les modules peuvent occasionner des concurrences durant des communications inter-modules.

3.3.5 | *Organisation en équipes*

Dans une organisation en équipes, les agents créent un groupe, une équipe, et définissent un objectif de groupe qui diffère du leur [Dorri et al., 2018]. Les agents d'une équipe collaborent donc pour atteindre l'objectif de l'équipe. Ce dernier peut être mis à jour, ce qui entraîne une modification des responsabilités, des rôles et des pouvoirs des agents de l'équipe. Chaque équipe peut demander des informations aux agents des autres équipes pour améliorer son propre processus décisionnel. Une équipe peut avoir une organisation interne (par exemple hiérarchique) pour améliorer les performances et l'efficacité dans la réalisation de son objectif. Par exemple, dans une équipe d'agents chargés d'analyser le trafic d'une ville, les agents créent une structure hiérarchique de sorte que les données relatives au trafic puissent être intégrées par les parents afin de réduire les frais de communication.

Le nombre d'agents dans une équipe est l'une des questions clés de ce type d'organisation. Une équipe plus nombreuse peut traiter davantage de données dans l'environnement. Cependant, l'intégration des données et des connaissances de plusieurs agents exige un traitement plus complexe. La décision finale du groupe est moins difficile à prendre dans les petites équipes, cependant, les données utilisées sont limitées. Un compromis entre les frais généraux de prise de décision et l'objectif à atteindre par une équipe doit être envisagé lors de la détermination de sa taille. Cette organisation est donc adaptée lorsque plusieurs agents tentent d'atteindre le

1. Séries de poupées de tailles décroissantes placées les unes à l'intérieur des autres. Aussi appelées : poupées russes.

même objectif.

3.3.6 | *Réflexion sur une organisation en constellations*

Maintenant que nous avons détaillé quelques-unes des principales techniques d'organisation rencontrées dans le domaine des systèmes multi-agents, nous pouvons mener une réflexion sur l'organisation adaptée pour notre approche IoT-a.

Nous avons évoqué dans le chapitre précédent les contraintes de communication posées par les objets connectés. La communication intra-système permet l'échange de données entre des composants matériels. Nous venons d'ailleurs de préciser dans les définitions 6 et 7, que ces composants peuvent être des agents. D'autre part, la communication inter-systèmes offre la possibilité aux objets de communiquer entre eux. Nous devons pour cela, proposer une organisation permettant des interactions intra et inter-systèmes de nos agents.

Bien qu'intéressantes pour leurs faibles surcharges en communication, les organisations hiérarchiques et de subsomption ne permettent pas suffisamment de dynamisme dans leur évolution. C'est donc sur les trois dernières organisations que nous nous concentrons.

L'organisation horizontale ouvre de nombreuses possibilités pour la communication intra-système. Elle permet aux agents de communiquer directement entre eux. Elle tolère une mise en application dans des systèmes hétérogènes, étant donné qu'elle n'impose pas de structure particulière. Et malgré la mise en place d'un mécanisme de localisation, cette organisation supporte une dynamique capable de satisfaire les contraintes de résilience que nous avons évoquées dans la section 2.3.

De plus, l'organisation modulaire ouvre des perspectives intéressantes en matière de dynamisme entre des IoT-a. Notamment, le fait qu'un module puisse être entièrement autonome en étant perçu comme un SMA. Cette particularité nous permet d'envisager un objet connecté comme le siège du fonctionnement d'un système multi-agents. L'objet disposerait de plusieurs agents répartis à différents niveaux matériels ou logiciels dont l'ensemble serait organisé par le système multi-agents.

Par ailleurs, l'organisation en équipe permet des résolutions groupées de problématiques plus ou moins complexes en fonction du nombre d'agents. Les interactions entre un grand nombre d'objets connectés, donneraient alors lieu à une mise en commun d'un grand nombre d'agents et donc à une plus grande force de réflexion et de décision collective.

C'est donc l'association des organisations horizontale, modulaire et d'équipes qui nous permettront de proposer, dans la suite de cette thèse, une organisation propre aux IoT-a : l'organisation en constellations.

3.4 | Normes de la FIPA

Pour atteindre ses objectifs, un agent peut avoir besoin d'interagir avec ses congénères, de les coordonner, de collaborer ou de leur faire concurrence. Pour faciliter ces capacités, l'architecture d'un agent doit inclure des constructions lui permettant des interactions sociales. Un élément de communication doit permettre l'envoi, la réception et le traitement des messages. Il doit également prendre en charge des langages de communication, des formats de message et des protocoles spécifiques.

La *Foundation for Intelligent Physical Agent* (FIPA) [Poslad, 2007] est un organisme de normalisation de l'IEEE Computer Society qui promeut la technologie basée sur les agents et l'interopérabilité de ses normes avec d'autres technologies. Elle a été créée en 1996 pour produire des spécifications de normes logicielles afin de permettre des interactions entre agents hétérogènes. Depuis sa fondation, la FIPA a joué un rôle crucial dans le développement de normes pour le paradigme multi-agents. En outre de très nombreuses spécifications proposées par la fondation sont aujourd'hui utilisées dans quantité de travaux du domaine.

Les spécifications apportées par la FIPA représentent un ensemble de normes qui visent à promouvoir l'interopérabilité d'agents hétérogènes et les services qu'ils peuvent représenter, avec des systèmes réels. Le point principal sur lequel la fondation établit des propositions est la communication entre les agents.

3.4.1 | *Communication*

Afin que des protagoniste puissent communiquer entre eux, il est nécessaire qu'ils disposent d'un même langage pour se comprende mutuellement. Il existe aujourd'hui deux langages principaux pour communiquer entre des agents. Le premier connu sous le nom de *Knowledge Query and Manipulation Language* (KQML) [Finin et al., 1994] n'est pas proposé par la fondation FIPA. Il permet l'échange de données entre des bases de connaissances distribuées. Il est constitué de plusieurs performatives pour la gestion des conversations. Le KQML est le plus ancien des deux langages et il est progressivement remplacé par le second : le *FIPA Agent Communication Language* (FIPA ACL) [FIPA, 2002a]. Cette évolution peut s'expliquer aisément, en effet, le FIPA ACL utilise une sémantique plus claire que KQML. Les actes de langage sont plus simples et plus restreints tout en conservant l'autonomie des agents dans leurs échanges. De plus, la fondation propose la mise en œuvre de son langage au travers de plusieurs protocoles, comme le HTTP par exemple. Enfin, le langage FIPA ACL peut être mis en place à plusieurs niveaux dans le modèle OSI. Il peut donc être utilisé dans un grand nombre d'applications et plus particulièrement dans le domaine des objets connectés, comme le propose le projet AIoT que nous avons présenté dans la section 3.2.3.

La communication entre des agents est basée sur les conversations humaines. Entre humains, nous échangeons des phrases contenant un verbe et un sujet. Les agents échangent des performatives ou des actes de langage en fonction des théories. Dans les années 1960, plusieurs travaux regroupent les actes de langage que nous avons entre êtres humains pour les catégoriser de la façon suivante :

— les représentations, qui donnent des informations,
— les directives, qui exigent quelque chose du destinataire,
— les commissions, qui engagent l'expéditeur à faire quelque chose à l'avenir,
— les expressions, qui décrivent l'état mental de l'expéditeur,
— les déclarations, qui traitent un acte en le prononçant simplement.

Les agents ne se servent pas de toutes les phrases que nous pouvons utiliser, mais seulement une partie. En l'occurrence, en fonction du langage choisi, seules les formes représentatives et directives sont utilisées. Les langues de communication utilisées dans le paradigme multi-agents ne peuvent pas être aussi riches que les langues naturelles, ces dernières nécessiteraient trop d'interprétation de la part des agents.

Performative	**Transmission d'informations**	**Demande d'informations**	**Négociation**	**Exécution d'actions**	**Traitement d'erreurs**
Accept-proposal			√		
Agree				√	
Cancel		√		√	
Cfp			√		
Confirm	√				
Disconfirm	√				
Failure					√
Inform	√				
Inform-if	√				
Inform-ref	√				
Not-understood					√
Propose			√		
Query-if		√			
Query-ref		√			
Refuse				√	
Reject-proposal			√		
Request				√	
Request-when				√	
Request-whenever				√	
Subscribe		√			

Tableau 3.1 – Performatives proposées par le standard de communication FIPA ACL

Les performatives de FIPA ACL sont classées en cinq catégories : la transmission d'informations, la demande d'informations, la négociation, l'exécution d'actions et le traitement des erreurs. Le tableau 3.1 résume les

différentes performatives et leurs catégories. Les performatives *Inform* et *Request* sont des primitives de FIPA ACL. En d'autres termes, différentes performatives peuvent être construites à partir de celles-ci. La performative *Inform* signifie qu'un agent informe les destinataires de quelque chose auquel il croit et que les destinataires ne connaissent pas encore. La performative *Request* demande aux destinataires d'exécuter des actions qu'il pense réalisables par ces derniers. La figure 3.2 illustre un exemple d'interaction entre deux agents. L'agent Alice émet une requête à Bob qui peut l'accepter ou la refuser. Lorsqu'il l'accepte, Bob exécute la tâche demandée et informe Alice à l'issue de son déroulement.

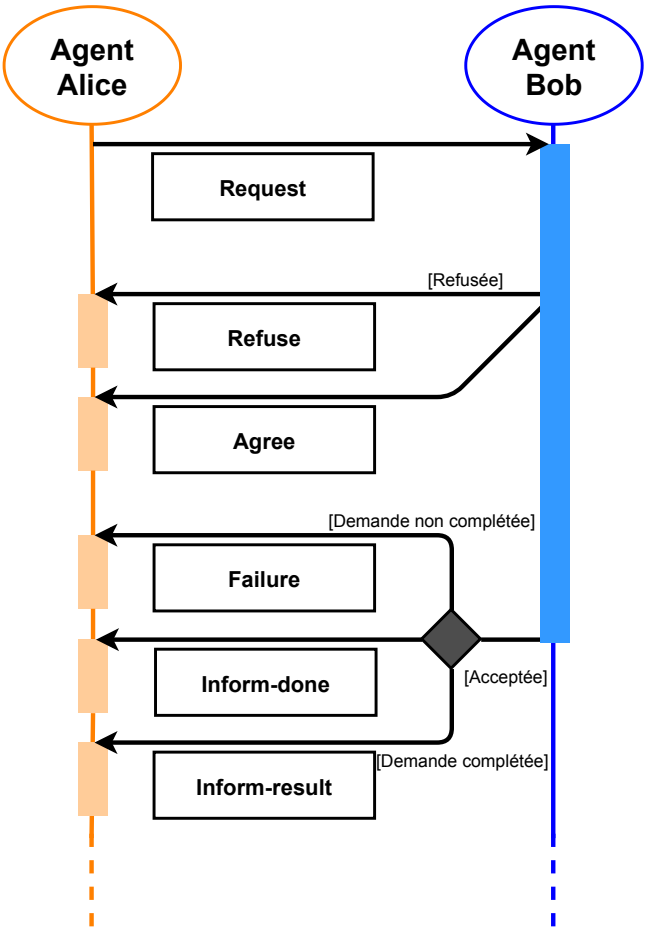


FIGURE 3.2 – Diagramme du protocole d'interaction FIPA dans le cadre d'une demande (request)

3.4.2 | *Transport*

Les messages ACL proposés par la FIPA représentent l'information à transiter entre les agents. Étant donné l'hétérogénéité des matériels dans lesquels les agents sont susceptibles d'évoluer, la FIPA propose un modèle de référence pour le transport de ces messages. La norme [FIPA, 2002c] définit le modèle MTPS pour *Message Transport Service Protocol*, constitué de trois niveaux, représenté sur la figure 3.3.

Le premier d'entre eux, est le protocole utilisé pour le transport de message, il est appelé *Message Transport Protocol*. Aucune restriction n'est donnée de la part de la FIPA. Aussi, les différents protocoles que nous avons définis dans le chapitre précédent sont compatibles avec cette norme. Mais un *protocole* au sens de la FIPA n'a pas la même signification que nous avons donné avec le modèle OSI. Il englobe toutes les méthodes de transport de l'information d'un message ACL, qu'elles soient bas niveau, comme les bus de communication intra-système, ou haut niveau comme les protocoles de communication.

Le second niveau est le *Message Transport Service* offrant les méthodes logicielles nécessaires pour accéder à un MTP. Il dispose aussi d'une fonction permettant d'encapsuler les messages à faire parvenir dans ce que la FIPA appelle une enveloppe. Cette enveloppe permet de préciser le destinataire, l'expéditeur, la date et le format du contenu. Elle a significativement le même rôle que la mise en paquets dans la norme IP. La méthode permet de faire transiter des données sans en révéler le contenu.

Le dernier niveau du modèle MTPS est donc le message de type FIPA ACL généré par un agent, à faire parvenir à d'autres. Ce modèle est particulièrement intéressant pour notre approche IoT-a. Elle permet une communication entre de nombreux types de technologies. Nous pouvons, avec cette méthode, proposer des interactions entre plusieurs agents présents dans un seul objet. Nous procédons, de cette façon, à une communication intra-système. De plus, l'approche permet des échanges de données entre des agents situés dans différents objets connectés. Nous obtenons une solution adaptée à une communication inter-systèmes.

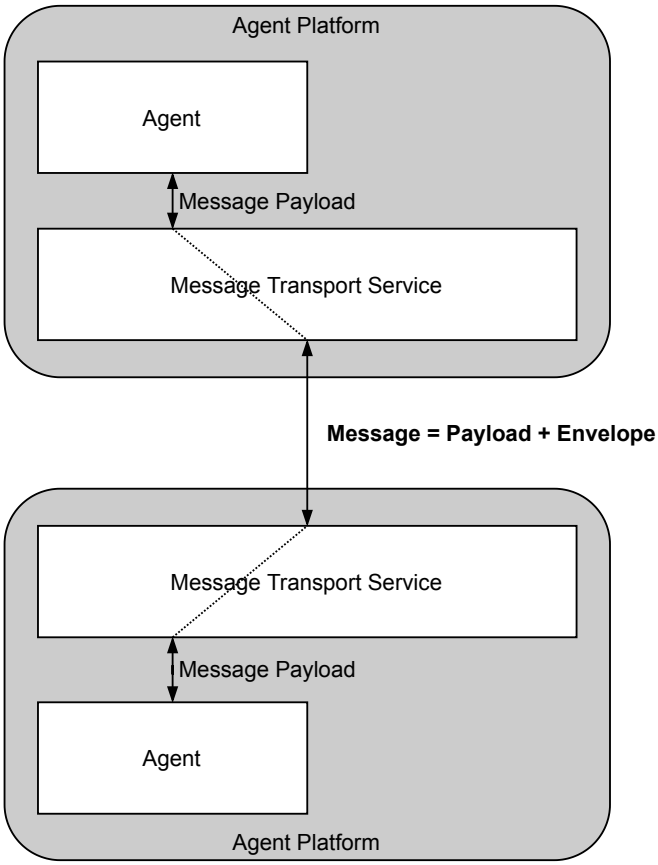


FIGURE 3.3 – Modèle de référence pour le transport de messages par la norme de la FIPA, [FIPA, 2002c]

3.4.3 | *Administration*

Nous avons constaté dans notre présentation des organisations de systèmes multi-agents, qu'une structure horizontale nécessite une administration pour renseigner la présence et les services proposés par les agents. La FIPA propose, là encore, une architecture permettant l'administration, la gestion et le référencement des agents dans un SMA [FIPA, 2004].

La gestion des agents selon la FIPA, fournit un cadre dans lequel ils existent et interagissent. La fondation établit le modèle de référence pour la création, l'enregistrement, la localisation, la communication et le retrait des agents. Ils sont notamment identifiés par un paramètre appelé *Agent Identifier (AID)*, constitué d'un nom et d'une adresse. Cela permet de distinguer chaque élément de l'écosystème et de pouvoir le contacter.

La figure 3.4 illustre l'architecture d'une plate-forme multi-agents proposée par la FIPA. L'élément appelé *Agent* est l'entité autonome et communicante que nous avons définie en début de chapitre. Il échange des informations, en utilisant le langage FIPA ACL, au travers d'un système de transport de messages, que nous avons présenté dans la section précédente et représenté dans la figure par le cadre appelé *Message Transport System*. Il est identifié par son AID et propose des services à sa communauté.

Pour être connu des autres et proposer ses services au sein d'une plate-forme, un agent doit s'enregistrer auprès d'une autorité de renseignement. En l'occurrence, la FIPA propose un second type d'agent appelé : "*agents de plate-forme*". Ils sont responsables de l'organisation des agents.

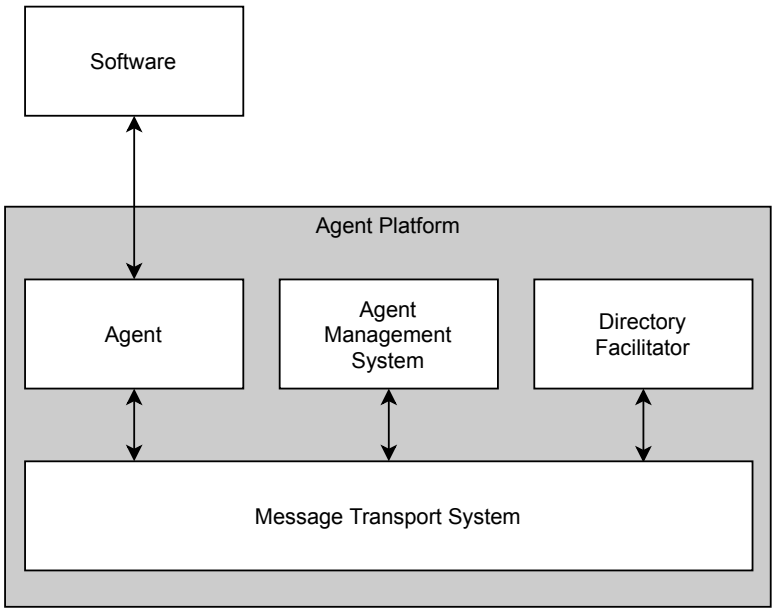


FIGURE 3.4 – Modèle de référence pour la gestion des agents, [FIPA, 2004]

AMS

Le responsable de gestion des agents est l'*Agent Management Systeme (AMS)*. Il est obligatoirement présent et unique. Il supervise et contrôle les membres d'une plate-forme.

Lorsqu'un agent naît ou arrive sur une plate-forme, il s'enregistre auprès d'AMS. Ce dernier répertorie l'AID du nouvel arrivant dans un registre. C'est d'ailleurs durant son inscription que l'agent obtient un AID valable. Les agents peuvent également contacter AMS pour lui demander de mettre à jour les informations les concernant (statut ou adresse) par le biais d'une fonction de modification. Quand un agent n'est plus actif, il en informe également AMS et ce dernier le retire de sa liste. AMS offre aussi un service de pages blanches. Les agents peuvent le contacter afin d'obtenir l'adresse d'un de leurs congénères.

Selon l'implémentation de la norme de la FIPA dans une plate-forme multi-agents, AMS peut se voir octroyer des fonctions de contrôle supplémentaires. Par exemple, il est possible de lui permettre de demander à un agent de s'arrêter, voire de forcer son arrêt en cas de nécessité. Une autre option est de lui donner la possibilité d'ajouter des agents dans le système.

DF

Le second agent de plate-forme représenté sur la figure 3.4 est le facilitateur d'annuaire : *Directory Facilitator (DF)*. Il tient une liste des services actifs sur la plate-forme et des agents auxquels ils sont reliés. Le maintien de cette liste permet de savoir, à tout instant, quelles capacités sont à disposition sur la plate-forme. Il est possible de disposer de plusieurs agents DF au sein d'un SMA. La répartition des services proposés sur différentes listes peut alors se faire selon leurs types, ou bien selon leurs localisations si le SMA s'étend à plusieurs machines. D'autres critères de répartition peuvent être choisis.

À l'instar d'AMS, DF propose des fonctions d'enregistrement, de désenregistrement et de modification des informations sauvegardées. Les agents l'informent des services qu'ils proposent lors de leur activation. Ils se désinscrivent de DF avant leur mort ou leur départ de la plate-forme. L'agent DF apporte également une fonction de recherche qui permet d'obtenir l'identité des agents proposant un service donné. Lorsque plusieurs agents DF sont présents sur un même SMA et que la fonction recherche de l'agent DF contacté ne donne pas de résultat, il va étendre cette recherche aux autres agents DF du système.

3.5 | Synthèse

Nous nous sommes concentrés dans ce chapitre, sur le paradigme multi-agents. Nous avons défini la notion d'agent et de système multi-agents, conformément aux travaux théoriques et pratiques de la littérature. Les quelques projets alliant ce paradigme avec le domaine des objets connectés, et les méta études que nous avons présentés montrent l'intérêt de la communauté scientifique à examiner cette association entre SMA et IOT.

Pour parvenir à répondre à notre problématique, nous nous sommes intéressés aux différentes organisations de système multi-agents. Nous avons conclu qu'une nouvelle organisation en constellation, associant trois d'entre elles, serait susceptible de convenir à la contrainte de communication intra et inter-systèmes, imposée par l'hétérogénéité des objets connectés.

La communication est donc le sujet que nous avons souhaité préciser. Nous avons présenté les normes de la fondation FIPA et notamment le langage FIPA ACL permettant des interactions homogènes entre les agents. Les méthodes de transport de messages proposées par la fondation ont aussi été abordées, elles sont d'ailleurs compatibles avec tout type de protocole de communication. Enfin, le modèle d'architecture et d'administration, avancé par la FIPA, permet de dresser un cadre dans la mise en place d'agents dans les objets connectés. Il va nous guider dans une proposition d'un modèle de l'IoT-a.

Nous pouvons maintenant relier les éléments de nos deux chapitres d'état de l'art pour échafauder nos propositions scientifiques. Nous pouvons affirmer que l'association du paradigme multi-agents avec le domaine des objets connectés est un sujet régulièrement abordé en science de l'informatique et qui suscite de l'intérêt. Le point principal à retenir de notre premier chapitre est que les objets connectés sont divers et intègrent une pluralité de technologies les rendant hétérogènes. Leur omniprésence et leur quantité font qu'ils représentent une société d'éléments, pouvant potentiellement faire preuve d'intelligence, mais encore faut-il les combiner convenablement.

D'après notre second chapitre, le paradigme multi-agents est assez vaste pour proposer des solutions permettant de créer des agents adaptés à l'hétérogénéité des objets connectés. Nous nous sommes principalement intéressés au sujet de la communication, car elle est, d'après de nombreux travaux, le cœur des systèmes distribués. Le partage, l'échange et la négociation entre les acteurs d'une société en font émerger une certaine intelligence. Nous souhaitons développer ces principes. C'est pourquoi nous nous concentrons à interconnecter un maximum de niveaux matériels dans les objets connectés, en y associant des agents capables de communiquer, à l'intérieur et vers l'extérieur d'un système.

Notre prochaine partie détaille nos propositions en ce sens. Elle présente, dans un premier chapitre, notre approche de l'IoT-a et notre proposition de constellations d'IoT-a. Dans un second chapitre, nous présentons une application mettant en œuvre notre approche.

Bibliographie

[Abras et al., 2009] Abras, S., Ploix, S., Pesty, S., and Jacomino, M. (2009). Apport d'une approche multi-agents pour la résolution d'un problème de gestion de l'énergie dans l'habitat. In *17emes Journées Francophones sur les Systèmes Multi-Agents*, pages 110–116, Lyon, France. Cepadues.

[Barbosa et al., 2018] Barbosa, J., Leitão, P., and Teixeira, J. (2018). Empowering a cyber-physical system for a modular conveyor system with self-organization. In Borangiu, T., Trentesaux, D., Thomas, A., and Cardin, O., editors, *Service Orientation in Holonic and Multi-Agent Manufacturing : Proceedings of SOHOMA 2017*, pages 157–170. Springer International Publishing, Cham.

[Belkaid and Sabouret, 2014] Belkaid, M. and Sabouret, N. (2014). Un modèle logique de théorie de l'esprit pour un agent virtuel dans le contexte de simulation d'entretien d'embauche. In *Workshop Affects, Compagnons Artificiels et Interaction*, Rouen, France.

[Bellifemine et al., 2001] Bellifemine, F., Poggi, A., and Rimassa, G. (2001). Developing multi-agent systems with a fipa-compliant agent framework. *Softw., Pract. Exper.*, 31 :103–128.

[Breivold and Sandström, 2015] Breivold, H. P. and Sandström, K. (2015). Internet of things for industrial automation – challenges and technical solutions. In *2015 IEEE International Conference on Data Science and Data Intensive Systems*, pages 532–539.

[Chouhan et al., 2013] Chouhan, S., Ghorbani, J., Inan, H., Feliachi, A., and Choudhry, M. A. (2013). Smart mas restoration for distribution system with microgrids. In *2013 IEEE Power Energy Society General Meeting*, pages 1–5.

[Dorri et al., 2018] Dorri, A., S. Kanhere, S., and Jurdak, R. (2018). Multi-agent systems : A survey. *IEEE Access*, 6 :28573–28593.

[Ferber et al., 2005] Ferber, J., Michel, F., and Baez-barranco, J.-A. (2005). AGRE : Integrating Environments with Organizations. In Weyns D., P. V. D. M. F., editor, *E4MAS'04 : Environments for Multiagent Systems*, number 3374 in Lecture Notes in Computer Science, pages 127–134, Melbourne (Australie).

[Finin et al., 1994] Finin, T., Fritzson, R., McKay, D. P., and McEntire, R. (1994). KQML - A Language and Protocol for Knowledge and Information Exchange. In *13th Int. Distributed Artificial Intelligence Workshop*, pages 93–103. AAAI Press.

[FIPA, 2002a] FIPA, C. (2002a). Fipa acl message structure specification - sc00061g. *FIPA TC C*.

[FIPA, 2002b] FIPA, C. (2002b). Fipa agent message transport service specification - sc00067f. *FIPA TC B*.

[FIPA, 2004] FIPA, C. (2004). Fipa agent management specification - sc00023k. *FIPA TC B*.

[Ghijsen et al., 2010] Ghijsen, M., Jansweijer, W. N. H., and Wielinga, B. J. (2010). Adaptive hierarchical multi-agent organizations. In Babuška, Robertand Groen, F. C. A., editor, *Interactive Collaborative Information Systems*, pages 375–400. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Horling and Lesser, 2004] Horling, B. and Lesser, V. (2004). A survey of multi-agent organizational paradigms. *The Knowledge Engineering Review*, 19 :281–316.

[Huraux et al., 2015] Huraux, T., Sabouret, N., Haradji, Y., and Sempé, F. (2015). Simulations multi-agents de l'activité humaine : application dans le contexte énergétique résidentiel français. In *Applications Pratiques de l'Intelligence Artificielle (APIA 2015)*, page 8, Rennes, France.

[Ioniţă and Ioniţă, 2014] Ioniţă, L. and Ioniţă, I. (2014). Nm-mas : A multi-agent system for network management in oil industry. In *2014 RoEduNet Conference 13th Edition : Networking in Education and Research Joint Event RENAM 8th Conference*, pages 1–6.

[Jamont, 2016] Jamont, J.-P. (2016). *Démarche, modèles et outils multi-agents pour l'ingénierie des collectifs cyber-physiques*. Habilitation à diriger des recherches, Université Grenoble Alpes.

[Kato et al., 2017] Kato, T., Takahashi, H., and Kinoshita, T. (2017). Multiagent-based autonomic and resilient service provisioning architecture for the internet of things. *International Journal of Computer Science and Network Security*, 17 :36–58.

[Kubera, 2010] Kubera, Y. (2010). *Simulations orientées-interaction des systèmes complexes*. Theses, Université des Sciences et Technologie de Lille - Lille I.

[Leitão, 2009] Leitão, P. (2009). Agent-based distributed manufacturing control : A state-of-the-art survey. *Engineering Applications of Artificial Intelligence*, 22(7) :979–991. Distributed Control of Production Systems.

[Leitão et al., 2016] Leitão, P., Karnouskos, S., Ribeiro, L., Lee, J., Strasser, T., and Colombo, A. W. (2016). Smart agents in industrial cyber physical systems. *Proceedings of the IEEE*, 104(5) :1086–1101.

[Madsen et al., 2015] Madsen, B., Rzevski, G., Skobelev, P., and Tsarev, A. (2015). A strategy for managing complexity of the global market and prototype real-time scheduler for lego supply chain. *International Journal of Software Innovation*, 1 :28–39.

[Mařík and Lažanský, 2007] Mařík, V. and Lažanský, J. (2007). Industrial applications of agent technologies. *Control Engineering Practice*, 15(11) :1364–1380. Special Issue on Manufacturing Plant Control : Challenges and Issues.

[Michel, 2004] Michel, F. (2004). *Formalisme, outils et éléments méthodologiques pour la modélisation et la simulation multi-agents*. Theses, Montpellier II.

[Mzahm et al., 2013] Mzahm, A. M., Ahmad, M. S., and Tang, A. Y. C. (2013). Agents of things (aot) : An intelligent operational concept of the internet of things (iot). In *2013 13th International Conference on Intellient Systems Design and Applications*, pages 159–164.

[Nam Bui and Jung, 2017] Nam Bui, K.-H. and Jung, J. J. (2017). Internet of agents framework for connected vehicles : A case study on distributed traffic control system. *Journal of Parallel and Distributed Computing*.

[Nguyen-Duc et al., 2003] Nguyen-Duc, M., Briot, J., and Drogoul, A. (2003). An application of multi-agent coordination techniques in air traffic management. In *IEEE/WIC International Conference on Intelligent Agent Technology, 2003. IAT 2003.*, pages 622–625.

[Othman et al., 2016] Othman, A., Zargayouna, M., Scémama, G., and Zeddini, B. (2016). Effets de l'information temps-réel des voyageurs : une simulation multi-agent.

[Picard, 2018] Picard, G. (2018). Optimisation sous contraintes distribuée : une introduction au domaine. In *26èmes Journées Francophones sur les Systèmes Multi-Agents (JFSMA 2018)*, pages 43–52, Métabief, France. Cépaduès.

[Pico-Valencia and Holgado-Terriza, 2018] Pico-Valencia, P. and Holgado-Terriza, J. (2018). Agentification of the internet of things : A systematic literature review. *International Journal of Distributed Sensor Networks*, 14.

[Poslad, 2007] Poslad, S. (2007). Specifying protocols for multi-agent systems interaction. *ACM Trans. Auton. Adapt. Syst.*, 2(4).

[Rocha et al., 2014] Rocha, A., Di Orio, G., Barata, J., Antzoulatos, N., Castro, E., Scrimieri, D., Ratchev, S., and Ribeiro, L. (2014). An agent based framework to support plug and produce. In *2014 12th IEEE International Conference on Industrial Informatics (INDIN)*, pages 504–510.

[Savaglio et al., 2020] Savaglio, C., Ganzha, M., Paprzycki, M., Bădică, C., Ivanović, M., and Fortino, G. (2020). Agent-based internet of things : State-of-the-art and research challenges. *Future Generation Computer Systems*, 102 :1038–1053.

[Scheutz et al., 2005] Scheutz, M., Madey, G., and Boyd, S. (2005). tmans - the multi-scale agent-based networked simulation for the study of multi-scale, multi-level biological and social phenomena. In *Proceedings of Spring Simulation Multiconference (SMC 05)*, Agent-Directed Simulation Symposium, San Diego.

[Shehory and Sturm, 2014] Shehory, O. and Sturm, A. (2014). Multi-agent systems : A software architecture viewpoint. In Shehory, Onnand Sturm, A., editor, *Agent-Oriented Software Engineering : Reflections on Architectures, Methodologies, Languages, and Frameworks*, pages 57–78. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Singh and Chopra, 2017] Singh, M. P. and Chopra, A. K. (2017). The internet of things and multiagent systems : Decentralized intelligence in distributed computing. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pages 1738–1747.

[Stratulat et al., 2009] Stratulat, T., Ferber, J., and Tranier, J. (2009). MASQ - Towards an Integral Approach to Agent-Based Interaction. In *AAMAS'09 : The Eighth International Conference on Autonomous Agents and MultiAgent Systems*, page 8, Budapest, Hungary.

[Wu et al., 2018] Wu, C., Liu, Z., Zhang, D., Yoshinaga, T., and Ji, Y. (2018). Spatial intelligence toward trustworthy vehicular iot. *IEEE Communications Magazine*, 56(10) :22–27.

[Yichuan and Zhaofeng, 2011] Yichuan, J. and Zhaofeng, L. (2011). Locality-sensitive task allocation and load balancing in networked multiagent systems : Talent versus centrality. *Journal of Parallel and Distributed Computing*, 71(6) :822–836. Special Issue on Cloud Computing.

[Yu et al., 2013] Yu, H., Shen, Z., and Leung, C. (2013). From internet of things to internet of agents. In *Proceedings of the 2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*, GREENCOM-ITHINGS-CPSCOM '13, pages 1054–1057, Washington, DC, USA. IEEE Computer Society.

Partie II

Propositions transversales

Introduction

Nous avons évoqué dans notre état de l'art les défis actuels apporté par les objets connectés. Nous avons relevé dans la section 2.2 les modèles économiques appliqués à leur gestion par de grandes entreprises qui centralisent l'information et leurs traitements. Les effets d'une telle organisation ont été observés et nous souhaitons proposer une nouvelle approche décentralisée de la gestion des objets connectés.

Notre problématique se concentrent alors, sur les adaptations nécessaires pour satisfaire une organisation autonome d'objets connectés tant pour la gestion de leurs données, de leurs comportements et de leurs communications. Nous faisons pour cela, recours au paradigme des systèmes multi-agents et émettons l'hypothèse qu'un certain nombres d'agents adaptés mis en relation et portés par les objets connectés, permet de répondre à une application donnée.

Nous proposons de replacer notre problématique par rapport à différents points de vue de l'approche IoT-a avancée par l'équipe Tifaifai, dont cette thèse fait partie. Ils sont illustrés par le figure 4.0.

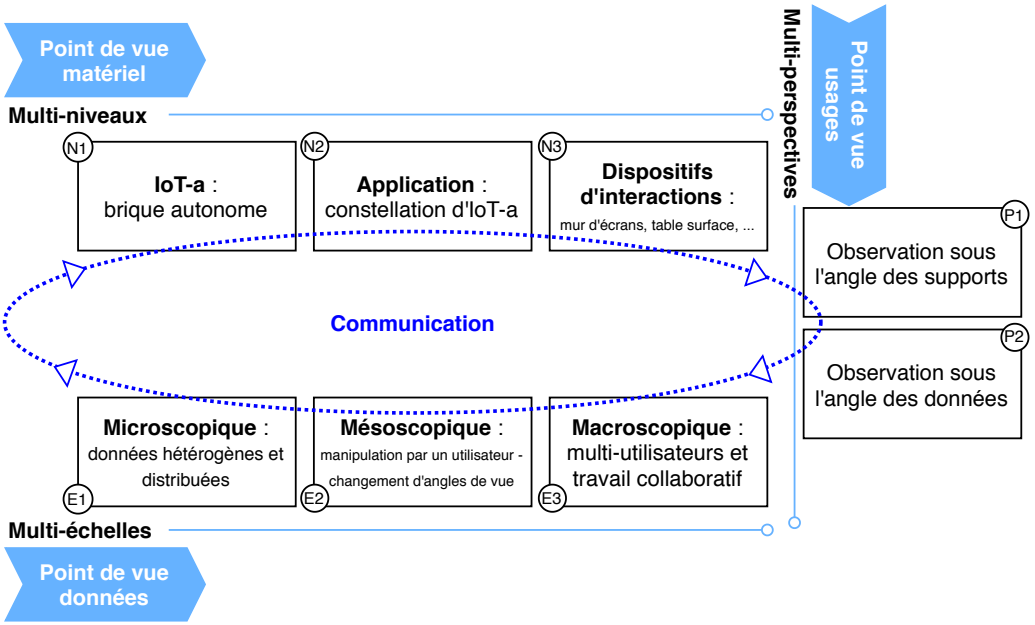


Figure 4.0 – Différents points de vue de l'approche IoT-a

Un premier point de vue évoque le sujet sous la perspective **du matériel**, ou autrement dit des supports physiques, les objets connectés. Nous parlons dans cette partie de l'organisation des systèmes multi-agents avec les matériels de façon à interagir, capter et agir sur l'environnement physique. La répartition des IoT-a selon leurs caractéristiques techniques nous a conduit à aborder la notion de multi-niveaux.

Un second point de vue expose la gestion multi-échelles **des données** par le biais des IoT-a. Les objets connectés produisent des données, mais sont aussi capables de les traiter collectivement afin de produire des informations. À une échelle microscopique, les données sont brutes, bruitées, difficiles à exploiter et présentes dans chaque objet. À une échelle mésoscopique, les données sont agrégées entre les objets, modelées, filtrées et donnent naissance à des informations exploitables et visualisables par les utilisateurs. Enfin, à une échelle macroscopique, un ou plusieurs utilisateurs peuvent visualiser, travailler et interagir avec les données.

Se pose alors la question de la perspective à adopter en fonction **des usages**. Le but du troisième point de vue est d'ouvrir la voie vers de nouveaux dispositifs permettant des usages sous l'angle des données et l'angle des supports physiques. Ce point de vue fait intervenir de nombreux domaines techniques mais aussi des compétences issues des sciences sociales et comportementales, la mise en place de nouveaux outils n'est pas sans effet auprès des utilisateurs.

Les travaux développés dans cette thèse ne s'intéressent qu'à une partie des questions soulevées par ces différents points de vue. Ils serait trop ambitieux de vouloir en apporter une réponse globale. Nous nous intéressons spécifiquement à la communication qui doit être compatible entre plusieurs niveaux de matériel. Elle doit permettre l'interaction entre les utilisateurs, les éléments matériels et les échelles de données. Nous détaillons pour cela nos proposition dans les chapitres suivants.

Dans le chapitre 4, nous abordons le concept de brique autonome IoT-a sous le point de vue matériel. Nous détaillons les éléments qui constituent un IoT-a puis nous voyons comment les agents sont répartis au sein des objets connectés et comment ils communiquent. Nous nous concentrons dans un second temps sur leur organisation en introduisant les notions de centralisation locale et décentralisation collective. Nous rassemblons ces notions au sein de notre proposition de constellation d'IoT-a.

Dans le chapitre 5, nous présentons un premier outil permettant de répondre au point de vue des usages tout en prenant compte des données. Cet outil repose sur la mise en place réelle de nos éléments théoriques développés dans le chapitre 4 et en lien direct avec un usage spécifique en entreprise.

Chapitre 4

De l'IoT à l'IoT-a

Nous présentons dans ce chapitre notre approche IoT-a associant le paradigme multi-agents et le domaine des objets connectés. Nous précisons nos travaux permettant une prise en compte de l'hétérogénéité des systèmes embarqués et voyons les éléments facilitant l'émergence de nouveaux comportements entre les objets connectés. Ainsi, nous dressons le modèle de l'IoT-a constituant une brique autonome capable d'interagir avec d'autres IoT-a.

Dans un second temps, nous nous intéressons plus précisément à l'organisation et aux interactions entre IoT-a. Nous définissons la constellation d'IoT-a et caractérisons les principes de centralisation locale et décentralisation collective pour évoquer les interactions intra et inter-constellations. Enfin, nous étudions comment les constellations d'IoT-a s'organisent dynamiquement dans l'objectif de les rendre résilientes face aux problématiques de mobilité ou de perte de connexion, propres au domaine des objets connectés.

Afin de construire notre discours, nous allons suivre les étapes présentées dans la figure 4.1, plus précisément le niveau $N1$ concernant le concept d'IoT-a, le niveau $N2$ se rapportant à la notion de constellation d'IoT-a et enfin le niveau $N3$ présentant l'idée de dispositifs d'interactions entre les utilisateurs et les objets physiques.

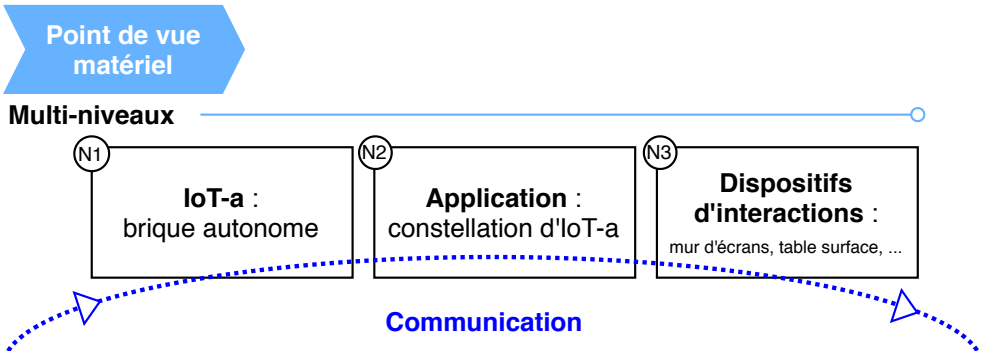


FIGURE 4.1 – Point de vue matériel de l'approche IoT-a

4.1 | Brique autonome IoT-a

Le marché actuel des objets connectés propose de nombreuses catégories de matériels et se pose alors la question de comment introduire le paradigme multi-agents au sein de ces objets industriels ? Nous l'avons vu dans la partie état de l'art, la mise en place du paradigme multi-agents en lien avec des systèmes communicants fait déjà l'étude de nombreux travaux dans la littérature. Nous souhaitons rapprocher les travaux issus du domaine des systèmes multi-agents ainsi que ceux proposés dans l'architecture *Service Oriented Architecture* (SOA) et plus précisément le paradigme des micro services. Cette vision a pour but de définir les agents susceptibles de correspondre à un cadre applicatif et s'insérer au sein d'un objet pour former un IoT-a.

4.1.1 | Contexte

La première définition du terme IoT-a apparaît dans [Renault and Carlier, 2016] et signifie : *Internet of Things - agent(s)*. Elle désigne la combinaison d'un objet connecté avec un ou plusieurs agents issus du paradigme des systèmes multi-agents. Les agents sont portés par l'objet sous la forme d'un composant logiciel ou matériel. L'objectif du concept d'IoT-a consiste à apporter des réponses à des besoins d'applications décentralisées. Cette approche s'inscrit dans une logique inverse à celle appliquée régulièrement dans l'industrie, ayant pour mission de centraliser un maximum les données et leurs traitements dans des fermes de serveurs. Le concept d'IoT-a aborde l'utilisation des systèmes multi-agents au sein d'objets connectés réels pour en faire émerger une intelligence collective portée par les objets eux-mêmes. Nous pourrons d'ailleurs utiliser dans la suite de ce chapitre, la notation d'*objet-agent(s)* désignant cette idée de relation 1 : N entre objet et agents. Nous définissons donc un IoT-a de la façon suivante :

Définition 8 : Un IoT-a

Un IoT-a est un objet connecté auquel est intégré un ou plusieurs agents. Les agents peuvent être logiciels ou matériels sous forme de composants électroniques. Un IoT-a dispose des mécanismes nécessaires pour permettre l'interaction de ses agents.
Un IoT-a peut évoluer seul mais son intérêt se justifie lorsqu'il interagit avec d'autres.

Tout comme un agent, un IoT-a n'a pas vocation à évoluer seul, c'est pourquoi il doit échanger avec d'autres IoT-a dans un environnement commun. Aussi, nous entendons par le terme "*les IoT-a*", la définition suivante :

Définition 9 : Un ensemble d'IoT-a

Un ensemble d'IoT-a est un réseau d'objets connectés hétérogènes dans lequel, chacun d'entre eux intègre un ou plusieurs agents évoluant dans un même environnement.
Un IoT-a est un objet-agent(s), alors que le terme "*les IoT-a*" désigne une organisation d'IoT-a.

Le lecteur habitué à évoquer le sujet de l'objet connecté en parlant du domaine par le terme "*IoT*", aura sûrement tendance à considérer l'IoT comme un réseau d'objets connectés et ne pas trouver pertinent l'utilisation du terme IoT-a pour ne désigner qu'un seul objet-agent(s). Néanmoins, nous insistons sur cet aspect bivalent du terme IoT-a, désignant à la fois un objet agentifié et un groupe d'objets-agent(s), car c'est la multiplication de ces objets pour former un groupe d'IoT-a cohérent qui justifie l'intérêt de ce concept.

4.1.2 | Comportements

Les tâches à réaliser par un agent sont conditionnées par son cycle de vie. C'est pourquoi nous devons détailler ce dernier avant la notion de comportement.

Nos réflexions en matière de comportements s'appuient, dans un premier temps, sur les recommandations FIPA définissant la gestion des agents. Dans ces travaux, la fondation précise le modèle d'un agent et les différents agents nécessaires à l'organisation d'un SMA. Nous retrouvons dans ce document la spécification de l'*Agent Life Cycle* (ALC), à ne pas confondre avec ACL pour *Agent Communication Language*. L'ALC est une machine d'états définissant le cycle de vie d'un agent. La figure 4.2 détaille les états dans lesquels un agent peut se trouver et les transitions pour passer d'un état à un autre. Le modèle ALC est indépendant de tout système d'application et il définit les états et les transitions d'un service de l'agent.

D'après l'ALC, les comportements des agents peuvent être administrés par l'agent lui-même, ou par un agent de gestion appelé *Agent Management System* : AMS. Ainsi, l'AMS est responsable des services proposés par les agents d'une plate-forme. Il a connaissance de leurs états et dispose des outils nécessaires pour les contrôler et satisfaire les demandes de gestion d'un système multi-agents.

L'ensemble des transitions illustrées dans la figure 4.2 sont définies dans le document [FIPA, 2004]. Notons que les deux transitions *Move* et *Execute* ne sont utilisées que par des agents mobiles. Les agents mobiles sont définis comme étant capables de changer d'environnement d'exécution pour passer d'une plate-forme multi-agents à une autre à la volée. Bien que les applications théoriques ouvrent de nombreuses perspectives intéressantes,

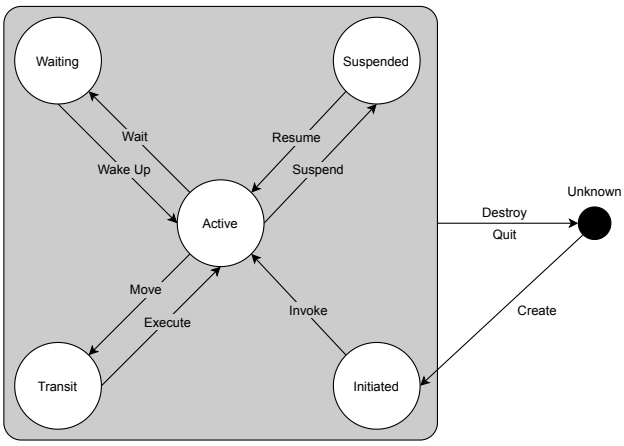


FIGURE 4.2 – Agent Life Cycle [FIPA, 2004]

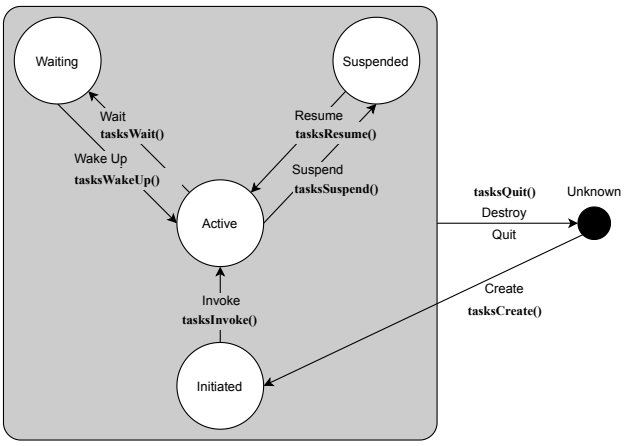


FIGURE 4.3 – Agent Life Cycle adapté pour les IoT-a

les applications pratiques sont beaucoup plus restreintes par les matériels. Le déplacement à la volée de nos agents, d'un matériel à un autre, demande un travail qui sort donc du cadre de cette thèse.

Mis à part les définitions des transitions développées dans le document [FIPA, 2004], FIPA ne donne pas de précision concernant le principe d'ALC dans les agents. Notons d'ailleurs que la fondation spécifie l'ALC comme une solution pour la gestion des messages, mais ne précise pas davantage l'inclusion de la gestion des comportements pour les agents. Aussi, nous précisons les cinq étapes -*Unknow*, *Initiated*, *Active*, *Waiting* et *Suspend*- de la figure 4.3 par nos notions de comportements avec les éléments suivants. Nous les complétons par les recommandations en matière de gestion des messages formalisées par FIPA, représentés ici en italique :

Unknow : Dans le cas où l'étape est suivie par la création d'un agent, l'élément créateur vérifie si les conditions de création sont suffisantes pour la vie du futur agent. Dans le cas d'une destruction ou d'un arrêt, l'agent est enregistré comme étant inactif auprès de la plate-forme. *Les messages sont enregistrés par un tiers ou rejetés.*

Initiated : Tous les paramètres des comportements sont initialisés et les comportements sont créés. *Les messages sont enregistrés en mémoire tampon dans l'agent puis traités au retour à l'état actif.*

Active : Les comportements exécutent les tâches suivant les paramètres d'initialisations. *Les messages sont traités normalement.*

Waiting : Tous les comportements sont mis en état d'attente. *Les messages sont enregistrés en mémoire tampon dans l'agent puis traités au retour à l'état actif.*

Suspend : Tous les comportements sont mis en état suspendu. *Les messages sont enregistrés en mémoire tampon dans l'agent puis traités au retour à l'état actif.*

Nous pouvons maintenant présenter plus précisément la notion de comportement et quels en sont leurs natures. Les travaux de [Renault and Carlier, 2016] exposent leur point de vue à ce sujet au travers du concept d'IoT-a. Les agents évoluent au moyen de différents types de comportements primaires. Certains travaux comme [Bellifemine et al., 2001] évoquent la notion de comportement dans les systèmes multi-agents. Cette formalisation se tourne vers une mise en application, dans le cadre du développement de la plate-forme SMA JADE. Ces travaux définissent un comportement comme étant une classe abstraite apportant un squelette de tâches à exécuter par les agents. Les comportements des agents d'un IoT-a reprennent cette philosophie.

Lorsqu'un agent souhaite réaliser des tâches, il doit nécessairement disposer d'au moins un comportement. Chacun d'entre eux peut être responsable d'une ou plusieurs tâches. Nous proposons quatre types de comportements "primaires" que nous détaillons ci-après. Ils correspondent aux types de réalisation à effectuer par l'agent. Les tâches sont donc encapsulées dans un comportement et s'effectuent selon son type et son paramétrage. Plusieurs comportements peuvent coexister et s'exécuter en parallèle en fonction des capacités techniques de l'objet connecté support. Pour chaque type de comportement, un agent a la possibilité de réaliser des tâches avant et après son exécution par les fonctions `tasksBegin()` et `tasksEnd()`. Cette particularité permet aux agents de lancer des procédures d'initialisations nécessaires à l'exécution du comportement et de finaliser les tâches qui le nécessitent avant l'arrêt du comportement. Nous noterons, de plus, dans le figure 4.3, notre proposition d'associer une fonction à chaque transition d'étape de l'ALC. Elles permettent l'exécution de tâches en amont de la prochaine étape.

Nous pouvons citer quatre types de comportements primaires.

One Shot est un comportement ne s'exécutant qu'une seule fois. L'agent se crée, effectue les tâches comprises dans le comportement puis se termine. Les tâches sont d'une durée indéterminée, mais ne s'exécutent qu'une seule fois.

Periodic permet d'exécuter des tâches de façon périodique. Avant la réalisation du comportement, l'agent spécifie l'intervalle de temps entre chaque itération. La fin du comportement périodique est engendrée par un seul des stimuli extérieurs ou un changement dans l'ALC.

Event est un comportement réalisé à partir de l'apparition d'un événement, il est apparenté au fonctionnement d'une interruption. Toutes les tâches de ce comportement s'effectuent une fois que l'événement paramétré a lieu. Ce comportement prend fin lorsque toutes les tâches sont réalisées, ou pour les mêmes raisons qu'un comportement périodique si l'événement déclencheur ne survient jamais.

TimeOut produit l'exécution de tâches à l'issue de la péremption d'une condition. L'exécution n'intervient que si le comportement n'est pas validé par une condition paramétrée par l'agent. Ce type de comportement permet de réaliser des tâches résultantes de la détection d'un problème causé par une trop longue attente. Si la condition est remplie alors aucune tâche n'est exécutée et le comportement se termine.

S'ajoute à ces comportements primaires la possibilité de les composer. La composition de plusieurs comportements primaires dans un seul élément fait naître des comportements complexes. Nous proposons un exemple de comportement complexe : **Finite State Machine**, permettant la réalisation d'une machine à états finis. La réalisation de ce comportement est issue de la combinaison de plusieurs comportements **One Shot** séquencés par des conditions. Ce type de comportement nécessite le paramétrage de l'ensemble des comportements **On Shot** à mettre en œuvre et de chaque condition associée permettant la validation de l'exécution. Toutes les formalisations théoriques propres aux machines à états finis sont accessibles au travers du paramétrage initial. Le comportement **Finite State Machine** se termine une fois le dernier comportement terminé. Il est possible que l'agent puisse disposer d'un graphe cyclique de tâches, dans ce cas, seul des stimuli extérieurs ou un changement dans l'ALC permettra d'arrêter l'agent.

L'utilisation de ces types permet une meilleure formalisation des comportements que doivent adopter les IoT-a. Elle donne un cadre pratique aux développeurs tout en se rapprochant de certaines formalisations du domaine des systèmes multi-agents [Poslad, 2007]. Cependant, nous notons l'absence des concepts de proactivité et réactivité dans ce formalisme. Il nous semble pertinent de proposer deux catégories de comportements pour cloisonner les activités de chaque agent et ainsi rajouter certains éléments de formalisme au concept d'IoT-a.

Nous avançons l'idée d'une première catégorie responsable uniquement des comportements proactifs de l'agent. En ce sens que, l'agent a la possibilité d'analyser son environnement par le biais d'appels systèmes ou de mesures physiques des capteurs de l'objet. Il est alors capable de prendre ses propres décisions. La catégorie proactive instancie tous les types de comportements uniquement restreints par leurs activités en lien avec la notion de proactivité. Nous proposons une seconde catégorie responsable des comportements réactifs. C'est-à-dire tout comportement en réponse à une demande extérieure de la part d'un agent ou d'une entité. Les comportements sont de tout type et doivent se limiter à la réaction à des stimuli extérieurs.

4.1.3 | *Micro services*

L'architecture SOA précise le découpage d'une application monolithe en différents services répartis dans un ensemble de matériels [Ribeiro et al., 2008] [Nazir Raja et al., 2008]. L'évolution de cette architecture depuis quelques années a donné naissance au concept de micro services, développé dans de nombreux domaines de la recherche et de l'industrie [Vural et al., 2017]. Initialement employé dans le domaine du web, où il prend d'ailleurs le nom de web-services dans certains cas, le paradigme des micro services est aujourd'hui introduit dans le secteur des objets connectés [Krivic et al., 2017] [Butzin et al., 2016]. Dans ces travaux, les objets répondent à des services au travers de protocoles définis par le domaine des micro services. Leurs but est de décentraliser les applications. Cette approche favorise l'agilité et le déploiement, une baisse de la consommation par le développement d'adaptateurs de charges (load balancer) [Dragoni et al., 2018] et une meilleure maintenance par l'uniformisation des traces entre les objets. Comme nous le montrent les travaux [Collier et al., 2019], les similitudes entre un agent et un micro service sont grandes. Il apparaît que la majeure différence entre ces deux approches est la capacité d'un agent à être proactif vis-à-vis de son environnement. Il doit être capable de prendre des "initiatives", chose qu'un micro service n'est pas en mesure de faire d'après les définitions actuelles. Aussi,

certains travaux [Greenwood et al., 2007] [Collier et al., 2019] avancent l'idée de passerelles entre des agents et des micro services afin de permettre aux agents de bénéficier des services et aux services de bénéficier des agents. Avec cette approche, nous disposons alors de deux univers qui cohabitent par une interface commune leur permettant des échanges. Les agents évoluent de leur côté jusqu'à éprouver le besoin d'un ou plusieurs services qu'ils appellent par le biais de cette interface.

L'IoT-a pourrait alors bénéficier des concepts de ce nouveau paradigme sans pour autant changer sa ligne directrice ayant pour but la mise en place d'agents dans les objets connectés pour réaliser des applications décentralisées. De ce fait, nous proposons une approche légèrement différente des travaux de [Greenwood et al., 2007] et [Collier et al., 2019]. Étant donnée la ressemblance étroite entre micro services et agents, pourquoi ne pas associer les caractéristiques de ces éléments ? Un agent est proactif, mais il a aussi la possibilité d'être réactif, il est capable d'effectuer une action en réponse à un élément extérieur. Si nous oublions un instant la capacité proactive d'un agent, toutes les autres caractéristiques entre lui et un micro service sont compatibles et nous pourrions utiliser l'un comme l'autre pour répondre à une application. Selon le point de vue adopté, seuls les protocoles de communication et d'interactions entre les éléments seraient susceptibles de marquer une différence.

Nous pourrions alors prendre deux chemins différents pour associer la capacité de proactivité à cet élément purement réactif. La première serait de proposer une solution pour instiller la notion de proactivité au sein du paradigme des micro services. Au même titre que les agents, les micro services seraient capables de prendre des décisions par eux mêmes pour générer des résultats de façon autonome. Néanmoins, cette approche semble remettre en cause de nombreux aspects dans la conception de micro services déjà existants. Il semble difficile de faire évoluer rétroactivement d'anciens micro services préalablement développés pour les doter de ce nouvel aspect.

C'est pour cela que nous proposons une seconde approche, ayant pour objectif d'inclure la notion de micro services au sein des agents. Les comportements proactifs d'un agent subsistent pour initier des réactions issues du libre arbitre de l'élément, mais l'aspect réactif répond aux recommandations établies dans le paradigme des micro services. Nous obtenons ainsi une enveloppe principale régie par la définition d'un agent avec ces comportements élémentaires, à laquelle sont introduits des comportements réactifs d'un micro service. Nous pensons que cette vision d'association des paradigmes permet une évolution des deux domaines par la possibilité de faire évoluer en même temps, des agents simples, sans micro service, des agents avec micro services associés et des micro services simples. Cette vision ne restreint pas l'idée de passerelles entre les deux domaines car elle permet le développement d'agents responsables de leur interface. Toutefois, elle ne cloisonne pas non plus les domaines dans leurs ensembles respectifs. Une vision réunifiée de ces deux paradigmes permet de générer de nouvelles notions, notamment le développement de la théorie de proactivité associée à des micro services. Néanmoins, cela doit se faire au détriment d'un effort d'harmonisation et de simplification des communications pour améliorer les performances issues d'interactions entre les éléments.

Dans le cadre de l'IoT-a, un micro service est donc une instance d'un comportement et bénéficie des mêmes caractéristiques présentées dans la section précédente. Il représente un comportement supplémentaire qu'un IoT-a peut instancier. Un agent présent dans un IoT-a dispose de plusieurs comportements, certains inscrits dans la catégorie proactive, et d'autres, réactifs, dont une partie à la possibilité d'être des micro services. L'intégration des micro services à notre modèle IoT-a est notamment détaillée dans la figure 4.5.

4.1.4 | *Communication*

La communication est au centre de l'idée d'IoT-a et nous y reviendrons à plusieurs reprises durant ce chapitre. Un agent seul n'a pas d'intérêt réel, ce n'est qu'une fois mis en relation avec d'autres qu'il prend tout son sens. Le domaine des objets connectés répond aux mêmes principes et il est nécessaire de faire correspondre les attentes de ces deux domaines pour en faire émerger des comportements intelligents entre nos IoT-a.

Dans un IoT-a, les agents ont besoin de communiquer avec d'autres éléments que des agents, notamment des microcontrôleurs ou microprocesseurs. Dans ce cas, l'utilisation des normalisations de la FIPA n'est pas entièrement compatible. Les composants disposent de protocoles de communication normés mais nécessitent une structure particulière pour échanger des informations. Par exemple : un capteur météo pouvant mesurer la température et la pression atmosphérique, attend une séquence de messages particulière pour diffuser ses informations. L'émission d'une trame de communication permet d'obtenir la température et une autre la pression. Les paramètres associés à ces trames peuvent modifier les unités de mesure ou encore étalonner les capteurs internes. Enfin, tous ces éléments diffèrent entre les fabricants et les produits.

Au vu de la pluralité de méthodes de communication entre ces matériels, nous proposons la mise en place d'une ontologie rassemblant leurs éléments de langage. Ces éléments sont définis par les fabricants et doivent être retranscrits algorithmiquement dans l'agent. Chaque famille de composants dispose d'une ontologie spécifique, il est donc nécessaire de les organiser pour simplifier les interactions entre un agent et des matériels. Nous proposons de les distinguer en deux catégories : une première propre aux interactions entre agents, aujourd'hui formalisée par la FIPA que nous nommons *ontologies SMA*, et une seconde permettant les échanges d'informations bas-niveaux avec des éléments matériels que nous nommons *ontologies matérielles*. Un agent a recours à l'une ou l'autre catégorie pour interagir, en fonction de ses besoins, avec un agent ou un composant.

Les agents d'un IoT-a évoluent dans un environnement réel, de surcroît sur des systèmes embarqués. Cette particularité, contraint la capacité de mémorisation des données, de calcul, mais aussi de communication. Un objet connecté dispose de bus de communication qu'un agent doit être capable d'utiliser pour communiquer avec d'autres agents, mais aussi avec des éléments électroniques tels que des capteurs ou des actionneurs. Pour répondre à cette particularité, nous proposons qu'un agent bénéficie de canaux de communication. Un canal de communication est un vecteur de messages entre des agents ou des matériels et nous nous appuyons sur les formalisations de la FIPA concernant les *Message Transport Protocol* (MTP) et *Message Transport Service* (MTS) [FIPA, 2002c]. Un MTP est défini comme étant un canal capable de transporter des messages, alors qu'un MTS est l'ensemble des services permettant l'acheminement des ACL-message au travers d'un MTP. Le concept de MTP énoncé par la FIPA précise que les agents communiquent au travers d'un réseau IP en annonçant l'URL d'émission et de réception pour chaque message. Dans le cadre des communications IoT-a, il n'est pas garanti qu'un agent puisse bénéficier d'un réseau IP. De plus, un agent présent dans un IoT-a doit avoir la possibilité d'échanger des informations à un niveau matériel le plus bas possible. C'est pourquoi un canal de communication prend en compte à la fois les protocoles matériels, au travers de l'utilisation de bus de communication matériel (I2C, Ethernet, etc.), et les protocoles fonctionnels portés par ces bus de communication (MQTT, HTTP, AMQP, etc.)

En résumé, la communication d'un agent est séparée en deux phases. La première consiste en l'exploitation d'une ontologie permettant de communiquer avec l'élément distant (agent ou composant). L'agent prend soin de constituer un message compréhensible pour celui qui en est destinataire. La seconde phase repose sur le choix du protocole adéquate à la communication. Ce découpage, permet aux agents d'échanger avec des éléments matériels, mais aussi avec d'autres agents présents sur le même IoT-a ou distants.

4.1.5 | *Réflexion sur la configuration d'IoT-a*

Nous proposons dans cette section une réflexion permettant l'intégration d'agents au sein des objets connectés. En l'occurrence, nous détaillons la notion d'agent matériel ou logiciel et nous nous intéressons aux questions de la répartition physique des agents dans un système.

Un IoT-a se veut reconfigurable et compatible avec différents types de matériels, du microcontrôleur 8-bits à l'ordinateur personnel, en passant par les SBC et autres systèmes embarqués du marché. Aussi, nous dressons dans cette section, un ensemble de configurations permettant la mise en place d'agents dans les différents types d'IoT-a en fonction de leurs caractéristiques technique et technologique.

D'un point de vue méthodologique, nous constatons que très peu de méthodes existent actuellement pour déterminer la localisation d'agents dans un système embarqué. Contrairement aux méthodes de co-design [Reza Naji et al., 2004] et [Jamont and Occello, 2007], ayant pour but d'adapter le matériel aux attentes des systèmes multi-agents, notre approche propose des solutions pour "agentifier" des objets déjà existants ou développés. Pour cela, nous adoptons une méthode de classification des systèmes embarqués comme le propose [Van Moergestel et al., 2016]. Cette méthode, bien que subjective, se base sur l'étude des nombreuses technologies d'objets connectés que nous avons présentées dans cette thèse, notre expérience issue de rencontres dans l'industrie et la recherche et enfin, les nombreux travaux réalisés dans le domaine des systèmes embarqués issus de la littérature. Cette classification s'articule selon trois critères : les caractéristiques techniques des objets connectés, le niveau matériel des applications auxquelles se dédie l'objet et leurs caractères temps réels. Cette classification permet de faire émerger des catégories et adapter les capacités des agents présents dans les systèmes.

Nous spécifions une configuration comme étant un ensemble de préconisations pour l'intégration d'agents dans un objet connecté. Chacune d'elle se veut indépendante et un IoT-a a la possibilité de cumuler plusieurs configurations. Étant donné qu'une configuration intègre au moins un agent, leur interopérabilité est assurée

par la correspondance de canaux de communication implémentés entre les agents de chacune d'entre elles. Un IoT-a disposant de plusieurs configurations interopérables est dans la possibilité d'échanger des informations entre différents composants matériels et logiciels.

Aussi, nous proposons dans la figure 4.4 quatre configurations pour la mise en place d'agents au sein d'un objet de l'IoT. Plus un objet est complexe, plus il peut en intégrer. À l'inverse, plus un objet est dédié à une tâche précise plus le nombre de configurations possibles est restreint.

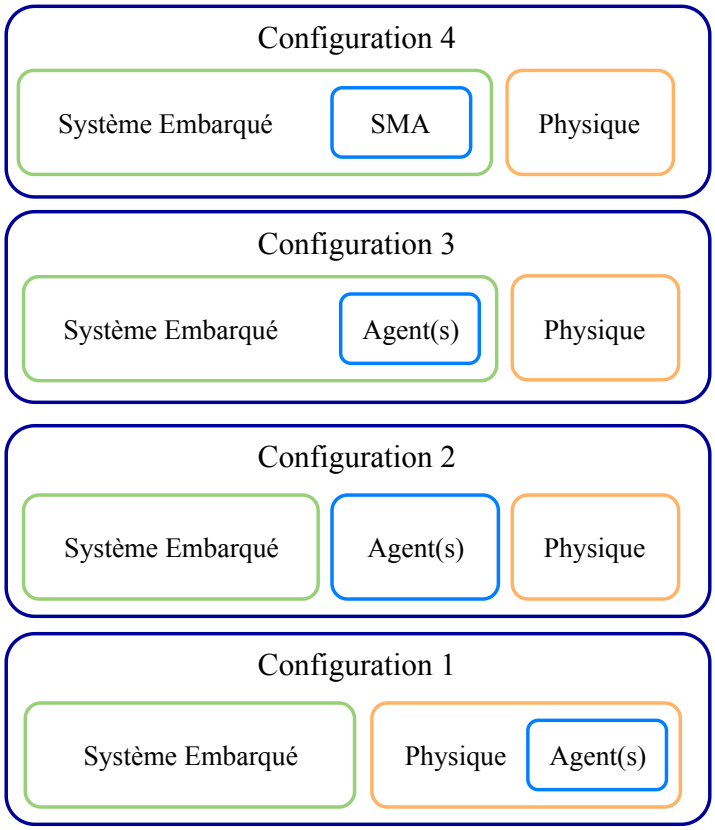


FIGURE 4.4 – Préconisations de configurations pour la constitution d'IoT-a

Configuration 1 : Elle propose l'intégration d'agents à un niveau physique. Un objet connecté dispose d'une architecture matérielle divisée en trois parties principales [Nurmi, 2007] : le processeur, la mémoire et les périphériques. L'ensemble est interconnecté au moyen d'un bus de données et d'un bus d'adresses. Un ou plusieurs périphériques agents peuvent être ajoutés au système et connectés sur les bus de données et d'adresses. Nous avons exposé dans notre état de l'art la méthode des NoC 2.4.1. Un agent, au sens où nous l'avons défini, est une entité capable de s'inscrire dans la démarche des NoC avec un composant capable de comportements proactifs et réactifs, communiquant avec d'autres composants du système. Le logiciel du système embarqué bénéficie des comportements des périphériques agents situés dans la partie matérielle. Les agents disposent des canaux de communication leur permettant de s'intégrer avec le système existant et d'autres canaux permettant de communiquer avec des agents situés à des configurations supérieures. Ils peuvent alors communiquer avec des agents répartis sur d'autres niveaux matériels. Cette proximité avec le matériel offre aux agents la possibilité de répondre à des applications très bas niveau, nécessitant des comportements temps réels durs avec des latences faibles et des débits importants.

Cette première configuration est applicable à des objets connectés pouvant reposer sur une architecture matérielle de type microcontrôleurs ou ASIC (Microchip PIC, Atmel Atmega, Espressif Systems ESP32, Texas Instrument MSP430 & C2000, ARM Cortex-M, etc.). Le composant intègre un agent matériel dans le silicium et devient une fonction supplémentaire et autonome. L'intégration d'un tel agent est aussi envisageable dans un composant programmable de type FPGA, car l'agent peut être directement programmé dans le composant [Reza Naji et al., 2004].

Configuration 2 : À partir de cette configuration, il est admis que l'objet connecté dispose de ressources matérielles suffisantes pour héberger un système d'exploitation. Dans cette configuration, un agent est présent en complément du système logiciel global (système d'exploitation). Pour cela nous avons recensé principalement deux technologies permettant ce type de configuration. Premièrement, le *Trusted Execution Environement* [Sabt et al., 2015] ou plus communément appelé *Trusted Zone* dans les processeurs ARM, est une partie matérielle dédiée à l'exécution de solutions de sécurité. Cette zone est inaccessible sans une procédure de sécurité depuis le reste du système et permet d'analyser certains comportements du système ou de réaliser certaines tâches sécurisées (manipulation de clés privées). Deuxièmement, un co-kernel [Barbalace et al., 2008] est un noyau léger ne s'occupant que de certaines parties bas-niveau spécifiques. Il laisse le noyau original se charger de tous les éléments habituels, mais passe prioritaire lorsque certaines actions lui sont réservées. Cette seconde configuration peut être mise en place sur des objets reposant sur des architectures de type ARM, PowerPC, RISC-V ou x86 (Broadcom, Motorola, Intel, etc.). Dans ces deux cas de figure, les agents sont de type logiciel, car ils ne font plus partie des composants matériels du système, mais sont des implémentations logicielles. Ils sont néanmoins, très proches du matériel, et ne sont pas ou peu impactés par les latences du système tout en offrant la possibilité de communiquer plus facilement avec le reste du système par des mécanismes de communication logiciels (pipe, socket, fichiers, etc.). L'exécution des agents peut être temps réel et/ou sécurisée.

Configuration 3 : Cette configuration avance l'idée d'intégrer un ou plusieurs agents au sein du système d'exploitation. L'intérêt d'un système d'exploitation est la parallélisation de l'exécution de tâches. C'est de cette propriété que la configuration 3 bénéficie par rapport à la configuration précédente. Il existe néanmoins une grande diversité de types de systèmes d'exploitation pour les systèmes embarqués, spécialisés dans les systèmes télécoms, de traitement de signal audio ou vidéo, les systèmes temps réels, réseau de type router, multimédia, mobile, etc. Prendre en compte toutes ses plate-formes est complexe, mais nous pouvons cependant prendre l'exemple de systèmes GNU/Linux particulièrement présent dans ce domaine. Ces systèmes présentent deux types d'environnement.

Un premier consacré à la gestion du matériel, des interactions avec l'environnement extérieur et l'ordonnancement des tâches à exécuter. On parle de *noyau* ou plus communément dans la littérature de *kernel*. Cette zone est limitée à certains types d'exécutions parallèles appelées *kernel threads*. L'ajout de code dans le noyau nécessite soit une re compilation prenant en compte les développements soit la mise au point de *kernel modules* pour une intégration à chaud de ces applications dans le noyau. La mise en place d'agents dans cet environnement peut faire intervenir des instabilités, mais permet d'accéder à tous les éléments logiciels ou matériels bas niveau du système. Un agent dans le noyau est néanmoins contraint par l'utilisation des mécanismes proposés par le noyau pour s'exécuter et communiquer avec d'autres agents.

Le second type d'environnement est appelé *espace utilisateur* ou *user space* dans la littérature. C'est dans cette zone que s'exécutent toutes les applications et qu'il est possible de bénéficier d'un système de fichiers. Dans cet environnement, les exécutions sont surveillées et ordonnancées par le noyau. Cette gestion confère une grande stabilité au système tout en lui permettant d'exécuter de nombreuses tâches en parallèle. Un objet connecté bénéficiant d'un système d'exploitation avec un espace utilisateur a la possibilité d'exécuter un grand nombre d'agents en parallèle avec de nombreux canaux de communication différents. Les agents peuvent être créés à la volée en fonction des besoins et limités en nombre par les capacités du système.

Il est possible de mettre en place cette configuration sur tout type d'architectures compatibles avec un système d'exploitation notamment les SBC de type : Raspberry Pi, Freescale Sabre SD, Intel UP, Rock64, BeagleBoard black, etc. Dans cette configuration, l'agent est compilé pour être compatible avec le système d'exploitation et subit les ralentissements induits par d'autres processus s'il est dédié à l'espace utilisateur et évolue avec d'autres exécutions dans l'espace noyau.

Configuration 4 : Les trois configurations précédentes n'évoquent pas le problème d'organisation entre les agents. Cette configuration propose que l'objet connecté héberge une plate-forme multi-agents. Les agents présents dans l'objet peuvent interagir de façon autonome et peuvent être en grand nombre. La plate-forme SMA porte des agents d'organisation, dits agents de plate-forme, et les canaux de communication nécessaires pour l'interaction des agents présents dans les configurations inférieures au sein d'un même objet, mais aussi avec d'autres IoT-a. Un IoT-a en configuration 4 n'est pas centralisateur, plusieurs IoT-a d'un même réseau ont la possibilité de se trouver en configuration 4. Une plate-forme sert de relais aux agents ou aux autres plate-formes SMA reparties sur d'autres types de configuration matérielle ou sur d'autres objets connectés. Cette configuration nécessite un objet avec des ressources matérielles suffisantes pour l'exécution de plusieurs agents et la prise en charge de différents protocoles de communication à différents niveaux matériels. La configuration 4 nécessite les mêmes contraintes d'intégration que la configuration 3, mais demande un dimensionnement des équipements en lien avec la quantité d'agents à administrer.

4.1.6 | *Modèle de l'IoT-a*

Nous avons défini dans les sections précédentes les différents aspects proposés pour le concept d'IoT-a. En outre, quatre configurations ont été proposées et ont pour but de guider les développeurs dans une démarche de mise en place d'agents et système multi-agents dans du matériel compatible. L'IoT-a se veut suffisamment ouvert pour correspondre aux grands nombres d'aspects qu'implique le domaine des objets connectés, tout en introduisant le paradigme multi-agents pour rendre les objets plus intelligents.

Aussi, nous dressons le modèle de l'IoT-a dans la figure 4.5. Nous nous appuyons, pour cela, sur les travaux de Carlier et Renault que nous complétons avec les éléments précédemment détaillés. Un IoT-a est un objet, ce que nous retrouvons au travers de l'encadré intitulé : Objet connecté. Cet objet dispose d'éléments matériels tels que des capteurs et des actionneurs, lui permettant d'interagir avec l'environnement physique. Il est aussi équipé, d'un ou plusieurs périphériques matériels de communication. Certains ne permettent de communiquer que vers des éléments internes alors que d'autres ouvrent les communications vers d'autres IoT-a. Le système embarqué permet le traitement et l'enregistrement des données tout en reliant l'ensemble des éléments.

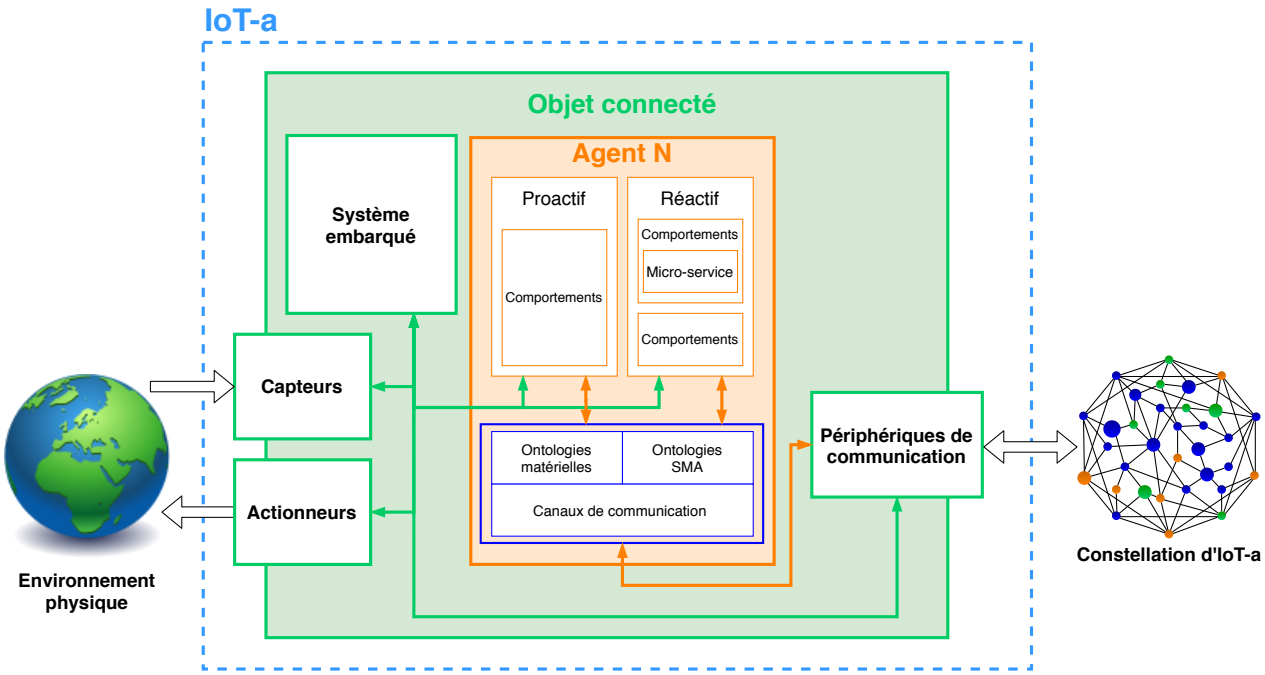


FIGURE 4.5 – Modèle d'un IoT-a

Mais un IoT-a est avant tout un objet connecté composé d'agents, c'est donc dans le cadre noté *Agent N* que nous retrouvons les formalisations agents. Nous noterons qu'un IoT-a peut être constitué d'un ou plusieurs agents. Ainsi, le paramètre N symbolise la multiplication possible de ce modèle d'agent dans l'objet. Deux catégories de comportements sont définies au sein d'un agent, l'une pour les comportements proactifs et l'autre pour les comportements réactifs. La catégorie réactive dispose de certains comportements implémentant un micro service en parallèle d'autres comportements. Chacun d'eux peut interagir directement avec le système embarqué. Enfin, un agent doit avoir la possibilité de communiquer avec d'autres agents et des matériels, c'est pourquoi nous retrouvons les éléments présents dans l'encadré bleu. Une première couche est responsable des ontologies permettant des interactions au sein d'un SMA et entre des matériels. Une seconde définit les canaux de communication. Ils sont en lien avec un périphérique physique de communication de l'objet connecté.

Un IoT-a est donc un élément autonome capable de répondre localement à des applications. Plus ses capacités techniques sont importantes, plus il est capable d'héberger un grand nombre d'agents et les organiser autour de lui. La pluralité des canaux de communication dans les agents, offre à l'objet de nombreuses voies de communication pour échanger à tous les niveaux matériels et réseau. Ainsi, des agents bas niveau, ayant la possibilité de communiquer avec des capteurs, des mémoires, des actionneurs, et des agents de haut niveau, responsables d'interfaces homme-machine ou de bases de données, ont les clés pour interagir entre eux et en faire émerger des comportements intelligents. Nous devons néanmoins, nous concentrer sur l'organisation de ces canaux pour envisager de faire interagir tous les agents au sein d'un grand nombre d'IoT-a hétérogènes.

4.2 | Constellations d'IoT-a

Nous avons abordé dans la section précédente les éléments qui constituent les IoT-a et avons dressé le modèle d'un IoT-a. Cette section se concentre sur l'interaction entre plusieurs IoT-a. En fonction de leur conception, les objets ont la capacité d'intégrer un, plusieurs agents ou un système multi-agents complet. Ces agents, présents à plusieurs niveaux matériels et répartis dans différents objets, n'interagissent pas tous de la même façon. Cette section présente les réflexions en matière de communication entre agents selon qu'elles soient internes ou externes à une constellation.

Qu'est-ce qu'une constellation d'IoT-a ? Nous avons évoqué dans la définition 1 le terme de constellation et l'avons défini comme un ensemble d'IoT-a répartis dans différents espaces. En l'occurrence, l'espace dans lequel évolue un agent est de nature différente selon les caractéristiques de l'objet connecté. Nous avons pour cela, proposé des configurations permettant d'orienter les emplacements physiques des agents dans les objets. Cette répartition offre l'opportunité d'une gestion d'applications de façon décentralisée, chaque agent ayant pour rôle de répondre à une partie de l'application globale. Certains agents répondent à des fonctions alors que d'autres émettent le résultat de calculs ou de mesures. La composition de plusieurs IoT-a, intégrant une ou plusieurs de nos configurations dans un même environnement forme une population d'IoT-a. Certains sont de petits systèmes intégrant quelques agents, alors que d'autres prennent en charge un plus grand nombre d'agents et disposent de plusieurs canaux de communication. Nous ne contraignons pas les bornes de cet environnement, car il est à l'appréciation de l'utilisateur qui choisit le nombre d'IoT-a et la taille de son système adéquat à ses applications. Notons simplement qu'une fois les bornes déterminées, cet ensemble doit pouvoir évoluer en autonomie et accepter l'arrivée ou le départ d' IoT-a. Dans ce cas nous qualifions cet ensemble de constellation d'IoT-a. Nous définissons donc une constellation de la façon suivante :

Définition 10 : Une constellation d'IoT-a - bis

Une constellation d'IoT-a est un regroupement autonome de plusieurs IoT-a réparties dans différents espaces matériels ou logiciels, capables de répondre à une fonction locale. Une constellation est dynamique et évolutive, elle doit pouvoir se développer dans le temps, tant par ses comportements que par sa communication.

En guise d'exemple du concept de constellation d'IoT-a, nous proposons d'évoquer le thème de la coupe de France et d'Europe de robotique : Eurobot [Vincenzo et al., 2006]. Chaque année se déroule un concours de robotique où des équipes de robots s'affrontent sur un terrain. Au travers de plusieurs matchs s'effectuant sur une table, deux équipes disposent chacune de deux robots autonomes s'opposant pendant un temps limité. Les robots tentent de remporter un maximum de points au travers de la réalisation d'actions (manipulation de blocs, déplacement d'éléments, etc.). Ils n'ont pas le droit de se toucher et ne peuvent avoir recours à aucune intervention humaine pour réaliser un maximum d'actions sur la table. Chaque robot dispose de son propre système mécanique de déplacement, le plus souvent par roues, et se déplace librement sur la table et les zones lui rapportant des points.

Cet exemple est intéressant, car il permet d'illustrer entièrement notre approche IoT-a. En effet, un robot est constitué de plusieurs systèmes électroniques, chacun responsable de certaines fonctions. Nous nous limiterons ici à seulement quatre des comportements les plus communs :

Le déplacement : Ce comportement est le plus élémentaire, il permet au robot de se déplacer sur la table de jeu. Il nécessite une grande précision grâce à une gestion en temps réel et de grandes vitesses de communication. Cette partie repose principalement sur des microcontrôleurs pour le contrôle des trajectoires, de la vélocité des moteurs et les positions à atteindre. Certains composants bas niveau, comme ceux nécessaires à l'odométrie, peuvent disposer d'une configuration 1 s'ils sont matériellement conçus pour l'occasion, comme nous l'avons proposé dans la section précédente. La configuration 2 reste la plus adaptée pour le reste des systèmes électroniques. L'utilisateur peut donc mettre en place les agents nécessaires dans les microcontrôleurs pour la réalisation de cette fonction.

La réalisation d'actions : Ce comportement permet de remporter des points grâce à la récupération d'éléments sur la table de jeu. Un grand nombre d'actionneurs et de capteurs sont mis de concert pour manipuler, déplacer et soulever les éléments. Chaque tâche est découpée en fonctions simples, incluant de la mécanique et des systèmes électroniques dédiés (servomoteurs, capteurs complexes, moteurs pas-à-pas). Les fonctions sont autonomes et chacune dispose de son microcontrôleur. Notre configuration 2 est ici la

plus adaptée pour répondre aux multiples tâches. L'utilisateur peut développer un par un ses modules en y intégrant les agents nécessaires.

La détection de l'adversaire : Ce comportement détermine la position de l'adversaire durant le match. L'information générée est alors primordiale pour éviter les chocs avec l'adversaire. Plusieurs capteurs, parfois complexes (lidar), sont analysés en continu pour déterminer s'il est nécessaire d'adapter la trajectoire du robot. Étant donné la complexité de la tâche, sa réalisation doit être effectuée par un système électronique performant et met en jeu plusieurs agents afin de découper les réalisations en tâches. Nous pouvons avoir ici une configuration 3 ou 4 en fonction des performances du système.

La stratégie : Enfin, le comportement responsable de la stratégie est le chef d'orchestre du robot. Il doit avoir un rôle omniscient sur tous les autres comportements. La stratégie évolue selon la détection de l'adversaire, la position du robot et des actions effectuées. Plusieurs agents peuvent superviser séparément les autres comportements et prendre des décisions en conséquence. Il semble évident que de telles fonctions doivent reposer sur un système électronique suffisamment performant et permettant de jouer l'interface entre les différents éléments précédemment énumérés. Pour cela, nous recommandons la configuration 4.

Ces quatre comportements illustrent les mises en œuvre nécessaires pour faire évoluer un robot sur une table de jeu durant la compétition. Ils mettent en lumière les configurations proposées dans la section précédente. Aussi, plusieurs IoT-a aux multiples configurations sont présents dans ce robot. Ils forment une constellation locale où chacun des points est relié aux autres par différents liens de communication physiques ou logiciels. Dans notre cas de figure, le règlement stipule qu'une équipe ne doit bénéficier que de deux robots maximum. Étant donné qu'un robot intègre une constellation, nous comprenons l'intérêt de pouvoir échanger d'un robot à l'autre à différents niveaux matériels. Disposer de liens entre constellations est particulièrement intéressant pour une coopération entre les robots. Chaque robot peut intervenir et notifier à l'autre les actions qu'il a menées et de celles qu'il lui reste à effectuer. Les données de localisation de l'adversaire mesurées dans les deux robots peuvent être mutualisées pour augmenter la précision des mesures. Il arrive qu'un des deux robots soit bloqué involontairement par un robot adverse et ne puisse plus effectuer ses actions sur la table. Avec notre approche, les agents de l'un peuvent partager leurs connaissances avec les agents de l'autre afin de réaliser des actions qu'il n'est plus capable de réaliser. La stratégie s'en trouve modifiée, le programmeur n'est plus simplement responsable de la programmation de deux machines indépendantes, mais de deux outils connectés, capables de réagir et modifier leurs comportements en fonctions de l'autre. Cette particularité est d'autant plus importante dans d'autres événements robotiques similaires, ayant pour but de faire jouer un match de football à deux équipes de robots humanoïdes [Allali et al., 2017]. Chaque robot centralise localement ses fonctions internes créant ainsi une constellation. Ils partagent collectivement leur constellation pour faire émerger des comportements de groupe complexes, au plus proche de ceux rencontrés dans une équipe de football humaine.

En résumé, les IoT-a ont pour objectif de répondre de façon décentralisée à une application. Cette application est découpée en fonctions et chacune nécessite l'exécution de tâches. La granularité des tâches est alors un facteur déterminant pour assurer une décentralisation maximale. Certains objets ont les capacités de résoudre plusieurs tâches, d'autres plus dédiés, se spécialisent dans les tâches de plus bas niveau, en nombre restreint. Nous disposons donc d'un environnement d'IoT-a hétérogènes, où chacun est capable de peu, mais où tout le monde doit pouvoir échanger, interagir, négocier ou marchander.

Nous avons présenté dans la section précédente la notion de canal de communication et avons défini que chaque agent en dispose d'au moins un pour échanger avec d'autres agents ou des capteurs et des actionneurs. Nous nous sommes concentrés sur différentes configurations pour répartir les agents dans les objets et avons évoqué une compatibilité entre les canaux pour permettre des échanges entre les configurations. Néanmoins, certains IoT-a ne permettent pas d'implémenter plusieurs canaux de communication. C'est pourquoi nous proposons une architecture satisfaisant cette particularité.

Dans un premier temps, il est important de mettre en place un espace de communication commun, offrant aux agents les possibilités d'interagir, quelle que soit la technologie de communication qu'ils exploitent. Pour cela, nous proposons une première approche que nous notons : "la centralisation locale". Nous entendons par ce terme la représentation d'un ensemble d'agents reliés les uns aux autres par différents liens de communication. La diversité de ces liens et l'hétérogénéité des agents font que l'organisation d'une constellation est complexe. Ces bornes ne sont pas fixes et sont susceptibles d'évoluer en fonction de l'environnement ou le choix des utilisateurs. Afin de répondre à cette problématique, nous faisons appel aux mécanismes apportés par la littérature dans le domaine des systèmes multi-agents. Nous nous intéressons plus particulièrement aux agents de plate-forme dédiés à la gestion des services et de leur cycle de vie. Dans le cas de la simulation, tous les agents d'une plate-forme passent par le même canal pour échanger. Ils sont uniformes, en grand nombre et capables de se

comprendre. Les IoT-a n'ont pas tous ces opportunités. Ils ont recours à de nombreux types de bus différents pour communiquer. C'est pourquoi nous pensons qu'une plate-forme multi-agents doit servir, en plus de la gestion des services, de passerelle aux différents canaux de communication pour devenir une constellation. L'échange de données est centralisée au niveau des agents de la plate-forme pour ensuite être redistribuées aux IoT-a. Le but de la centralisation locale est donc de mettre à disposition des agents les éléments nécessaires à leurs interactions. Enfin, cette approche permet à une constellation d'être autonome. Le fonctionnement et les réflexions internes au groupe d'IoT-a ne nécessitent pas des liens vers des éléments extérieurs à la constellation. Toutes les informations circulent entre les agents et leur permettent de répondre localement aux fonctions pour lesquelles ils ont été mis en place.

Néanmoins, cette approche n'est pas suffisante pour permettre la décentralisation des systèmes. La taille d'une constellation n'est pas fixée, mais elle ne doit pas rendre l'organisation omnisciente sur l'ensemble des environnements, sous peine d'une centralisation globale. Lorsque nous reprenons notre exemple des robots de compétition, il est difficile de penser qu'un seul robot puisse orchestrer l'ensemble des agents de toute une équipe sans que cela représente un risque. Le robot centralisateur ne doit connaître aucun problème logiciel ou rupture de communication, sans quoi les autres robots ne pourraient plus effectuer aucune action. Plus généralement, une plate-forme SMA seule responsable de l'ensemble des communications introduit un risque non négligeable de pannes et fait émerger des goulots d'étranglement. Si le matériel portant les agents de plate-forme subit un dysfonctionnement, c'est l'ensemble des agents qui cesse de communiquer. Ce problème de centralisation est bien connu du domaine des télécommunications et des réseaux, c'est pour cela que la centralisation n'est que rarement mise en place. Au contraire, la décentralisation et la redondance sont les approches les plus utilisées dans ce domaine. Tous les éléments sont reliés entre eux par plusieurs liens distincts et si l'un d'entre eux vient à ne plus fonctionner, les autres prennent le relai.

Nous nous inspirons de cette approche pour la gestion des échanges de données entre nos constellations d'IoT-a. Notre hypothèse est qu'une plate-forme ne prenne en charge qu'un nombre réduit d'agents pour limiter les pertes d'informations lors d'un dysfonctionnement. Ainsi, tous les IoT-a ne sont pas administrés par la même plate-forme. La mise en place de plusieurs plate-formes SMA au sein d'une population d'IoT-a, favorise un découpage fin des fonctions à réaliser, une redondance et une meilleure stabilité face aux erreurs. Nous sommes alors face à plusieurs constellations autonomes, chacune organisée par une plate-forme SMA, ayant besoin de communiquer. Le défi est de les relier, pour que les IoT-a de l'une puissent échanger avec les autres. Dans notre exemple, un robot doit pouvoir communiquer avec ses partenaires pour augmenter ses chances de réussite durant un match. Il partage ses positions, ses actions et la vision qu'il a des adversaires. Nous proposons donc dans la suite de cette section, une approche reposant sur des passerelles entre les constellations. Cela permet aux IoT-a, quelle que soit leur nature technologique, d'avoir des échanges collectifs avec l'ensemble de la population. La décentralisation avec un grand nombre de constellations assure une continuité dans les communications même dans le cas du dysfonctionnement d'un équipement. Nous appelons cette approche : la décentralisation collective.

La figure 4.6 illustre des constellations dans lesquelles des agents interconnectés interagissent au travers de différents canaux de communication. Les constellations reposent chacune sur une plate-forme multi-agents et communiquent les unes avec les autres par l'intermédiaire d'une passerelle. Dans chacune d'entre elles, l'approche de centralisation locale offre aux éléments les interconnexions et l'organisation nécessaire à la réalisation des fonctions locales. L'approche de décentralisation collective, quant à elle, permet à tous les IoT-a d'échanger d'une constellation à une autre. Elle offre ainsi plus de connaissances aux IoT-a et donne lieu à de nouvelles collaborations plus complexes. Nous allons maintenant détailler ces approches et expliquer les mécanismes mis en place pour les satisfaire.

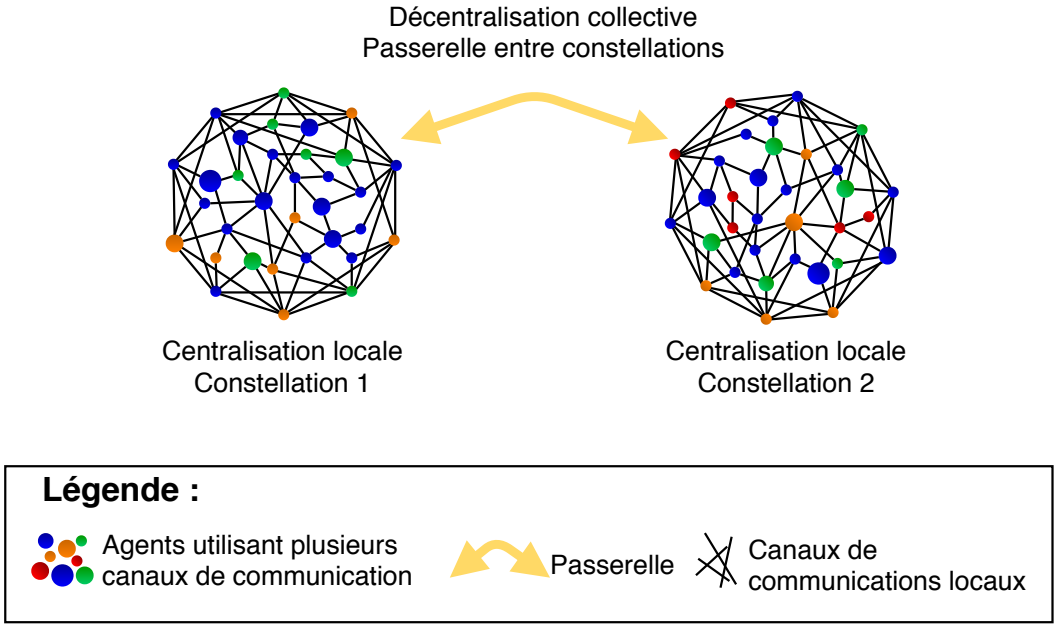


FIGURE 4.6 – Représentation du concept de centralisation locale et décentralisation collective

4.2.1 | *Communication intra-constellation : centralisation locale*

Cette section développe les mécanismes permettant les interactions entre agents au sein d'une constellation d'IoT-a. Une constellation intègre une pluralité d'IoT-a techniquement hétérogènes et offrant des services multiples. Elle est autonome face aux responsabilités qui lui sont fixées et permet aux agents de s'organiser et communiquer. Dans le cas de notre exemple de robotique, lorsque nous nous plaçons du point de vue d'un robot unique, tous les IoT-a interagissent pour faire évoluer le robot sur la table de jeu. Ceux responsables de la stratégie définissent, entre autres, les coordonnées cibles utiles au déplacement et la réalisation d'actions particulières. Sans une certaine organisation entre les IoT-a, aucun déplacement ou réalisation d'action ne peut avoir lieu. La problématique soulevée est donc de réussir à organiser et faire communiquer des IoT-a au travers de différents canaux. Notre hypothèse propose la mise en place d'un espace de communication commun, ou tous les agents auraient la possibilité d'échanger librement. L'ensemble étant organisé par les mécanismes communs de la littérature dans le domaine des systèmes multi-agents.

Cette approche ouvre donc deux volets différents.

Le premier tend à organiser les IoT-a entre eux pour qu'ils puissent interagir. Pour cela, nous nous appuyons sur les formalismes de la FIPA, normalisant les nombreux aspects qu'impliquent les communications entre agents dans un système. Nous nous intéressons notamment à l'agent de contrôle Agent Management Systeme (AMS) qui a la charge d'administrer les agents d'une plate-forme SMA. D'après la FIPA, chaque agent connaît l'adresse d'AMS à sa création. Pour rappel, vis-à-vis de ses administrés, AMS joue le rôle d'un annuaire des pages blanches, capable de renseigner les noms et adresses de chacun. Ainsi, un agent interrogera AMS pour disposer d'une liste des noms et adresses de ses congénères.

Le second volet est la communication et sur ce point, nous constatons que la fondation manque de formalisations offrant aux IoT-a la possibilité de communiquer. En effet, l'hétérogénéité des matériels sur lesquels évoluent les IoT-a ouvre de nouvelles questions sur ce sujet. Les agents de plate-forme communiquent tous sur le même bus et doivent, pour se comprendre, utiliser le même protocole. La FIPA propose, entre autres, l'utilisation du protocole HTTP [FIPA, 2002b]. Cependant, nous avons vu dans la section 2 de l'état de l'art que les protocoles de type *request/response* n'étaient pas des plus adaptés pour notre approche. Dans notre comparatif 2.1, nous avions présenté plusieurs protocoles de type *publish/subscribe* permettant de décentraliser la gestion des communications au sein des clients. De nombreux travaux justifient les qualités de ce type d'approche dans le domaine des objets connectés, c'est pourquoi nous nous y intéressons pour les interactions entre IoT-a. Aussi, nous définissons un *canal de communication META* afin d'évoquer un protocole d'échange de messages de type *publish/subscribe* entre nos objets-agents.

Ce canal META a pour objectif de centraliser les communications entre les agents d'une constellation. Tous les IoT-a doivent disposer des moyens pour communiquer directement ou indirectement par ce canal, sans quoi ils n'auraient pas les possibilité d'interagir avec leurs congénères. Aussi, un élément supplémentaire vient se rajouter à notre définition d'une constellation d'IoT-a. Étant donné que le type de transmission *publish/subscribe* nécessite un broker pour les échanges entre les clients, la présence d'au moins un broker au sein d'une constellation est indispensable. Cette centralisation, locale à une constellation, permet de créer un espace de communication commun pour l'ensemble des IoT-a. Espace dans lequel, le broker assure l'aiguillage des échanges. Nous pouvons désormais dresser notre définition de la centralisation locale, de part les éléments que nous venons de présenter :

Définition 11 : La centralisation locale

La centralisation locale assure l'organisation horizontale et autonome des agents d'une constellation d'IoT-a. Elle propose les outils nécessaires à l'inter-connexion des agents indépendamment de leurs technologies de communication. Un canal de communication META permet l'harmonisation des interactions entre différentes catégories d'agents.

L'utilisation par les IoT-a, du type *publish/subscribe*, ouvre un nouveau paradigme dans la façon d'identifier les agents. En effet, il est nécessaire d'organiser les règles d'abonnement et de publication entre chacun. Aussi, chaque agent disposant des capacités pour communiquer sur le canal META, s'abonne lors de sa création à minima au topic suivant : `ACC/nom_agent/#`. Cette règle lui permet d'être notifié de tous les messages concernant son nom et ainsi pouvoir déclencher les comportements adéquats. Bien entendu, un agent est dans la capacité d'écouter tous les autres topics nécessaires à son fonctionnement. Si certains éléments de l'agent le requiert, alors ils doivent s'organiser selon une arborescence commune. Par exemple : `ACC/nom_agent/nom_service/nom_fonction/nom_tâche`. Cette organisation permet de mettre en place facilement des agents de télémétrie ou de log en écoute sur un ou plusieurs niveaux de l'arborescence.

Reprenons maintenant, les éléments de la norme de la FIPA lors de la création d'un agent et voyons comment les communications doivent être adaptées pour correspondre à notre point de vue. Par exemple, la figure 4.7 illustre la création et l'enregistrement d'un agent *Alice* au sein d'une plate-forme SMA. Nous pouvons observer les échanges entre elle et AMS. Ce dernier vérifie la validité de l'AID d'*Alice* puis indexe ses informations sous la forme d'un tableau de champs du type : `nom_agent@adresse_agent`. Une fois l'indexation terminée, AMS renvoie un message à *Alice* pour lui confirmer son enregistrement comme agent de la plate-forme.

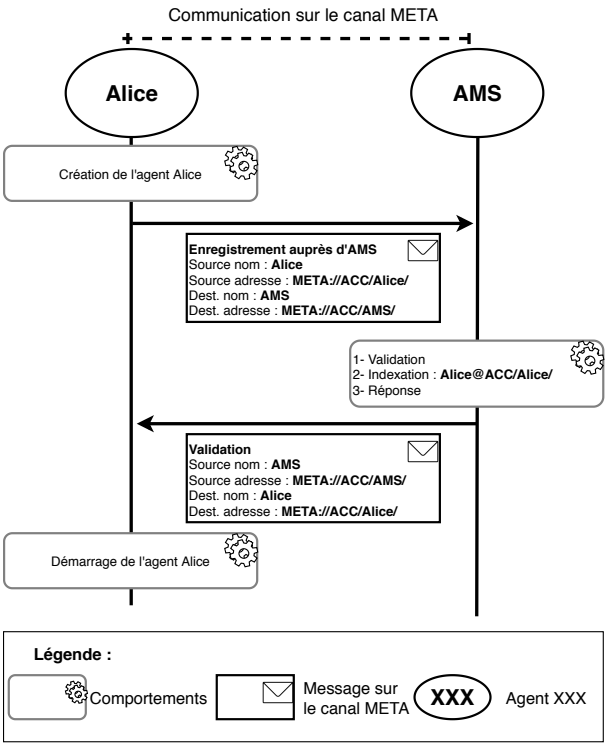


FIGURE 4.7 – Diagramme de temps pour l'inscription d'un agent Alice selon la norme de la FIPA

Alice et AMS utilisent ici le même bus de communication pour échanger et s'enregistrer. Dans le cas où un agent serait dans l'incapacité de communiquer sur le même canal qu'AMS, il n'aurait pas la possibilité de s'y enregistrer. Comment *Alice* ferait-elle pour échanger avec AMS et avec d'autres agents au travers de plusieurs bus de communication ou protocoles différents ?

4.2.2 | *Proposition de l'Agent Mediation Control*

Pour répondre à ces questions, nous proposons l'ajout d'un nouveau type d'agent de plate-forme : **Agent Mediation Control, AMC**, que nous illustrons au sein d'une constellation d'IoT-a, dans la figure 4.8. Cet agent est responsable de l'interface entre un ou plusieurs canaux de communication et le canal META. Plusieurs AMCs peuvent être mis en place dans une constellation pour relayer les messages entrants et sortants de différents types de canaux. Ils permettent alors, le transfère d'informations depuis et vers plusieurs canaux au sein d'une même plate-forme. Tous les AMCs sont créés après AMS et doivent s'enregistrer auprès de lui. Comme pour l'adresse de leur AMS, chaque agent intègre l'adresse de son AMC.

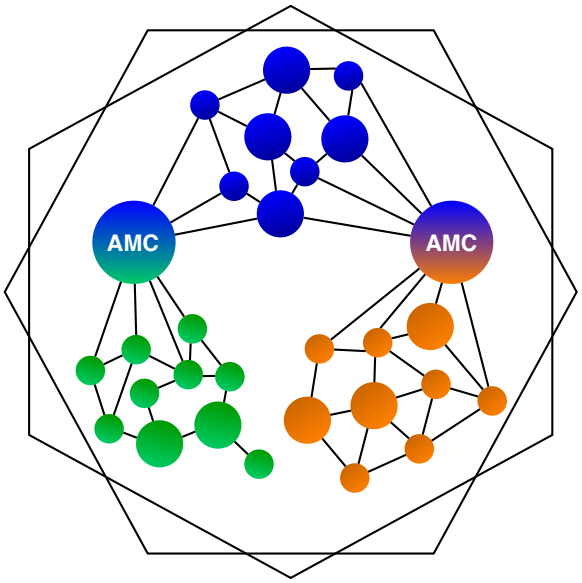


FIGURE 4.8 – Zoom sur une constellation avec l'intégration de plusieurs Agent Mediation Control

La figure 4.9 présente le modèle de l'Agent Mediation Control. Nous pouvons noter que cet agent est purement réactif et dispose de 5 comportements. Un AMC intègre deux types de canaux de communication. Un premier, appelé *canaux locaux*, permettent de communiquer avec les bus ou les protocoles propres à une population d'agents ne disposant pas d'un canal META. Le second type est par conséquent le *canal META* permettant de communiquer avec le reste des agents de la constellation. Un AMC n'a pas vocation à interagir avec d'autres éléments que des agents, c'est pourquoi il est uniquement doté d'une ontologie SMA pour l'interprétation des messages. Dans la suite de cette section, nous faisons le choix de faire correspondre un AMC à un seul canal local. Ce choix permet d'illustrer plus simplement notre point de vue et correspond mieux à une gestion décentralisée des communications. Ce faisant, le nom d'un AMC est alors contraint par le nom du canal qu'il interface. Un AMC d'un canal X aura pour nom : `AMC_X`, exemple pour le bus I2C : `AMC_I2C`.

Un AMC fonctionne selon les 5 comportements suivants :

Analyse : À la réception d'un message depuis l'un des deux canaux de communication dont dispose un AMC, l'agent de plate-forme analyse le type du message. Un AMC est fait pour recevoir des messages au format ACL et a la capacité d'en analyser le contenu. Dans le cas de l'enregistrement ou du désenregistrement d'un agent, l'AMC procède au comportement d'indexation ou de désindexation de l'agent concerné. Pour tous les autres types de messages, l'AMC n'a pas de comportement particulier.

Indexation/Désindexation : Suite à sa création, un agent procède à son enregistrement auprès d'AMS par l'envoi d'un message ACL de type *request* comprenant son nom et son adresse sur son canal local. Pour cela, il envoie ce message à son AMC spécifiant que le destinataire final est AMS. À la réception du message,

AMC interprète le type et procède à l'indexation du nom et de l'adresse du nouvel agent. L'indexation se fait sous la forme : `nom_agent@adresse_agent_canal_local`. Une fois l'agent indexé, AMC procède au comportement d'abonnement. Si toutefois l'enregistrement de l'agent n'est pas validé par AMS, le message reçu par AMC provoque la désindexation et le désabonnement de l'agent.

Lors de son arrêt, un agent doit se désenregistrer auprès d'AMS. De manière analogue à son enregistrement, AMC interprète ce message et procède à la désindexation de l'agent puis au comportement de désabonnement.

Abonnement/Désabonnement : Une fois l'agent indexé, AMC s'abonne au topic META correspondant aux règles d'abonnement qu'un agent de l'IoT-a doit avoir, en l'occurrence : `ACC/nom_agent/#`.

Si AMC détecte un désenregistrement d'agent, alors il se désabonne du topic correspondant à cet agent.

Modification source/destination : Pour chaque message, AMC procède à un changement de la source ou de la destination en fonction du sens de la communication. Si le message provient du canal local, AMC modifie l'adresse source pour correspondre à l'adresse META de l'agent source. Si au contraire, le message émane du canal META, AMC modifie l'adresse de destination pour correspondre à l'adresse de l'agent destinataire du canal local.

Transfert : Enfin, AMC procède au transfert du message. Si le message provient du canal local, AMC envoie le message modifié à l'adresse META de destination. Si AMC reçoit un message depuis le canal META, alors, après modification, il le transfère à l'adresse locale de destination.

La figure 4.10 illustre l'algorithme simplifié d'un Agent Mediation Control. Nous y retrouvons les 5 comportements permettant l'interfaçage des communications d'agents depuis un canal local vers un canal META.

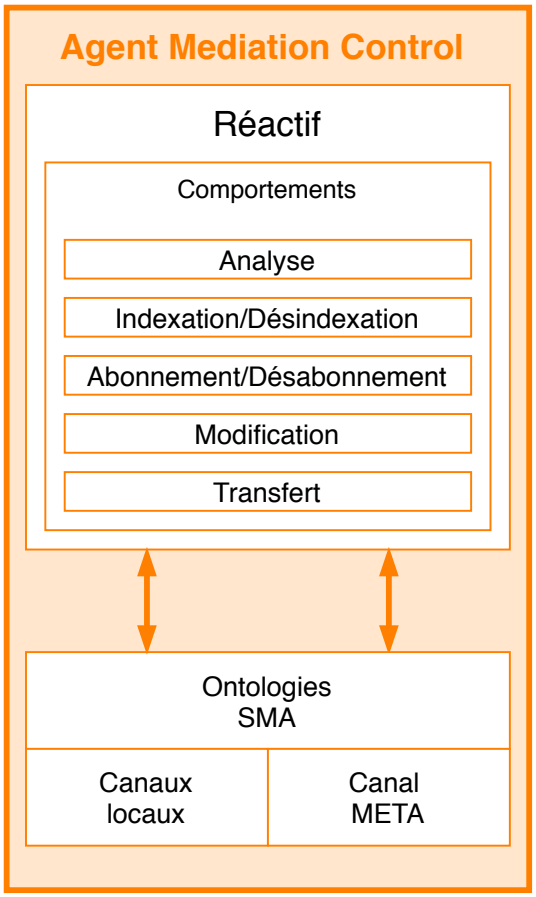


FIGURE 4.9 – Modèle de l'Agent Mediation Control

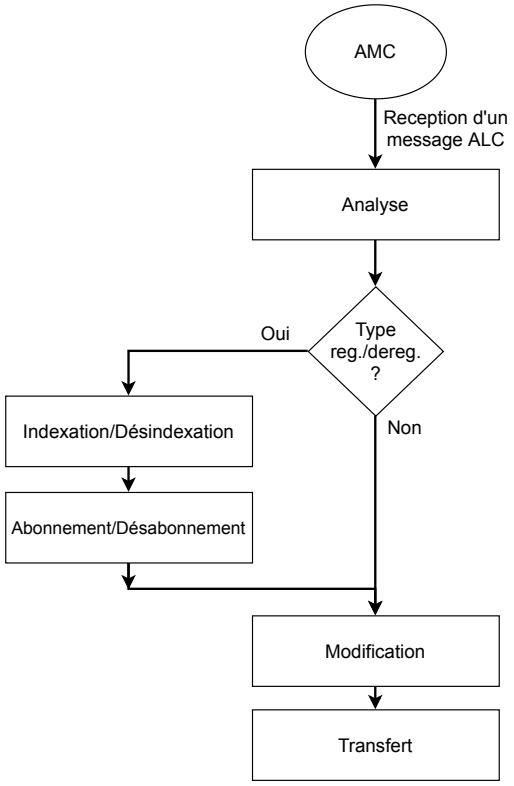


FIGURE 4.10 – Algorithme simplifié d'un Agent Mediation Control

AMC et centralisation locale

Expliquons maintenant le fonctionnement d'un AMC au travers de la situation décrite dans la figure 4.11.

Lorsqu'un agent *Alice*, utilisant un canal local X est créé, il doit s'enregistrer auprès d'AMS, utilisant lui, le canal META. Un AMC prévu pour le canal X sert de relais vers le canal META. À sa création, *Alice* doit s'enregistrer auprès d'AMS par l'envoie d'un ACL message de type *request* comprenant son nom et son adresse sur son canal. Pour cela, *Alice* envoie ce message à son AMC spécifiant que le destinataire final est AMS. À la réception, l'AMC analyse le message provenant de l'agent *Alice*. Il en extrait le type, les destinataires et la source. Lors d'une demande d'enregistrement, l'AMC indexe le nom et l'adresse sur le canal X de l'agent. Il s'abonne ensuite au topic META correspondant au nom de l'agent, en l'occurrence `ACC/Alice/#`, pour pouvoir recevoir les messages des agents par le canal META et les transférer vers le canal X. Enfin, AMC change l'adresse source du message pour y mettre à la place le topic abonné qui lui correspond. Par cette manipulation, l'agent *Alice* est enregistré auprès d'AMS avec une adresse META et non une adresse correspondant à son canal d'origine. Cette abstraction permet à l'ensemble des agents de communiquer avec *Alice* de façon transparente en s'affranchissant de son canal de communication.

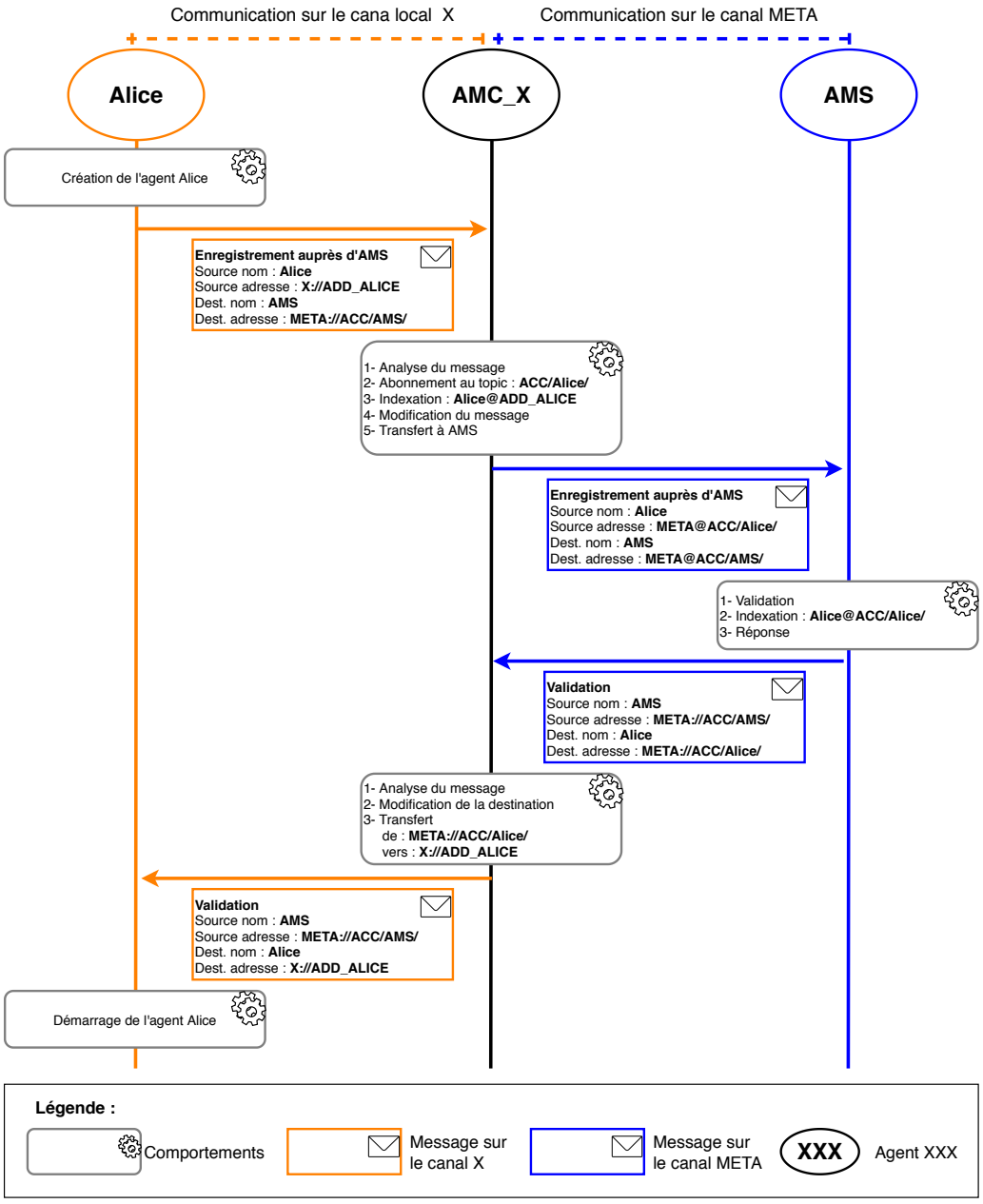


FIGURE 4.11 – Diagramme de temps pour la création d'un agent *Alice* depuis un canal local X

Une fois enregistré, AMS renvoie un message à *Alice* sur `ACC/Alice/` parvenant à son AMC, le message lui est transféré et elle peut désormais communiquer avec tous les agents de la plate-forme. Si AMS répond négativement à la demande d'enregistrement, le message est transféré, l'abonnement du topic par l'AMC est arrêté et l'AMC désindexe les informations de l'agent *Alice*.

Pour développer notre point de vue et l'intérêt de notre agent AMC, étudions un cas pratique énuméré par la fondation FIPA dans le document [FIPA, 2004], illustré figure 4.12.

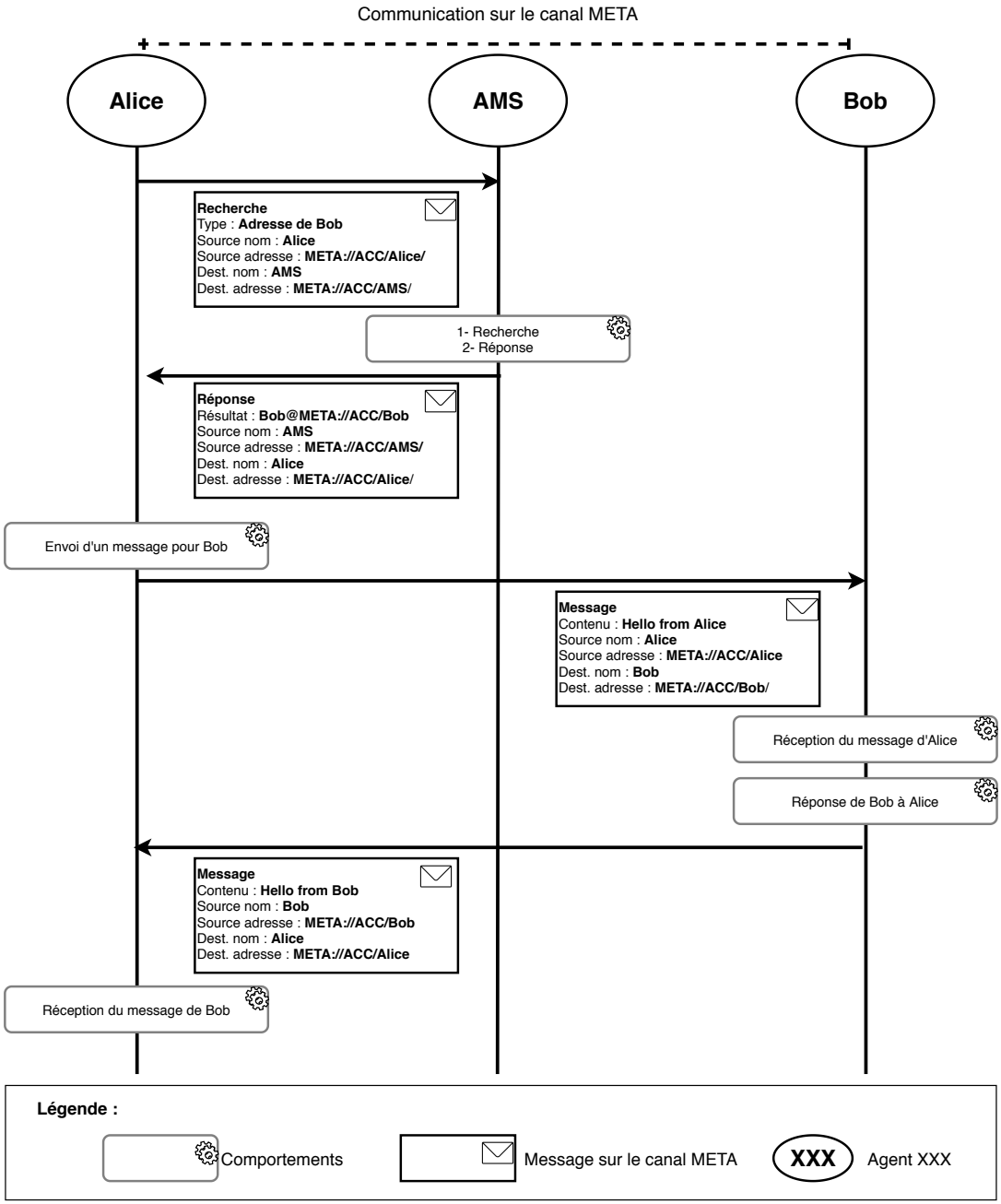


FIGURE 4.12 – Diagramme de temps d'une interaction entre deux agents selon la norme de la FIPA

Alice et *Bob* sont deux IoT-a dans une constellation bénéficiant d'une plate-forme SMA avec les agents de plate-forme AMS et DF. Dans cette étude de cas, l'agent DF n'est pas utile. *Alice* souhaite communiquer avec *Bob*. Dans un premier temps elle ne connaît pas l'adresse de *Bob* et doit pour cela en faire la demande à AMS. À leur création les deux agents connaissent l'adresse d'AMS et ont les compétences suffisantes pour communiquer avec lui. *Alice* contacte AMS en lui envoyant un message de type *research* en précisant le paramètre *resolvers* avec le nom de *Bob*. Le comportement d'AMS est alors de chercher dans son index, les informations concernant *Bob*. Il renvoie à Alice, un message de type *result* avec comme contenu, les informations au sujet de *Bob*. Une fois qu'elle reçoit le message, elle en extrait l'adresse de *Bob*. *Alice* peut désormais communiquer avec *Bob* car elle connaît son adresse.

Tout comme pour notre première situation, *Alice* et *Bob* utilisent ici le même bus de communication pour échanger. Seule l'adresse de *Bob* est inconnue pour *Alice* mais une fois en possession de son adresse, *Alice* est capable de le contacter par le moyen de communication adéquat. Cependant, comment faire si, en plus de son adresse, *Alice* ne connaissait pas le type de canal de communication de *Bob* ? Comment ferait-elle pour échanger avec *Bob* au travers de deux canaux X et Y différents ?

Nous bénéficions désormais d'un agent AMC capable d'interfacer plusieurs canaux de communication. Nous avons vu qu'une plate-forme multi-agents est nécessaire pour bénéficier des agents de plate-forme AMS, DF et AMC. Pour permettre aux agents *Alice* et *Bob* de communiquer, nous faisons évoluer la situation précédente. Désormais, deux agents médiateurs `AMC_X` et`AMC_Y` interviennent, offrant une interface entre leur canal local et le canal META.

Reprenons l'étude de cas par notre nouvelle approche détaillée dans la figure 4.14. L'agent *Alice* souhaite communiquer avec l'agent *Bob*. Les deux agents existent et sont enregistrés auprès d'AMS. *Alice* demande à AMS l'adresse de *Bob* par l'intermédiaire de son `AMC_X`. AMS répond à *Alice* par un message contenant l'adresse de *Bob*. `AMC_X` interprète le message, procède au changement d'adresse de destination pour correspondre à celle d'*Alice* sur le canal local et transfère le message. *Alice* crée alors un nouveau message à destination de *Bob* avec comme adresse de destination `META@ACC/Bob/` et l'envoie à son AMC. L'agent de médiation responsable du canal X reçoit le message, il y modifie l'adresse source et le transfère, sans le savoir, à l'`AMC_Y`. `AMC_Y` modifie l'adresse de destination puis transfère le message à *Bob* sur le canal Y. *Bob* finit par recevoir le message d'*Alice*, il connaît donc son adresse sur le canal META et lui renvoie un message pour se présenter en retour. *Bob* fait parvenir ce message à son `AMC_Y` qui le transfère à l'`AMC_X`. L'AMC d'*Alice* reçoit le message et lui transfère. Les deux agents ont la possibilité d'échanger des messages par l'intermédiaire de la plate-forme SMA.

Lorsque nous reprenons la figure 4.6, nous comprenons désormais comment les IoT-a d'une constellation réussissent à communiquer malgré leurs différences technologiques. Nous pouvons relier notre concept de configurations d'IoT-a énoncé dans la section précédente, avec les éléments que nous venons de présenter. En effet, comme l'illustre la figure 4.13 plusieurs IoT-a ayant des configurations différentes, disposent maintenant des outils nécessaires pour communiquer. Dans cet exemple, deux IoT-a, l'un en configuration 1 ne disposant que de peu de ressources bas niveau, notamment un seul bus I2C pour communiquer, souhaite échanger avec d'autres agents présents dans un IoT-a différent. Ce second IoT-a est lui dans une configuration 4 et dispose de plus de ressources lui permettant de porter les agents de plate-forme nécessaires à l'organisation. Avec AMC, les IoT-a peuvent communiquer et former une constellation. Les agents sont de toutes natures, matériel ou logiciel et peuvent échanger leurs informations grâce aux formalismes apportés par la fondation FIPA et les ajouts complémentaires que nous venons de présenter. Maintenant que les IoT-a d'une constellation forment une organisation centralisée localement, nous devons nous concentrer sur les mécanismes nécessaires pour relier plusieurs constellations les unes aux autres, c'est donc l'objet de la prochaine section.

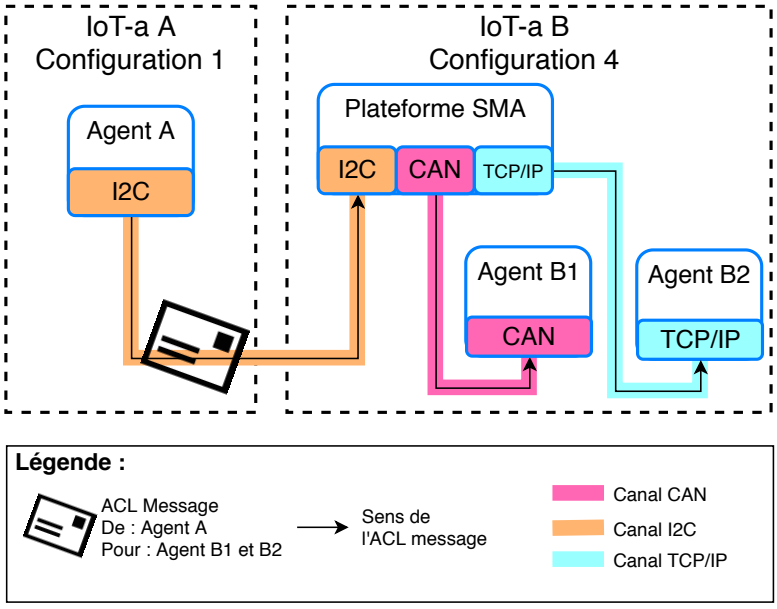


FIGURE 4.13 – Interaction entre deux IoT-a (A et B) avec différentes configurations 1 et 4

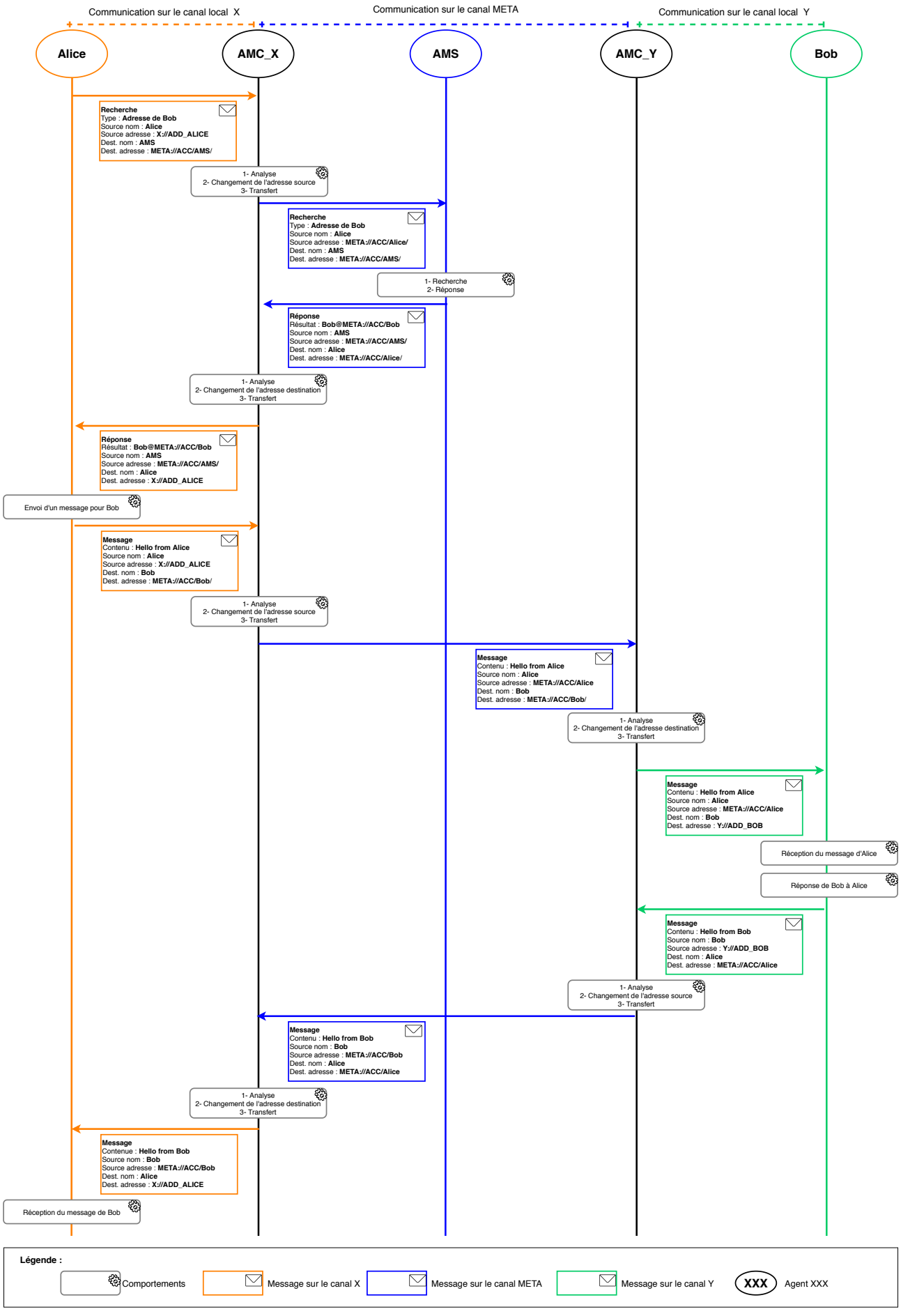


FIGURE 4.14 – Diagramme de temps d'une interaction entre deux agents (canal X vers canal Y)

4.2.3 | *Communication inter-constellations : décentralisation collective*

Nous allons maintenant évoquer dans cette section notre concept de décentralisation collective. En effet, la communication au sein d'une plate-forme entre plusieurs IoT-a avec des configurations différentes est rendue possible grâce à notre concept de centralisation locale. Néanmoins, cette approche ne donne pas toutes les clés nécessaires aux interactions entre l'ensemble des IoT-a répartis dans plusieurs constellations. Nous proposons alors, le concept de décentralisation collective dans la définition 12 afin d'organiser et faire interagir nos constellations d'IoT-a.

Définition 12 : La décentralisation collective

La décentralisation collective assure une organisation dynamique et résiliente des constellations d'IoT-a par l'intermédiaire de différentes topologies de liens. Elle permet l'harmonisation des interactions entre tous les agents, à l'échelle des constellations, en offrant un espace commun de communication.

La centralisation locale fait émerger des agrégats autonomes d'agents que nous appelons constellation. Les méthodes pour mettre en place des liens entre plusieurs plate-formes multi-agents sont évoquées dans certains travaux et notamment dans le projet JADE [Bellifemine et al., 2007]. Toutes ces approches reposent aujourd'hui sur des protocoles réseau de type *Request/Response* et ne profitent pas des avantages qu'apportent des protocoles basés sur le type *Publish/Subscribe*. En effet, nous avons vu précédemment que ce type de protocole permet de décentraliser la gestion des messages au sein des clients, le broker n'ayant qu'un rôle d'aiguilleur. Dans le cas des IoT-a, chaque constellation utilise son propre broker et tous les agents s'y connectent directement ou au travers de leur AMC. Cette méthode permet de rendre une constellation autonome, néanmoins elle impose qu'au moins un IoT-a du groupe soit capable d'exécuter un broker.

L'utilisation de protocoles basés sur le type *Publish/Subscribe* modifie le paradigme de lien inter plate-forme. Avec cette approche, ce sont les clients qui s'abonnent aux topics qu'ils souhaitent auprès de leur broker. Lorsqu'un message est publié sur un topic, tous les clients abonnés reçoivent le message. Aussi, la publication de messages à un groupe est simplifiée côté client et transparente pour le broker. L'organisation des messages repose alors sur une mise en place cohérente d'une arborescence de topics entre tous les clients. Cette arborescence permet de répartir l'information à des agents uniques ou à des groupes d'agents sans faire appel à des fonctions spécifiques dans le broker.

Parmi les différents protocoles aujourd'hui formalisés pour l'échange de messages de type *Publish/Subscribe*, certains jouissent de projets proposant la mise en place de passerelles entre brokers. L'un d'eux, le projet open source Mosquitto [Light, 2017], implémente un broker MQTT répondant à toutes les normes du protocole. Il propose en plus, la génération de *bridges* afin de relier plusieurs instances de brokers et ainsi partager des messages entre les clients de l'ensemble des brokers. Nous parlons bien ici d'un projet et non pas du protocole MQTT directement. En effet, le protocole MQTT ainsi que d'autres protocoles comme le CoAP ou le XMPP, ne proposent aucune formalisation pour la gestion des liens entre plusieurs brokers. Aussi, c'est au niveau des implémentations logicielles que nous pouvons trouver des mécanismes proposant ce type de formalisme. Nous notons d'ailleurs le manque d'harmonisation, sur ce point, entre les projets implémentant le même protocole. Pour exemple, la non-compatibilité des liens entre les projets HiveMQ et Mosquitto/Eclipse ou l'un parle de cluster alors que l'autre met en place des bridges. En l'occurrence, le choix de développer les formalismes de bridge du projet Mosquitto tient à son point de vue proche de la logique des formalismes officiels de la norme MQTT.

Nous détaillons dans cette section les mécanismes permettant de connecter des constellations d'IoT-a afin qu'elles puissent interagir. Notre démarche va plus loin que la présentation des concepts répondant à cette problématique. En effet, nous avons développé le sujet de la résilience dans notre état de l'art, en section 2.3, et avons avancé les atouts d'une telle approche dans le domaine de l'ingénierie et plus particulièrement des objets connectés. Les IoT-a sont des systèmes susceptibles d'être mis en place dans des environnements sensibles, bas niveaux et temps réel. Ils doivent être résilients face aux différents aléas, comme la mobilité des objets physiques, la perte de connexion, ou des performances limitées. C'est dans cet objectif que nous présentons les limites d'une première approche concernant les bridges. Nous la faisons évoluer afin de rendre nos constellations d'IoT-a plus résilientes. En l'occurrence, nous voyons comment les interconnecter à la volée tout en proposant des stratégies de liens répondant aux différentes étapes de gestion d'erreurs proposées dans les travaux de [Pujolle, 2006a].

Cette section est donc organisée de la façon suivante. Dans un premier temps, nous présentons la technologie de bridge mise en place dans le projet Mosquitto. Nous voyons son fonctionnement et quels sont les éléments susceptibles de nous aider pour interconnecter des constellations d'IoT-a. Nous présentons ensuite les limites des bridges dans leur fonctionnement initial, au travers de deux applications mettant en réseau de multiple broker MQTT. Nous exposons à la suite, nos propositions permettant des liens dynamiques capables d'être réagencés à la volée entre les systèmes. Puis, nous voyons comment les IoT-a interagissent pour organiser ces liens de façon décentralisée et résiliente. Enfin, nous nous penchons de nouveau sur nos applications et étudions théoriquement comment elles se comportent avec cette nouvelle approche dynamique.

Notion de bridge

Un bridge permet de mutualiser l'arborescence de topics entre plusieurs brokers. La figure 4.15 illustre deux brokers reliés par un bridge partageant la même arborescence de topics. Avec cette méthode, les agents des deux plate-formes peuvent communiquer de façon transparente sans avoir recours à aucun autre agent d'organisation. Nous pouvons alors extrapoler cette idée et parler du bridge comme d'une solution permettant de créer un environnement commun de communication entre plusieurs brokers. Chaque broker se bridgeant sur un autre, partage ses clients avec lui et leur offre la possibilité de communiquer avec tous les autres clients déjà interconnectés. La figure 4.16 illustre cette idée. Nous observons une population d'agents au centre d'un espace commun de communication à tous les agents. Le nombre de brokers joue sur le nombre d'agents, mais une fois bridgés, la communication s'effectue de façon transparente pour tous. Lorsque nous appliquons cette particularité des bridges avec nos principes de centralisation locale, nous offrons aux agents de chaque agrégat la possibilité d'échanger dans un environnement homogène, décentralisé et disposant des mécanismes de communication qu'ils possèdent déjà avec la centralisation locale.

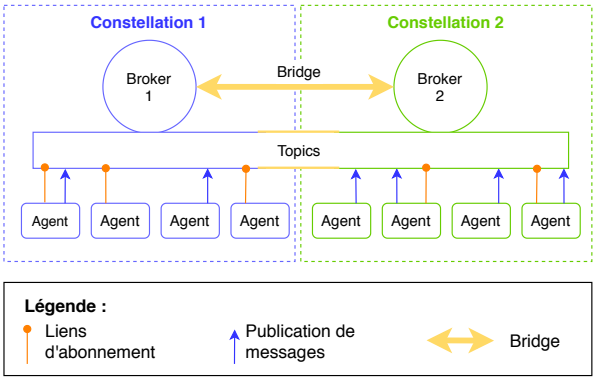


FIGURE 4.15 – Bridge entre deux brokers

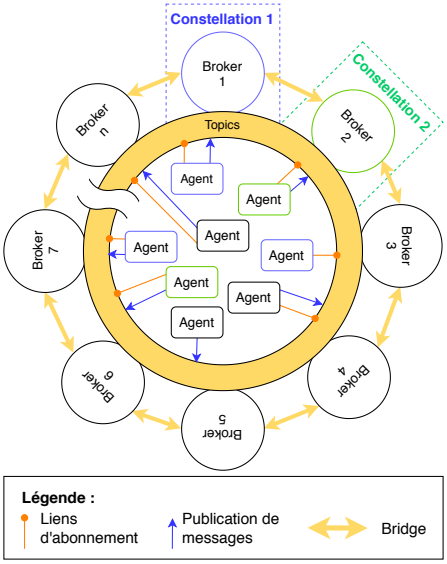


FIGURE 4.16 – Bridges multiples entre plusieurs brokers

Un bridge se caractérise par la mise en place de liens d'abonnement entre un broker source et un second cible. Vis-à-vis du broker cible, le broker source est considéré comme un client et bénéficie des mêmes procédures d'abonnement et de publication. En fonction des règles d'abonnement configurées, les liens entre les brokers peuvent être bidirectionnels afin d'échanger des messages depuis et vers chacun ou unidirectionnels avec un sens pour ne permettre que certains échanges. Chaque bridge est qualifié par un nom et correspond à un broker cible auquel il souhaite partager ses messages. Un broker peut mettre en place plusieurs bridges vers un ou plusieurs brokers. Nous détaillons dans la figure 4.17 les étapes de connexion intervenant dans la création d'un bridge. Dans un premier temps, le broker source 1 initialise le bridge en ajoutant une règle d'abonnement locale à ses propres topics. Il se connecte ensuite au broker cible. Ce dernier réagit de la même façon à la connexion de la source que pour la connexion d'un nouveau client. Une fois connecté, le broker source s'inscrit aux topics cibles configurés, crée le bridge en mémoire et les échanges peuvent alors commencer. Lorsqu'un client connecté au broker 1 publie un message sur le topic source, le broker 1 republie le message vers le broker 2 sur le topic cible.

Inversement, lorsqu'un client connecté au broker 2 publie un message sur le topic cible, le boker 1, abonné à ce topic, reçoit le message et le republie sur le topic source.

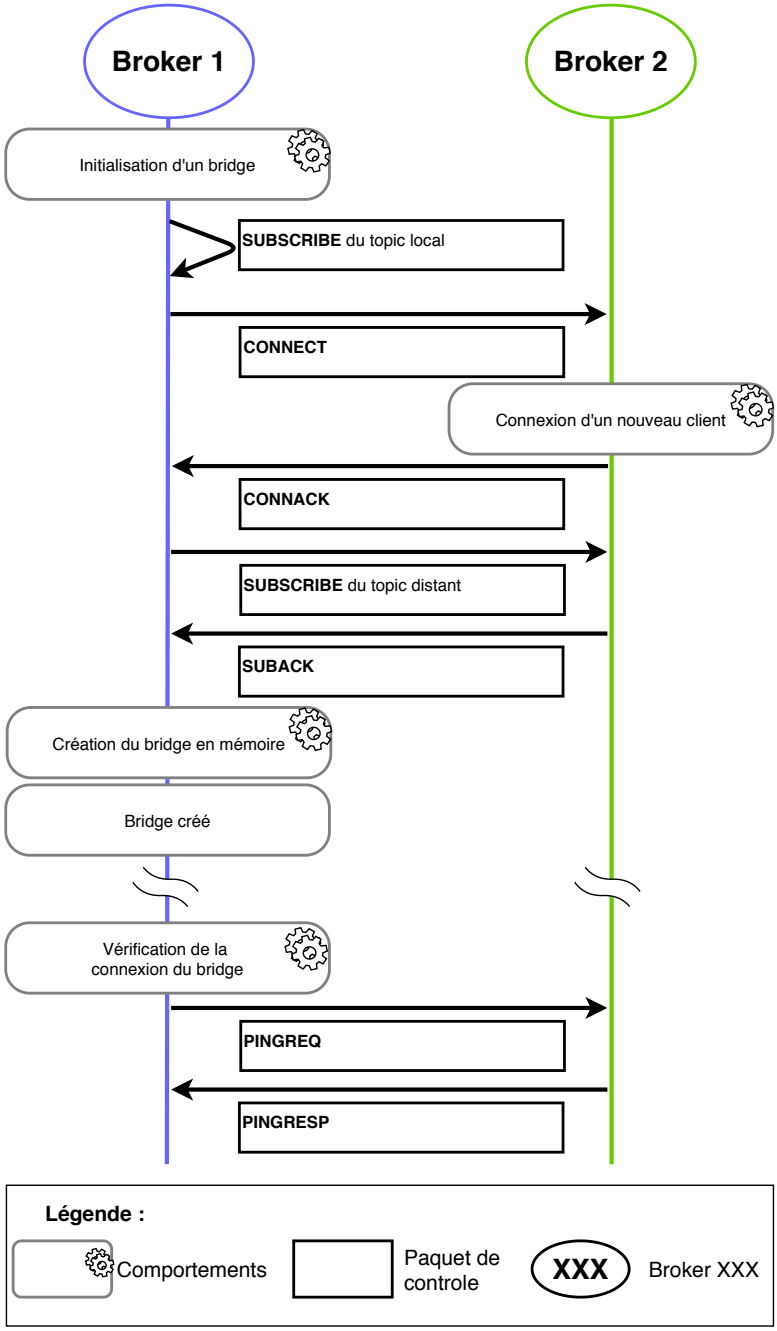


FIGURE 4.17 – Diagramme de temps de la connexion d'un bridge entre deux brokers

Limites des bridges statiques

Disposer de liens directs entre IoT-a, par l'intermédiaire des bridges, permet de nombreuses possibilités de gestion de l'information, comme l'organisation hiérarchique d'agents en fonction de leur emplacement physique, de leur nature ou de leur criticité. Ils permettent de mettre en place une redondance ou de disperser l'information sur différents sites géographiques. Cependant, pour que les plate-formes SMA puissent répondre aux critères de tolérance aux pannes, de dynamisme et d'autonomie, les bridges doivent pouvoir évoluer en fonction de différents indicateurs. Nous constatons que les différents projets de broker MQTT open source disponibles, proposent principalement l'utilisation d'un fichier de configuration où l'ensemble des options nécessaires à la mise en place de bridges y sont définies. Cette solution va à l'encontre du paradigme multi-agents, car elle contraint le dynamisme et l'autonomie de l'environnement de communication de nos IoT-a. La mise en place

de bridges par cette méthode est statique et nous constatons ses limites au travers de différents cas de figure. Détaillons l'un d'entre eux.

L'étude du cas illustré au travers de la figure 4.18 nous permet de décrire les contraintes d'une gestion statique des bridges. Dans cette étude nous retrouvons cinq brokers ayant pour identifiant un chiffre unique de 1 à 5. À l'état initial, chacun d'entre eux se relie par l'intermédiaire d'un bridge à son voisin disposant de l'identifiant directement supérieur. Chaque broker démarre normalement et crée le bridge en fonction des paramètres présents dans sa configuration. Parmi les différentes topologies de liens réseau existantes dans la littérature [Takabatake, 2011], la topologie en ligne établie entre les brokers de la figure 4.18 soulève un problème en cas de défaillance d'un des nœuds. En effet, à l'étape suivante, l'arrêt du broker 3 isole le réseau en deux parties. D'un côté les brokers 1 et 2 n'ont plus les capacités de communiquer avec l'autre côté, exploitant les brokers 4 et 5.

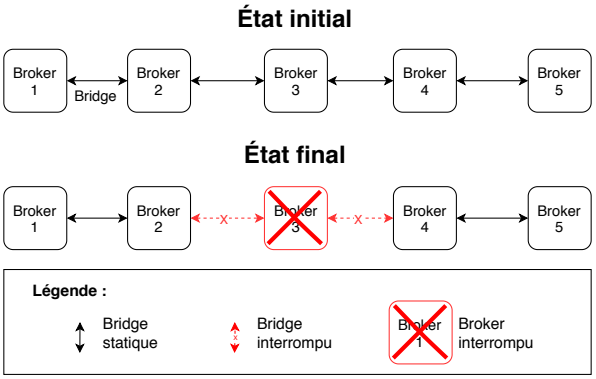


FIGURE 4.18 – Topologie en ligne

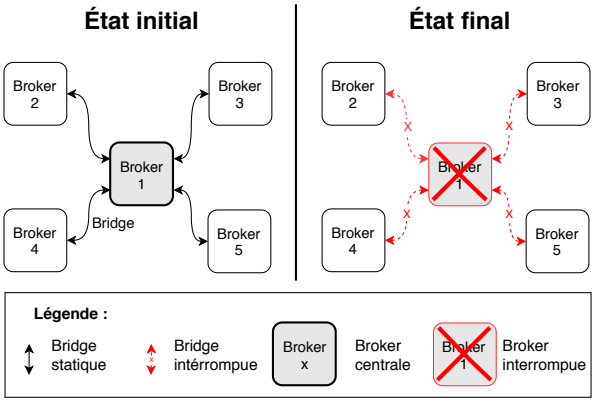


FIGURE 4.19 – Topologie en étoile

Un second cas de figure illustré figure 4.19, montre encore une fois les limites des bridges statiques face à l'arrêt ou la déconnexion d'un des éléments du réseau. En l'occurrence, les brokers de la figure 4.19 sont ici reliés dans une topologie en étoile, ayant pour nœud central le broker identifié par le chiffre le plus faible. À l'étape initiale, tous les brokers sont reliés au broker 1. Lorsque celui-ci vient à s'arrêter, plus aucun broker ne peut échanger de messages avec ceux encore connectés.

Proposition de bridges dynamiques

Dans ces cas de figure, les bridges statiques contraignent l'évolution du réseau de brokers et restreignent la tolérance aux pannes. Nous proposons alors une nouvelle approche pour la gestion des bridges de façon à la rendre dynamique. Pour se faire, l'analyse de la norme MQTT met en lumière les particularités de certains topics réservés. La norme définit succinctement l'utilisation de topics pour les besoins du broker. Ils sont appelés : *$ topics*. Ils proposent des informations concernant le serveur, comme le nombre de connexions actives, la charge réseau, le nombre de paquets transmis, etc. Un agent s'abonnant à ces topics dispose d'indicateurs propres aux communications et peut ainsi alimenter ses comportements proactifs en informations. La norme est plutôt ouverte sur ce point et ne donne pas de précision sur l'utilisation concrète de ce type de topics. Ce faisant, il est possible de les utiliser pour un usage différent de la récupération d'information. En l'occurrence, rien n'est précisé au sujet de l'utilisation de ces topics pour la publication de messages sur le serveur. En effet, un broker a les moyens d'en écouter certains pour procéder à des comportements spécifiques. Sous couvert de certains mécanismes de sécurité pour éviter des modifications de la part de clients indésirables que nous détaillerons par la suite, les agents pourraient avoir un certain contrôle de leur broker et donc faire évoluer des paramètres de leur réseau.

Nous profitons de cette particularité pour proposer deux $ topics au sein d'un broker, responsables des comportements de création et suppression de bridges. Les paramètres d'un bridge étant initialement contraints par un fichier de configuration pris en compte au début de l'exécution du broker, cette approche permet de créer et supprimer des bridges dynamiquement par des agents. De plus, la mise en place de topics de ce type, permet à tout client MQTT de modifier les liens entre brokers sans nécessiter un accès privilégié au système hôte. Un client n'a plus besoin de disposer de droits spécifiques pour modifier le fichier de configuration et redémarrer le

broker pour modifier ses bridges. La figure 4.20 détaille notre approche pour la création dynamique de bridges en parallèle du fonctionnement initial.

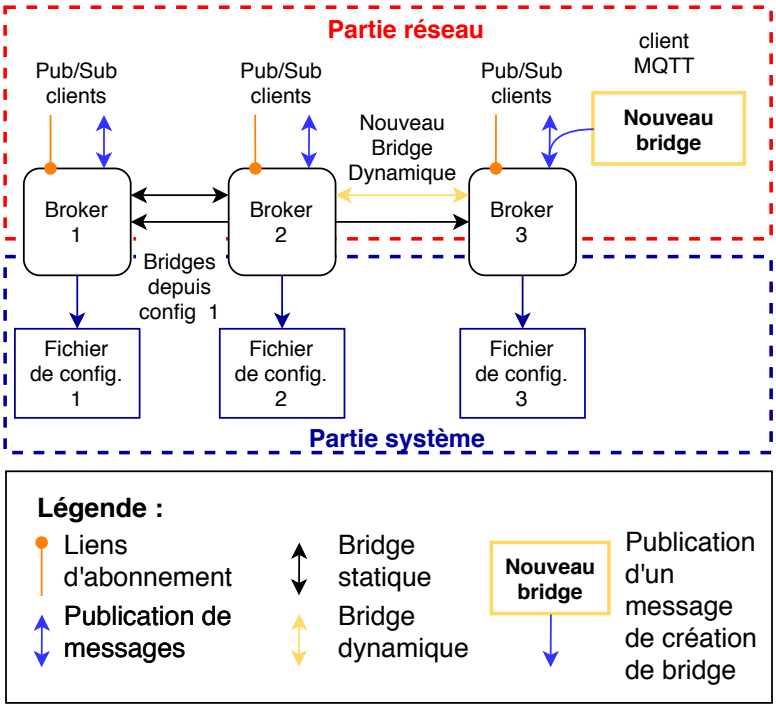


FIGURE 4.20 – Procédure de création d'un bridge dynamique

Côté formalisation, nous nous inspirons des travaux de [Light, 2017] dans le projet Mosquitto. Le projet open source propose l'utilisation de topics réservés ayant pour base le pattern `$SYS/`. Le broker envoie périodiquement des informations au sujet de son fonctionnement sur ces canaux. Les topics utilisés respectent la norme MQTT et nous estimons, de par notre expérience, que l'arborescence proposée par Mosquitto organise correctement les indicateurs et permet de mettre à disposition des informations claires pour les clients. Comme nous venons de le voir, nous proposons qu'un broker dispose de 2 topics pour la création et la suppression dynamique de bridges. Aussi, nous complétons l'arborescence proposée par Mosquitto, avec l'ajout de deux nouveaux topics pour la gestion des bridges dynamiques.

— `$SYS/broker/bridge/new`, topic utilisé pour la création d'un ou plusieurs nouveaux bridges.
— `$SYS/broker/bridge/del`, topic utilisé pour la suppression de bridges dynamiques ou statiques.

Pour ce faire, nous avons modifié le projet Mosquitto afin d'intégrer la gestion dynamique des bridges au sein du broker. Cet ajout de code est aujourd'hui disponible librement sur notre compte Github : `https://github.com/Tifaifai/mosquitto`. Après la publication de cet apport auprès de la communauté Mosquitto, nous attendons une validation de cette dernière pour son intégration dans le projet officiel.

Formalisation des liens inter-constellations

Afin d'intégrer la notion de bridges dynamiques à notre concept de constellations d'IoT-a, nous proposons de la modéliser par deux types de connecteurs. La figure 4.21 illustre une constellation avec un connecteur de sortie et un autre d'entrée. Un connecteur d'entrée est représenté par une cavité circulaire capable de se conformer à un connecteur de sortie, représenté par une excroissance circulaire. Un bridge entre deux constellations se verra symbolisé par un couple de connecteurs d'entrée-sortie. Une constellation étend à l'origine de la mise en place d'un bridge, symbolise sa connexion par un connecteur de sortie relié au connecteur d'entrée de la constellation cible. Inversement, elle symbolise ce lien par un connecteur d'entrée lorsqu'elle reçoit la connexion d'un bridge. Autant de connecteurs que nécessaires sont possibles pour répondre aux différents liens essentiels aux connexions inter-constellations. Par exemple, la figure 4.22 met en scène une topologie arbitraire de liens entre plusieurs constellations. Certaines partagent plusieurs couples de connecteurs alors que d'autres n'en ont qu'un.

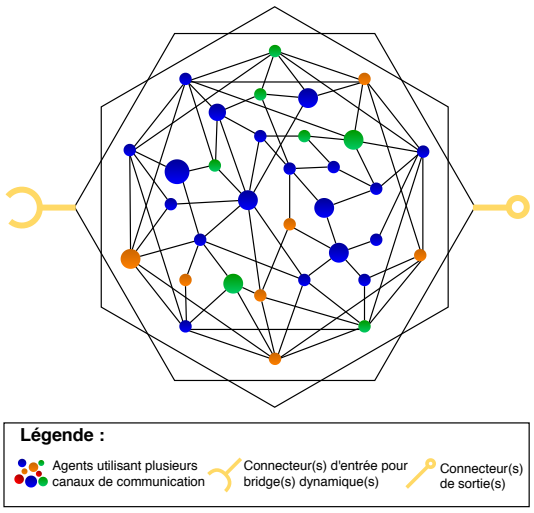


FIGURE 4.21 – Connecteurs inter-constellations

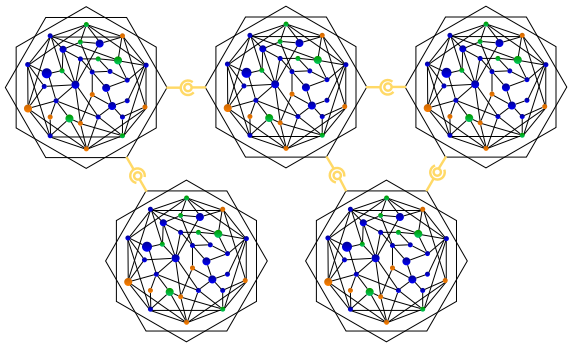


FIGURE 4.22 – Exemple de topologie d'interconnexions entre constellations d'IoT-a

Agentification de liens inter-constellations

Au travers des topics que nous avons définis précédemment, les clients ont la possibilité de publier un message au broker afin de créer ou supprimer un bridge. Dans notre cas, tous les agents d'une constellation d'IoT-a ont donc les clés pour administrer dynamiquement les liens avec d'autres constellations. Néanmoins, la capacité de créer des liens n'est qu'un outil pour relier nos constellations entre elles. Il est nécessaire d'organiser les politiques de liens afin de ne pas créer des réseaux trop complexes ou surchargés. Nous proposons une approche multi-agents pour la mise en place de liens dynamiques entre nos constellations.

Dans les conditions initiales, nous disposons donc de plusieurs constellations d'IoT-a ne pouvant pas échanger d'information. Notre objectif est de les relier selon une stratégie de topologie prédéfinie de façon à ce que tous les IoT-a puissent échanger, quelle que soit leur constellation. Pour cela, nous proposons la mise en place de plusieurs agents capables de créer dynamiquement les liens nécessaires au moyen de notre gestion dynamique de bridges.

La première fonction à laquelle ces agents doivent répondre est la découverte. Elle permet à chaque constellation de déterminer la présence de ses voisines pour pouvoir s'y relier. En effet, les constellations ne se connaissent pas entre elles et évoluent dans le temps. Ces dernières peuvent apparaître sur le réseau, dû à l'ajout de nouveau IoT-a ou d'un nouvel équipement rassemblant une ou plusieurs constellations. De la même façon, des dysfonctionnements comme une perte d'alimentation ou de connectivité réseau peuvent survenir et faire évoluer le nombre de constellations.

La seconde fonction est la stratégie de liens employée pour connecter les constellations entre elles. Elle se traduit par un algorithme prenant en compte les éléments découverts et la position de la constellations par rapport aux autres. De nombreuses stratégies existent dans la littérature et nous faisons le choix de nous restreindre à deux des plus simples pour illustrer notre propos. Les deux topologies *ligne* et *étoile* que nous avons évoquées dans les figures 4.18 et 4.19 nous servent de cadre. Elles sont considérées comme les références en matière de relation entre éléments dans un réseau. Nous proposons donc notre point de vue afin que nos constellations puissent s'organiser selon ces deux types de topologies.

Enfin, la dernière fonction est la création et la suppression effective des liens entre nos constellations. Pour cela, il est nécessaire de produire les messages adéquats pour chaque broker afin de créer le lien vers la nouvelle constellation. Il est notamment important que chaque constellation ait conscience des liens qui la relient avec le reste de la communauté. En effet, lors de la rupture de certains de ses liens, une constellation doit pouvoir supprimer ses liens en mémoire pour ne pas être victimes de certains effets de bords. Il est difficile de connaître la raison de la perte d'un lien. La constellation disparue est-elle simplement déconnectée du réseau ou connaît-elle une anomalie logicielle ou une coupure de courant ? Sans ces informations, il est nécessaire de prendre une décision unique, quel que soit le type de perte d'un lien. Si jamais cela n'est pas effectué, nous avons mis en évidence un problème appelé *liens fantômes* entre les constellations [Schmitt et al., 2018]. Ces derniers nuisent aux communications en surchargeant le nombre de messages et en créant plusieurs liens non désirés.

De par ces fonctions, nous proposons deux agents responsables des liens entre constellations. Ils doivent être présents dans chaque constellation et uniques. Le but est de rendre une constellation autonome de façon à ce qu'elle n'ait pas besoin d'avoir recours à d'autres constellations pour créer ses liens. L'emplacement de ces deux agents reste libre. Pour des raisons de performance l'utilisateur pourra intégrer ces agents dans des configurations bas-niveau, s'il le souhaite.

Le premier de ces agents est l'**agent Bridge**. Il est en capacité de créer et supprimer des bridges depuis sa constellation vers une ou plusieurs constellations. Le comportement proactif de cet agent a pour charge d'attendre un ordre de création ou de suppression d'un bridge depuis un autre agent. Il dispose néanmoins de comportements proactifs lui permettant d'adapter la gestion de bridges si jamais il en ressent le besoin (surcharge réseau, batterie critique, etc.). L'agent est responsable des liens de sa constellation. Aussi, il prend garde de supprimer les liens inactifs ou fantômes pour éviter les anomalies durant les allées et venues des autres constellations.

Le second agent est l'**agent Ping**, responsable des deux fonctions que nous avons citées précédemment. La première est la découverte. Les méthodes pour effectuer cette fonction sont variées. L'utilisateur est libre d'implémenter le protocole qu'il souhaite pour déterminer la présence des autres constellations sur le réseau. Néanmoins, ce comportement doit être périodique et permettre la mise à jour régulière d'une liste de constellations. Le second comportement est la gestion des stratégies de liens. Ces stratégies sont alimentées par les données perçues par le comportement de découverte. Elles déterminent les liens qui doivent être élaborés depuis la constellation vers ses voisines. Une fois les meilleures routes décidées, l'agent Ping est capable de transmettre à l'agent Bridge, les liens qu'il doit mettre en place.

Illustration de liens dynamiques inter-constellations

Nous pouvons désormais reprendre les deux topologies en ligne et étoile pour illustrer notre approche avec nos nouveaux agents Ping et Bridge. Dans les figures 4.18 et 4.19 nous avons remarqué les difficultés imposées par le bridge statique lors de la déconnexion d'un des éléments du réseau. Dans le cas de constellations munies des agents Bridge en Ping, les bridges ont la possibilité d'évoluer en fonction des constellations présentes. Dans un premier temps, nous présentons notre approche pour le cas de la topologie en ligne puis celle en étoile.

Prenons l'exemple d'une population de cinq constellations, chacune associée à un identifiant de 1 à 5. La stratégie en ligne régit l'agent Ping au sein de chacune d'elle. Aussi, le comportement interne de l'agent répond à l'algorithme 1 dans lequel nous retrouvons les paramètres suivants : n, p et i représentent respectivement, les identifiants minimum, maximum et courant des constellations. $CONSTELLATION\_ID$ exprime l'identifiant de la constellation responsable de l'agent Ping et Bridge en cours, alors que *idToSearch* est l'identifiant de la constellation à chercher sur le réseau. La fonction *search*() est responsable de l'interrogation du réseau pour déterminer si une constellation est joignable. Elle retourne une valeur booléenne `vrai` si elle détermine qu'une constellation est connectable par un bridge et `faux` dans la cas contraire. Enfin, la variable *constellationToConnect* sauvegarde l'identifiant de la constellation à connecter. Si l'agentPing estime nécessaire de créer un nouveau lien entre sa constellation et une autre, il contacte l'agent Bridge et lui spécifie les paramètres d'une nouvelle connexion dynamique de bridge. La figure 4.23 illustre l'évolution des liens entre les constellations. À l'étape initiale, chaque constellation est reliée à sa voisine disposant d'un identifiant directement supérieur. L'agent Ping analyse constamment la présence de chaque constellation et détecte à l'étape finale que celle identifiée par le numéro 3 est défaillante. Chaque constellation détecte cette anomalie, mais de par la stratégie appliquée, seule la 2 réagit pour résoudre la brèche. L'agent Ping du groupe 2 estime les paramètres nécessaires et notifie son agent Bridge afin de créer un nouveau bridge entre les constellations 2 et 3. À l'étape finale, toutes les constellations bénéficient à nouveau de liens leur permettant les interactions entre les différents IoT-a. Nous notons, de plus, la cohabitation des approches statique et dynamique pour relier les éléments.

De manière analogue à la stratégie en ligne, nous pouvons montrer l'intérêt de notre approche de liens dynamiques entre constellations avec une stratégie en étoile. Pour ce faire, nous retrouvons dans l'algorithme 2 les mêmes paramètres que précédemment. Nous pouvons constater dans la figure 4.24 l'étape initiale impliquant 5 constellations reliées en étoile par des bridges statiques. Le broker de la constellation 1 finit par ne plus fonctionner et l'ensemble du groupes d'IoT-a se retrouve sans solution de communication. Suite à cela, chaque agent Ping détecte l'avarie et détermine le broker 2 comme étant le nouveau centre des communications puis notifie son agent Bridge. À l'étape finale, toutes les constellations sont de nouveau reliées à un point central qui est la constellation 2.

Algorithm 1 Stratégie en ligne

```
1: procedure LineLinker(constellationToConnect)
2:     for i ← CONSTELLATION_ID − 1, i >= n, i ← i − 1 do
3:         idToSearch ← i
4:         k ← search(idToSearch)
5:         if k = true then
6:             constellationToConnect ← idToSearch
7:             return true
8:         end if
9:     end for
10:    return false
11: end procedure
```

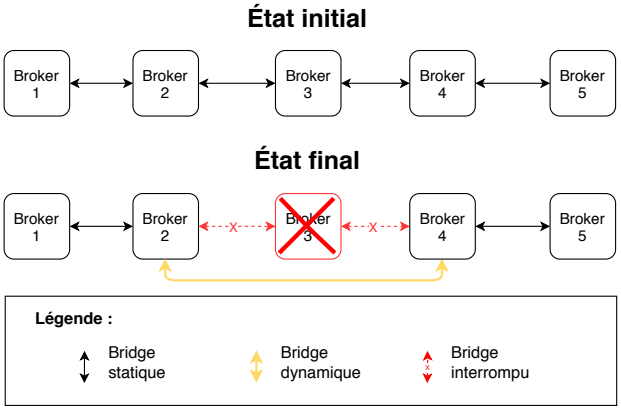


FIGURE 4.23 – Topologie en ligne de constellations avec une gestion dynamique des liens

Algorithm 2 Stratégie en étoile

```
1: procedure StarLinker(constellationToConnect)
2:     for i ← n, i <= p, i ← i + 1 do
3:         idToSearch ← i
4:         if idToSearch = CONSTELLATION_ID then
5:             return false
6:         end if
7:         k ← search(idToSearch)
8:         if k = true then
9:             constellationToConnect ← idToSearch
10:            return true
11:        end if
12:    end for
13:    return false
14: end procedure
```

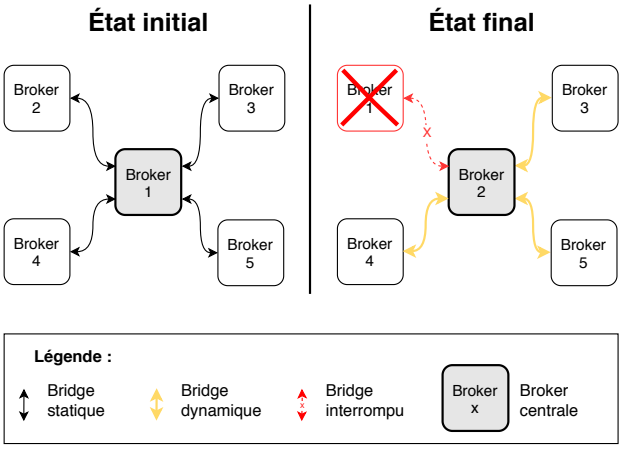


FIGURE 4.24 – Topologie en étoile de constellations avec une gestion dynamique des liens

Organisation des agents entre constellations

Les agents de liaison Ping et Bridges assurent les liens dynamiques entre les différentes constellations d'un réseau d'IoT-a. Afin que les échanges entre les constellations n'entrent pas en conflit, nous proposons certaines modifications dans les standards de la FIPA. Notamment, la mise en place d'un nouveau paramètre au sein des agents et de leur procédure d'enregistrement. Certains comportements de l'agent de contrôle AMS sont aussi impactés.

À l'issue de sa création un agent s'enregistre auprès d'AMS par l'émission d'un tuple de clé-valeurs que sont, son nom et son adresse. Afin de permettre à un AMS de connaître la provenance d'un agent parmi plusieurs constellations, nous ajoutons à ce tuple un troisième paramètre, identifiant le nom unique de la constellation hôte. Désormais, un agent s'enregistre avec le tuple présent table 4.1.

Proposition 1 : IDENTIFICATION D'UN AGENT

Clé	Valeur
Agent-name	Un nom unique global pour l'agent
Agent-locator	Une ou plusieurs descriptions de transport permettant de contacter l'agent. Chacune d'entre elle décrit une structure avec le type de transport, l'adresse de transport spécifique et les propriétés si nécessaire du type de transport
Agent-constellation-name	Un nom unique global de la constellation dont il fait partie

Tableau 4.1 – Tuple de clés-valeurs nécessaires à l'identification d'un agent auprès de son AMS dans le cadre d'une mise en place de constellations d'IoT-a

Concentrons-nous maintenant sur les éléments à modifier concernant AMS. Pour rappel, cet agent est présent au sein de chaque constellation de façon à ce qu'elles puissent évoluer en autonomie lorsqu'elles ne sont pas reliées aux autres. Tant qu'elles sont seules, notre approche respecte le standard de la FIPA qui stipule qu'il ne doit exister qu'un seul agent AMS par plate-forme multi-agents. Cette règle a pour but de restreindre de potentiels conflits de communication entre les agents de la plate-forme. Néanmoins, lorsque les constellations se relient collectivement, plusieurs AMS évoluent simultanément sur la même arborescence de topics. En effet, les bridges créent un espace commun ou tous les agents peuvent communiquer et notamment les AMS. Ils entrent donc en conflit. Nous constatons encore une fois certaines limites de la norme de la FIPA dans le cadre de nos travaux et proposons, pour ce faire, des éléments supplémentaires pour répondre à notre approche de décentralisation collective.

Les éléments que nous apportons à l'Agent Management System annulent et remplacent ceux de la norme de la FIPA concernant ce point. De ce fait, nous en dressons une nouvelle définition initialement précisée dans le document [FIPA, 2004], alinéa 4.2.1.

Proposition 2 : AMS constellations

Un AMS est un composant obligatoire dans une constellation et il n'en existe qu'une seule instance. Il est responsable de la gestion des opérations d'une constellation, comme la création ou la suppression d'agents sans pour autant en être l'unique responsable. Étant donné les différentes compétences que peuvent avoir des constellations, un AMS peut être interrogé pour obtenir une description de sa constellation. Un cycle de vie est associé à chaque agent d'une constellation et est maintenu par son AMS.

AMS représente l'autorité de gestion de tout agent enregistré auprès de lui. Il peut demander ou ordonner un agent à effectuer certaines fonctions, comme celle de se terminer. AMS tient un index de tous les agents lui notifiant sa présence. Il est capable d'indexer ou désindexer des agents d'une autre constellation. Chaque agent, en accord avec le modèle de référence de la FIPA, doit s'enregistrer auprès de l'AMS de sa constellation. Il effectue son enregistrement en contactant le topic `ACC/AMS/register`. La description d'un agent peut être modifiée après son enregistrement sans limite de temps ou de raison. Elle est néanmoins restreinte par l'autorisation d'AMS. La vie d'un agent au sein d'une constellation se termine par un désenregistrement auprès d'AMS sur le topic `ACC/AMS/deregister`. Après celui-ci, l'AID présent dans l'index est libéré et mis à disposition pour d'autres agents souhaitant l'utiliser. La recherche de la description d'un agent peut être effectuée par une demande auprès d'AMS sur le topic `ACC/AMS/search`.

AMS est toujours nommé AMS quel que soit le nom de sa constellation. Il est le seul agent pouvant exister plusieurs fois parmi de multiples constellations. AMS doit néanmoins s'abonner obligatoirement à deux topics dont `ACC/AMS/#` et `ACC/AMS_nom_constellation/#`, où `nom_constellation` est le nom unique de la constellation dont il fait partie. Cette dernière précision permet aux agents de communiquer avec leur AMS local sans notifier les autres. Lorsqu'une constellation est reliée à une ou plusieurs autres par l'intermédiaire d'un bridge, AMS continue d'effectuer ses comportements normalement. Aussi, il indexera un nouvel agent même si celui-ci n'appartient pas à sa constellation.

Dans son index, AMS tient à jour la disponibilité de ses agents. Un agent peut de lui-même modifier sa disponibilité en émettant une notification `modify` de type : "`:availability available/unavailable`". Lors d'un enregistrement, si un agent n'existe pas encore ou s'il est déjà enregistré, mais noté comme indisponible, alors il est considéré comme disponible.

Tout agent peut notifier AMS de la disponibilité d'une constellation. Lorsqu'un agent effectue une modification de ce type, AMS met à jour la disponibilité des agents concernés dans son index. AMS décide quand il le souhaite de désenregistrer un agent après un certain temps d'indisponibilité.

Afin de prendre en compte la présence des agents d'une nouvelle constellation, nous ajoutons un comportement à l'**agentBridge**. En effet, l'AMS d'une constellation A n'est pas au courant des agents présents dans une constellation B et inversement, au moment de leurs ralliements par un bridge. Nous proposons alors que l'agentBridge, une fois qu'il a établi le lien entre les constellations, effectue une demande de la liste des agents enregistrés, aux AMS respectifs. Une fois leurs réponses obtenues, l'agentBridge effectue sur le topic `ACC/AMS/`, autant d'enregistrements que d'agents initialement enregistrés. Ainsi, tous les AMS reliés ne disposant pas de l'enregistrement des agents pourront les indexer. Les AMS ayant déjà indexé ces agents répondront par un message d'impossibilité d'indexation, ne gênant pas les autres.

De plus, lorsque l'**agentPing** détecte la perte d'un lien avec une constellation, il notifie son AMS de la non-disponibilité de cette constellation. AMS met alors à jour son index en renseignant tous les agents de cette constellation comme étant indisponibles. Après un temps défini par l'utilisateur, AMS peut faire le choix de désenregistrer les agents concernés.

Nous noterons que par ces derniers ajouts, des constellations d'IoT-a peuvent désormais se découvrir, se relier selon plusieurs stratégies, connaître les agents disponibles et évoluer selon l'arrivée ou le départ d'autres constellations. Reprenons notre exemple dans le domaine de la robotique pour illustrer à nouveau notre approche, mais cette fois prenons le point de vue de la décentralisation collective. Des robots footballeurs évoluent sur un terrain. Chacun d'entre eux porte en lui une constellation autonome. Au début du match, ils se reconnaissent et se relient dynamiquement les uns aux autres par des bridges. Durant le match, tous les agents partagés entre les robots évoluent de concert pour optimiser le style de jeux. A un moment, l'un d'entre eux rencontre une difficulté et l'arbitre autorise son remplacement. Un nouveau robot est alors introduit, en remplacement de l'ancien. Tous les autres membres de l'équipe ont détecté le départ de l'ancien et sont sur le point de se

relier au nouveau, grâce à leur bridge dynamique. Pour le moment, les agents de l'ancien robot sont renseignés comme indisponibles auprès de l'équipe. Le nouveau robot est ensuite dynamiquement relié aux autres et peut s'intégrer à la partie. Tous ses agents sont agglomérés avec les autres et après quelques minutes, les agents de l'ancien robot sont désenregistrés, car ils ne sont plus utiles.

4.2.4 | *Modèle d'une constellation d'IoT-a*

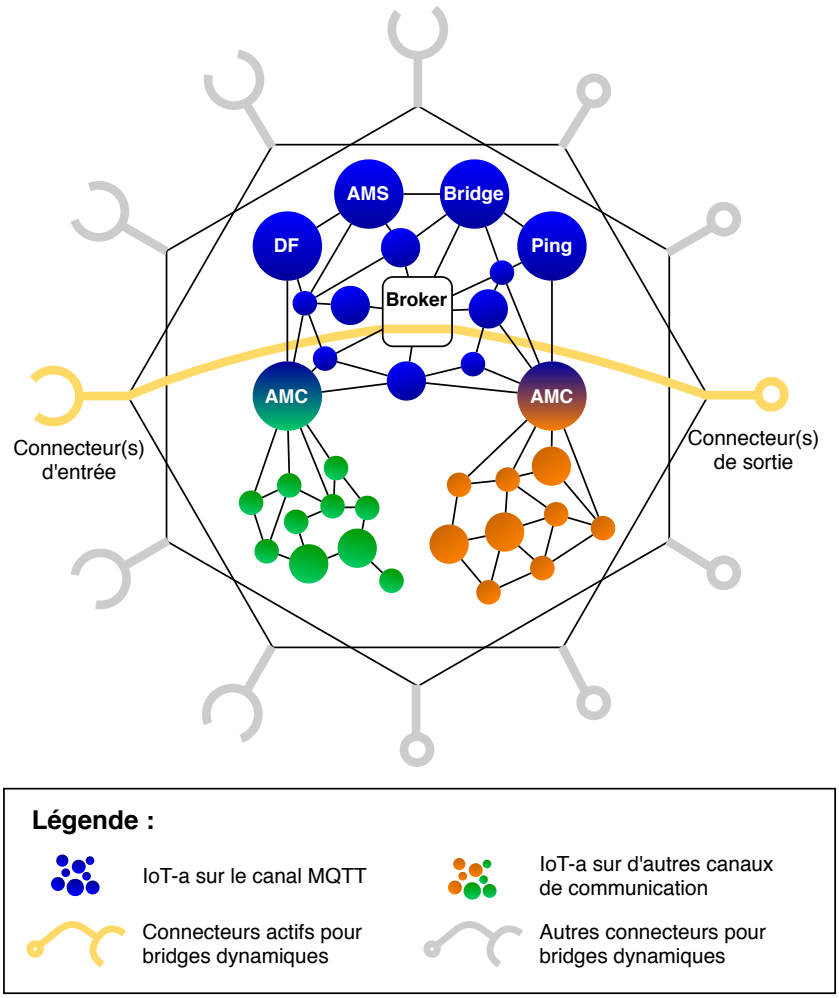


FIGURE 4.25 – Modèle d'une constellation d'IoT-a

L'association des éléments présentés dans les sections précédentes permet de dresser le modèle d'une constellation, illustré figue 4.25. Aussi, elle est définie comme étant un regroupement autonome d'IoT-a représentés par les multiples cercles de couleur. Ceux-ci sont de nature hétérogène tant par leur architecture, leur emplacement matériel ou encore, leur canaux de communication que nous représentons sous différentes couleurs dans notre modèle.

Une constellation permet une centralisation locale des IoT-a qui la constituent. Les agents de plate-forme AMS et DF permettent d'administrer et réguler la population et leurs comportements. Les agents AMC assurent le relais des communications entre des agents ne disposant pas du canal META et le reste de la population. Le canal META permet, d'ailleurs, l'harmonisation des communications et ouvre la possibilité à tous les agents d'échanger sur le même plan quelque soit leur emplacement physique. Toutes les communications sur le canal META s'effectuent au travers d'un unique broker par constellation.

Le choix du protocole MQTT comme technologie pour le canal META, ouvre un champ de possibilités pour ce qui est de la décentralisation collective des constellations d'IoT-a. En effet, ce protocole nous permet de proposer la notion de bridge dynamique, symbolisée dans la figure 4.25 par les deux types de connecteurs d'entrées ou de sorties. Cette particularité permet la mise en relation de nos constellations et fait évoluer la norme de la FIPA pour permettre aux agents d'évoluer sans conflits.

4.3 | Synthèse

Nous avons présenté dans ce chapitre nos travaux concernant l'IoT-a. Nous avons défini son modèle à un niveau microscopique et vu en détail les éléments nécessaires à la constitution d'un IoT-a. Une réflexion sur différentes configurations permet d'appréhender les niveaux matériels et logiciels dans lesquels ce modèle s'inscrit. Nous nous sommes concentrés ensuite sur l'organisation de ces multiples IoT-a que nous définissons sous le nom de constellation d'IoT-a et qui introduit nos deux notions de centralisation locale et décentralisation collective. La première décrit le fonctionnement des communications internes à une constellation alors que l'autre s'attache aux interactions de groupe à groupe. Avec la proposition de l'Agent Mediation Control, les IoT-a intégrant des protocoles de communication discordants sont désormais capables de se comprendre et d'interagir avec l'ensemble de leurs congénères. La décentralisation collective, quant à elle, se concentre sur les liens inter constellations en prenant compte des questionnements soulevés par le domaine de la résilience dans les systèmes critiques. En réponse, nous proposons des bridges dynamiques, administrés par de nouveaux agents Ping et Bridge, responsables de la mise en place de topologies de liens entre les constellations. Enfin, nous présentons à un niveau macroscopique notre modèle d'une constellation d'IoT-a, dotée de l'ensemble des propriétés exposées.

Nous pouvons désormais reprendre nos hypothèses afin de mettre en perspectives nos travaux présentés dans ce chapitre.

Rappel de l'hypothèse 1

L'intégration d'un ou plusieurs agents au sein d'un objet connecté, répartis à différents niveaux matériels ou logiciels, permet la réponse à des besoins applicatifs spécifiques.

Nous avons proposé, dans notre modèle de l'IoT-a, qu'un objet connecté puisse intégrer un ou plusieurs agents. Lors de notre réflexion sur les différentes configurations possibles de l'IoT-a, nous avons établi qu'un objet peut disposer d'agents matériel ou logiciel en fonction de ses caractéristiques. Nous avons de plus, spécifié que ces configurations sont cumulables selon les performances des objets. Ainsi, de multiples agents peuvent être mis en place au sein d'un objet connecté. Ils permettent alors, le découpage en entités autonomes responsables des fonctions à réaliser pour un besoin applicatif. La mise en place de ces agents permet la réponse à des problèmes complexes, car des algorithmes centralisés sont plus complexes à mettre en œuvre, à déployer et à maintenir. Le défi d'une telle approche est la communication permettant l'interaction entre tous les agents. Pour cela, nous avançons notre première notion de centralisations locales. Grâce à cela, les agents répartis sur différents niveaux matériels ou logiciels, peuvent communiquer et faciliter d'autant la réponse à des besoins applicatifs bas-niveau ou temps réel.

Aussi, avant d'observer les mises en œuvre permettant d'attester de la véracité de l'hypothèse 1 dans notre chapitre dédié aux expérimentations, nous pouvons valider théoriquement les éléments suivants : nous proposons un modèle permettant la mise en place d'agents à plusieurs niveaux matériels ou logiciels, au sein des objets connectés, et la littérature confirme que le paradigme multi-agents permet de répondre à des applications complexes en facilitant les démarches des architectes, développeurs et des intégrateurs.

Rappel de l'hypothèse 2

Une organisation en constellations des IoT-a permet leur adaptation au contexte matériel ou logiciel des objets connectés et ce quelque soit le cadre applicatif. La centralisation locale d'agents donne lieu à des interactions intra-objet alors que la décentralisation collective permet des échanges inter-objets.

Cette seconde hypothèse se concentre sur les aspects de communication et les liens entre les agents. En l'occurrence, notre travail sur l'AMC et la centralisation locale montre l'intérêt d'une organisation permettant l'homogénéisation des échanges entre les agents. Le cadre applicatif impose l'utilisation de certains types d'objets disposants chacun de leurs caractéristiques. L'intégration d'IoT-a bénéficie d'AMC permettant de les interfacer dans le cas où ils ne pourraient pas communiquer sur le canal META propre aux interactions des agents plus haut niveau. Dans un second temps, notre proposition concernant la décentralisation collective apporte une solution pour des échanges dynamiques de données inter-objets et offre un espace commun de communication pour l'ensemble des agents, quelque soit leur niveau matériel.

Bibliographie

[Allali et al., 2017] Allali, J., Deguillaume, L., Fabre, R., Gondry, L., Hofer, L., Ly, O., N'Guyen, S., Passault, G., Pirrone, A., and Rouxel, Q. (2017). Rhoban football club : Robocup humanoid kid-size 2016 champion team paper. In Behnke, S., Sheh, R., Sarıel, S., and Lee, D. D., editors, *RoboCup 2016 : Robot World Cup XX*, pages 491–502, Cham. Springer International Publishing.

[Barbalace et al., 2008] Barbalace, A., Luchetta, A., Manduchi, G., Moro, M., Soppelsa, A., and Taliercio, C. (2008). Performance comparison of vxworks, linux, rtai and xenomai in a hard real-time application. *Nuclear Science, IEEE Transactions on*, 55 :435–439.

[Bellifemine et al., 2001] Bellifemine, F., Poggi, A., and Rimassa, G. (2001). Developing multi-agent systems with a fipa-compliant agent framework. *Softw., Pract. Exper.*, 31 :103–128.

[Bellifemine et al., 2007] Bellifemine, F. L., Caire, G., and Greenwood, D. (2007). *Developing Multi-Agent Systems with JADE (Wiley Series in Agent Technology)*. John Wiley & Sons.

[Butzin et al., 2016] Butzin, B., Golatowski, F., and Timmermann, D. (2016). Microservices approach for the internet of things. In *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–6.

[Collier et al., 2019] Collier, R. W., O'Neill, E., Lillis, D., and O'Hare, G. (2019). Mams : Multi-agent microservices. In *Companion Proceedings of The 2019 World Wide Web Conference*, WWW '19, pages 655–662, New York, NY, USA. Association for Computing Machinery.

[Dragoni et al., 2018] Dragoni, N., Lanese, I., Larsen, S., Thordaland Mazzara, M., Mustafin, R., and Safina, L. (2018). Microservices : How to make your application scale. In Petrenko, A. K. and Voronkov, A., editors, *Perspectives of System Informatics*, pages 95–104, Cham. Springer International Publishing.

[FIPA, 2002a] FIPA, C. (2002a). Fipa agent message transport protocol for http specification - sc00084f. *Transport WG*.

[FIPA, 2002b] FIPA, C. (2002b). Fipa agent message transport service specification - sc00067f. *FIPA TC B*.

[FIPA, 2004] FIPA, C. (2004). Fipa agent management specification - sc00023k. *FIPA TC B*.

[Greenwood et al., 2007] Greenwood, D., Lyell, M., Mallya, A., and Suguri, H. (2007). The ieee fipa approach to integrating software agents and web services. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '07, New York, NY, USA. Association for Computing Machinery.

[Jamont and Occello, 2007] Jamont, J.-P. and Occello, M. (2007). Designing embedded collective systems : The DIAMOND multiagent method. In *IEEE International Conference on Tools with Artificial Intelligence - ICTAI 07*, pages 91–94, Patras, Greece. IEEE Computer Society.

[Krivic et al., 2017] Krivic, P., Skocir, P., Kusek, M., and Jezic, G. (2017). Microservices as agents in iot systems. In Jezic, G., Kusek, M., Chen-Burger, Y.-H., Jessicaand Howlett, R. J., and Jain, L. C., editors, *Agent and Multi-Agent Systems : Technology and Applications*, pages 22–31, Cham. Springer International Publishing.

[Light, 2017] Light, R. (2017). Mosquitto : server and client implementation of the MQTT protocol. *The Journal of Open Source Software*, 2(13).

[Nazir Raja et al., 2008] Nazir Raja, M. A., Farooq Ahmad, H., Suguri, H., Bloodsworth, P., and Khalid, N. (2008). Soa compliant fipa agent communication language. In *2008 First International Conference on the Applications of Digital Information and Web Technologies (ICADIWT)*, pages 470–477.

[Nurmi, 2007] Nurmi, J. (2007). *Processor design : System-on-chip computing for ASIDs and FPGAs*. Springer.

[Poslad, 2007] Poslad, S. (2007). Specifying protocols for multi-agent systems interaction. *ACM Trans. Auton. Adapt. Syst.*, 2(4).

[Pujolle, 2006] Pujolle, G. (2006). Les réseaux. chapter Partie X, Le contrôle et la gestion, pages 737–866. Eyrolles, cinquième edition.

[Renault and Carlier, 2016] Renault, V. and Carlier, F. (2016). Triskell3S, une plate-forme embarquée multi-agents pour les IoT-a. In *Journées Francophones sur les Systèmes Multi-Agents (JFSMA 2016)*, pages 181–190, Rouen, Saint-Martin-du-Vivier, France. Fabien Michel and Julien Saunier.

[Reza Naji et al., 2004] Reza Naji, H., Wells, B. E., and Etzkorn, L. (2004). Creating an adaptive embedded system by applying multi-agent techniques to reconfigurable hardware. *Future Generation Computer Systems*, 20(6) :1055–1081. Computational science of lattice Boltzmann modelling.

[Ribeiro et al., 2008] Ribeiro, L., Barata, J., and Mendes, P. (2008). Mas and soa : Complementary automation paradigms. In Azevedo, A., editor, *Innovation in Manufacturing Networks*, pages 259–268, Boston, MA. Springer US.

[Sabt et al., 2015] Sabt, M., Achemlal, M., and Bouabdallah, A. (2015). Trusted execution environment : What it is, and what it is not. In *2015 IEEE Trustcom/BigDataSE/ISPA*, volume 1, pages 57–64.

[Schmitt et al., 2018] Schmitt, A., Carlier, F., and Renault, V. (2018). Dynamic bridge generation for iot data exchange via the mqtt protocol. In *The 9th International Conference on Ambient Systems, Networks and Technologies (ANT 2018)*, volume 130, pages 90–97, Porto, Portugal.

[Takabatake, 2011] Takabatake, T. (2011). Simulations of noc topologies for generalized hierarchical completely-connected networks. In *6th International Workshop on Reconfigurable Communication-Centric Systems-on-Chip (ReCoSoC)*, pages 1–5.

[Van Moergestel et al., 2016] Van Moergestel, L., Berg, M., Knol, M., Paauw, R., Voorst, K., Puik, E., Telgen, D., and Meyer, J.-j. (2016). Internet of smart things, a study on embedding agents and information as a service. *ICAART Proceedings*, 1 :102–109.

[Vincenzo et al., 2006] Vincenzo, N., Concetto, S., and Corrado, S. (2006). Software agents for autonomous robots : the eurobot 2006 experience. In *Proceedings of the 7th WOA 2006 Workshop, From Objects to Agents*.

[Vural et al., 2017] Vural, H., Koyuncu, M., and Guney, S. (2017). A systematic literature review on micro-services. In Gervasi, O., Murgante, B., Misra, S., Borruso, G., Torre, C. M., Rocha, A. M. A., Taniar, D., Apduhan, B. O., Stankova, E., and Cuzzocrea, A., editors, *Computational Science and Its Applications – ICCSA 2017*, pages 203–217, Cham. Springer International Publishing.

Chapitre 5

Cadre applicatif

Ce chapitre se concentre sur un cas d'usage de notre approche IoT-a dans le domaine industriel. Cette thèse a été liée par un contrat CIFRE avec l'entreprise SARP-VEOLIA. Son besoin est d'obtenir un système permettant une visualisation dynamique d'informations pour faciliter l'activité de certaines équipes de collaborateurs. Face à la diversité des activités du groupe, cet outil doit être modulaire, robuste et évolutif. L'approche IoT-a a été mise en œuvre pour répondre à ces attentes.

Nous allons présenter dans un premier temps le contexte de l'entreprise afin de mieux appréhender leurs attentes. Nous décrivons ensuite la nature des informations à traiter par le système et de quel façon celles-ci doivent être visualisées. L'approche IoT-a deviendra alors le support de ce cas d'usage et nous analyserons comment nous l'avons mis en œuvre.

5.1 | Contexte industriel

L'entreprise SARP faisant partie du groupe VEOLIA, est spécialisée dans la gestion de déchets d'assainissement et l'entretien de canalisations. L'assainissement se réalise en plusieurs étapes. Dans un premier temps, il consiste en la mise en œuvre de techniques adaptées pour assurer la collecte des eaux usées et pluviales. Dans un second temps, il permet l'épuration de ces eaux avant leur rejet dans le milieu naturel ou de leur acheminement auprès d'usines de traitement des déchets. L'entreprise est constituée de plus de 120 agences sur le territoire français. Elles sont de tailles variées et ne disposent pas toutes du même fonctionnement selon les régions. Cela s'explique en partie par le tissu industriel et l'administration des collectivités qui diffèrent selon les localités.

Malgré la diversité des agences, nous avons constaté des invariants dans le fonctionnement du groupe. Les activités se déroulent au travers de chantiers faisant appel à trois types de ressources. La première est la ressource humaine, le personnel, constituée d'opérateurs de chantier qui disposent de certaines qualifications en fonction des formations suivies ou de l'ancienneté. La seconde se compose de véhicules. Chaque agence en dispose de plusieurs. Certains d'entre eux sont spécialisés dans un spectre d'interventions alors que d'autres sont plus polyvalents et proposent d'intervenir sur plusieurs types de chantiers. Enfin, la dernière ressource est matérielle. Le matériel est spécifique et nécessaire pour chaque chantier. Il comprend des outils comme des caméras, des tuyaux, des bennes, etc.

La disponibilité des ressources adéquates à un chantier est nécessaire pour intervenir. D'autres données contraintes sont aussi à prendre en compte pour pouvoir intervenir chez un client. La nature du déchet et sa quantité, la distance du lieu d'intervention, les contraintes horaires du client, la sécurité à appliquer sur le chantier, le trafic dans la zone d'intervention, la configuration du lieu d'intervention, etc. Toutes ces données, ainsi que les ressources énoncées ci-dessus doivent être orchestrées au mieux, de façon à obtenir un maximum de sécurité pour les opérateurs en intervention et organiser la tournée des véhicules au plus juste pour qu'ils puissent répondre à un maximum d'interventions et ainsi dégager plus de rentabilité. Au-delà des données techniques, viennent donc se rajouter les données de rendement économique et de coût d'intervention interne à l'entreprise.

L'organisation des chantiers passe par la mise en place d'un planning complexe, sans cesse modifié par de nombreux acteurs : assistants, responsables techniques, directeurs, commerciaux, etc. La planification est l'une des tâches les plus complexes aujourd'hui dans l'entreprise, elle requiert un savoir-faire fort de plusieurs années d'expérience afin d'agencer correctement tous les paramètres nécessaires.

Le service d'exploitation est responsable de l'organisation quotidienne des tournées. Dans ce service, en fonction de la taille de l'agence, un ou plusieurs assistants d'exploitation se chargent de répondre aux appels téléphoniques et prennent des notes concernant les demandes clients. Ils sont chargés de trouver un créneau dans le planning puis d'y faire correspondre les personnels et matériels nécessaires. Les assistants sont supervisés par un responsable d'exploitation ayant une expertise technique du terrain et des coûts d'intervention. Les différents flux intervenant dans l'organisation du planning d'une agence sont synthétisés dans la figure 5.1.

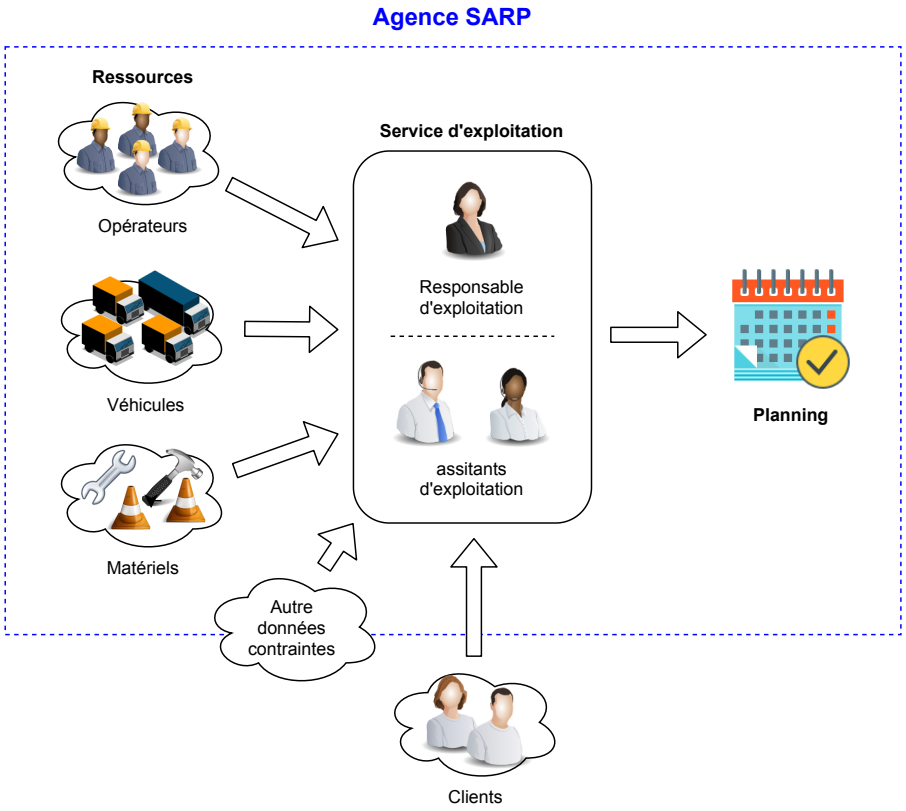


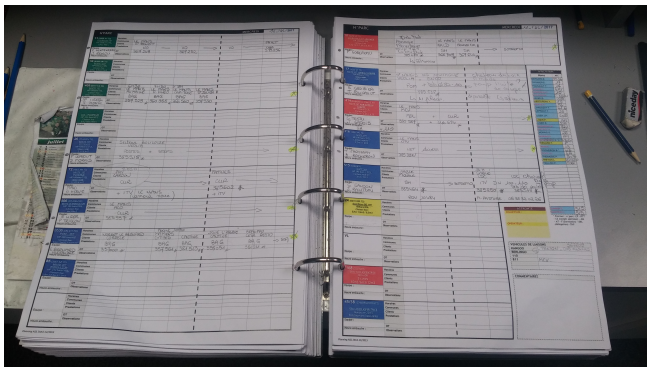
FIGURE 5.1 – Flux nécessaires à l'organisation du planning dans la SARP

La planification est une tâche dynamique. Elle évolue tout au long de la journée et prend en compte les nombreux aléas qui apparaissent au fur et à mesure que les chantiers avancent. Les opérateurs intervenant sur les chantiers font état, plusieurs fois dans la journée, de leur avancement et des difficultés qu'ils ont pu rencontrer. Ces difficultés sont parfois à l'origine du report d'une intervention et font systématiquement intervenir le responsable d'exploitation. Le responsable et ses équipes déterminent le problème et statuent sur la solution à apporter. Dans certains cas, le planning s'en trouve bouleversé et doit être adapté.

En parallèle de l'exploitation, le service commercial est chargé du démarchage et de la prise en charge de clients nécessitant une intervention non standardisée. Il est entendu par le terme *intervention non standardisée*, toute intervention ne pouvant être traitée par un assistant d'exploitation au travers d'un unique appel téléphonique. Les commerciaux se déplacent sur les lieux du chantier pour échanger avec le client et apprécier, sur le terrain, les besoins en matière de sécurité, matériels et effectifs. Une fois le devis établi par le commercial et validé par le client, l'intervention doit être ajoutée au planning. Ce type d'intervention peut être, dans certains cas, complexe à organiser, car réparti sur plusieurs jours consécutifs, dans des lieux nécessitant des accès particuliers (établissement pénitencier, aéroports, zones militaires, etc.), ou ayant une récurrence dans le temps (tous les mois, tous les trimestres, etc.). Là encore, les contraintes sont multiples et elles s'ajoutent aux missions du service d'exploitation pour l'organisation du planning.

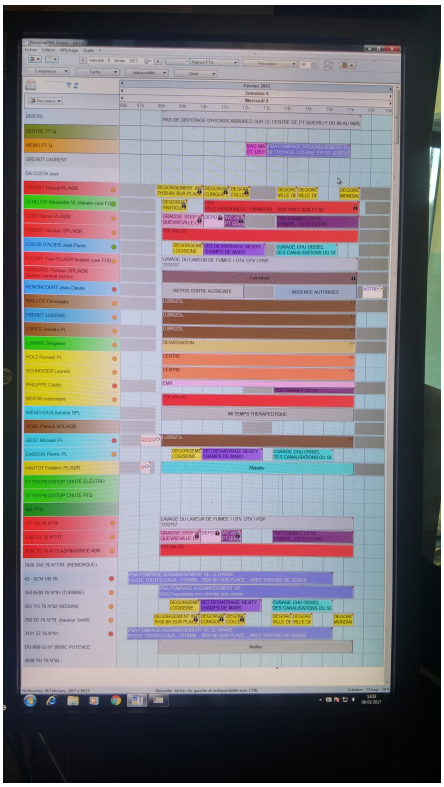
La planification est donc au carrefour de nombreuses contraintes. Elle prend en compte des modifications à différentes échelles de temps, à l'heure, à la journée, à la semaine et même au mois ou au trimestre. Les interventions doivent être organisées au plus juste de façon à pouvoir en dégager un maximum de rentabilité. La casse de matériel, les congés du personnel, les intempéries ne sont que quelques-uns des nombreux paramètres à prendre en compte pour une organisation optimale.

Quels sont les outils actuels permettant à l'exploitation de répondre à une tâche aussi complexe que la planification ? Dans toutes les agences, la planification s'effectue autour d'un outil capable de centraliser toutes les interventions avec les noms des opérateurs, les véhicules associés et le matériel nécessaire. La nature de l'outil diffère en fonction des agences et peut se matérialiser sous la forme d'un cahier papier (figure 5.2a), un tableau à fiches en T (figure 5.2b), des tableurs numériques (figure 5.2c) ou encore une association de plusieurs techniques manuscrites et numériques.



(a) Cahier papier

(b) Tableau à fiches en T



(c) Tableur

FIGURE 5.2 – Trois types de planning utilisés dans trois agences SARP différentes

Le souhait du groupe SARP est de numériser l'activité de planification pour les raisons listées ci-dessous :

Arrêter l'archivage papier trop coûteux.
Lorsque la saisie du planning est manuscrite, une copie doit-être conservée pour une consultation ultérieure des activités de l'agence. Il peut être nécessaire à un membre d'exploitation de revenir sur l'organisation d'un ancien créneau pour gagner du temps dans une organisation future.

Diminuer le nombre de saisies de la part des assistants d'exploitation.
Pour chaque intervention, l'information doit être saisie sur plusieurs supports : le planning, la fiche d'intervention pour l'opérateur, la facturation, etc. Dans le cas d'un système informatique centralisé, l'information est automatiquement ventilée sur les différents supports dès le positionnement d'un créneau sur le planning et peut évoluer.

Faciliter la modification et la saisie de créneaux complexes.
Les décisions pour la répartition d'un créneau sur le planning évoluent tout au long d'une journée ou de la semaine. Comme nous l'avons vu, certains aléas peuvent modifier la réalisation dans les temps d'un chantier. Il est donc nécessaire de modifier rapidement le planning dans ce cas. De plus, les commerciaux sont aussi générateurs de créneaux. Les clients souscrivent à des contrats qui font intervenir des récurrences, la saisie manuscrite est alors complexe à effectuer sur plusieurs mois.

Automatiser la disponibilité des ressources humaines et techniques.
Connaître la disponibilité des ressources au moment de la planification est la principale demande du personnel d'exploitation. Un opérateur absent, un camion en panne, un matériel cassé, tous ces éléments sont à prendre en compte pour mettre en place les créneaux dans le planning.

La SARP dispose depuis peu d'un nouvel outil de gestion intégrée. JASON est un *Entreprise Ressource Planning* (ERP) développé en interne depuis plusieurs années et qui prend en compte un grand nombre d'aspects qu'impose le métier de collecte de déchets. Ce logiciel intervient dans plusieurs filiales du groupe Véolia et plus particulièrement dans la SARP depuis 2018. JASON permet le suivi complet d'un client depuis la saisie d'un devis jusqu'à l'édition de la facture et l'archivage des informations quotidiennes. Il est composé de plusieurs modules dont certains sont encore en cours de développement, comme un module de planification ayant pour but de répondre aux raisons listées ci-dessus. Aussi, JASON est un élément centralisateur de nombreuses informations nécessaires à la planification et permettra demain de planifier des interventions.

De plus, la SARP met en place depuis quelques mois, un outil pour les opérateurs sur les chantiers. Ces derniers se voient équipés d'une tablette afin de connaître les différentes interventions qu'ils ont à effectuer dans la journée. Cet outil leur permet aussi de remplir les compte-rendus d'interventions et de saisir les différents retours d'expériences concernant le chantier. Étant donné l'environnement dans lequel les opérateurs évoluent, il est nécessaire qu'ils disposent d'un outil capable de fonctionner sans réseau et résistant à des conditions de travail difficiles (eau, boue, poussière, etc.). L'application présente sur la tablette et centralisant les différentes données est aussi développée par SARP et ce nomme MySARP. Elle permet la lecture d'informations saisies en agence par les équipes d'exploitation, et d'enregistrer localement les informations saisies par l'opérateur. Lorsque l'opérateur est de nouveau dans une zone disposant d'un réseau suffisant, l'application synchronise les données saisies avec un serveur et les met à disposition de l'agence. En fin de journée, quand toutes les interventions sont terminées, une synchronisation automatique s'opère entre les serveurs MySARP et JASON pour l'archivage.

Les équipe d'exploitation font régulièrement face à des demandes urgentes. Afin d'y répondre au plus vite, il leur est nécessaire de connaître la position des véhicules durant la journée. C'est pourquoi chacun d'entre eux est équipé d'un émetteur sans fil longue portée, capable de partager leur position GPS ainsi que des données concernant le véhicule (taux de remplissage de la cuve, utilisation des outils de pompage, etc.). Les informations issues des véhicules sont centralisées sur un serveur et est mises à disposition des agences au travers d'une page web. Chaque agence dispose d'un code d'accès lui permettant de suivre son parc de véhicules en direct à tout instant de la journée. Les exploitants disposent ainsi d'une information précieuse, qu'est la position d'un véhicule par rapport à la position du client ayant une demande urgente. L'assistant est capable de déduire un temps d'intervention et donc de proposer rapidement une solution au client.

Nous pouvons dès lors, constater un nombre important d'informations à manipuler dans le métier de l'exploitation. Aujourd'hui, la prise en compte de tous ces paramètres passe par un affichage de nombreuses interfaces informatiques réparties sur des supports épars. Certaines agences affichent la géolocalisation de leurs véhicules sur un téléviseur, alors que d'autres la rendent disponible dans une fenêtre de leur ordinateur. Une page JASON permet de connaître la présence des opérateurs, et une autre, les compétences de chaque opérateur. Enfin, c'est bien souvent par échanges oraux que les assistants d'exploitation apprennent la panne d'un véhicule ou la casse d'un matériel qu'ils reportent sur un document informatique de type tableur. La multiplication des informations et l'hétérogénéité de leurs supports font que l'entreprise perd en rentabilité. C'est pourquoi la SARP souhaite mettre en place un outil permettant la visualisation de tous ces éléments, pour le service d'exploitation.

5.2 | Objectifs d'une surface de visualisation

Devant le nombre important de visuels et la quantité d'informations à prendre en compte pour accomplir les tâches d'exploitation, la SARP a jugé nécessaire de disposer d'une surface globale de visualisation de tous ces paramètres pour faciliter la tâche de planification.

Les 120 agences du groupe n'ont pas toutes les mêmes activités, la même taille, ou les mêmes gestions internes. Il est donc nécessaire de proposer un outil capable de s'adapter, quels que soient les paramètres qui régissent une agence. L'outil doit pouvoir afficher un grand nombre de visuels de façon dynamique afin de correspondre aux activités d'une agence au fur et à mesure de la journée. Afin d'analyser et comprendre les besoins du personnel au sein d'une agence, nous avons travaillé en concertation avec un étudiant de Master 2 spécialisé dans les études en psychologie du travail et des organisations. À l'issue d'un stage de six mois en immersion dans différentes agences, l'étudiant a dressé un ensemble d'indicateurs et de préconisations concernant les interfaces à diffuser pour le personnel d'exploitation et les autres membres de l'entreprise [Boulongne, 2018].

Les principales interfaces à visualiser sont les suivantes :

— planning informatisé,
— disponibilité des ressources,
— géolocalisation des véhicules,
— flux vidéo,
— pages Web diverses.

Trois contraintes sont établies par l'entreprise :

La surface d'affichage doit permettre un travail collaboratif. Plusieurs personnes doivent pouvoir se tenir devant la surface d'affichage afin d'échanger et créer de l'émulation.

La surface d'affichage doit s'adapter aux configurations d'une agence. Le nombre d'écrans et leurs tailles doivent permettre une mise en place dans une petite agence (quelques véhicules), mais aussi dans une grande agence (plusieurs dizaines de véhicules).

La surface d'affichage doit être dynamique et permettre différents scénarios d'affichage. L'affichage doit évoluer en fonction du nombre de personnes, des indicateurs à visualiser et de la nature de la situation de travail.

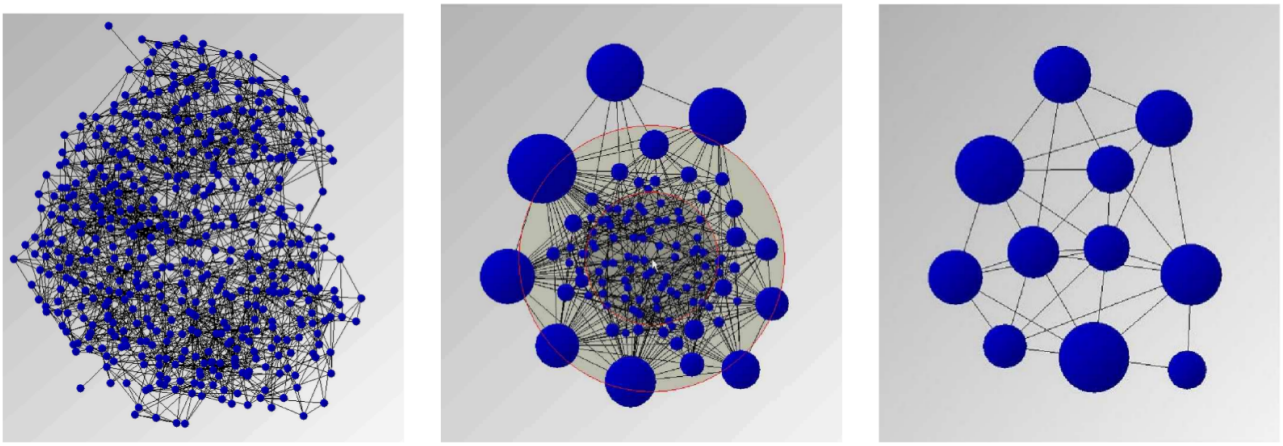
5.3 | Travaux dans le domaine de la visualisation

L'étude de la littérature, dans le domaine de la visualisation d'informations, nous permet de prendre du recul face aux contraintes énoncées précédemment. Tout d'abord, essayons de mieux comprendre ce que représente l'information et comment la recherche scientifique permet de la visualiser et la manipuler.

Depuis les années 1990 de très importantes quantités d'informations sont stockées dans des parcs de bases de données ce qui a donné naissance à un nouveau domaine : le *Big Data*. Certains travaux le définissent comme étant l'ensemble des données brutes qui ne peuvent être transformées ou analysées en utilisant les outils et applications traditionnelles de traitement de l'information [Zikopoulos et al., 2012]. Le volume d'informations brutes est aujourd'hui conditionné par des entreprises et des collectivités capables de collecter toutes données générées par des utilisateurs, des services ou des capteurs. Il est estimé que des entreprises comme Twitter ou Facebook collectent plus de 7 à 10 To de données par jour. Leur variété est due aux multiples vecteurs de génération comme les échanges de documents, les traces de logiciels, les données marketings clients, les e-mails, ou encore les mesures et analyses de l'IoT.

La complexité et la quantité de ces données nécessitent une approche spécifique afin d'en retirer de l'information utile. Le *Data Mining* rassemble les domaines conjoints d'exploration, tri, traitement, analyse et restitution de données brutes. Dans son livre [Han et al., 2011], l'auteur définit le data mining comme étant un *"processus d'exploration de données qui consiste à découvrir des tendances et des connaissances intéressantes à partir de grandes quantités de données"*. Lorsque les outils du big data font émerger des informations exploitables et intéressantes à analyser, un troisième domaine vient alors appuyer l'analyse par les utilisateurs, nous parlons de la *visualisation*.

La visualisation se concentre sur la présentation graphique des informations pour qu'elles puissent être analysées et manipulées par des acteurs humains. Ainsi, elle s'intéresse, entre autre, aux couleurs, aux formes, aux échelles et aux liens que peuvent avoir les données entre elles. La littérature foisonne de modèles et d'outils de visualisation. Par exemple, Koenig avance dans sa thèse, une méthode "focus+contexte" pour la visualisation de graphes [Koenig, 2009]. Son approche consiste en la visualisation de données locales avec un maximum de détails tout en conservant une vue globale du contexte dans lequel ces données sont présentes. Disposant de graphes de liens entre personnes ou sociétés, il génère dans un premier temps une hiérarchie entre les nœuds en utilisant le principe de degré d'abstraction (Degree Of Abstraction, DOA) pour agréger les nœuds entre eux. Cette technique prend en compte la distance en pixels de chaque nœud et agrège les plus proches. Son procédé consiste alors à appliquer une méthode de *fish eyes* pour moduler localement le paramètre de DOA afin de bénéficier d'une vue schématique du contexte, avec des nœuds agrégés, et une vue détaillée du focus. Il est possible de voir sur la figure 5.3 l'approche présentée avec à gauche (a) le graphe brut, à droite (c) le graphe abstrait et au centre (b) la proposition de focus+contexte permettant une vue détaillée du graphe en fonction de la position de la souris de l'utilisateur sur le graphe. Dans cette étude, la visualisation permet d'analyser un



(a) Graphe d'origine (b) Vue "focus+contexte" (c) Graphe abstrait

FIGURE 5.3 – Travaux sur la vue "focus+contexte" proposé par [Koenig, 2009]

grand nombre de liens entre des entités hétérogènes et apporte une aide à la décision.

Le méta modèle avancé dans [Ware, 2012] et illustré par la figure 5.4, modélise les différentes étapes que nous avons présentées précédemment. Tout à gauche sont représentées les données brutes, c'est le big data. Au centre, le bloc *"Data transformations"* symbolise les outils du data mining. Accolés à sa droite, nous retrouvons les outils de visualisation permettant de faire l'interface entre les données et les utilisateurs. Nous observons des boucles de rétroaction émanant de l'utilisateur vers chacun des niveaux du modèle. Elles permettent de manipuler les informations à plusieurs niveaux. Le premier, offre l'opportunité à l'utilisateur, de manipuler la visualisation, il peut agrandir, tourner, sélectionner, changer de perspective, etc., afin de visualiser ce qu'il analyse. La seconde boucle lui permet de modifier les paramètres de traitement, il peut alors changer d'échelle, de lieu, de conditions, de protagonistes, etc. La dernière boucle autorise l'utilisateur à modifier des valeurs dans les jeux de données. Il peut ainsi comparer des visualisations avec différentes données. Enfin, le dernier point que nous pouvons noter dans ce modèle est la présence des contraintes de l'environnement social. L'analyste, devant une interface de visualisation, ne perçoit pas les informations qui lui sont exposées sans certains biais sociaux. Nous verrons, dans le cas de notre étude, que ce paramètre joue un rôle non négligeable durant un travail de groupe face à une surface de visualisation.

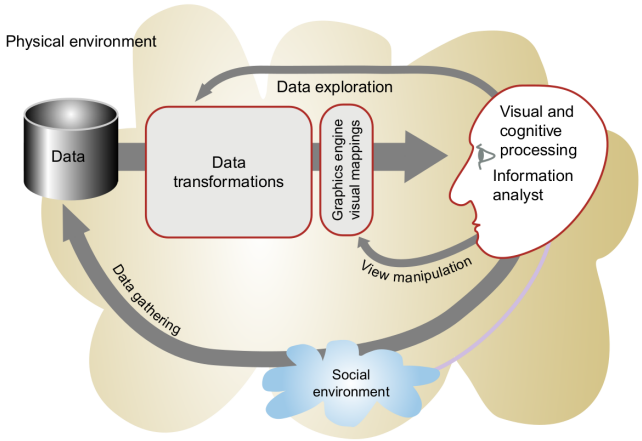


FIGURE 5.4 – Processus de visualisation de l'information, [Ware, 2012]

Nous proposons une synthèse du modèle développé par Ware, afin de le faire correspondre à notre cas d'étude. Aussi, nous retrouvons dans la figure 5.5 les trois premiers éléments que sont : le big data, le data mining et la visualisation. Néanmoins, nous avons matérialisé l'ensemble des boucles de rétroaction par la fonction de *Manipulation*, pouvant agir sur les données brutes ou transformées. En outre, nous proposons aussi l'ajout des éléments *Écran(s)* et *Élément(s) d'interaction(s)*. Cette proposition résulte des travaux de [Bethel et al., 2012], dans lesquels le processus de visualisation intègre aussi les outils matériels comme les écrans. Par conséquent, les éléments matériels nécessaires à la visualisation et aux interactions servent d'interfaces tangibles entre le monde virtuel des données et le monde réel de l'utilisateur.

Nous avons présenté dans les sections précédentes, les éléments que l'entreprise souhaite visualiser pour améliorer sa compétitivité. Différentes interfaces résumant ses activités sont mises à disposition et sont issues de logiciels internes à l'entreprise. Nous pouvons constater que les trois étapes principales, de big data, data mining et visualisation/manipulation, sont ici administrées par le groupe SARP et bénéficient de points d'entrée permettant les rétroactions. La nouveauté, face à la synthèse que nous avons proposée en figure 5.5, est la multiplication des flux de visuels et de manipulation. De plus, les approches de [Ware, 2012] et [Bethel et al., 2012] prennent en compte un seul utilisateur (ou un groupe restreint d'experts) face à la visualisation. Notre système a pour but de mettre à disposition une pluralité de visuels pour un ou plusieurs groupes d'utilisateurs.

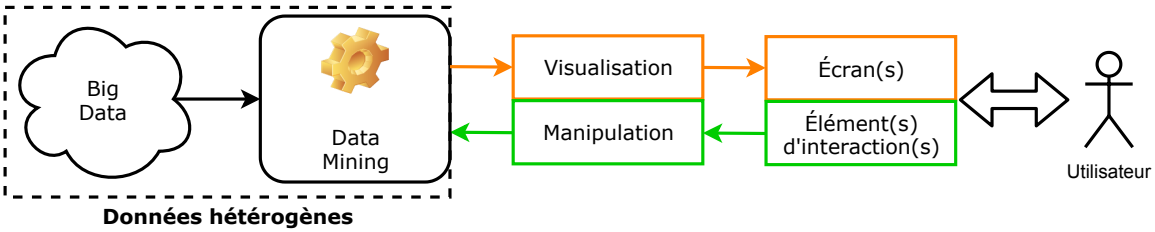


FIGURE 5.5 – Synthèse du modèle de visualisation

5.4 | Modèle 5M

Nous avons étudié dans les sections précédentes les modèles s'appliquant au domaine de la visualisation. Nous en avons proposé un résumé et avons relevé certains leviers possibles par rapport à l'approche que nous présentons ci-dessous.

Le modèle **5M** : **M**ulti-visualisations, **M**ulti-manipulations, **M**ulti-utilisateurs, **M**ulti-agents et **M**ulti-IoT-a offre une formalisation face aux défis de la visualisation d'une pluralité de visuels devant plusieurs groupes d'utilisateurs. Ce modèle, illustré figure 5.6, repose sur notre approche IoT-a et permet une gestion décentralisée des fonctions de visualisation et de manipulation.

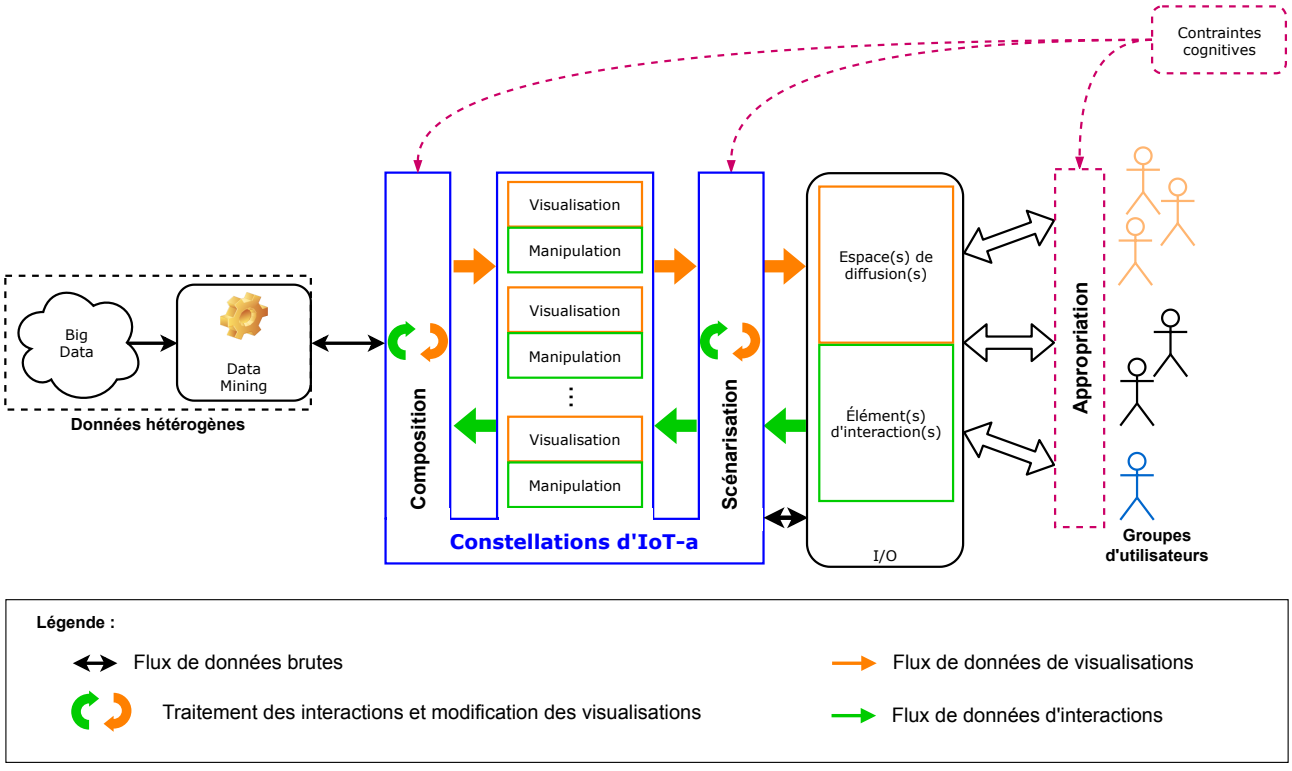


FIGURE 5.6 – Modèle 5M reposant sur l'approche IoT-a

5.4.1 | *Présentation du modèle*

La figure 5.6 illustre le modèle 5M. Tout à gauche, les éléments de big data et de data mining représentent les données hétérogènes traitées par une administration externe. Ces flux de données brutes sont visualisés et manipulés au travers d'interfaces utilisateur. En bleu, au centre, les constellations d'IoT-a forment un environnement commun de communication et de traitement de l'information. À leur gauche, la fonction de *composition* amalgame les éléments nécessaires à la mise en place de visuels interactifs. Ils constituent un vivier d'interfaces, visibles au centre, par la multiplication de blocs symbolisant la *Visualisation* et la *Manipulation* associée. À droite, la fonction de *scénarisation* permet l'affichage effectif de visuels sur les espaces de diffusion. Elle paramètre la taille, la position, le facteur de zoom, ou encore la priorité de chaque visuel par rapport au public présent devant les espaces de diffusion.

L'encadré noir représente le matériel associé aux fonctions administrées par les IoT-a. En l'occurrence, les espaces de diffusions peuvent être incarnés par tout type de matériel permettant la diffusion de flux d'images (écrans LCD, vidéoprojecteurs, dalles LED, etc.). Nous entendons par le terme *espace de diffusion*, un lieu dans lequel plusieurs utilisateurs ont les capacités de visualiser une pluralité de visuels au travers d'un ensemble de matériels de diffusion. Plusieurs espaces de diffusion sont possibles pour permettre à divers groupes d'utilisateurs de visualiser parallèlement des interfaces différentes.
Les éléments d'interaction sont liés aux espaces de diffusion et permettent de modifier les interfaces visualisées ou leurs contenus. Ils sont par exemple représentés par des caméras, une table surface, une commande vocale, des lecteurs de badge RFID, des téléphones portables, etc.

Enfin, le dernier élément de notre modèle est le public face aux espaces de travail. Plusieurs utilisateurs ont la possibilité de travailler simultanément avec les multiples interfaces à leur disposition. Au même titre que Ware, qui le souligne dans ses travaux [Ware, 2012], la prise en compte d'un public humain dans ce type de modèle fait nécessairement intervenir certaines contraintes cognitives. Bien que ce domaine sorte du cadre du sujet de cette thèse, nous intégrons des briques supplémentaires à notre modèle afin de les prendre en compte dans de futurs travaux, pour adapter les fonctions de composition et de scénarisation face à l'appropriation des utilisateurs devant de tels systèmes.

5.4.2 | *Gestion distribuée par IoT-a*

Les espaces de diffusion et les éléments d'interaction représentent une multitude d'éléments matériels autonomes. Par exemple, un espace de diffusion est constitué de plusieurs modules d'affichage agissant sur l'environnement réel au travers de la diffusion d'un flux d'images. Nous le détaillons ci-après. Les éléments d'interaction sont eux, des capteurs de l'environnement physique. Ils recueillent les demandes utilisateurs afin de modifier la visualisation.

Notre approche IoT-a permet donc de relier de façon homogène tous ces matériels. Nous les organisons sous la forme de constellations d'IoT-a afin de faire émerger des comportements de visualisation et manipulation traités collectivement et non centralisés dans un système de traitement global.

Module d'affichage

Un module d'affichage est un des constituants d'un espace de diffusion. Il est unique et n'appartient qu'à un seul espace de diffusion à la fois. Il est éventuellement mobile d'un espace à un autre et se compose de deux éléments principaux.

Le premier est un objet connecté intégrant une constellation d'IoT-a. Le système dispose de caractéristiques suffisantes pour traiter un flux d'images et organiser une constellation d'IoT-a. Pour cela, le recours aux SBC, comme nous l'avons présenté dans notre section 2.4.2, permet la mise en œuvre d'un système simple à déployer, compact et polyvalent. De plus, ces systèmes sont ouverts et offrent pour certains des modules de contrôle graphique (GPU), permettant le décodage de flux vidéo haute qualité.

Le second élément est le matériel de restitution de l'image. Différentes technologies sont envisageables en

fonction de l'application. Par exemple, un vidéoprojecteur permet de grandes surfaces d'affichage mais nécessite une faible luminosité dans la pièce où il est installé. Une dalle LED offre une forte luminosité mais ne propose que d'une résolution faible. Un écran LCD dispose d'une surface d'affichage lumineuse, avec une grande densité de pixels, mais fait naître un problème d'effet Bezel, autrement dit, de bords noirs entre plusieurs écrans mis les uns à côté des autres [De Almeida et al., 2012].

Un module d'affichage seul a un intérêt limité, la quantité d'information diffusée est faible et les interactions de travail entre les utilisateurs sont pauvres. La multiplication des modules d'affichage ouvre le champ des possibles. Un espace de diffusion est donc un ensemble de modules d'affichage contigus les uns aux autres et synchronisés. Les éléments de visualisation sont répartis sur un ou plusieurs modules. Notre proposition se rapproche en cela, des travaux sur les murs d'images, composant plusieurs matériels de diffusion pour augmenter la définition et la surface de visualisation [Sandstrom et al., 2003] [Liu and Anshus, 2009] [Liu et al., 2009] [Moreno and Gomes Soares, 2015]. Nos travaux diffèrent néanmoins de ces derniers par notre approche IoT-a qui décentralise le traitement plutôt qu'un système complexe et centralisé, bénéficiant d'une évolutivité limitée. Il offre aussi la perspective d'interactions présentielles et distancielles entre les utilisateurs, grâce à la multiplicité des espaces de diffusion.

5.4.3 | *Validation des objectifs de l'étude*

Reprenons les objectifs avancés dans la section 5.2 et observons si notre modèle 5M permet d'y répondre.

Les **flux d'entrée** proposées par l'entreprise correspondent à différents types de données métier. Elles sont traitées et analysées, au travers des logiciels du groupe. Certaines d'entre elles sont mises à disposition au travers de ces logiciels et d'autres par l'intermédiaire de pages web ou flux vidéo.

Comme dans notre modèle, le traitement des données est effectué en amont par leur propriétaire. Il en résulte des interfaces adaptées à nos fonctions de composition et scénarisation.

“ La surface d'affichage doit permettre un travail collaboratif ”

Notre proposition se concentre sur la multiplication d'espaces numériques collaboratifs. Au même titre que pour les travaux sur les murs d'images, faire évoluer un groupe d'utilisateurs devant de grandes surfaces d'affichage favorise l'émulation entre les collaborateurs et permet l'affichage d'un grand nombre d'informations.

“ La surface d'affichage doit s'adapter aux configurations d'une agence ”

L'approche IoT-a est au centre de cette contrainte. L'organisation distribuée des matériels et logiciels rend l'ensemble naturellement évolutif. Des modules d'affichage peuvent être rajoutés ou retirés en fonction de la configuration d'une agence. Lorsqu'une agence évolue, le système a la capacité d'évoluer avec elle.

“ La surface d'affichage doit être dynamique et permettre différents scénarios d'affichage ”

De manière analogue à la contrainte précédente, l'approche IoT-a décentralise aussi la gestion de ce qui doit être affiché par les fonctions de composition et de scénarisation. Ainsi, notre modèle offre la possibilité de modifier à la volée certains paramètres pour réorganiser les interfaces à diffuser. La présence d'un utilisateur, l'heure de la journée, la survenue d'une alerte urgente sont autant de déclencheurs potentiels d'une réorganisation des données à visualiser.

5.5 | Mise en œuvre d'un système de visualisation

Afin d'expérimenter notre modèle 5M et proposer un prototype de nos travaux à l'entreprise, nous avons mis en œuvre un système de visualisation reproduisant les éléments présentés dans la section précédente.

5.5.1 | *Création d'un espace de visualisation*

L'entreprise souhaite un prototype lui permettant de tester puis préparer ses agences afin qu'elles puissent, à leur tour, en être équipées. Notre réalisation doit donc tenir compte des coûts, mais aussi des contraintes d'industrialisation, de montage et de maintenance. Elle doit de plus, être compatible mécaniquement et techniquement avec les infrastructures des futurs lieux ou elle sera mise en place.

Cette thèse s'inscrit dans l'équipe Tifiafai du laboratoire CREN. En 2017 elle publie un brevet d'invention intégrant leurs actions de recherche [Carlier and Renault, 2017]. Le brevet détaille les éléments d'un système d'affichage multi-agents. Il s'intéresse particulièrement aux dispositifs de restitution de contenus audiovisuels et leur affichage sur une pluralité de ces dispositifs. L'invention s'appuie sur un prototype de mur de seize écrans, réalisé quelques années avant la publication du brevet. Il est illustré figure 5.7. Il permet la diffusion de plusieurs flux vidéo simultanément sur différentes portions du dispositif.

FIGURE 5.7 – Premier prototype d'un mur de 16 écrans mis en œuvre par l'équipe Tifaifai

Nous l'avons constaté durant sa présentation, l'élément central de notre modèle 5M est le module d'affichage. Nous nous sommes donc concentrés sur ce point pour compléter les travaux sur le mur d'écrans multi-agents. Notre objectif est de proposer un module d'affichage répondant aux caractéristiques que nous avons présentées dans notre modèle. Après une étude des éléments techniques existants et un temps de modélisation mécanique, nous avons proposé une évolution d'un module d'affichage dont une vue en trois dimensions est illustrée par la figure 5.8.

Ainsi, nous avons assemblé, dans un premier temps, seize de nos nouveaux modules d'affichage pour former un second prototype de système de visualisation Dans un second temps nous avons mis en place notre livrable en entreprise au travers d'un système de 22 modules correspondant aux besoins des utilisateurs. Les figures 5.9 et 5.10 représentent nos réalisations.

Nous disposons donc d'une nouvelle version du dispositif proposé par l'équipe Tifaifai. Il permet la mise en œuvre pratique de notre modèle 5M et intègre quelques caractéristiques facilitant une future industrialisation comme nous l'avons précédemment présenté. En outre, il se veut compatible avec une mise en place en agence, dans les salles de travail où se déroule le métier d'exploitant.

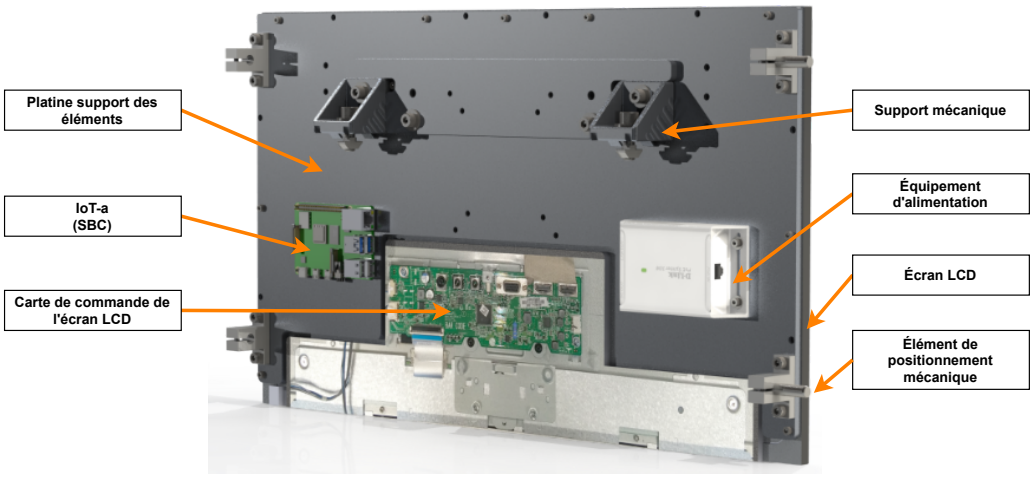


FIGURE 5.8 – Modélisation 3D d'un module d'affichage, vue arrière

FIGURE 5.9 – Second prototype d'un mur d'écrans de 16 modules d'affichage

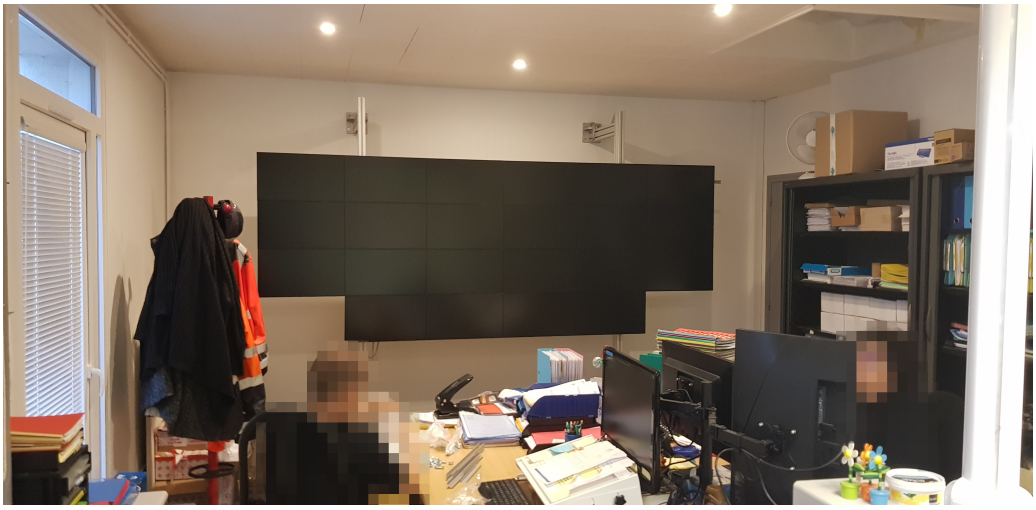


FIGURE 5.10 – Livrable du système de visualisation en agence SARP (22 modules d'affichage)

5.5.2 | *Application du modèle 5M*

Nous avons présenté les éléments techniques de notre système de visualisation, nous pouvons détailler l'organisation de nos IoT-a.

Chaque module d'affichage est équipé d'un IoT-a. Comme nous l'avons évoqué dans la section précédente, cet IoT-a doit disposer de caractéristiques techniques suffisantes pour répondre à de nombreuses fonctions, dont la gestion d'un flux continu d'images. Nous avons donc fait le choix d'un SBC, en l'occurrence un système embarqué Raspberry Pi sur lequel nous avons intégré nos agents. Nous présentons dans notre chapitre 6 la plate-forme et le framework Triskell3S nous permettant la mise en place de ces agents.

L'IoT-a compris dans un module d'affichage est responsable d'une constellation. Aussi, les agents AMS, DF, Ping, Bridge et, si besoin, plusieurs AMC, y sont implémentés. Pour échanger avec d'autres modules et ainsi mutualiser les agents du système de visualisation, les constellations se bridgent entre elles dynamiquement. Lorsqu'un module d'affichage est ajouté ou retiré du système, la visualisation peut être adaptée. Chaque module d'affichage intègre dans sa constellation un agent de visualisation. Il est responsable de la diffusion à l'écran du flux d'images proposé par la fonction de composition.

Le modèle 5M définit les fonctions de composition et de scénarisation. Ces dernières sont portées par les IoT-a du système. Pour rappel, la fonction de composition génère les visuels à afficher et la scénarisation les organise sur la surface d'affichage. La procédure à suivre pour les appliquer dans le système n'est pas définie, et pour cause, de multiples approches sont envisageables. Nous pouvons, par exemple, envisager une granularité fine de nos agent et distribuer ces fonctions dans chacun d'eux. Il est aussi possible de mettre en place un seul agent responsable des deux fonctions dans un élément extérieur aux modules d'affichage. Notre choix est encore différent, car nous proposons que ces fonctions soient portées par deux agents, chacun responsable d'une fonction. Nous disposons donc de deux agents supplémentaires : l'agent composition et l'agent scénarisation.

L'agent composition sert de passerelle entre notre système et les informations de l'entreprise. Il doit reposer sur une machine dimensionnée pour administrer les données et générer les flux d'images associés aux modules d'affichage. Dans cette logique, la puissance de calcul et la taille de la mémoire de cette machine, fixe les performances en termes de qualité d'image et de visualisation sur l'espace d'affichage. Nous ajoutons ainsi, en plus des modules d'affichage, une machine supplémentaire, de type ordinateur personnel, responsable d'une constellation et intégrant l'agent composition. Comme les modules d'affichage, cet élément complémentaire a la faculté de se bridger dynamiquement au reste du système.

Comment l'agent composition génère-t-il les visuels ? Nous avons vu dans la section 5.2 les types d'interfaces à visualiser par l'entreprise. L'agent composition est donc capable d'instancier des fenêtres graphiques dans un bureau virtuel. Il en contrôle le contenu et la disponibilité par différents mécanismes. L'agent est par exemple compétent pour remplir les champs d'un formulaire afin de connecter un utilisateur automatiquement à une interface protégée par mot de passe. Il permet aussi d'interagir avec les éléments d'un site ou encore de parcourir une page internet. Chaque fenêtre graphique, ouverte par l'agent, dispose d'un point d'entrée permettant sa modifications. En l'occurrence, ce point se traduit par l'abonnement au topic MQTT dédié, en écoute des ordres de modification. L'agent scénarisation peut alors organiser les visuels en publiant sur leur différents topics. L'agent composition est aussi responsable de la diffusion des flux d'images pour les modules d'affichage. Nous détaillons certaines de ces techniques dans la section suivante.

L'agent scénarisation est compatible avec les éléments déjà en place. Il a la capacité d'être présent dans un des modules d'affichage, mais pour des raisons de déploiement et de maintenance il est au côté de l'agent composition dans la machine responsable de la diffusion de flux d'images. Son principal comportement est d'agencer les visuels de façon à maximiser l'affichage sur un espace de diffusion. Il fait référence, pour cela, à un fichier de description des scénarios établis par les utilisateurs. Ce dernier définit des scénarios d'affichage chacun composé d'une répartition spatiale de visuels et des poids caractérisant leur priorité. Ce poids joue un rôle dans le comportement proactif de l'agent scénarisation, il lui permet de favoriser un visuel par rapport à un autre, lors de modifications de l'espace de visualisation par exemple. Ces poids et les informations de répartition peuvent changer à tout moment, rendant l'affichage dynamique. Pour le contrôler, l'agent scénarisation contacte les visuels, proposés par l'agent composition, sur leurs topics MQTT. Il paramètre ainsi, leurs tailles en pixels, leurs positions, leurs facteurs de zoom pour la lisibilité et leurs activations.

Dans la figure 5.11 nous retrouvons à gauche, un équipement avec des performances supérieures à un module d'affichage. Il est intègre les agents composition et scénarisation en plus des constituants d'une constellation. L'agent composition compile les données hétérogènes de l'entreprise pour en générer des visuels accessibles

au travers d'un flux continu d'images. L'agent scénarisation organise les visuels. À droite, plusieurs modules d'affichage ont bridgés leurs constellations entre eux et avec l'équipement supplémentaire. Ils récupèrent la partie du flux d'images qui les concerne par l'intermédiaire de leur agent visualisation et le diffuse sur leur écran.

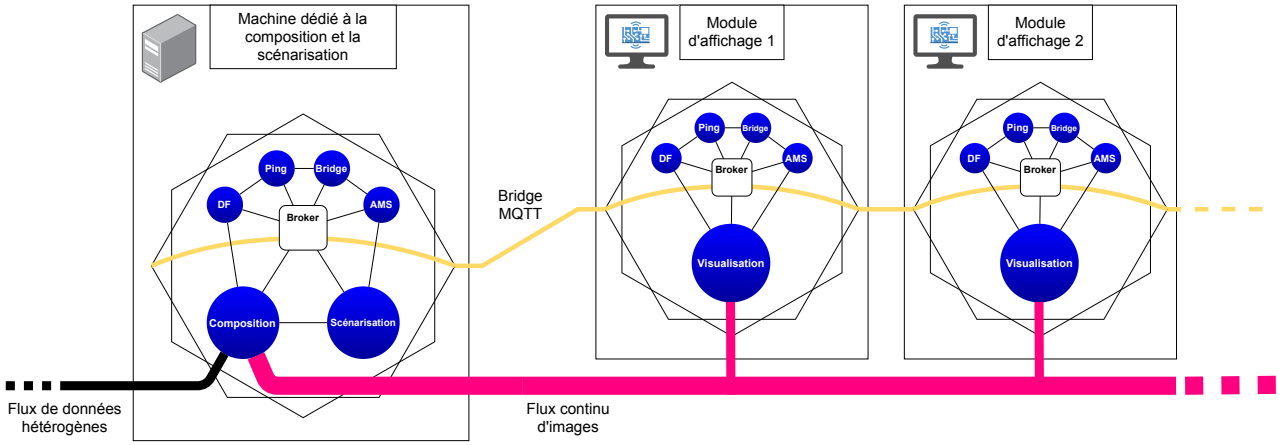


FIGURE 5.11 – Architecture des éléments constituant notre système de visualisation basé sur le modèle 5M

5.5.3 | *Technologies de diffusion des flux d'images*

Nous avons étudié dans la section précédente la présentation du projet de mur d'écrans détaillé dans le brevet [Carlier and Renault, 2017]. L'une des limites de ces travaux réside dans le fait que le système permet uniquement la visualisation de flux d'images de type vidéo. Nous allons détailler dans cette section l'évolution de cette méthode permettant de visualiser à la fois des flux vidéo synchronisés et des applications de bureau. Nous entendons par le terme applications de bureau, toutes applications portées par un environnement graphique. Ces applications se présentent sous la forme d'une ou plusieurs fenêtres administrées par un gestionnaire de fenêtres. Chaque système d'exploitation propose une gestion de ses fenêtres au travers d'une couche logicielle dédiée. En l'occurrence, nous nous concentrerons sur le fonctionnement des applications fenêtrées sur les systèmes d'exploitation GNU/Linux.

Le concept de fenêtrage sur les systèmes Linux débute à la fin des années 1980 par l'implémentation du logiciel X Window System développé par le Massachusetts Institute of Technology (MIT) avec le support du Digital Equipement Corporatioin (DEC) [Douglas, 1994]. C'est en 1987 que la version 11 parait pour donner naissance à X11 dans le but de proposer une solution d'affichage graphique indépendant du matériel support. X11 est un ensemble de spécifications permettant la manipulation d'informations graphiques au travers d'un protocole de communication. Ce protocole permet aux applications de bureau de bénéficier de la création de fenêtres graphiques, d'interactions avec les périphériques comme le clavier ou la souris et les événements du système d'exploitation. Le protocole X11 repose sur l'utilisation de sockets pour ses interactions. Il est compatible avec des échanges réseau et son cœur est d'ailleurs appelé serveur graphique. En effet, chaque application graphique est considérée comme un client qui doit se connecter au serveur graphique.

Notre objectif est de pouvoir afficher ces applications sur notre dispositif de visualisation. Le projet X11 avait comme objectif de partager la diffusion de fenêtres graphiques au travers d'un réseau. Les documents de références de l'époque donnent l'exemple d'une diffusion dans un cadre scolaire. Une machine principale serveur serait dans la capacité de diffuser une application sur chaque poste client devant les étudiants. L'ensemble de la classe aurait alors la possibilité de visualiser et d'interagir avec l'application. Le professeur pourrait, depuis son poste, voir les modifications instantanément et aider les étudiants en direct.

Cet objectif est proche de notre champ d'application, car nous observons une similarité entre la répartition des affichages devant plusieurs étudiants et nos multiples modules d'affichage contigus. Dans les deux cas, l'affichage d'une application peut s'effectuer sur plusieurs écrans. Nous avons donc expérimenté cette solution comme première approche. La mise en place de cette méthode fait appel à quelques éléments de configuration du serveur graphique pour autoriser les échanges sur le réseau. Les échanges entre clients et serveur ne connaissent aucune compression de données et l'affichage d'une interface ayant une grande résolution (Full HD ou +) génère une quantité de données trop importante pour qu'elle puisse être partagée sur un réseau Ethernet. Cette solution n'a donc pas été retenue.

Nous devons prendre en compte la bande passante comme un vecteur d'optimisation de notre système. Pour cela, certaines approches permettent le partage d'écrans depuis une machine source vers plusieurs autres. Plusieurs termes se cachent derrière cette approche : partage de bureau, partage d'écrans, bureau à distance, etc. L'une des techniques régulièrement rencontrées dans la littérature propose l'utilisation du protocole *Virtual Network Computing* (VNC) [Yazdanipour et al., 2012]. Il permet la diffusion d'images au travers d'un réseau utilisant un minimum de bande passante grâce à une compression d'images dynamiques. D'autres travaux montrent d'ailleurs la pertinence de cette approche en évaluant les métriques associées à une visualisation multi écrans [Liu and Anshus, 2009] [Liu et al., 2009]. Le VNC est une technologie permettant le partage d'écrans au travers d'un réseau. Il a été créé dans les années 1990 et connaît aujourd'hui encore une grande activité de développement au sein de plusieurs acteurs industriels (RealVNC, Teamviewer, Skype) et du monde du logiciel libre [LibVNC, 2020]. L'utilisation de VNC s'articule autour d'une architecture clients-serveur. Le serveur capte la surface graphique (pixels) à visualiser, la compresse et la diffuse. Le client quant à lui est appelé visualiseur. Il a pour rôle de se connecter au serveur, de décompresser le flux et l'afficher à l'écran de la machine hôte.

Le protocole VNC encapsule un protocole de plus bas niveau appelé Remote FrameBuffer (RFB) [Richardson et al., 2011]. Ce dernier est un protocole d'accès à distance pour des interfaces graphiques. Il permet à un client de visualiser et de contrôler un système de fenêtres sur un autre ordinateur.

Un serveur VNC détermine, sur l'hôte dont l'écran est partagé, les pixels différents de l'image précédemment envoyée pour en former un vecteur de pixels nouveaux. Cette méthode permet d'envoyer uniquement la différence de pixels entre une image N et $N-1$ et n'occasionne aucune transmission de données si l'image reste fixe. Le client a les compétences de sauvegarder à l'écran l'image précédemment envoyée. Un client peut contrôler les applications à distance par l'envoi de primitives de périphériques (touches du clavier, macro clavier, position et état des boutons de la sourie). La figure 5.12 illustre les différentes étapes nécessaires à la mise à jour d'une transmission d'images entre un serveur et un client VNC. Cette mise à jour intervient lorsque de nouveaux pixels apparaissent sur l'interface graphique à transférer.

Le protocole RFB normalise plusieurs méthodes de compression pour l'échange des vecteurs de pixels. Les échanges entre clients et serveur déterminent la bande passante disponible et une adaptation automatique de la compression, entre les deux protagonistes, s'effectue pour réduire les débits de transfert.

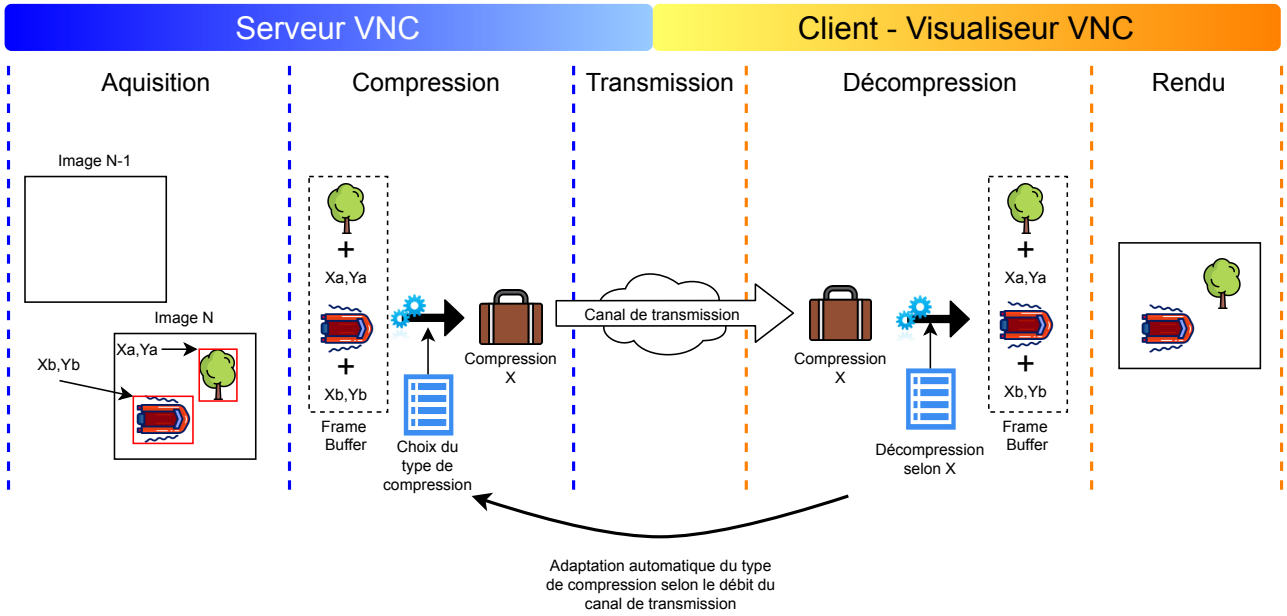


FIGURE 5.12 – Transmission VNC depuis un serveur vers un client visualiseur

Bénéficiant du protocole VNC comme vecteur de transmission d'interfaces graphiques, nous avons repris l'approche de [Yazdanipour et al., 2012] pour scinder le signal VNC en différents canaux compatibles avec nos modules d'affichage. Ainsi, nous avons mis en place le procédé illustré sur la figure 5.13 dans notre agent composition.

Un élément α est capable de générer une surface graphique (bureau virtuel) dans laquelle peuvent évoluer différentes applications graphiques. L'élément α instancie autant de serveurs VNC ($Svnc$) que de modules d'affichage (Ma) : $Svnc_0 ... Svnc_n, n = \sum Ma$. Chaque serveur VNC ouvre un port sur le réseau et émet une partie de la surface graphique restreinte par la géométrie réelle du module d'affichage. Enfin, les agents visualisation, dans chacun des modules d'affichage, instancient un client visualiseur, $Cvnc$, se connectant au serveur VNC associé : $Cvnc_p \Leftrightarrow Svnc_p$. Nous obtenons alors une visualisation complète du bureau disponible dans α, généré par l'agent composition, grâce à l'agrégation des modules d'affichage Ma_n.

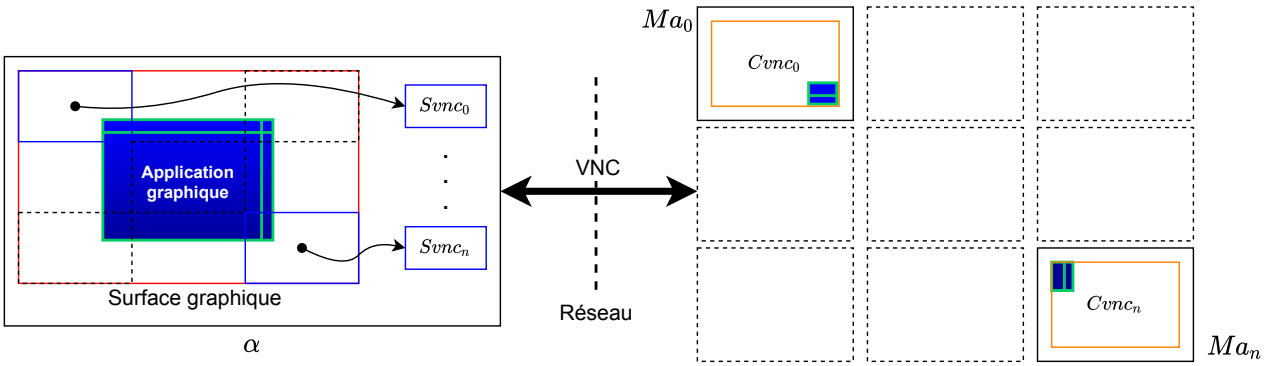


FIGURE 5.13 – Répartition des affichages par l'utilisation du protocole VNC

L'intégration de nos propositions et du protocole VNC aux travaux de l'équipe Tifaifai, permet à notre dispositif de visualisation d'être capable d'afficher des contenus de différentes natures. L'agent composition génère les flux nécessaires en fonction des visuels à diffuser. Les interactions entre l'agent scénarisation et les agents de visualisation, présents dans les modules d'affichage, permettent la diffusion des différents types de visuels sur le dispositif. La figure 5.14 illustre un exemple de visualisation hybride, avec la diffusion de flux vidéo de caméras sur la ligne d'écrans du bas et des interfaces métier sur le reste des écrans.

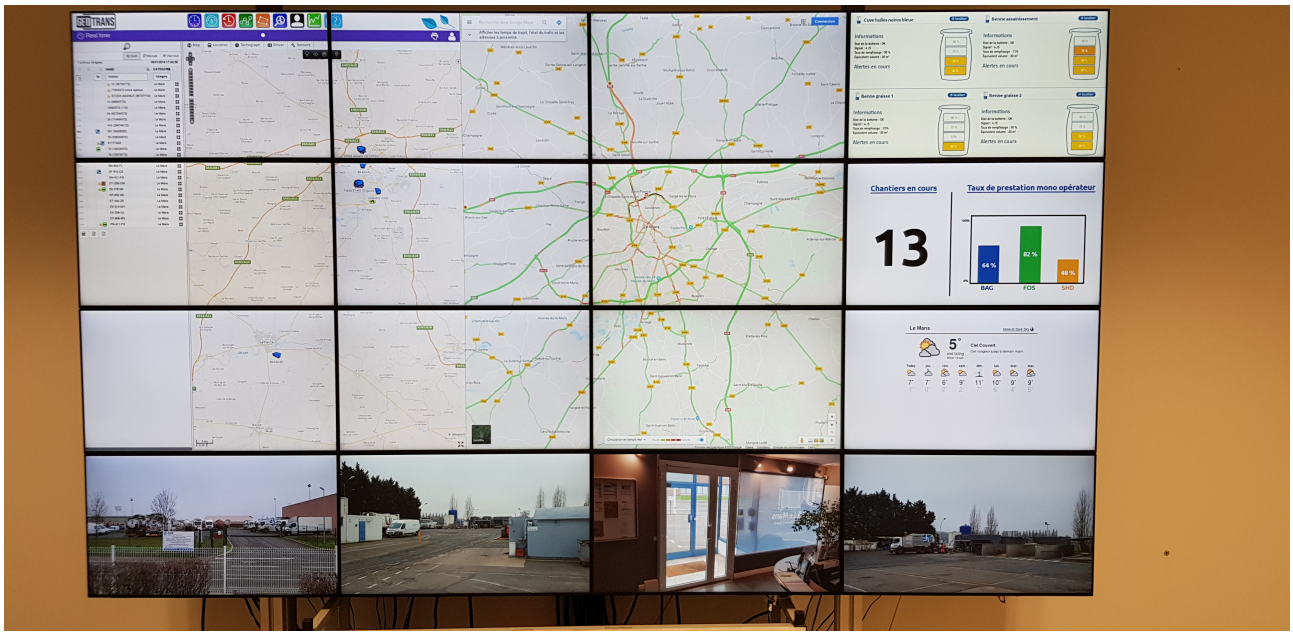


FIGURE 5.14 – Exemple de scénario d'affichage hybride vidéo + interfaces graphiques

5.6 | Synthèse

Nous avons présenté, dans ce chapitre, le cadre applicatif dans lequel s'inscrivent nos travaux. L'entreprise SARP-Véolia, avec laquelle cette thèse est en collaboration CIFRE, souhaite bénéficier d'un nouvel outil pour ses activités d'exploitation. Son besoin correspond à la diffusion dynamique d'une multitude de visuels métier, face à différentes équipes de collaborateurs.

Nous avons ainsi proposé le modèle 5M, reposant sur nos propositions de l'approche IoT-a. Il avance la notion d'espace de diffusion, dans laquelle les IoT-a composent et scénarisent les interfaces à visualiser. L'ensemble des informations est alors réparti sur une pluralité de modules d'affichage composant un mur d'écrans. Un module d'affichage est défini comme l'association d'un IoT-a avec un dispositif de diffusion.

Ce chapitre présente principalement un cas d'application de notre approche IoT-a. Le mur d'écrans nous permet de bénéficier d'un outil pour expérimenter nos concepts. D'autres applications pourront être envisagées dans de futurs travaux. Aussi, verrons-nous dans le chapitre suivant, différentes preuves de concept illustrant l'intérêt de nos propositions.

Bibliographie

[Bethel et al., 2012] Bethel, E. W., Childs, H., and Hansen, C. (2012). *High Performance Visualization : Enabling Extreme-Scale Scientific Insight*. Chapman and Hall/CRC, 1st edition.

[Boulongne, 2018] Boulongne, A. (2018). Faire face à la transformation numérique : l'exemple de la collaboration entre le cren et la sarp. Master's thesis, Univeristé Rennes 2.

[Carlier and Renault, 2017] Carlier, F. and Renault, V. (2017). Systeme d'affichage multi-agents et methode d'affichage associee. Brevet, Notice.

[De Almeida et al., 2012] De Almeida, R. A., Pillias, C., Pietriga, E., and Cubaud, P. (2012). Looking behind Bezels : French Windows for Wall Displays. In ACM, editor, *AVI - 11th working conference on Advanced visual interfaces - 2012*, Proceedings of the 11th working conference on Advanced visual interfaces, pages 124–131, Capri, Italy.

[Douglas, 1994] Douglas, A. Y. (1994). *The X Window System : Programming and Applications with Xt, OSF/Motif edition*. Prentice Hall.

[Han et al., 2011] Han, J., Kamber, M., and Pei, J. (2011). *Data Mining : Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition.

[Koenig, 2009] Koenig, P.-Y. (2009). *Information Visualization : multiscale navigation paradigms and focus+context approaches*. Theses, Université Montpellier II - Sciences et Techniques du Languedoc.

[LibVNC, 2020] LibVNC (2020). Libvnc software. `https://libvnc.github.io/`. Accès le : 28/01/2020.

[Liu and Anshus, 2009] Liu, Y. and Anshus, O. J. (2009). Improving the performance of vnc for high-resolution display walls. In *2009 International Symposium on Collaborative Technologies and Systems*, pages 376–383.

[Liu et al., 2009] Liu, Y., Bjorndalen, J. M., and Anshus, O. J. (2009). Using multi-threading and server update pushing to improve the performance of vnc for a wall-sized tiled display wall. In Mueller, P., Cao, J.-N., and Wang, C.-L., editors, *Scalable Information Systems*, pages 306–321, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Moreno and Gomes Soares, 2015] Moreno, F. M. and Gomes Soares, L. F. (2015). Support to multimedia presentations on multi-head setups. In *2015 IEEE International Symposium on Multimedia (ISM)*, pages 381–384.

[Richardson et al., 2011] Richardson, T., Levine, J., and Ltd., R. (2011). The remote framebuffer protocol. RFC 6143, RFC Editor. http ://www.rfc-editor.org/rfc/rfc6143.txt.

[Sandstrom et al., 2003] Sandstrom, T., Henze, C., and Levit, C. (2003). The hyperwall. In *Proceedings International Conference on Coordinated and Multiple Views in Exploratory Visualization - CMV 2003* -, pages 124–133.

[Ware, 2012] Ware, C. (2012). *Information visualization : perception for design*. Elsevier, third edition.

[Yazdanipour et al., 2012] Yazdanipour, M., Mahmoudi, D., Yazdanipour, A., Yazdanipour, M., and Mehdipour, A. (2012). Comprehensive review and selection criteria for virtual network computing technology. In *2012 Ninth International Conference on Wireless and Optical Communications Networks (WOCN)*, pages 1–5.

[Zikopoulos et al., 2012] Zikopoulos, P. C., Eaton, C., deRoos, D., Deutsch, T., and Lapis, G. (2012). *Understanding Big Data : Analytics for Enterprise Class Hadoop and Streaming Data*. McGraw-Hill Osborne Media.

Partie III

Expérimentations et analyses

Chapitre 6

Expérimentations et plate-forme Triskell3S

Ce chapitre présente nos expérimentations et les outils utilisés pour y parvenir. Nous souhaitons illustrer différents cas pratiques du mur d'écrans que nous avons présenté dans le chapitre précédent. Pour rappel, un module d'affichage représente un écran dans le mur d'écrans. Il est constitué d'un IoT-a intégrant une constellation dont un broker MQTT permettant de centraliser localement les communications des différents agents présents.

Dans une première section nous présentons la plate-forme SMA embarquée Triskell3S, utilisée pour l'implémentation de nos agents dans les différents IoT-a. Nous poursuivons avec la présentation de trois preuves de concept. Nous commençons par la mise en place de bridges dynamiques entre plusieurs constellations d'IoT-a. Nous validons son fonctionnement et voyons comment nos agents Ping et Bridge réagissent face à des aléas de connexion. Nous nous servons alors de cette base expérimentale pour les preuves de concept suivantes. Nous analysons, dans une seconde preuve de concept, le retrait ou de l'ajout, à la volée, d'écrans dans notre mur d'écrans grâce à l'utilisation de liens dynamiques entre constellations. Nous vérifions ainsi que notre système est résilient face la perte d'éléments. Afin d'éprouver notre modèle 5M, nous présentons une dernière preuve de concept ayant pour but d'adapter les flux d'images diffusés en fonction des profils des utilisateurs présents devant le mur d'écrans.

6.1 | Framework et plate-forme Triskell3S

La plate-forme Triskell3S a été développée en 2016 par l'équipe Tifaifai [Renault and Carlier, 2016]. Elle offre une architecture permettant le développement d'agents embarqués dans des environnements matériels hétérogènes. L'acronyme "3S" repose sur **S**ystèmes embarqués, **S**ystèmes multi-agents et **S**ystèmes mobiles.

À la fois framework et plate-forme multi-agents, Triskell3S, implémente un ensemble de modules indépendants tenant compte des contraintes de programmation du domaine des systèmes embarqués. Ce choix s'explique par la mise en application d'agents au plus proche des composants électroniques. Triskell3S se démarque d'une partie des plate-formes SMA reposant sur un middleware faisant abstraction du matériel. Dans le cas des IoT-a, il est intéressant que les agents puissent accéder à l'ensemble des caractéristiques matérielles des objets. De plus, les performances d'une solution par middleware ne sont pas toujours adaptées avec des dispositifs ayant des ressources limitées [Bergenti et al., 2014].

6.1.1 | *Framework Triskell3S*

Triskell3S repose sur les standards de la FIPA que nous avons abordés précédemment. Néanmoins, voyons comment la plate-forme est organisée pour y répondre et notamment, intéressons-nous à son framework et ses composants.

La notion de framework est particulièrement présente dans de nombreux projets actuels. Elle est définie comme étant un ensemble d'éléments offrant de la modularité, de l'extensibilité et des comportements minimaux [Rapicault, 2002]. Le framework doit avoir un aspect structurel, en proposant une architecture d'objets logiciels, et comportemental, en définissant les fonctionnalités fournies par ces objets. Nous retrouvons dans la littérature la méthode de méta-modélisation [Rapicault, 2002]. Elle consiste en la réification d'éléments struc-

turels d'un framework pour en simplifier les dépendances. Ainsi, le framework dispose d'une méta-description de ses dépendances. Au fur et à mesure de sa création, la description de l'application est ajoutée aux côtés de celle de la modélisation. Cette méthode permet de découper la conception en trois étapes : une première dite de méta-framework, une seconde issue d'une transition des réalisations conceptuelles donnant naissance au framework et une dernière décrivant exhaustivement les objets logiciels du projet au sein d'une architecture.

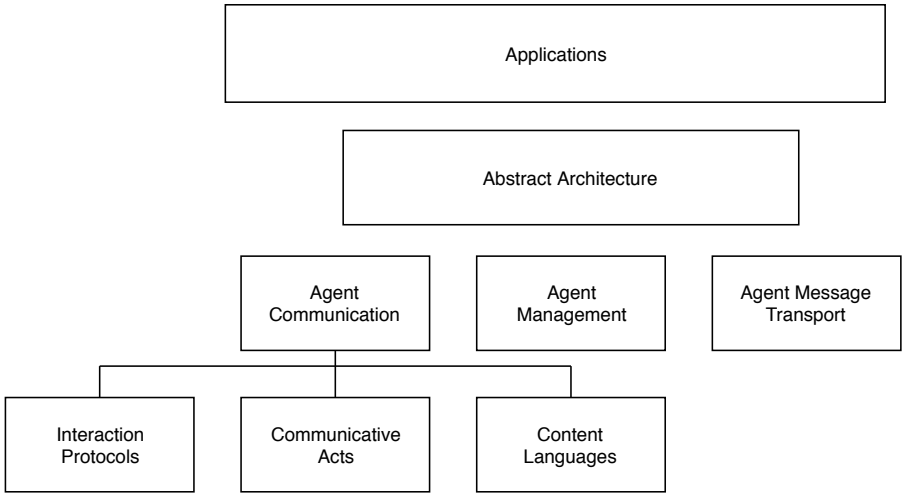


FIGURE 6.1 – Méta-framework Triskell3S : modélisation des spécifications des standards de la FIPA

Triskell3S s'appuie sur la méthode du méta-framework pour son développement tout en respectant la structure théorique des standards de la FIPA. Dans la figure 6.1 nous distinguons quarte niveaux de description. Le premier, tout en haut, concerne les applications "métier" vues par l'utilisateur. Le second regroupe les éléments abstraits pour l'architecture des agents. Dans le niveau suivant, nous observons les trois fonctions principales qu'un framework SMA doit proposer selon la FIPA : les aspects de communication, de transport des messages et d'administration des agents. Enfin, la dernière couche détaille la partie communication.

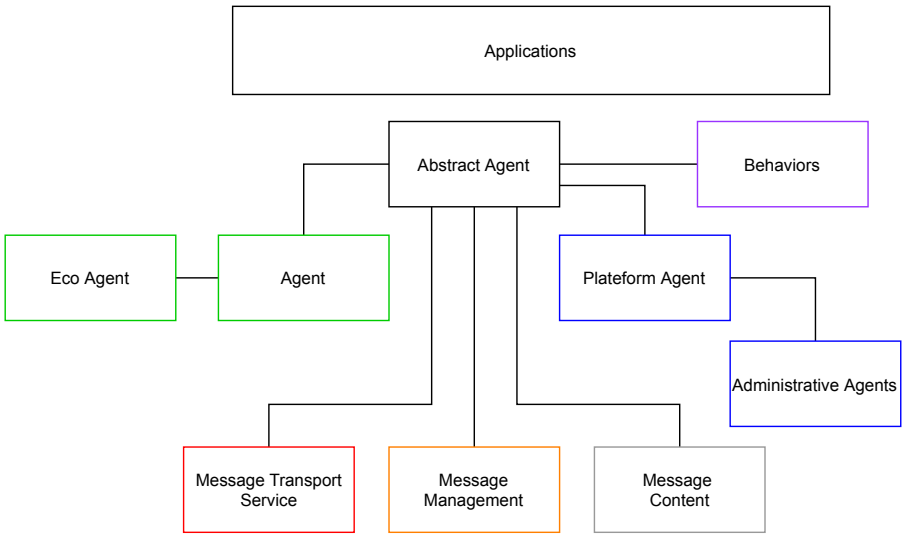


FIGURE 6.2 – Framework Triskell3S

La méta-modélisation avancée dans la figure 6.1, pose le cadre des éléments nécessaires dans Triskell3S. Ainsi, nous proposons le framework illustré en figure 6.2 détaillant la structure des agents et leurs méthodes de communication. Cette représentation illustre les éléments correspondant aux recommandations de la FIPA pour laisser libre l'implémentation concrète à la phase suivante. L'ajout, par rapport à la méta-modélisation, de l'objet *Behaviors*, symbolise les comportements que nous avons définis dans la section 4.1.2. L'objet *AbstractAgent* rassemble toutes les fonctionnalités nécessaires à l'évolution d'un agent dans un environnement, notamment les éléments de communication et les comportements.

La figure 6.3 détaille l'architecture des objets dans le framework Triskell3S. Les éléments de structure, dans le cadre noir, permettent le développement d'agents, mais aussi d'agents de plate-forme, comme AMS, DF ou AMC. Ils héritent de l'objet *AbstractAgent* qui décrit les mécanismes de communication et les comportements.

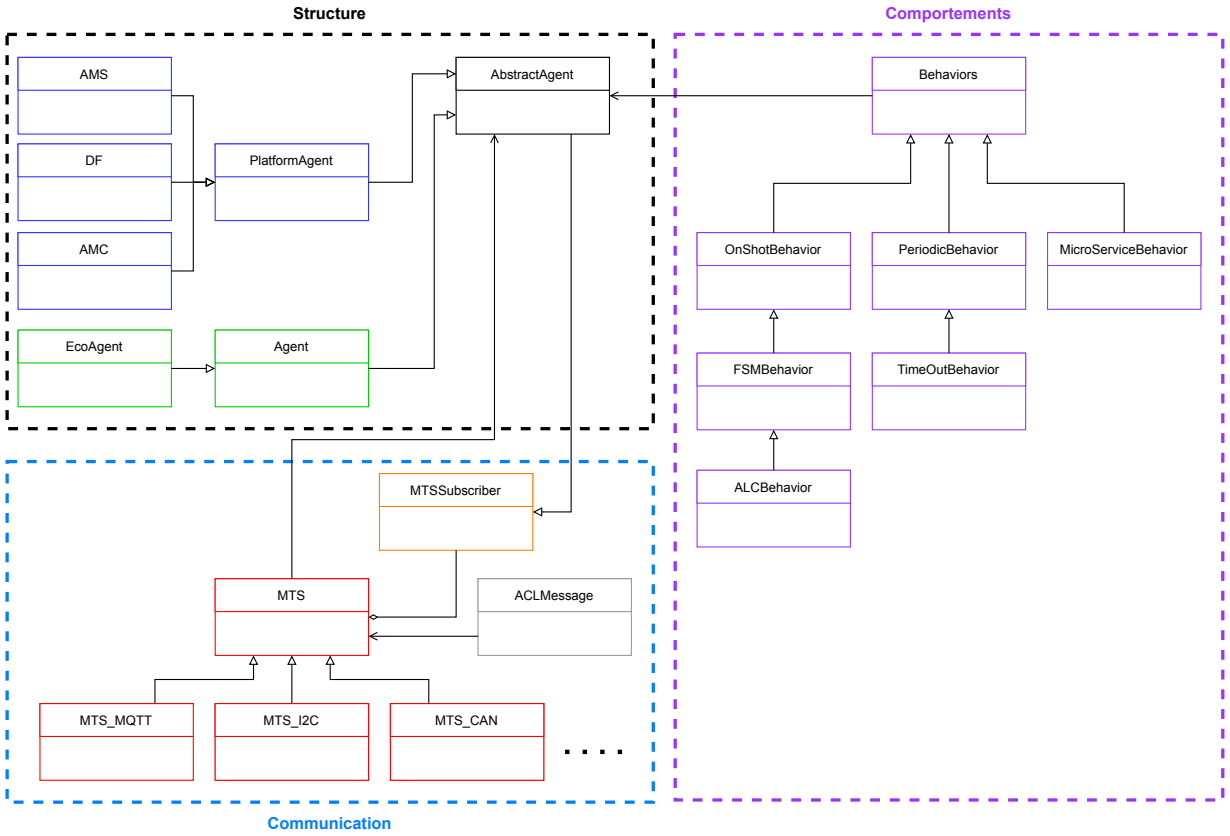


FIGURE 6.3 – Architecture embarquée multi-agents du framework Triskell3S

Dans le cadre de Triskell3S, les différences entre un agent et un agent de plate-forme sont leurs mécanismes internes. Un agent intègre les éléments lui permettant d'organiser son cycle de vie. Il propose des services au travers de l'exécution de comportements et a la possibilité de naître ou mourir à tout moment. Les agents de plate-forme sont toujours vivants, car ils organisent les agents, ils disposent pour certains des mécanismes permettant de modifier le cycle de vie des agents. Ils peuvent créer ou tuer des agents à la volée en fonction des paramètres fixés par le concepteur.

La section communication de la figure 6.3 détaille les objets propres aux interactions entre les agents. L'élément principal de ce cadre est l'objet *MTS* reprenant les définitions du standard de la FIPA [FIPA, 2002c]. Nous avions évoqué ce sujet dans la section 4.1.4. La classe *MTS* permet la définition de plusieurs classes filles. Chacune implémente les fonctions nécessaires à la mise en place d'un canal de communication spécifique. En l'occurrence, nous avons seulement détaillé ici, les canaux MQTT, I2C et CAN, mais il en existe d'autres. Elle définit, de plus, les méthodes abstraites permettant de communiquer uniformément, quel que soit le protocole utilisé. Le tableau 6.1 en détaille les trois principales. Le concepteur a le choix d'instancier un ou plusieurs canaux de communication dans un agent.

Méthode	**Description**
MTSsubscribe	Permet l'abonnement d'un agent à un topic. Chaque réception de message sur le canal associé appel la méthode *receivedMessage* défini dans l'objet *MTSSubscriber*.
MTSunsubscribe	Supprime un topic dans la liste des abonnements.
MTSpublish	Publie un message sur un topic.

Tableau 6.1 – Méthodes abstraites définies dans l'objet MTS

Un agent a aussi la possibilité d'intégrer autant de comportements que nécessaire pour réaliser ses fonctions. La section "comportements" de la figure 6.3 rassemble les objets définissant les comportements que nous avions évoqués en section 4.1.2. La classe mère *Behaviors* définit les méthodes de contrôle de chacun.

Une architecture modulaire est primordiale dans le domaine des systèmes embarqués. Au même titre que l'organisation modulaire d'un SMA que nous avons développé en section 3.3.4, ce type d'architecture permet de faciliter la mise œuvre dans différents cadres applicatifs. Une telle architecture permet de faire évoluer les couches afin de prendre en charge un nouveau système, sans modifier l'ensemble du framework. L'architecture de Triskell3S offre alors une grande souplesse lors de l'implémentation. Lorsqu'un concepteur souhaite développer une série d'agents, il a la possibilité de sélectionner uniquement les éléments qu'il souhaite mettre en œuvre. La conception d'un agent s'effectue en trois étapes : la première permet de déterminer la nature de l'agent. Est-il un agent ou un agent de plate-forme ? La seconde, dresse l'ensemble des comportements de l'agent parmi les six actuellement proposés. Plusieurs d'entre eux peuvent être composés. Enfin, dans la dernière, le concepteur doit préciser les canaux de communication de l'agent et les paramètres lui permettant d'échanger avec ses agents de plate-forme. Une fois ces paramètres renseignés, le concepteur peut se concentrer sur le développement du contenu des comportements puis envisager par la suite l'implémentation sur une cible matérielle.

6.1.2 | *Plate-forme Triskell3S multi-agents embarqués*

Le moteur de communication au sein de la plate-forme Trikell3S repose sur le protocole MQTT de type *Publish/Subscribe*. Tous les agents de plate-forme disposent d'un canal de communication mettant en œuvre ce protocole. En l'occurrence, une plate-forme Triskell3S intègre un broker Mosquitto modifié, incluant notre proposition de bridges dynamiques.

Nous avons présenté dans le chapitre 4 quels sont les topics auxquels un agent doit s'abonner. Trikell3S développe cette organisation. Chaque agent s'abonne aux topics correspondants à son nom et aux fonctions nécessitant des interactions avec les autres agents. Par exemple, les agents AMS et DF s'abonnent respectivement aux topics : `ACC/AMS/register` et `ACC/DF/register`. Contrairement à l'approche qui consiste à analyser le contenu du message pour déterminer son type, la mise en place de topics intégrant le nom de la fonction permet plusieurs évolutions. Cette technique permet de découper, grâce aux fonctionnalités du protocole, la gestion des événements à mettre en place lors de la réception des messages. En d'autres termes, elle offre aux agents la possibilité de ne pas analyser le contenu d'un message ou de le faire suivre si le sujet le nécessite. Elle ouvre aussi la possibilité d'adresser un message ayant le même contenu à un groupe d'agents. Enfin, elle permet une meilleure gestion du débogage. Il est rendu possible au concepteur de créer un agent analysant certains types de comportement. En ayant recours aux *Wildacrds* ("`#`" ou "`+`"), l'agent peut s'abonner à des groupes de topics mettant en œuvre les mêmes fonctions. Il est donc possible d'abonner notre agent d'analyse au topic `ACC/+/register` pour écouter et analyser l'enregistrement de tous les agents auprès des deux agents administratifs.

En plus des aspects de communication, la plate-forme Trikell3S intègre trois types d'agents de plate-forme :

- AMS existe une seule fois par plate-forme. Il est responsable de la gestion des opérations, comme la création ou la suppression d'agents. Il peut être interrogé pour diffuser les informations de la constellation dans laquelle il se trouve. Il peut aussi donner les adresses des agents demandés. AMS représente l'autorité de gestion des agents enregistrés auprès de lui. Il dispose d'une table de trois champs dans laquelle est enregistrée chaque AID. Le tableau 6.2 précise les topics auxquels AMS est abonné permettant de mettre à jour sa table.

- DF est le second agent de plate-forme au sein de Triskell3S. Il a pour rôle d'enregistrer dans une table les services que peuvent rendre les agents. En fonction des paramètres précisés par le concepteur, plusieurs agents DF peuvent coexister et indépendamment de leur nombre, ils s'enregistrent auprès d'AMS. Dans le cas où plusieurs constellations sont reliées les unes aux autres, les DF partagent leurs services. Après s'être enregistrés auprès d'AMS les agents s'enregistrent auprès de DF. Le tableau 6.3 précise les topics auxquels DF est abonné permettant de mettre à jour sa table.

- AMC, défini en section 4.2.2, permet de relayer vers le canal MQTT, les communications émanant d'IoT-a utilisant des canaux de communication différents. Plusieurs agents de médiation peuvent être mis en place. Triskell3S impose d'ailleurs qu'un AMC ne soit responsable que d'un seul canal local pour éviter des incohérences dans les noms. Si plusieurs groupes d'IoT-a ne disposent pas du protocole MQTT, alors

autant d'AMC que le nombre de groupes doivent être présents sur la plate-forme. L'agent AMC intègre lui aussi une table des agents enregistrés. Contrairement à la présentation des deux agents de plate-forme précédents, AMC ne dispose pas d'abonnement fixe à des topics. Ils sont modulés en fonction des agents présents sur le canal local. Lorsqu'un agent s'enregistre auprès d'AMS par l'intermédiaire d'un AMC, ce dernier s'abonne automatiquement au topic associé au nom de l'agent.

Topics	**Description**
`ACC/AMS/register` `ACC/AMS_nom_constellation/register`	Enregistre un agent. L'agent renseigne son *Agent Identifier Description* et son adresse sous la forme : `protocol@add` `protocol` signifie le protocole par lequel l'agent s'est enregistré. `add` détermine l'adresse sur ce protocole. Un agent est automatiquement renseigné comme disponible à sa création.
`ACC/AMS/deregister` `ACC/AMS_nom_constellation/deregister`	Désenregistre un agent de la plate-forme. L'agent spécifie son AID pour être reconnu.
`ACC/AMS/modify` `ACC/AMS_nom_constellation/modify`	Change les paramètres d'un agent.
`ACC/AMS/search` `ACC/AMS_nom_constellation/search`	Recherche l'AID, les coordonnées et la disponibilité d'un agent.
`ACC/AMS/info` `ACC/AMS_nom_constellation/info`	Renvoie les informations sur la constellation.
`ACC/AMS/#` `ACC/AMS_nom_constellation/#`	Permet la perception de tout autres messages non définis.

Tableau 6.2 – Topics abonnés par AMS

Topics	**Description**
`ACC/DF/register`	Enregistre un agent et ses services.
`ACC/DF/deregister`	Désenregistre un agent de la plate-forme. L'agent spécifie son AID pour être reconnu.
`ACC/DF/modify`	Change les paramètres d'un agent.

Tableau 6.3 – Topics abonnés par DF

Triksell3S est donc un environnement polyvalent mettant à disposition des outils propices à la mise en œuvre de nos expérimentations. Nous proposons désormais, dans les sections suivantes, trois preuves de concept afin de mesurer la pertinence de nos apports et ainsi vérifier nos hypothèses :

- Liens dynamiques entre constellations d'IoT-a, section 6.2,
- Système de visualisation résilient, section 6.3,
- Système de visualisation multi-utilisateurs, section 6.4.

6.2 | Liens dynamiques entre constellations d'IoT-a

Le chapitre précédent décrit un cas d'usage des IoT-a au travers de la mise en place d'un mur d'écrans connectés. Ce dispositif a vocation à être installé dans différentes agences de taille et fonctionnement différents. Le mur d'écrans doit donc être modulaire et évolutif pour s'adapter aux futures agences et à leurs activités. Pour rappel, chaque module d'affichage qui compose le mur d'écrans est composé d'un système embarqué responsable d'une constellation d'IoT-a et d'un écran. Les liens entre les constellations confèrent au système des solutions de communication entre les agents des différents modules d'affichage. Néanmoins, nous avons présenté dans la section 4.2.3, les limites des liens statiques dans un système évolutif comme le mur d'écrans. Pour répondre à cela, nous avons proposé les bridges dynamiques et les agents Ping et Bridge afin d'organiser les liens dynamiquement en fonction de l'évolution du système.

6.2.1 | *Contexte expérimental*

Ces premières expérimentations se focalisent sur la mise en œuvre de liens entre des constellations sous la forme de topologies en ligne et en étoile, comme nous les avons présentées dans la section 4.2.3. L'objectif principal est d'assurer le plus haut niveau de connectivité entre nos constellations afin de maintenir la transmission d'un maximum de messages lorsque des aléas techniques surviennent.

Le support de ces expérimentations est donc le second prototype de mur d'écrans que nous avons présenté dans notre chapitre précédent. Seize modules d'affichage jouent le rôle de constellation, chaque module représente un écran. Chacun des modules est architecturé selon la structure présentée dans la figure 6.4. Ils intègrent une plate-forme Triskell3S avec les agents de plate-forme AMS et DF, un agent Ping et un agent Bridge, d'autres agents responsables des tâches de visualisation et un broker MQTT. Les conditions initiales sont les suivantes : aucune constellation ne se connaît sur le réseau, aucun bridge ne les relie, toutes les constellations cherchent à se connecter aux autres et aucune configuration de bridge statique n'est mise en place pour éviter les biais d'expérimentation.

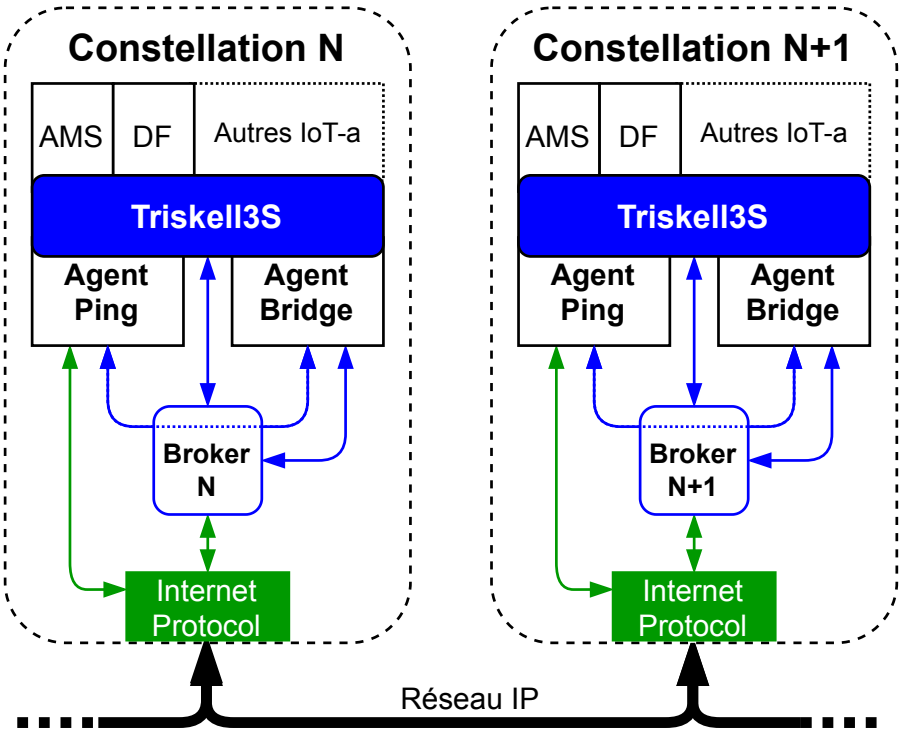


FIGURE 6.4 – Architecture IoT-a de plusieurs modules d'affichage

6.2.2 | *Agents Ping et Bridge*

L'agent Ping est responsable de la découverte des constellations voisines et de la stratégie permettant leur raccordement. Nous avons décrit ses comportements dans la section 4.2.3. Néanmoins, nous avions laissé libre la technologie permettant la découverte des constellations voisines. Dans cette expérimentation, nous avons fait le choix de lui intégrer le protocole *Internet Control Message Protocol*(ICMP)[Postel, 1981]. Ce protocole de niveau 3 sur le modèle OSI permet de véhiculer des messages de contrôle et d'erreur. Il est à l'origine de la commande *ping* sur les systèmes d'exploitation. Lorsqu'un hôte destinataire reçoit un message de la sorte et qu'il est en capacité de répondre, il renvoie un message avec le type *Echo-reply*. Dans le cas où l'hôte destinataire n'est pas joignable correctement, un message retourne un code caractérisant le problème rencontré.

Comme évoqué dans la figure 6.4 l'interconnexion des constellations repose sur le standard IP. Ainsi, la mise en place du protocole ICMP dans notre agent Ping permet de disposer d'informations bas niveaux et concrètes sur l'état du matériel testé. Ce choix ouvre le champ des possibles en ce qui concerne la découverte d'éléments sur le réseau et ne limite pas les comportements de l'agent à cette seule expérimentation.

Concernant son architecture, l'agent Ping intègre un comportement de type *Periodic*. Il vérifie, à chaque pas de temps, l'état de connexion de sa constellation ainsi que la disponibilité des autres constellations en fonction de la stratégie de ping. Pour rappel, les deux stratégies correspondent à la mise en place d'une topologie en ligne ou en étoile. Pour des raisons pratiques, l'agent Ping ne peut proposer qu'une seule stratégie à la fois.

L'agent Ping a connaissance de sa constellation d'appartenance. Il connaît l'adresse IP de la constellation la plus basse (x.x.x.0) et la plus haute (x.x.x.255) sur le réseau. Cela permet de borner le nombre d'adresses à analyser et améliorer les latences. À chaque pas de temps, il commence par déterminer si sa constellation n'est pas celle qui a été déconnectée. Si c'est le cas, il notifie la plate-forme et se met en attente jusqu'au prochain pas de temps. Une fois la constellation de nouveau connectée, l'agent Ping la détecte puis notifie la plate-forme avant de poursuivre son exécution. Lorsque la constellation est correctement connectée, l'agent Ping réalise l'analyse de ses voisins afin de connaître son environnement. Les algorithmes 1 et 2 présentés dans la section 4.2.3 sont alors mis en œuvre pour déterminer quels liens il est nécessaire de créer ou de supprimer. À l'issue de cette analyse, l'agent Ping fait parvenir un message à l'agent Bridge afin de créer effectivement les bridges dynamiquement.

L'agent Bridge est responsable de la création et de la suppression des bridges dynamiques au sein d'une constellation. Il a conscience des liens qu'il a déjà mis en place. Il régule les demandes de création ou de suppression en n'effectuant que les liens nécessaires. Si par exemple l'agent Ping demande la création d'un bridge alors qu'il existe déjà, l'agent Bridge n'effectuera pas de modification.

L'agent Bridge intègre un comportement de type *Event* lui permettant d'effectuer des actions à la réception de certains types de messages. Il est en écoute du topic `ACC/Bridge/inform` et s'attend à recevoir une liste d'adresses IP sur lesquelles le broker doit établir des bridges. Lorsqu'il est sollicité pour créer un bridge, il publie un message sur le topic `$SYS/broker/bridge/new`. Pour supprimer un bridge, il publie sur le topic `$SYS/broker/bridge/del`.

6.2.3 | *Étude de la résilience*

Nous souhaitons étudier dans cette preuve de concept, l'apport en résilience de notre approche de bridges dynamiques. Dans le cas où des écrans tomberaient en panne lors de l'utilisation du mur d'écrans en agence, une solution doit permettre d'assurer le fonctionnement du système, le rendant alors résilient. Pour cela, nous effectuons un enchaînement de phases, illustrées par la figure 6.6, durant lesquelles les liens entre écrans sont mis à l'épreuve au travers de la déconnexion et la reconnexion, à la volée, d'un écran. Les contraintes données dans cette expérimentation sont de créer et supprimer le minimum de bridges et d'assurer au maximum un bridge par constellation. Pour simplifier l'analyse, nous nous concentrons dans cette première expérimentation sur une stratégie de lien permettant une topologie en étoile. Dans une phase initiale, non représentée sur la figure, aucune constellation ne se connaît. Leurs agents Ping et Bridge initialisent alors une connexion vers la constellation 1.

Plusieurs ruptures de lien de communication entre les modules d'affichage sont effectuées durant cette expérimentation. Afin de mesurer leurs incidences sur le fonctionnement du système, une constellation de référence,

ne subissant aucune déconnexion, intègre un script publiant un flux continu de messages à l'ensemble du système. Chaque constellation dispose alors d'un programme comptabilisant les messages envoyés par ce script. Durant un fonctionnement normal, lorsqu'aucun module d'affichage ne connaît de rupture de connectivité, tous les programmes de comptabilisation perçoivent le même nombre de messages de la part du script. Néanmoins, si une rupture de liens apparaît, alors les constellations concernées ne reçoivent plus les messages de la constellation de référence. Une différence dans le nombre de messages qu'elles reçoivent est alors observable. Dans ce cas de figure, les constellations réagissent en réorganisant dynamiquement leurs bridges afin de recevoir de nouveau le flux de messages du script ayant pour effet de stabiliser les réceptions. Les résultats de ces observations sont illustrés par la figure 6.5 dans laquelle la constellation 5 est la référence.

Durant la phase 1, tous les modules d'affichage sont reliés à la première constellation, elle est le nœud de la topologie en étoile. Les liens sont fonctionnels et les messages émis par le script sont reçus normalement. À partir de la seconde phase, et ce jusqu'à la quatrième, le système subit les déconnexions successives des modules d'affichage 1, 2 puis 3. Lors de ces phases, le système réagit automatiquement aux déconnexions. Les agents Ping et Bridges des constellations toujours fonctionnelles, détectent la perte de l'élément principal de la topologie en étoile et créent les liens nécessaires vers un nouveau nœud. Nous pouvons observer sur le graphique représenté par la figure 6.5 un temps de détection α due à l'analyse périodique de l'agent Ping.

À compter de la phase 5, les trois premiers modules d'affichage sont de nouveau connectés au système. D'après les contraintes que nous avons présentées précédemment, le module 1 redevient donc l'élément au centre de la topologie en étoile. Les constellations suppriment leurs liens vers le module 4 pour se raccorder de nouveau au premier. Cette caractéristique fait émerger un temps γ non négligeable sur la figure 6.5. Il est dû à un comportement des brokers que nous appelons "bridges fantômes". En effet, les bridges mis en place précédemment ne sont pas supprimés, ils sont donc toujours présents en mémoire dans le broker de chaque constellation. Lorsque qu'un module d'affichage anciennement élu comme nœud des liens du système est de nouveau présent, les anciens bridges enregistrés dans les constellations se raccordent. Cette particularité permet de limiter le nombre de créations et de suppressions de bridges. Néanmoins, la gestion des connexions d'anciens bridges par les brokers provoque une latence importante au moment de reconnecter les modules. Le temps γ résulte donc du temps d'analyse α et du temps de traitement des bridges fantômes pouvant prendre plusieurs secondes. Ce paramètre explique les retards successifs de reconnexions des constellations 2 puis 3. Elles sont pourtant physiquement raccordées au système mais elles ne se reconnectent à nouveau qu'à 51 et 64 secondes.

Afin d'obtenir un indicateur validant la pertinence de notre apport, nous comparons notre approche par bridges dynamiques avec une méthode par bridges conventionnelles, sans reconnexion automatique. Nous parlerons dans ce cas de bridges statiques. L'évolution du nombre de messages reçus par le broker 1 correspond aux résultats par bridge statique. En effet, si un broker était reliée statiquement au broker 1, il ne pourrait que recevoir un nombre de messages inférieur ou égal à ceux du broker 1, car il est le premier à subir une déconnexion. En fin d'expérimentation, une perte d'environ 43% est constatée par rapport aux résultats obtenus sur le broker 5 de référence. Par ailleurs, le broker 4 est celui qui connaît le plus de connexions et de déconnexions dynamiques de bridges, c'est donc lui le plus représentatif de cette approche dynamique. Lorsque nous analysons le nombre de messages qu'il a reçu en fin d'expérimentation, nous constatons une perte d'environs 20% par rapport à la référence donnée par le broker 5.

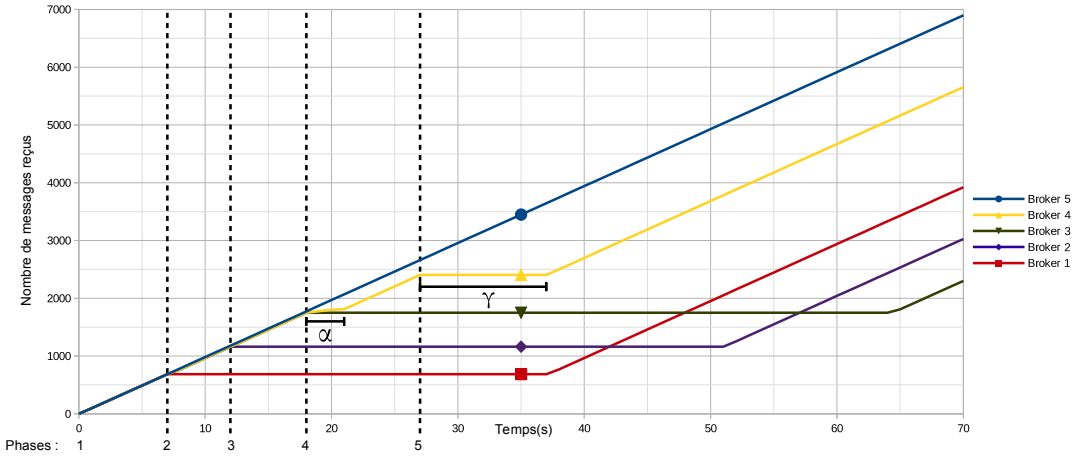
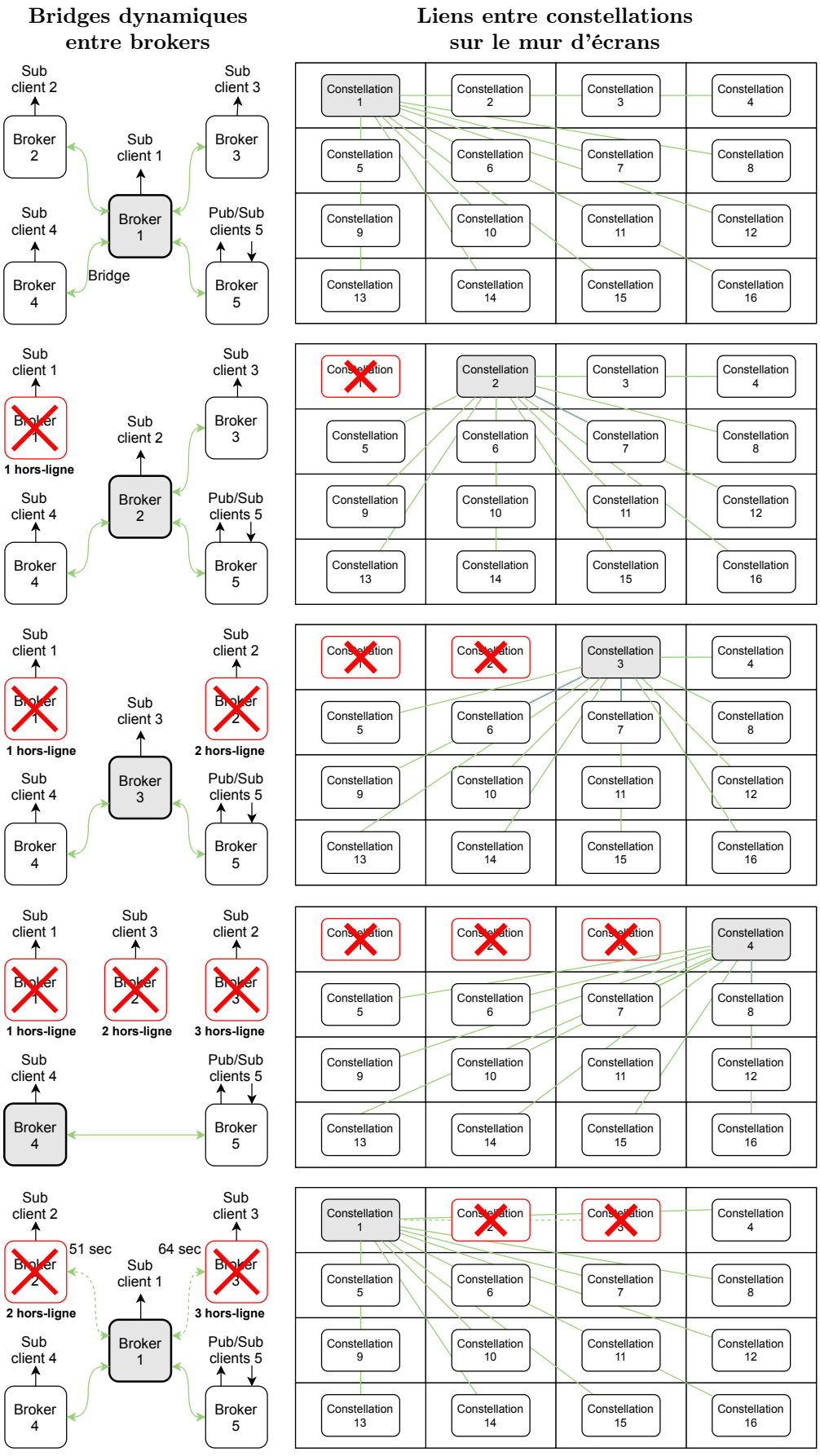


FIGURE 6.5 – Nombre de messages reçus par les programmes de comptabilisation pour chaque module d'affichage



(a) Phase 1 : Toutes les constellations sont reliées en étoile avec comme nœud central la première d'entre elles.

(b) Phase 2 : La constellation 1 est déconnectée. Les autres se connectent alors sur la seconde.

(c) Phase 3 : Les constellations 1 et 2 ne sont plus fonctionnelles. La constellation 3 devient le nouveau nœud central de l'étoile des liens.

(d) Phase 4 : Les trois premières constellations sont inopérantes. Toutes les constellations se relient à 4.

(e) Phase 5 : Les constellations 1, 2 et 3 sont de nouveau opérationnelles. Les autres constellation abandonnent la 4 comme nœud centrale et se connectent de nouveau à 1.

FIGURE 6.6 – Phases successives de l'expérimentation étudiant la résilience des liens entre constellations

6.2.4 | *Analyse des performances de connexion*

Dans cette seconde étude, nous souhaitons analyser les performances de connexion des liens dynamiques. Nous avons déterminé que les bridges fantômes peuvent être un facteur contraignant les performances. Aussi, nous ne limitons plus le nombre de créations et suppressions de bridges dans cette étude. L'agent Bridge est désormais capable de vérifier si sa constellation est le nœud principal d'une topologie en étoile. Dans ce cas, il supprime les liens vers l'ancien nœud.

Nous étudions, de plus, la topologie de liens dynamiques en ligne afin de la comparer à celle en étoile. Nous pouvons ainsi expérimenter nos algorithmes de stratégie de ping entre constellations d'IoT-a.

Deux mesures ont été effectuées pour caractériser chaque stratégie. Elles illustrent la déconnexion puis la re-connexion d'un seul broker. Dans la figure 6.7, cinq brokers sont reliés initialement en ligne. Un défaut est occasionné (déconnexion d'un module d'affichage). Le broker 3 est alors déconnecté. Les liens se réorganisent puis après un temps suffisant pour l'analyse, il est de nouveau connecté au système. Dans la figure 6.8, cinq brokers sont connectés en étoile sur le broker 1. Le nœud central est déconnecté, les liens se réorganisent puis après quelques secondes le broker 1 est de nouveau connecté au système. Avec cette procédure il devient ainsi possible d'analyser les performances de connexion obtenues durant la réorganisation des liens en mesurant le débit de chaque broker. Comme pour l'étude précédente, un agent témoin publie un flux continu de messages depuis la constellation 5. Au début de chaque mesure, aucune constellation n'a connaissance de l'état de ses voisines et aucune configuration ne leur permet d'établir un bridge.

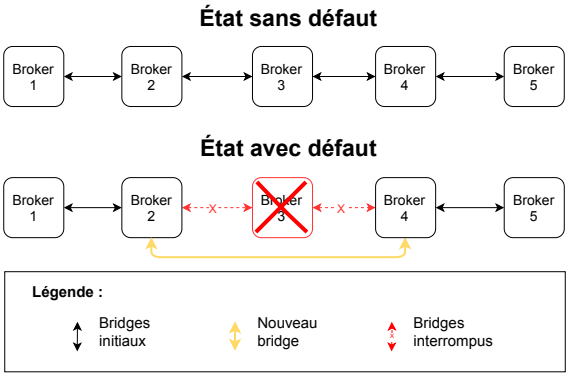


FIGURE 6.7 – Expérimentation d'une topologie en ligne

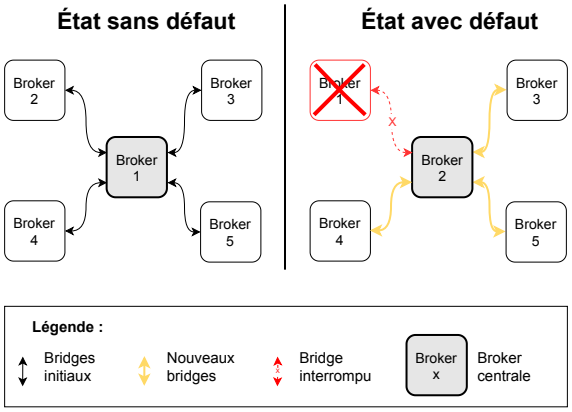


FIGURE 6.8 – Expérimentation d'une topologie en étoile

Les figures 6.9 et 6.10 détaillent les résultats obtenus pour la stratégie en ligne puis en étoile. Afin de simplifier la lecture des résultats, nous ne faisons apparaître que les données significatives des constellations concernées par la déconnexion et la re-connexion. Nous constatons trois temps importants : la connexion de l'ensemble des constellations sur leur cible déterminée par leur stratégie (marqueur C), la déconnexion de la constellation centrale (marqueur D) et sa re-connexion (marqueur R).

Nous pouvons observer sur les deux graphiques les pics de connexion au niveau des marqueurs C. Ils sont dus aux échanges pour la création de bridges durant l'initialisation du système.

Dans le cas de la stratégie en ligne, nous observons que l'agent Ping, présent sur la constellation 4, détecte la déconnexion de la constellation 3 en ≈100ms. Il propose alors à l'agent Bridge une solution de connexion vers le broker de la constellation 2. Il est possible de constater pour la stratégie en étoile qu'au moment de la déconnexion de la première constellation, les agents Ping présents sur les modules d'affichage 3, 4 et 5 proposent un nouveau nœud vers la constellation 2. Dans les deux types de stratégie, la reprise des communications s'effectue en moins d'une seconde et la perte de messages est limitée. Enfin, le recouvrement des courbes aux marqueurs R montre que la reconnexion ne provoque aucune perte de message dans les deux types de stratégies.

Les résultats que nous obtenons confirment encore une fois la résilience de notre système, quelle que soit la stratégie adoptée. Chacune de nos constellations s'adapte aux autres disponibles sur le réseau. Bien que responsable d'un faible surcoût de communication, le coût en bande passante pour la création dynamique de bridges reste négligeable par rapport au reste des communications. Cette caractéristique permet d'ailleurs de

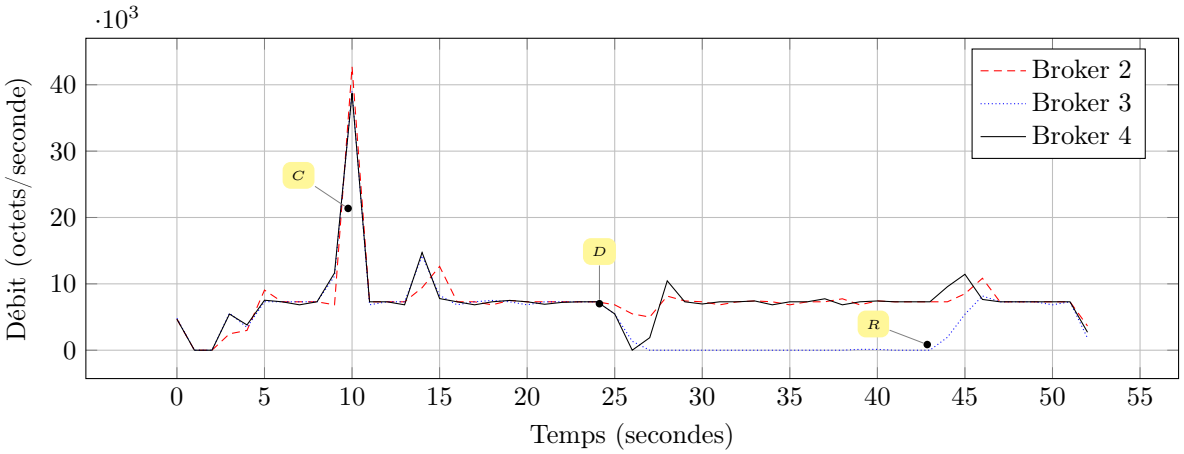


FIGURE 6.9 – Débit par broker pour l'expérimentation en ligne

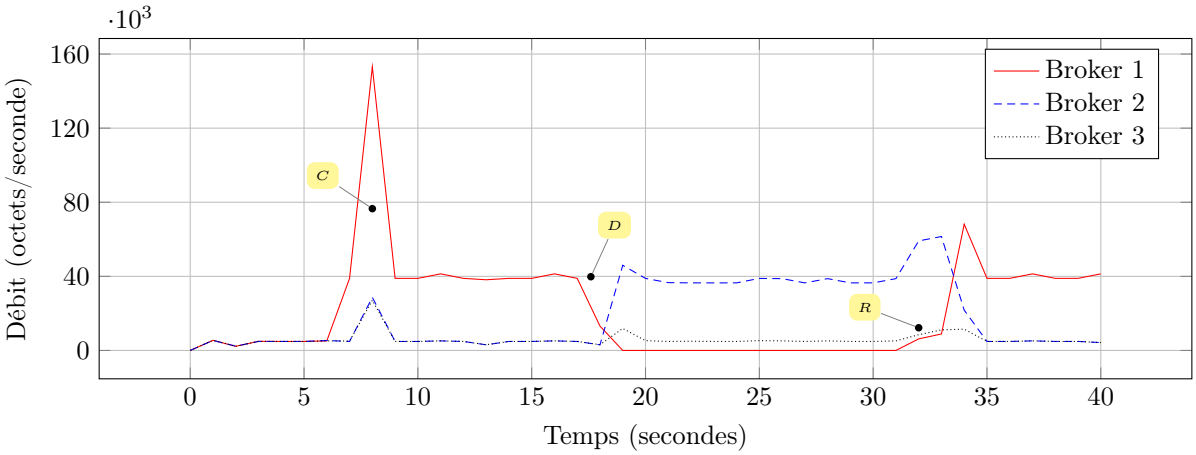


FIGURE 6.10 – Débit par broker pour l'expérimentation en étoile

correspondre aux contraintes des systèmes embarqués disposant de performances de communication limitées. Nous notons aussi la réactivité de notre solution qui réorganise l'ensemble du maillage du réseau en moins d'une seconde pour chaque événement.

	Débit moyen sur le nœud central	Débit moyen d'un élément	Nombre total de bits perdus
Topologie en ligne	7.1 Ko/s	7.1 Ko/s	60.1 Ko
Topologie en étoile	38.4 Ko/s	4.6 Ko/s	21.8 Ko

Tableau 6.4 – Comparaison de la bande passante moyenne entre les deux topologies

Nous sommes donc en présence de deux types de topologies capables de s'adapter aux aléas de connexion qui surviennent dans notre système de visualisation. La topologie en étoile est statistiquement moins sensible qu'une topologie en ligne, car son nœud central est moins exposé. Elle assure une meilleure résilience en perdant moins de messages mais déséquilibre fortement les débits de chaque constellation, comme nous le montre le tableau 6.4. Une topologie en ligne équilibre mieux les charges de communication et correspond donc mieux à notre approche de décentralisation collective. Cet équilibrage est essentiel pour les IoT-a, car leurs performances sont souvent limitées. De plus, notre proposition de décentralisation collective souhaite harmoniser la communication des objets. Une topologie de ce type offre les mêmes caractéristiques de communication quel que soit la constellation concernée.

La réactivité de notre proposition permet d'assurer une résilience suffisante pour une topologie en ligne et nous permet de l'adopter comme topologie de référence pour les autres expérimentations.

Notre proposition de bridges dynamiques permet donc aux constellations de conserver une connectivité même si l'un des éléments du système vient à manquer. Le mécanisme permettant aux constellations de traiter le départ d'un élément est le même que pour l'arrivée d'une nouvelle constellation. Il est donc possible d'intégrer, à la volée, de nouvelles constellations dans le système, comme des modules d'affichage supplémentaires par exemple.

Enfin, notre expérimentation soulève deux aspects.

D'une part notre approche consistant à relier des constellations d'IoT-a dynamiquement permet effectivement d'augmenter la résilience du système. Les liens dynamiques entre constellations majorent de plus de 20%, dans le cas de notre expérimentation, le nombre de messages véhiculés lors de coupures de connexion de certains éléments du système. Ces résultats sont néanmoins à mettre en perspective, car ils peuvent fortement varier en fonction de l'expérimentation, mais restent toujours supérieurs par rapport à une méthode par bridges statiques.

D'autre part, la contrainte concernant la limitation de créations et suppressions de bridges semble restreindre les performances de connexion. En effet, cette contrainte met en lumière la question des bridges fantômes qui ralentissent un retour à un fonctionnement initial, comme évoqué en phase 5. Nous constatons que, si nos agents Ping et Bridge suppriment les bridges d'un précédent nœud à chaque modification des liens, alors l'effet de bridge fantôme n'a plus lieu. Les performances obtenues sont donc supérieures malgré une augmentations des communications entre les agents Ping et Bridge, néanmoins négligeables par rapport au reste des communications.

Page laissée intentionnellement vide.

6.3 | Système de visualisation résilient

Nous avons observé dans notre première preuve de concept comment les constellations arrivent à s'adapter face à la perte de certains éléments réseau en fonction de certaines topologies. Nous avons d'ailleurs déterminé que la topologie en ligne était la plus adaptée pour correspondre à notre approche de décentralisation collective. Ces travaux représentent la base des communications entre nos IoT-a et ils ouvrent les portes d'autres expérimentations. Nous pouvons désormais nous intéresser à la mise en œuvre d'agents de plus haut niveau matériel, prenant en compte les informations bas niveau de nos constellations concernant les liens dynamiques.

Dans cette seconde preuve de concept, nous poursuivons nos travaux concernant la résilience de notre système et nous nous intéressons plus particulièrement à une application portée par nos constellations. Pour cela, nous souhaitons tester la réactivité de notre système face à la perte d'un ou plusieurs modules d'affichage durant la diffusion de visuels. L'objectif est que le système puisse adapter la visualisation en fonction de la surface d'affichage disponible. Nous proposons ainsi une étude de cas dans laquelle notre système est soumis à plusieurs modules d'affichage défectueux.

6.3.1 | *Etude de cas*

Un système de visualisation composé de seize modules d'affichage répartis sous la forme d'une grille de 4×4, retransmet plusieurs visuels à des utilisateurs (logiciel professionnel, interface web, flux vidéo, etc.). Ces visuels sont au nombre de quatre (de V1 à V4) et ils sont répartis uniformément sur le mur d'écrans comme l'illustre la figure 6.11. Les pointillés représentent les modules contigus d'affichage formant le mur d'écrans.

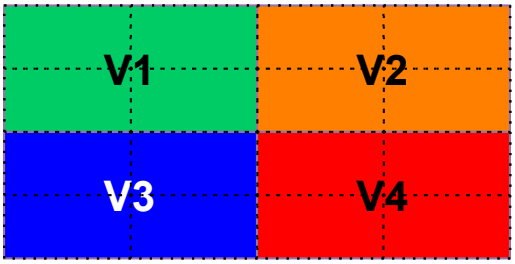


FIGURE 6.11 – Quatre visuels diffusés sur le mur d'écrans

Comme l'illustre la figure 6.12 plusieurs défauts surviennent sur des modules d'affichage, impactant l'observation de certains visuels. Face à ces défauts, nous souhaitons que le système puisse déterminer une solution afin de continuer à diffuser correctement l'ensemble des visuels. La stratégie envisagée est alors d'adapter la géométrie des visuels pour leur laisser la possibilité d'être diffusés sur les modules d'affichage toujours fonctionnels. Chaque visuel dispose d'un paramètre qualifiant sa priorité par rapport aux autres. Si le système fonctionne dans son mode nominal, le paramètre n'est pas pris en compte. Par contre, si le système connaît un défaut, alors il favorise l'affichage des visuels ayant la plus haute priorité. Si tous les visuels sont régis par la même valeur, alors le système équilibre leur géométrie tout en maximisant leurs proportions en hauteur et largeur. Dans notre cas d'étude, nous configurons tous les visuels au même niveau de priorité.

Dans le cas du premier défaut, l'affichage du visuel V1 est compromis. Sa géométrie et son emplacement doivent donc être modifiés pour continuer à être affiché. Le second visuel V2 est alors impacté pour maximiser les proportions de V1. Il modifie légèrement sa géométrie et son emplacement afin de laisser la place à V1 pour s'adapter à la place résultante. V1 et V2 peuvent ainsi être affichés l'un à côté de l'autre sur la partie supérieure du mur d'écrans, en laissant inutilisé l'écran supérieur gauche.

De manière analogue, le visuel V3 impacte V4 pour pallier le second défaut. Les deux visuels s'adaptent aux six écrans inférieurs gauche disponibles, en laissant le douzième module inactif. Si l'un des visuels V2 ou V3 avait été plus prioritaire que leur voisin, alors le système ne les aurait pas impactés. Il aurait modifié la géométrie du visuel concerné par le défaut pour le faire correspondre à l'un des écrans fonctionnel.

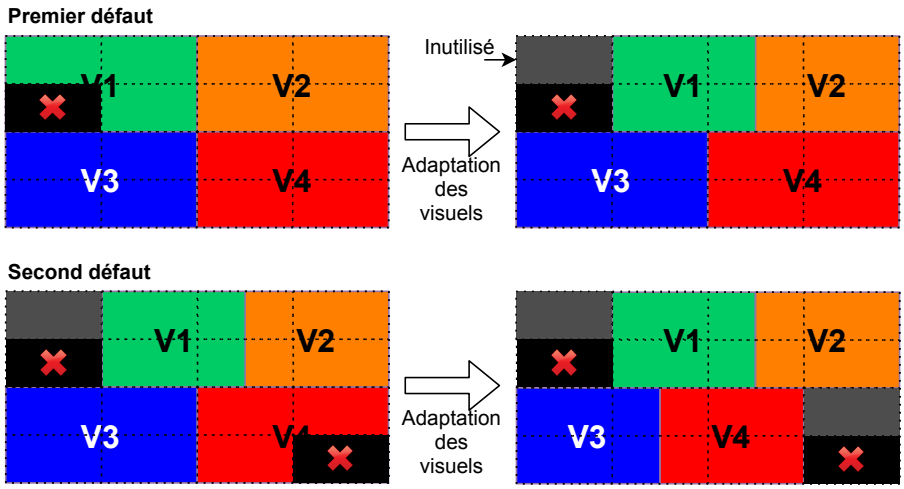


FIGURE 6.12 – Adaptation des visuels après deux défauts successifs dans les modules d'affichage

6.3.2 | *Adaptation dynamique de la visualisation*

Afin de répondre à notre étude de cas au moyen de notre modèle 5M nous devons détailler les agents nécessaires à notre expérimentation. Nous avons présenté dans le chapitre 5 le modèle 5M avec les fonctions de composition et scénarisation responsables de la génération et l'organisation de visuels sur un espace de diffusion. Ces fonctions sont chacune implémentées sous la forme d'un agent dans une constellation. Ces deux agents communiquent par le canal de communication MQTT et permettent la diffusion de flux d'images vers des modules d'affichage. Chaque module intègre d'ailleurs un agent visualiseur capable d'interpréter et diffuser ce flux d'image sur l'écran du dispositif. Nous nous reporterons à la figure 5.11 pour un rappel de l'architecture de ces agents.

Dans une situation statique, l'agent de scénarisation disposerait dans un fichier de configuration des caractéristiques d'affichage à propos de chaque module afin d'organiser les visuels à diffuser. Aucune interaction entre la scénarisation et la présence des modules d'affichage ne serait alors nécessaire pour maximiser l'organisation des visuels sur l'espace de diffusion. Néanmoins, dans le cas d'un élément défectueux ou de l'ajout d'un nouveau module d'affichage aucune rétroaction vers la fonction de scénarisation n'est possible. Nous pouvons donc percevoir l'importance d'une approche dynamique de la scénarisation si nous voulons rendre notre système plus résilient.

Augmenter la résilience d'un système passe d'abord par permettre à ces composants de prendre connaissance de leur environnement. C'est pourquoi l'agent de scénarisation doit être en mesure de connaître les éléments qui constituent un espace de diffusion. Pour répondre à cet objectif, nous ajoutons un comportement de type *Periodic* permettant d'interroger périodiquement les modules d'affichage au sujet de leur présence, mais aussi de leurs caractéristiques spatiale et technique (résolution, position absolue, luminosité, etc.). Les modules d'affichage se doivent alors de répondre à l'agent de scénarisation. C'est le rôle de l'agent visualisation présent dans chacun d'eux. En plus d'être chargé de la récupération et la diffusion du flux d'images, proposé par l'agent composition, l'agent visualisation est capable de répondre à une demande de présence et d'informations.

Disposant de cette connaissance, l'agent de scénarisation peut donc réorganiser les visuels en fonction de la présence des modules d'affichage et de leurs caractéristiques. La figure 6.13 illustre le flux de messages permettant à l'agent de scénarisation de connaître les agents visualisation. Dans le cas où un module n'est plus connecté, les constellations voisines réorganisent leurs bridges et les modules d'affichage fonctionnels sont de nouveau joignables. Lorsque l'agent de scénarisation interroge les agents de visualisation, seuls ceux portés par des modules fonctionnels répondent. La visualisation peut donc être adaptée en conséquence. Si jamais le module déconnecté est de nouveau disponible, alors l'agent de scénarisation le détecte à l'occasion de sa prochaine interrogation et organise les visuels comme ils étaient initialement.

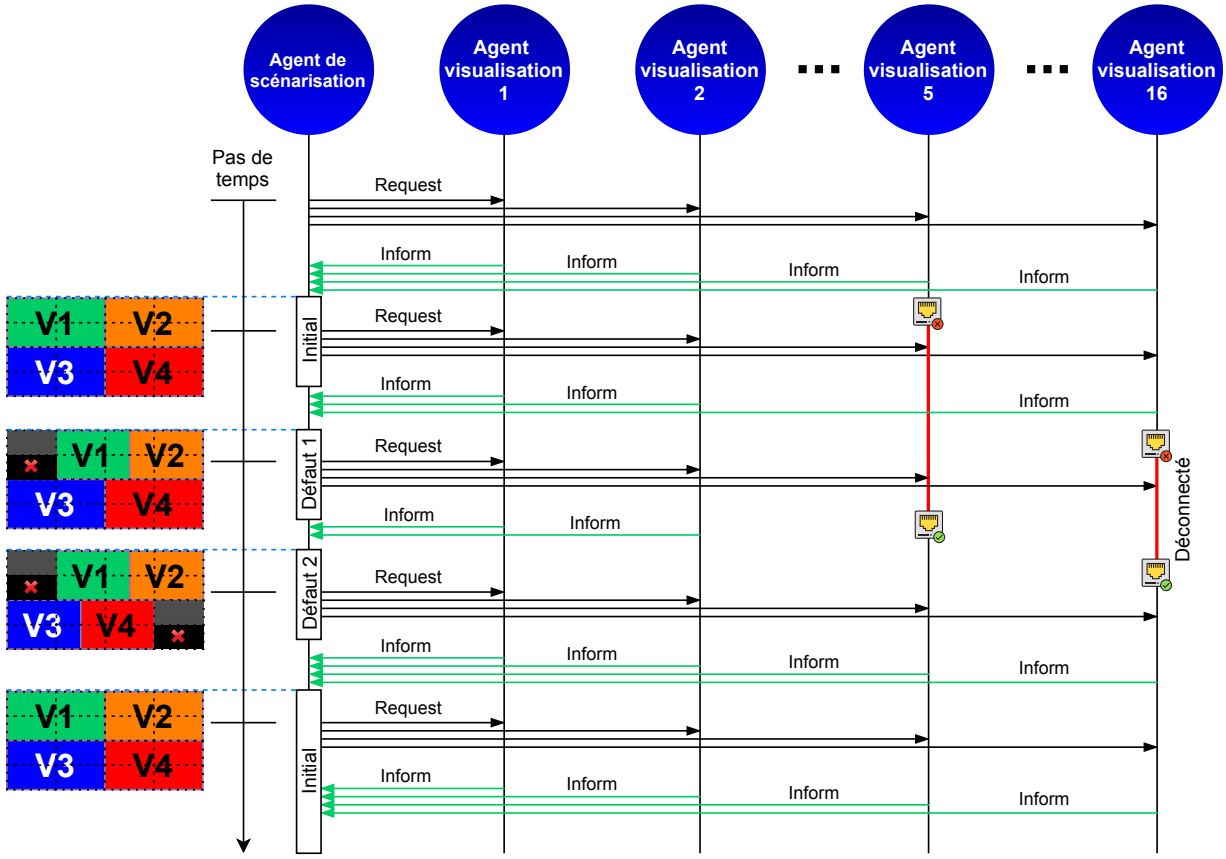


FIGURE 6.13 – Flux de messages pour l'adaptation des visuels en fonction des modules d'affichage disponible

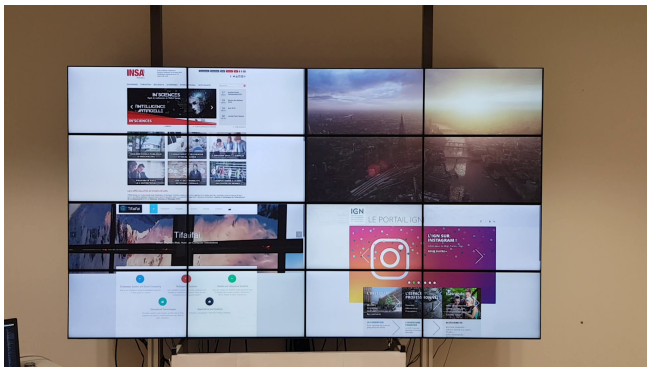
6.3.3 | *Résultats*

La figure 6.14 illustre les cinq phases de l'expérimentation menée sur notre mur d'écrans. En phase initiale quatre visuels différents sont diffusés. Dans la seconde phase de l'expérimentation, le cinquième module d'affichage est arrêté, il n'est plus capable de diffuser des contenus. Le module d'affichage six détecte l'anomalie, il supprime son bridge vers le module cinq et se raccorde au quatrième afin de reconnecter la ligne de constellations. L'agent de scénarisation envoie ses ACL messages de type *Request* aux seize agents visualiseur et seulement quinze lui répondent. Il exécute alors sa fonction d'adaptation des visuels pour les faire correspondre aux modules disponibles, puis fait part des géométries de chacun à l'agent composition qui les modifie. Nous obtenons alors le résultat proposé dans la troisième phase. Les deux visuels impactés sont adaptés aux six écrans supérieurs disponibles tout en laissant le premier module d'affichage inactif, car aucune solution n'a été déterminée pour l'utiliser.

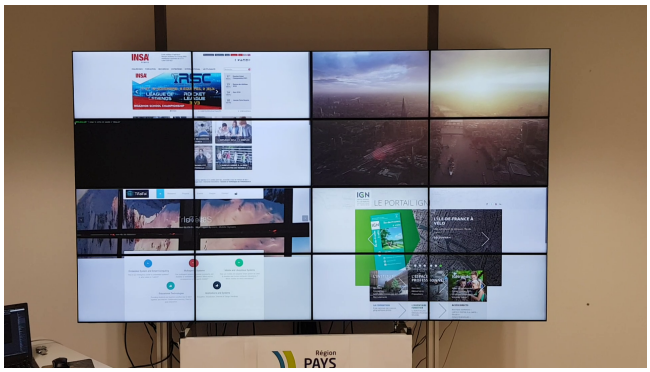
Durant la quatrième phase, c'est le module d'affichage 16 qui est déconnecté. L'agent Ping du quinzième module détecte alors qu'il est le module de queue dans la topologie en ligne qui relie toutes les constellations. L'agent scénarisation détecte aussi l'absence du module et adapte une seconde fois la visualisation pour donner le résultat obtenu dans la cinquième phase.

Enfin, les modules cinq et seize sont de nouveau connectés et le système revient dans son état initial comme illustré sur la phase 1 de l'expérimentation.

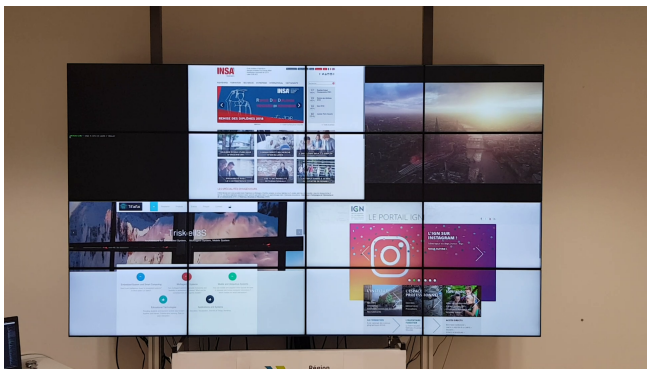
Ces résultats montrent la résilience et la dynamique de notre système de visualisation. Il permet la prise en compte, à la volée, des modules d'affichage disponibles et modifie, sans interruption de service, les données affichées. L'espace de communication commun, développé par notre approche de décentralisation collective, permet à l'agent scénarisation, placé sur une machine distante, de communiquer de façon transparente avec les différents agents visualisation présents sur d'autres constellations. Enfin, notre proposition de bridge dynamique permet une réorganisation des liens entre constellations sans impacter le fonctionnement des agents des plus hauts niveaux comme montré ici avec les agents visualisation.



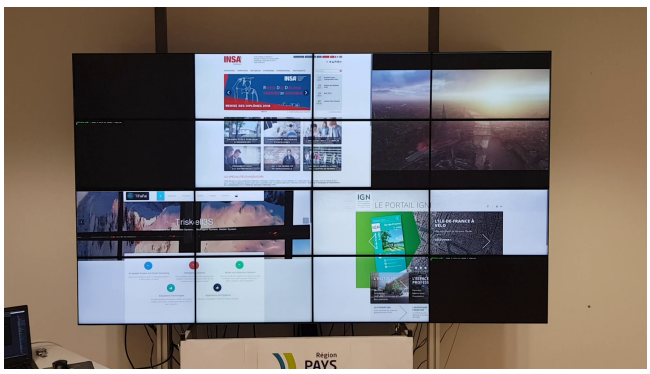
(a) Phase 1 : Quatre visuels sont affichés sur un mur de 16 modules d'affichage, repartis selon une matrice 4×4.



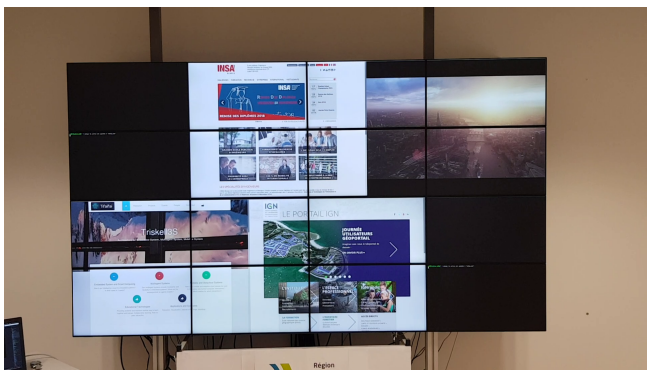
(b) Phase 2 : Le module d'affichage 5 est arrêté. Il n'est plus capable d'afficher du contenu. Une partie du visuel suppérieur gauche n'est plus visible.



(c) Phase 3 : Le système détecte l'anomalie. Les connexions réseau se réorganisent puis l'affichage des visuels s'adapte à la nouvelle surface d'affichage.



(d) Phase 4 : Un second module d'affichage est arrêté. Le visuel inférieur droit n'est plus visible entièrement.



(e) Phase 5 : Le système détecte l'arrêt du module 16. Les connexions réseau se réorganisent une seconde fois et les visuels s'adaptent pour proposer un nouvel affichage avec les modules d'affichage disponible.

FIGURE 6.14 – Phases successives de l'expérimentation étudiant la résilience du mur d'écrans

6.4 | Système de visualisation multi-utilisateurs

Dans cette troisième preuve de concept, nous poursuivons nos analyses concernant l'adaptation dynamique de visuels sur notre système d'affichage. Nous nous intéressons plus particulièrement aux questions soulevées par la visualisation multi-utilisateurs.

Nous avons présenté dans le chapitre 5 les types d'utilisateurs susceptibles d'avoir recours à notre système d'affichage dans une agence SARP afin qu'ils puissent améliorer leur vision d'indicateurs donnés. Par exemple, un assistant d'exploitation peut visualiser la position en temps réel des opérateurs sur le terrain, en plus d'une vue globale du planning pour modifier certaines informations en conséquence. En parallèle, un directeur d'agence a la possibilité de focaliser son activité sur des données financières et commerciales. Nous faisons face, dans cette situation, à différents scénarios d'affichage organisé en fonction de la qualité du personnel devant le mur d'écrans. Comme pour la preuve de concept précédente, nous nous intéressons à une étude de cas concret.

6.4.1 | *Étude de cas*

Un premier utilisateur souhaite visualiser quatre interfaces concernant son activité sur le mur d'écrans. Il estime qu'elles sont d'égale importance et désire qu'elles soient réparties uniformément sur l'espace d'affichage. Nous appelons cette première demande le *Scénario 1* comme l'illustre la figure 6.15.

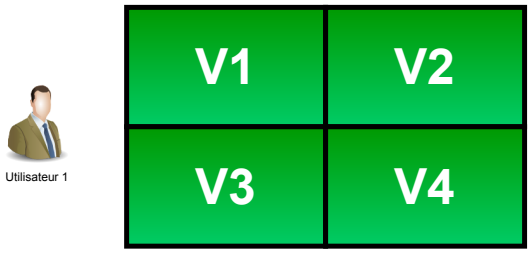


FIGURE 6.15 – Scénario 1

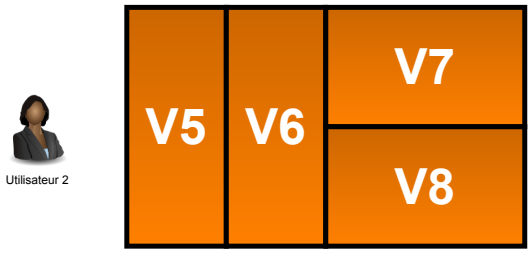


FIGURE 6.16 – Scénario 2

Un second utilisateur, ayant un poste différent du premier, souhaite lui aussi visualiser quatre interfaces différentes sur le mur d'écrans. Deux d'entre elles nécessitent d'être affichées verticalement pour maximiser la vue de certaines informations. À la différence du premier utilisateur, ce dernier détermine certains des visuels comme étant plus ou moins importants. Il qualifie notamment les visuels V5 et V8 comme étant moins prioritaires par rapport à V6 et V7. Cette seconde demande est nommée *Scénario 2* comme l'illustre la figure 6.16.

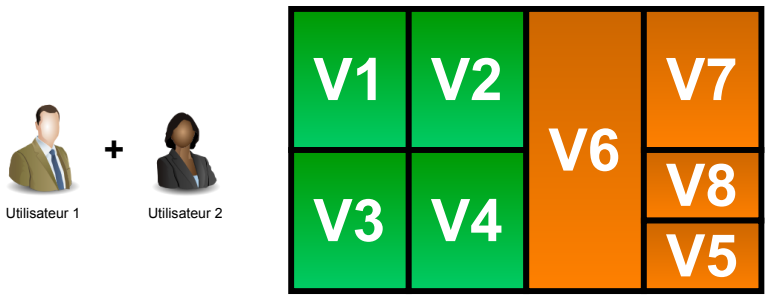


FIGURE 6.17 – Association des scénarios de l'utilisateur 1 avec celui de l'utilisateur 2

Un dispositif d'interaction permet de déterminer quel utilisateur est présent devant le mur d'écrans. Lorsqu'un des deux utilisateurs se trouve face au système, il est détecté et l'affichage s'adapte à la volée pour diffuser l'un ou l'autre des scénarios présentés. Cependant, si les deux utilisateurs sont présents en même temps, devant le mur d'écrans, alors une adaptation de leurs deux scénarios doit pouvoir être diffusée sur les modules d'affichage. Nous parlons alors d'une association de leurs scénarios comme proposés dans la figure 6.17. Cela permet de favoriser l'affichage des visuels prioritaires et minimiser les autres.

6.4.2 | *Mise en œuvre*

Pour mettre en œuvre notre preuve de concept, l'agent de scénarisation doit avoir conscience des utilisateurs présents devant l'espace de diffusion. Pour cela, nous introduisons un nouvel agent *Utilisateur*, représentant chaque personne physique. Lorsqu'une personne est détectée par un dispositif captation, se dernier crée un agent Utilisateur à son nom. Le rôle principal de cet agent est donc de prévenir l'agent de scénarisation de sa présence. À l'issue de sa création et de son enregistrement auprès d'AMS, il contacte l'agent de scénarisation en lui faisant parvenir un ACL message de type *Inform*. Le message contient son nom et une description du scénario et des priorités associées aux visuels qu'il souhaite voir. L'agent de scénarisation intègre alors la présence et les demandes de l'utilisateur pour adapter la diffusion. Au départ de l'utilisateur, le dispositif de captation avertit l'agent représentant cet utilisateur. L'agent concerné envoie alors un message de type *Inform* à l'agent de scénarisation pour qu'il soit conscient de son départ, puis il se termine. La figure 6.18 illustre les différentes étapes de notre preuve de concept avec l'arrivée puis le départ des deux utilisateurs.

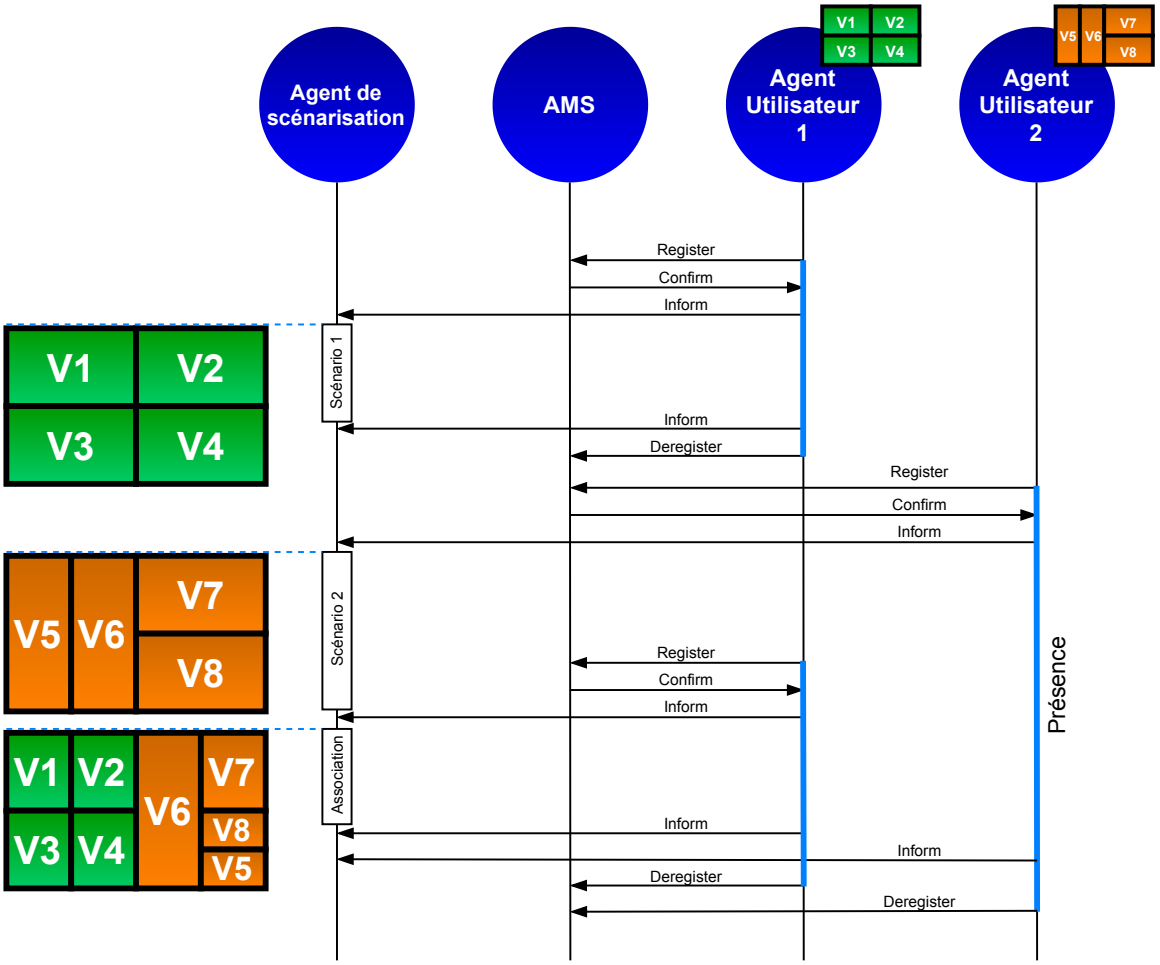
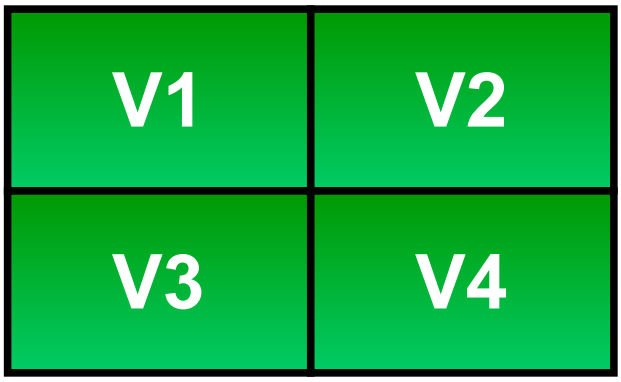


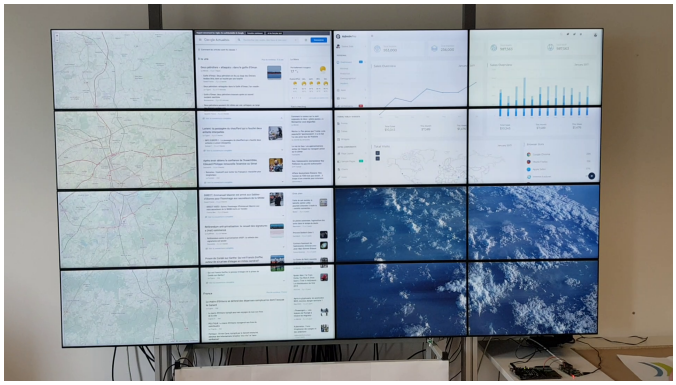
FIGURE 6.18 – Flux de messages pour la gestion de l'adaptation des visuels en fonction des utilisateurs présents

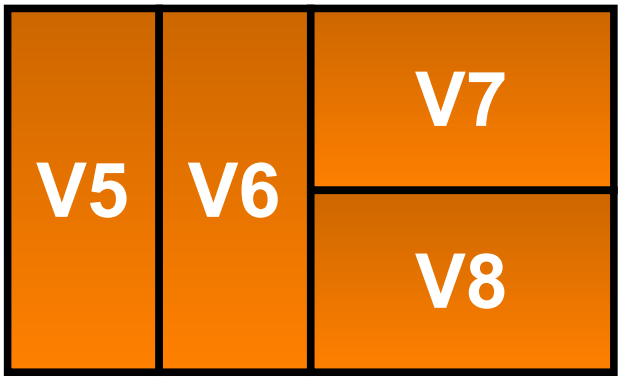
Nous retrouvons dans la figure 6.19 les trois phases de l'expérimentation menée sur notre mur d'écrans. Dans une première phase, l'utilisateur 1 est reconnu par le dispositif qui crée son agent correspondant. Se dernier informe l'agent de scénarisation de sa présence et les visuels correspondants sont alors diffusés sur le mur d'écrans. Quelques temps plus tard, il se retire et cède sa place à l'utilisateur 2. De manière analogue à la première phase, le système diffuse le second scénario. Durant la troisième phase, le premier utilisateur rejoint le second devant le dispositif. Une fois les deux personnes identifiées par le système, l'agent de scénarisation doit faire un choix pour associer les deux scénarios tout en prenant en compte les priorités de chaque visuel. Alors que les visuels continuent d'être diffusés, les proportions des quatre premiers sont adaptées pour correspondre à la moitié gauche du mur d'écrans. Le scénario de l'utilisateur 2 intégrant une priorisation de certains visuels, les proportions et géométries des visuels 6 et 7 sont maximisées sur la seconde moitié du système. Les visuels les moins prioritaires s'organisent ensuite pour cadrer à l'espace restant en se plaçant sur les deux derniers modules d'affichage disponible.

Visuels sur le mur d'écrans **Représentations schématiques**

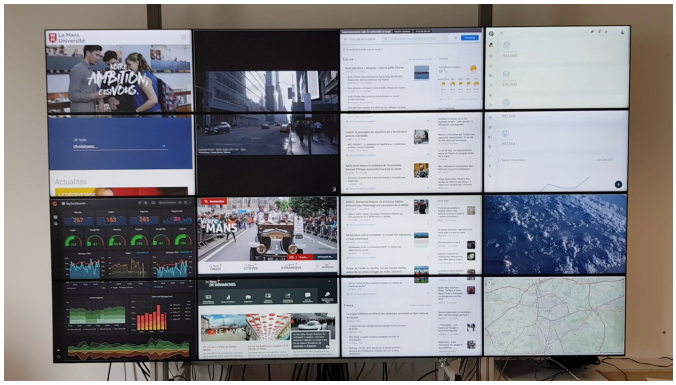


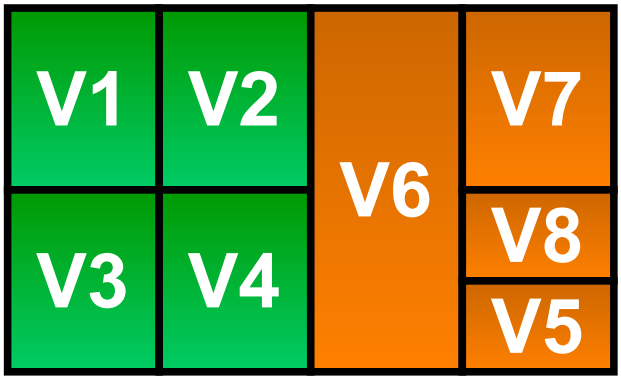
(a) Phase 1 : L'utilisateur 1 visualise un scénario de 4 affichages répartis uniformément : Deux sites internet (V1 et V4), des indicateurs numériques (V3) et une vidéo V2





(b) Phase 2 : L'utilisateur 2 visualise un scénario de 4 affichages différents de l'utilisateur 1. Le scénario se compose : d'une carte sur les quatre écrans de gauche (V5), puis d'un flux d'actualité sur les quatre écrans suivants à droite (V6), d'indicateur techniques dans le coin supérieur droit (V7) et d'une vidéo dans le coin inférieur droit (V8).





(c) Phase 3 : Les deux utilisateurs travaillent simultanément devant le mur d'écrans. Leur deux scénarios d'affichages sont associés pour afficher un maximum d'informations.

FIGURE 6.19 – Phases successives de l'expérimentation étudiant la visualisation multi-utilisateurs

6.5 | Synthèse

Dans ce dernier chapitre, consacré aux expérimentations, nous avons présenté la plate-forme Triskell3S responsable de l'implémentation des agents. Nous avons ensuite mis en œuvre trois preuves de concept ayant pour but d'expérimenter le dynamisme des communications intra et inter-constellation, la résilience d'un environnement IoT-a et la capacité de cette approche à répondre à un cadre applicatif.

La première preuve de concept présente les mécanismes de liens dynamiques entre constellations. Nos agents responsables de l'interconnexion sont capables de créer différentes topologies de liens à la volée. Nous avons pu montrer le gain en résilience de notre proposition au regard des résultats que nous avons obtenus, tout en constatant des leviers d'amélioration. Dans une seconde expérimentation, certains de ces leviers ont été actionnés pour satisfaire les performances en termes de rapidité de reconnexion. Nous avons constaté une augmentation des communications liées à la gestion par bridges dynamiques. Ce coût reste néanmoins négligeable face aux informations préservées en cas d'aléas de connexion entre les constellations.

La seconde preuve de concept teste la résilience du mur d'écrans face à la déconnexion de certains écrans. Nous avons ainsi présenté comment nos agents s'adaptent aux contraintes matérielles, à la volée, pour répondre au cadre applicatif. L'affichage est réorganisé en fonction du nombre d'écrans nouvellement disponibles. Les résultats obtenus à l'issue de cette expérimentation permettent ainsi de valider la résilience du mur d'écrans à un second niveau d'interaction.

La troisième preuve de concept étudie l'adaptabilité de notre approche IoT-a pour répondre à différents types d'applications. En l'occurrence, nous avons analysé comment nos agents peuvent prendre en charge l'affichage de plusieurs scénarios utilisateurs. Des agents symbolisent la présence de chaque personne devant le dispositif et l'affichage y est alors adapté pour correspondre aux scénarios souhaités. Les trois phases de notre expérimentation permettent de valider l'adaptabilité de notre solution.

Nous pouvons désormais reprendre nos hypothèses afin de mettre en perspectives nos travaux présentés dans ce chapitre.

Rappel de l'hypothèse 1

L'intégration d'un ou plusieurs agents au sein d'un objet connecté, répartis à différents niveaux matériels ou logiciels, permet la réponse à des besoins applicatifs spécifiques.

Afin de valider notre première hypothèse, nous avons pu observer et mesurer l'intérêt de notre approche IoT-a au travers de nos trois preuves de concept. L'intégration de plusieurs agents au sein de nos modules d'affichage connectés, nous a permis de répondre à différentes facettes de notre cadre applicatif. Comme nous le montrent nos expérimentations, l'évolutivité des IoT-a facilite la mise en place de systèmes ouverts, résilients et dynamiques.

Rappel de l'hypothèse 2

Une organisation en constellations des IoT-a permet leur adaptation au contexte matériel ou logiciel des objets connectés et ce quelque soit le cadre applicatif. La centralisation locale d'agents donne lieu à des interactions intra-objet alors que la décentralisation collective permet des échanges inter-objets.

Nos différentes expérimentations permettent de valider l'harmonisation des communications proposées par l'organisation en constellations. En effet, elle offre aux IoT-a la possibilité d'interagir dans différents contextes matériels et logiciels. La centralisation locale permet aux agents d'un module d'affichage d'évoluer en autonomie alors que la décentralisation collective permet la communication entre plusieurs modules de façon à répondre à un cadre applicatif.

Pour conclure nous pouvons observer que nos expérimentations ont fait naître de nombreuses questions face à l'utilisation du mur d'écrans devant un public. Premièrement, les méthodes observées pour associer différents visuels issus de plusieurs scénarios se basent aujourd'hui sur une gestion centralisée dans l'agent de scénarisation. Cependant, différentes approches par résolution distribuées seraient envisageables pour répondre à cette fonction. L'agent de composition pourrait, par exemple, créer un agent par visuel conscient de ses caractéristiques géométriques dans l'espace d'affichage. Leur coopération par certaines méthodes à base de forces ou de négociation DCOP [Picard, 2018], initiée par l'agent de scénarisation, serait alors susceptible de répondre au besoin d'adaptation.

Deuxièmement, les éléments que nous avons présentés sont aujourd'hui des points d'entrée pour de futurs travaux. Les expérimentations étudiées ne prennent, par exemple, pas en compte des contraintes d'IHM ou d'ergonomie. La surcharge cognitive des utilisateurs, face à autant de visuels, doit aussi faire l'objet d'études pour proposer un système qui n'est pas fatiguant ou stressant.

Enfin, notre dispositif de détection de la présence des utilisateurs est actuellement une boîte noire qui répond uniquement aux contraintes de l'expérimentation. Elle ouvre néanmoins la question de l'interaction avec le mur d'écrans. Par quelles méthodes détecter et interagir avec les utilisateurs sur un mur d'écrans ? De futurs travaux sont encore nécessaires pour répondre à cette question.

Bibliographie

[Bergenti et al., 2014] Bergenti, F., Caire, G., and Gotta, D. (2014). Agents on the move : Jade for android devices. *CEUR Workshop Proceedings*, 1260.

[Picard, 2018] Picard, G. (2018). Optimisation sous contraintes distribuée : une introduction au domaine. In *26èmes Journées Francophones sur les Systèmes Multi-Agents (JFSMA 2018)*, pages 43–52, Métabief, France. Cépaduès.

[Postel, 1981] Postel, J. (1981). Internet control message protocol. STD 5, RFC Editor. `http://www.rfc-editor.org/rfc/rfc792.txt`.

[Rapicault, 2002] Rapicault, P. (2002). *Modèles et techniques pour spécifier, développer et utiliser un framework : une approche par méta-modélisation*. Theses, Université Nice Sophia Antipolis.

[Renault and Carlier, 2016] Renault, V. and Carlier, F. (2016). Triskell3S, une plate-forme embarquée multi-agents pour les IoT-a. In *Journées Francophones sur les Systèmes Multi-Agents (JFSMA 2016)*, pages 181–190, Rouen, Saint-Martin-du-Vivier, France. Fabien Michel and Julien Saunier.

Chapitre 7

Conclusion

Le travail présenté dans cette thèse révèle les premières marches de l'escalier nous conduisant à l'étage de l'intelligence et l'autonomie totale des objets connectés. Il n'a pas pour prétention d'aboutir à une finalité, mais plutôt de proposer certaines pistes en ce qui concerne l'association du domaine des objets connectés avec celui des systèmes multi-agents. Les objets connectés sont à la croisée de nombreux domaines scientifiques qui se tissent, tels les morceaux d'un tifaifai [1]. Les systémiciens s'intéressent à cette logique depuis longtemps et nous l'expliquent : le rapprochement entre les domaines de la science fait émerger de nouvelles disciplines et des perspectives surprenantes. *"À la notion de complexité s'attache celle de variété des éléments et des interactions [...]. Elle se caractérise par l'émergence de propriétés nouvelles"*[2]. Nous pourrions, en ce sens, évoquer les travaux sur l'industrie 4.0 (usine du future) qui rassemblent des compétences en télécommunication, en réseau, en logistique et en intelligence artificielle, donnant naissance à des usines plus efficientes, plus écologiques et durables.

Les différents éléments présentés au cours de ce mémoire sont nés de réflexions avec les membres de l'équipe Tifaifai face à la révolution engendrée par les objets connectés. Ces réflexions ont abouti à une constatation : de nombreuses caractéristiques des objets connectés semblent similaires avec celles des systèmes multi-agents. Les contours de l'approche IoT-a se révèlent peu à peu et nos réflexions se cristallisent autour d'un ensemble de points de vue présentés dans la figure 7.1 et introduits dans notre première partie.

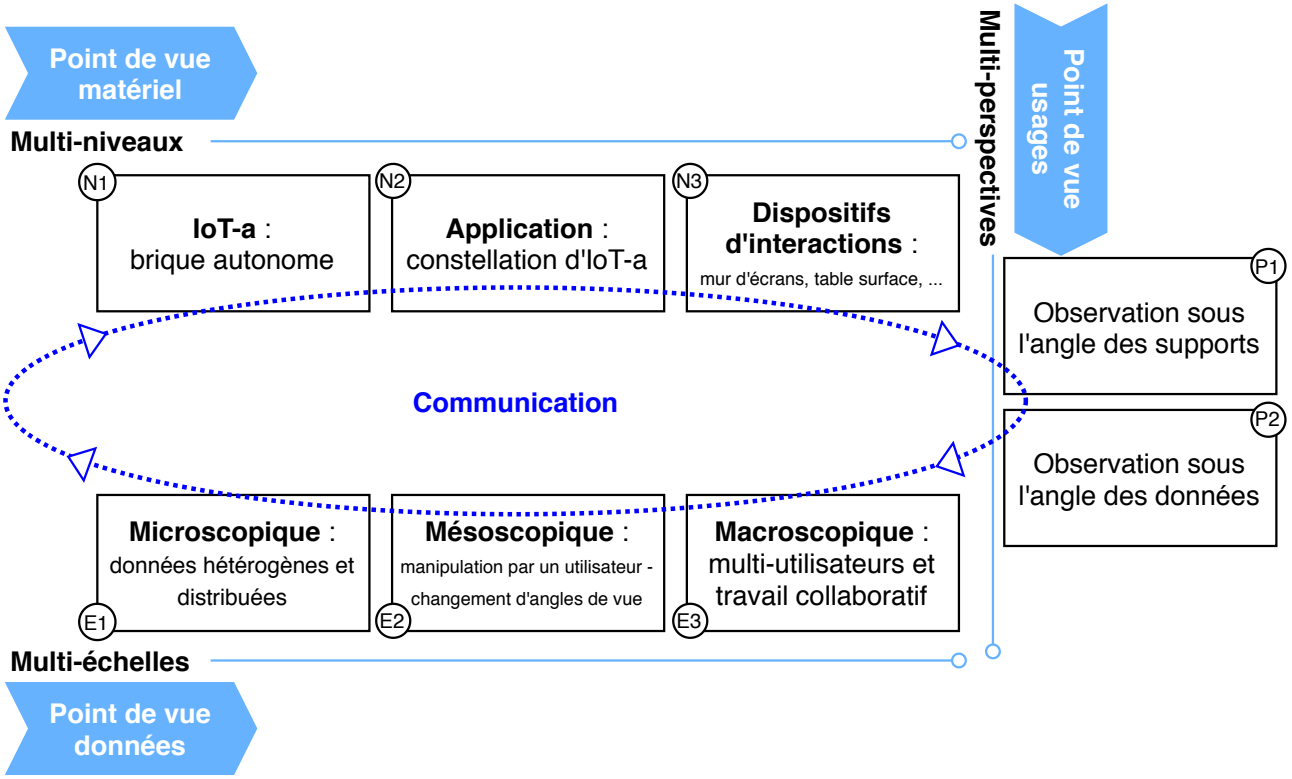


FIGURE 7.1 – Différents points de vue de l'approche IoT-a

1. Patchwork de tissu décoré d'origine polynésienne
2. Le Macroscope, J. de Rosnay, 1975

7.1 | De l'IoT à l'IoT-a

S'il est bien une réponse formelle pouvant être formulée à l'issue de cet ouvrage, c'est que cette seule thèse ne suffit pas pour répondre à toutes les questions soulevées par la figure 7.1. Ce sont aussi des spécialistes, dans des domaines bien éloignés de l'informatique, qui seront sollicités pour y parvenir dans les prochaines années. De prochains travaux permettront probablement de prolonger nos propositions en croisant les compétences variées de ces experts.

Ces raisons nous ont conduit à nous focaliser sur seulement quelques sujets. Notamment ceux concernant les notions de briques autonomes et de constellations, représentées dans le point de vue matériel de la figure 7.1. Nous avons ainsi avancé des propositions à la fois dans le domaine des objets connectés et dans le domaine des systèmes multi-agents. Bien que des formalisations restent encore à mener, le rapprochement de certaines normes de l'IoT avec les quelques standards proposés par la fondation FIPA nous a permis de définir les modèles de l'IoT-a et de la constellation d'IoT-a. Nous nous reporterons à la synthèse du chapitre 4 pour constater le bilan théorique de nos apports au regard de nos hypothèses proposées au chapitre 1. Nous avons aussi ébauché une première version d'un modèle 5M pour la prise en compte de certaines échelles du point de vue des données exposé dans la figure 7.1. Ce dernier permet de répondre à un cas d'usage dans le domaine industriel au moyen de la conception d'un mur d'écrans connectés. La collaboration de cette thèse CIFRE avec l'entreprise SARP-Véolia a permis de préciser notre modèle grâce à un cadre applicatif en lien avec ses activités. Enfin, ces propositions ont été expérimentées sur un mur d'écrans où chaque écran représente un objet connecté *agentifié* (IoT-a), capable de comportements distribués (ajout et retrait, à la volée, d'écrans dans le système).

Nous pouvons résumer en quelques mots notre travail par : l'introduction d'une approche basée sur les systèmes multi-agents, appelée IoT-a, ayant pour objectif l'organisation d'agents au sein des objets connectés et l'harmonisation de leurs communications intra et inter-objets.

7.2 | Sur la voie du FIPA embedded ?

Et si nous nous projetions un peu plus loin ? Pourrions-nous envisager plus qu'une association entre deux domaines, l'émergence d'un nouveau paradigme ? C'est bien l'idée que nous souhaiterions impulser à l'issue de cette thèse. Au-delà des détails de conception et des résultats d'expérimentations présentés, nous pressentons, pour un avenir proche, que les SMA et les IoT sont susceptibles d'aller plus loin qu'un simple apport de notions mutuelles. Quelques études commencent d'ailleurs à le relever, la proximité des caractéristiques et des applications entre les deux domaines fait sens. Les travaux sur les agents industriels et les systèmes de production holoniques multi-agents montrent bien les possibilités qu'offre l'intrication de matériels autonomes avec le paradigme multi-agents.

Emboîter le pas dans cet objectif demanderait alors un certain effort de formalisation, de modélisation et de standardisation. Les deux domaines ont chacun leurs spécificités, mais n'est-il pas possible d'en prélever les notions essentielles afin de construire de nouvelles normes ? Par exemple, nous avons relevé dans cette thèse le protocole communication bas niveau MQTT, conçu pour une utilisation dans les systèmes embarqués. Nous en avons rapproché les fonctionnalités avec certains standards de la FIPA nous ouvrant les portes de nouvelles applications. La prochaine étape serait alors d'adapter ces standards aux interactions entre agents portés par des systèmes embarqués. Plutôt que de se servir du MQTT comme simple protocole de transport de messages ACL, l'utiliser comme solution d'interaction. Comme, par exemple, formaliser une arborescence de topics en lien avec les performatives FIPA. Mais l'inverse est aussi envisageable, reprendre certains éléments de FIPA pour les adapter au MQTT, comme réduire certaines procédures d'interactions en utilisant les méthodes de qualité de services.

Avec ces avancées et d'autres encore à venir, nous disposerions d'un *FIPA embedded*, permettant des interactions légères, robustes et sécurisées, entre les agents, comme le sont aujourd'hui les communications bas niveau entre les objets connectés.

7.3 | Cadre applicatif

Le dernier sujet que nous souhaitions développer dans cette conclusion concerne notre cadre applicatif. Il a soulevé de nombreuses questions dans des domaines assez éloignés de notre sujet de recherche et de nos compétences. Il relie nos trois points de vue dans la figure 7.1 en exposant un cas d'usage. Ainsi, ce sont certains sujets concernant les IHM, l'ergonomie, la psychologie du travail ou encore la surcharge cognitive, auxquels nous nous sommes confrontés.

Interactions

L'un des points à soulever dans nos travaux concernant notre cadre applicatif est qu'il se concentre principalement sur la compétence de visualisation. Néanmoins, nous l'avons évoqué dans notre chapitre 5, l'entreprise souhaite bénéficier d'un outil lui permettant, certes de visualiser ses interfaces métier, mais aussi d'interagir avec les informations affichées.

Les éléments présentés dans cette thèse offrent, au moyen du modèle 5M, des points d'entrée logiciel pour déplacer, redimensionner, zoomer, modifier et afficher des flux d'images sur un mur d'écrans. Des travaux sont donc à mener pour déterminer comment ces points doivent être manipulés. L'état de l'art dans les recherches sur les murs d'images sera un point de départ. Certaines technologies comme les écrans tactiles sur tout ou partie du mur d'écrans, la captation d'images et l'analyse des gestes des utilisateurs, les commandes vocales, les pupitres de contrôle déportés, sont quelques exemples d'outils à étudier pour déterminer comment manipuler l'information sur une surface d'affichage aussi importante. La mise à l'échelle de certaines techniques que nous employons avec des écrans tactiles de la taille de nos téléphones, ne semble pas totalement adaptée dans le cadre d'une utilisation industrielle au quotidien. Le niveau d'interaction est d'ailleurs un point à prendre en compte. La manipulation de scénarios de visuels, comme la modification de la position d'une interface sur le mur d'écrans, semble faire intervenir des paradigmes différents de ceux intervenant dans la modification de données à l'intérieur d'un des visuels. Il sera pertinent de les étudier.

Toutes ces questions feront l'objet d'une seconde thèse CIFRE avec l'entreprise SARP-Véolia, en continuité de nos travaux. Elle aura pour objectif d'analyser, tester et formaliser les concepts d'interaction avec un mur d'écrans pour plusieurs utilisateurs.

Intégration du dispositif en agence

La mise en place d'un mur d'écrans dans une entreprise ayant pour rôle d'être un nouvel outil de travail modifie les habitudes des équipes impactées. Pour mesurer et anticiper ces effets, nous avons travaillé en collaboration avec une équipe de psychologues du travail. Nous avons encadré conjointement un stagiaire de master 2 durant six mois. Il avait pour objectif l'analyse des méthodes de travail des équipes au sein de différentes agences SARP. Il a mené une campagne d'entretiens auprès du personnel pour déterminer les attentes et les craintes face à l'arrivée d'un nouveau dispositif dans leur travail. À l'issue de son analyse, le stagiaire a fourni un ensemble de recommandations permettant de favoriser l'acceptation du mur d'écrans en agence.

De plus, l'ergonomie des postes de travail en agence a été profondément remise en cause avec l'apparition du mur d'écrans. L'effectif des agences SARP se compose en moyenne d'une vingtaine de salariés répartis dans des bureaux de quatre à six personnes maximum. Elles ne disposent généralement pas de salle suffisamment grande pour accueillir un mur d'écrans en plus des équipes. Des travaux sont alors nécessaires dans les agences pour réorganiser les bureaux et ainsi mettre en place une pièce suffisamment grande pour accueillir le personnel et le mur d'écrans.

En amont de ces changements profonds dans la première agence pilote concernée et afin d'évoquer ce sujet avec les équipes, nous avons mené deux expérimentations indépendantes avec deux groupes de personnel dans notre Salle Innovation Pédagogique - UserLab. Le but était que les personnes puissent échanger leurs ressentis face au prototype de mur d'écrans installé à Le Mans Université et qu'ils organisent en conditions expérimentales leur futur bureau. La salle où se situe notre prototype dispose de mobiliers mobiles pouvant être déplacés facilement. Notre expérimentation a donc consisté à présenter le mur d'écrans aux groupes tout en leur proposant de réagencer le mobilier en se plaçant dans la perspective d'une future intégration du dispositif dans leur agence.

Les résultats de ces expérimentations ont permis aux équipes de s'approprier le mur d'écrans en proposant certaines dispositions des postes de travail. Il est à noter que les deux groupes sans concertation ont convergé vers les mêmes résultats en proposant la même disposition de mobilier. Ces expérimentations ont ainsi ouvert de nombreuses discussions concernant la réorganisation de l'agence pilote et donnant ainsi des pistes pour les travaux à réaliser par la suite.

7.4 | Épilogue

Nous avons atteint, au cours de cette thèse, le rivage de la contribution scientifique face à une mer houleuse de connaissances encore inexplorées. Nous nous sommes efforcés de rassembler certains éléments provenant d'une nature luxuriante de formalismes et de paradigmes pour constituer un vaisseau à la conquête de nouveaux territoires. Il nécessite sûrement encore quelques évolutions, mais suffira peut-être à de futurs aventuriers de la recherche à prendre le large pour découvrir de nouveaux concepts au sujet des objets connectés intelligents.

Publications réalisées pendant la thèse

[Renault et al., 2017] Renault, V., Carlier, F., and Schmitt, A. (2017). Framework sma pour visualisation multi-écrans. In *Journée Interaction Homme-Machine et Intelligence Artificielle*, Université Pierre et Marie-Curie, Paris, France.

[Schmitt et al., 2017a] Schmitt, A., Carlier, F., and Renault, V. (2017a). Communication multi-niveaux pour des IoT-a. Interactions autour d'un mur d'écrans connectés. In *Livre "Conférence Nationale d'Intelligence Artificielle"*. Caen, France.

[Schmitt et al., 2018] Schmitt, A., Carlier, F., and Renault, V. (2018). Dynamic bridge generation for iot data exchange via the mqtt protocol. In *The 9th International Conference on Ambient Systems, Networks and Technologies (ANT 2018)*, volume 130, pages 90–97, Porto, Portugal.

[Schmitt et al., 2019a] Schmitt, A., Carlier, F., and Renault, V. (2019a). Data exchange with the mqtt protocol : Dynamic bridge approach. In *2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*, pages 1–5, Kuala Lumpur, Malaisie.

[Schmitt et al., 2017b] Schmitt, A., Carlier, F., Renault, V., and Leroux, P. (2017b). Communication multi-niveaux pour des IoT-a. Interactions autour d'un mur d'écrans connectés. In *Rencontres des Jeunes Chercheurs en Intelligence Artificielle (RJCIA 2017)*, Caen, France.

[Schmitt et al., 2019b] Schmitt, A., Carlier, F., Renault, V., and Leroux, P. (2019b). De l'iot à l'iot-a : une approche pour des communications dynamiques. In *Journées Francophones sur les Systèmes Multi-Agents (JFSMA 2019). Systèmes distribués, embarqués et diffus*, Toulouse, France.

Bibliographie complète

[Abras et al., 2009] Abras, S., Ploix, S., Pesty, S., and Jacomino, M. (2009). Apport d'une approche multi-agents pour la résolution d'un problème de gestion de l'énergie dans l'habitat. In *17emes Journées Francophones sur les Systèmes Multi-Agents*, pages 110–116, Lyon, France. Cepadues.

[Al-Fuqaha et al., 2015] Al-Fuqaha, A., Aledhari, K., Mohsen, G., Ammar, R., and Mehdi, M. (2015). Toward better horizontal integration among iot services. *IEEE Communications Magazine*, 53(9) :72–79.

[Allali et al., 2017] Allali, J., Deguillaume, L., Fabre, R., Gondry, L., Hofer, L., Ly, O., N'Guyen, S., Passault, G., Pirrone, A., and Rouxel, Q. (2017). Rhoban football club : Robocup humanoid kid-size 2016 champion team paper. In Behnke, S., Sheh, R., Sarıel, S., and Lee, D. D., editors, *RoboCup 2016 : Robot World Cup XX*, pages 491–502, Cham. Springer International Publishing.

[Barbalace et al., 2008] Barbalace, A., Luchetta, A., Manduchi, G., Moro, M., Soppelsa, A., and Taliercio, C. (2008). Performance comparison of vxworks, linux, rtai and xenomai in a hard real-time application. *Nuclear Science, IEEE Transactions on*, 55 :435–439.

[Barbosa et al., 2018] Barbosa, J., Leitão, P., and Teixeira, J. (2018). Empowering a cyber-physical system for a modular conveyor system with self-organization. In Borangiu, T., Trentesaux, D., Thomas, A., and Cardin, O., editors, *Service Orientation in Holonic and Multi-Agent Manufacturing : Proceedings of SOHOMA 2017*, pages 157–170. Springer International Publishing, Cham.

[Belkaid and Sabouret, 2014] Belkaid, M. and Sabouret, N. (2014). Un modèle logique de théorie de l'esprit pour un agent virtuel dans le contexte de simulation d'entretien d'embauche. In *Workshop Affects, Compagnons Artificiels et Interaction*, Rouen, France.

[Bellifemine et al., 2001] Bellifemine, F., Poggi, A., and Rimassa, G. (2001). Developing multi-agent systems with a fipa-compliant agent framework. *Softw., Pract. Exper.*, 31 :103–128.

[Bellifemine et al., 2007] Bellifemine, F. L., Caire, G., and Greenwood, D. (2007). *Developing Multi-Agent Systems with JADE (Wiley Series in Agent Technology)*. John Wiley & Sons.

[Ben Abdallah, 2017a] Ben Abdallah, A. (2017a). Introduction to multicore systems on-chip. In *Advanced Multicore Systems-On-Chip : Architecture, On-Chip Network, Design*, pages 1–18. Springer Singapore, Singapore.

[Ben Abdallah, 2017b] Ben Abdallah, A. (2017b). Multicore soc on-chip interconnection networks. In *Advanced Multicore Systems-On-Chip : Architecture, On-Chip Network, Design*, pages 67–106. Springer Singapore, Singapore.

[Ben Abdallah, 2017c] Ben Abdallah, A. (2017c). Multicore soc organization. In *Advanced Multicore Systems-On-Chip : Architecture, On-Chip Network, Design*, pages 39–66. Springer Singapore, Singapore.

[Bendel et al., 2013] Bendel, S., Springer, T., Schuster, D., Schill, A., Ackermann, R., and Ameling, M. (2013). A service infrastructure for the internet of things based on xmpp. In *2013 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pages 385–388.

[Bergenti et al., 2014] Bergenti, F., Caire, G., and Gotta, D. (2014). Agents on the move : Jade for android devices. *CEUR Workshop Proceedings*, 1260.

[Bethel et al., 2012] Bethel, E. W., Childs, H., and Hansen, C. (2012). *High Performance Visualization : Enabling Extreme-Scale Scientific Insight*. Chapman and Hall/CRC, 1st edition.

[Boulongne, 2018] Boulongne, A. (2018). Faire face à la transformation numérique : l'exemple de la collaboration entre le cren et la sarp. Master's thesis, Univeristé Rennes 2.

[Breivold and Sandström, 2015] Breivold, H. P. and Sandström, K. (2015). Internet of things for industrial automation – challenges and technical solutions. In *2015 IEEE International Conference on Data Science and Data Intensive Systems*, pages 532–539.

[Butzin et al., 2016] Butzin, B., Golatowski, F., and Timmermann, D. (2016). Microservices approach for the internet of things. In *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–6.

[Cao and Liu, 2016] Cao, J. and Liu, X. (2016). Introduction. In *Wireless Sensor Networks for Structural Health Monitoring*, chapter 1, pages 1–6. Springer International Publishing, Cham.

[Carlier and Renault, 2016] Carlier, F. and Renault, V. (2016). Iot-a, embedded agents for smart internet of things : Application on a display wall. In *2016 IEEE/WIC/ACM International Conference on Web Intelligence, The First International Workshop on the Internet of Agents (IoA)*, pages 80–83, Omaha, Nebraska, USA. IEEE Computer Society.

[Carlier and Renault, 2017] Carlier, F. and Renault, V. (2017). Systeme d'affichage multi-agents et methode d'affichage associee. Brevet, Notice.

[Chang, 2014] Chang, K. (2014). Bluetooth : a viable solution for iot ? [industry perspectives]. *IEEE Wireless Communications*, 21(6) :6–7.

[Chiang and Zhang, 2016] Chiang, M. and Zhang, T. (2016). Fog and iot : An overview of research opportunities. *IEEE Internet of Things Journal*, 3(6) :854–864.

[Chouhan et al., 2013] Chouhan, S., Ghorbani, J., Inan, H., Feliachi, A., and Choudhry, M. A. (2013). Smart mas restoration for distribution system with microgrids. In *2013 IEEE Power Energy Society General Meeting*, pages 1–5.

[Chouhan et al., 2009] Chouhan, S., Wan, H., Lai, H. J., Feliachi, A., and Choudhry, M. A. (2009). Intelligent reconfiguration of smart distribution network using multi-agent technology. In *2009 IEEE Power Energy Society General Meeting*, pages 1–6.

[Collier et al., 2019] Collier, R. W., O'Neill, E., Lillis, D., and O'Hare, G. (2019). Mams : Multi-agent microservices. In *Companion Proceedings of The 2019 World Wide Web Conference*, WWW '19, pages 655–662, New York, NY, USA. Association for Computing Machinery.

[Collotta et al., 2018] Collotta, M., Pau, G., Talty, T., and Tonguz, O. K. (2018). Bluetooth 5 : A concrete step forward toward the iot. *IEEE Communications Magazine*, 56(7) :125–131.

[Coskun et al., 2013] Coskun, V., Ozdenizci, B., and Ok, K. (2013). A survey on near field communication (nfc) technology. *Wireless Personal Communications*, 71.

[Coyle et al., 2010] Coyle, S., Lau, K., Moyna, N., O'Gorman, D., Diamond, D., Di Francesco, F., Costanzo, D., Salvo, P., Trivella, M. G., De Rossi, D. E., Taccini, N., Paradiso, R., Porchet, J., Ridolfi, A., Luprano, J., Chuzel, C., Lanier, T., Revol-Cavalier, F., Schoumacker, S., Mourier, V., Chartier, I., Convert, R., De-Moncuit, H., and Bini, C. (2010). Biotex—biosensing textiles for personalised healthcare management. *IEEE Transactions on Information Technology in Biomedicine*, 14(2) :364–370.

[Cyrulnik et al., 2012] Cyrulnik, B., Jorland, G., et al. (2012). *Résilience : Connaissances de bases*. Odile Jacob.

[D'Angelo et al., 2017] D'Angelo, G., Ferretti, S., and Ghini, V. (2017). Modeling the internet of things : a simulation perspective. In *2017 International Conference on High Performance Computing Simulation (HPCS)*, pages 18–27.

[De Almeida et al., 2012] De Almeida, R. A., Pillias, C., Pietriga, E., and Cubaud, P. (2012). Looking behind Bezels : French Windows for Wall Displays. In ACM, editor, *AVI - 11th working conference on Advanced visual interfaces - 2012*, Proceedings of the 11th working conference on Advanced visual interfaces, pages 124–131, Capri, Italy.

[De Rosnay, 1975] De Rosnay, J. (1975). *Le macroscope, vers une vision globale*. Editions du Seuil.

[Delic, 2016] Delic, K. A. (2016). On resilience of iot systems : The internet of things (ubiquity symposium). *Ubiquity*, 2016(February).

[Dorri et al., 2018] Dorri, A., S. Kanhere, S., and Jurdak, R. (2018). Multi-agent systems : A survey. *IEEE Access*, 6 :28573–28593.

[Douglas, 1994] Douglas, A. Y. (1994). *The X Window System : Programming and Applications with Xt, OSF/Motif edition*. Prentice Hall.

[Dragoni et al., 2018] Dragoni, N., Lanese, I., Larsen, S., Thordaland Mazzara, M., Mustafin, R., and Safina, L. (2018). Microservices : How to make your application scale. In Petrenko, A. K. and Voronkov, A., editors, *Perspectives of System Informatics*, pages 95–104, Cham. Springer International Publishing.

[Du and Swamy, 2010] Du, K.-L. and Swamy, M. N. S. (2010). *Wireless Communication Systems : From RF Subsystems to 4G Enabling Technologies*. Cambridge University Press.

[eMQTT, 2019] eMQTT (2019). The massively scalable mqtt broker for iot and mobile applications. `https://emqtt.io/`. Accès le : 10/11/2019.

[Espinosa-Aranda et al., 2015] Espinosa-Aranda, J. L., Vallez, N., Sanchez-Bueno, C., Aguado-Araujo, D., Bueno, G., and Deniz, O. (2015). Pulga, a tiny open-source mqtt broker for flexible and secure iot deployments. In *2015 IEEE Conference on Communications and Network Security (CNS)*, pages 690–694.

[Ferber et al., 2005] Ferber, J., Michel, F., and Baez-barranco, J.-A. (2005). AGRE : Integrating Environments with Organizations. In Weyns D., P. V. D. M. F., editor, *E4MAS'04 : Environments for Multiagent Systems*, number 3374 in Lecture Notes in Computer Science, pages 127–134, Melbourne (Australie).

[Finin et al., 1994] Finin, T., Fritzson, R., McKay, D. P., and McEntire, R. (1994). KQML - A Language and Protocol for Knowledge and Information Exchange. In *13th Int. Distributed Artificial Intelligence Workshop*, pages 93–103. AAAI Press.

[FIPA, 2002a] FIPA, C. (2002a). Fipa acl message structure specification - sc00061g. *FIPA TC C*.

[FIPA, 2002b] FIPA, C. (2002b). Fipa agent message transport protocol for http specification - sc00084f. *Transport WG*.

[FIPA, 2002c] FIPA, C. (2002c). Fipa agent message transport service specification - sc00067f. *FIPA TC B*.

[FIPA, 2004] FIPA, C. (2004). Fipa agent management specification - sc00023k. *FIPA TC B*.

[Ghijsen et al., 2010] Ghijsen, M., Jansweijer, W. N. H., and Wielinga, B. J. (2010). Adaptive hierarchical multi-agent organizations. In Babuška, Robertand Groen, F. C. A., editor, *Interactive Collaborative Information Systems*, pages 375–400. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Greenwood et al., 2007] Greenwood, D., Lyell, M., Mallya, A., and Suguri, H. (2007). The ieee fipa approach to integrating software agents and web services. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '07, New York, NY, USA. Association for Computing Machinery.

[Grigorik, 2013] Grigorik, I. (2013). Making the web faster with http 2.0. *Commun. ACM*, 56(12) :42–49.

[Han et al., 2011] Han, J., Kamber, M., and Pei, J. (2011). *Data Mining : Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition.

[HiveMQ, 2019] HiveMQ (2019). Reliable data movement for connected devices. `https://www.hivemq.com/`. Accès le : 10/11/2019.

[Hollnagel et al., 2006] Hollnagel, E., Wood, D. D., and Leveson, N. (2006). *Resilience Engineering : Concepts and Precepts*. ASHGATE.

[Horling and Lesser, 2004] Horling, B. and Lesser, V. (2004). A survey of multi-agent organizational paradigms. *The Knowledge Engineering Review*, 19 :281–316.

[Huraux et al., 2015] Huraux, T., Sabouret, N., Haradji, Y., and Sempé, F. (2015). Simulations multi-agents de l'activité humaine : application dans le contexte énergétique résidentiel français. In *Applications Pratiques de l'Intelligence Artificielle (APIA 2015)*, page 8, Rennes, France.

[Inguere, 2018] Inguere, T. (2018). *Multi-agents systems integration within embedded systems for tasks delegation*. Theses, Université du Maine.

[Ioniţă and Ioniţă, 2014] Ioniţă, L. and Ioniţă, I. (2014). Nm-mas : A multi-agent system for network management in oil industry. In *2014 RoEduNet Conference 13th Edition : Networking in Education and Research Joint Event RENAM 8th Conference*, pages 1–6.

[Isikdag, 2015] Isikdag, U. (2015). Internet of things : Single-board computers. In *Enhanced Building Information Models : Using IoT Services and Integration Patterns*, chapter 4, pages 43–53. Springer International Publishing, Cham.

[Jamont, 2016] Jamont, J.-P. (2016). *Démarche, modèles et outils multi-agents pour l'ingénierie des collectifs cyber-physiques*. Habilitation à diriger des recherches, Université Grenoble Alpes.

[Jamont and Occello, 2007] Jamont, J.-P. and Occello, M. (2007). Designing embedded collective systems : The DIAMOND multiagent method. In *IEEE International Conference on Tools with Artificial Intelligence - ICTAI 07*, pages 91–94, Patras, Greece. IEEE Computer Society.

[Joachim et al., 2011] Joachim, W., Kadereit, B., and Baldwin, G. (2011). Systematics, phylogeny, and evolution of papaver californicum and stylomecon heterophylla (papaveraceae). *Madroño*, 58(2) :92–100.

[Johnston et al., 2018] Johnston, S. J., Basford, P. J., Perkins, C. S., Herry, H., Po Tso, F., Pezaros, D., Mullins, R. D., Yoneki, E., Cox, S. J., and Singer, J. (2018). Commodity single board computer clusters and their applications. *Future Generation Computer Systems*, 89 :201–212.

[Kato et al., 2017] Kato, T., Takahashi, H., and Kinoshita, T. (2017). Multiagent-based autonomic and resilient service provisioning architecture for the internet of things. *International Journal of Computer Science and Network Security*, 17 :36–58.

[Kennedy and Hunt, 2008] Kennedy, T. and Hunt, R. (2008). A review of wpan security : Attacks and prevention. In *Proceedings of the International Conference on Mobile Technology, Applications, and Systems*, Mobility '08, pages 56 :1–56 :8, New York, NY, USA. ACM.

[Khanh Tuan, 2006] Khanh Tuan, L. (2006). Zigbee system-on-chip (soc) design. *High Frequency Electronics*, pages 16–25.

[Kodali et al., 2016] Kodali, R. K., Jain, V., Bose, S., and Boppana, L. (2016). Iot based smart security and home automation system. In *2016 International Conference on Computing, Communication and Automation (ICCCA)*, pages 1286–1289.

[Koenig, 2009] Koenig, P.-Y. (2009). *Information Visualization : multiscale navigation paradigms and focus+context approaches.* Theses, Université Montpellier II - Sciences et Techniques du Languedoc.

[Krivic et al., 2017] Krivic, P., Skocir, P., Kusek, M., and Jezic, G. (2017). Microservices as agents in iot systems. In Jezic, G., Kusek, M., Chen-Burger, Y.-H., Jessicaand Howlett, R. J., and Jain, L. C., editors, *Agent and Multi-Agent Systems : Technology and Applications*, pages 22–31, Cham. Springer International Publishing.

[Kubera, 2010] Kubera, Y. (2010). *Simulations orientées-interaction des systèmes complexes.* Theses, Université des Sciences et Technologie de Lille - Lille I.

[Lee, 2008] Lee, E. A. (2008). Cyber physical systems : Design challenges. In *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, pages 363–369.

[Lee et al., 2007] Lee, J., Su, Y., and Shen, C. (2007). A comparative study of wireless protocols : Bluetooth, uwb, zigbee, and wi-fi. In *IECON 2007 - 33rd Annual Conference of the IEEE Industrial Electronics Society*, pages 46–51.

[Leitão, 2009] Leitão, P. (2009). Agent-based distributed manufacturing control : A state-of-the-art survey. *Engineering Applications of Artificial Intelligence*, 22(7) :979–991. Distributed Control of Production Systems.

[Leitão et al., 2016] Leitão, P., Karnouskos, S., Ribeiro, L., Lee, J., Strasser, T., and Colombo, A. W. (2016). Smart agents in industrial cyber physical systems. *Proceedings of the IEEE*, 104(5) :1086–1101.

[Leminen et al., 2012] Leminen, S., Westerlund, M., Rajahonka, M., and Siuruainen, R. (2012). Towards iot ecosystems and business models. In Andreev, S., Balandin, S., and Koucheryavy, Y., editors, *Internet of Things, Smart Spaces, and Next Generation Networking*, pages 15–26, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Liaropoulos, 2017] Liaropoulos, A. N. (2017). Cyberspace governance and state sovereignty. In Bitros, George C.and Kyriazis, N. C., editor, *Democracy and an Open-Economy World Order*, pages 25–35. Springer International Publishing, Cham.

[LibVNC, 2020] LibVNC (2020). Libvnc software. `https://libvnc.github.io/`. Accès le : 28/01/2020.

[Light, 2017] Light, R. (2017). Mosquitto : server and client implementation of the MQTT protocol. *The Journal of Open Source Software*, 2(13).

[Liu and Anshus, 2009] Liu, Y. and Anshus, O. J. (2009). Improving the performance of vnc for high-resolution display walls. In *2009 International Symposium on Collaborative Technologies and Systems*, pages 376–383.

[Liu et al., 2009] Liu, Y., Bjorndalen, J. M., and Anshus, O. J. (2009). Using multi-threading and server update pushing to improve the performance of vnc for a wall-sized tiled display wall. In Mueller, P., Cao, J.-N., and Wang, C.-L., editors, *Scalable Information Systems*, pages 306–321, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Luzuriaga et al., 2015] Luzuriaga, J. E., Cano, J. C., Calafate, C., Manzoni, P., Perez, M., and Boronat, P. (2015). Handling mobility in iot applications using the mqtt protocol. In *2015 Internet Technologies and Applications (ITA)*, pages 245–250.

[Madsen et al., 2015] Madsen, B., Rzevski, G., Skobelev, P., and Tsarev, A. (2015). A strategy for managing complexity of the global market and prototype real-time scheduler for lego supply chain. *International Journal of Software Innovation*, 1 :28–39.

[Marwedel, 2018] Marwedel, P. (2018). *Embedded System Design : Embedded Systems Foundations of Cyber-Physical Systems and the Internet of Things.* Springer Publishing Company, Incorporated, 3rd edition.

[Mařík and Lažanský, 2007] Mařík, V. and Lažanský, J. (2007). Industrial applications of agent technologies. *Control Engineering Practice*, 15(11) :1364–1380. Special Issue on Manufacturing Plant Control : Challenges and Issues.

[Mekki et al., 2019] Mekki, K., Bajic, E., Chaxel, F., and Meyer, F. (2019). A comparative study of lpwan technologies for large-scale iot deployment. *ICT Express*, 5(1) :1–7.

[Mendoza et al., 2018] Mendoza, F., Alonso, L., López, A., Patricia Arias Cabarcos, D., et al. (2018). Assessment of fitness tracker security : A case of study. In *Multidisciplinary Digital Publishing Institute Proceedings*, volume 2, pages 1235–1254.

[Michel, 2004] Michel, F. (2004). *Formalisme, outils et éléments méthodologiques pour la modélisation et la simulation multi-agents.* Theses, Montpellier II.

[Mihailescu et al., 2017] Mihailescu, R.-C., Spalazzese, R., Davidsson, P., and Heyer, C. (2017). A role-based approach for orchestrating emergent configurations in the internet of things. *Internet of Agents Workshop 2017 (IoA@AAMAS 2017).*

[Moquette, 2019] Moquette (2019). Moquette mqtt broker. `https://moquette-io.github.io/moquette/`. Accès le : 10/11/2019.

[Moreno and Gomes Soares, 2015] Moreno, F. M. and Gomes Soares, L. F. (2015). Support to multimedia presentations on multi-head setups. In *2015 IEEE International Symposium on Multimedia (ISM)*, pages 381–384.

[Morin, 1986] Morin, E. (1986). *La méthode : La connaissance de la connaissance 3.* Éditions du Seuil, Paris.

[Morin, 1995] Morin, E. (1995). La stratégie de reliance pour l'intelligence de la complexité. In *Revue Internationale de Sytémique*, volume 9 of *2*, pages 105–112.

[MQTT, 2014] MQTT (2014). Version 3.1.1. Edited by Andrew Banks and Rahul Gupta. OASIS Standard. Latest version : `http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html`.

[Mzahm et al., 2013] Mzahm, A. M., Ahmad, M. S., and Tang, A. Y. C. (2013). Agents of things (aot) : An intelligent operational concept of the internet of things (iot). In *2013 13th International Conference on Intellient Systems Design and Applications*, pages 159–164.

[Naik, 2017] Naik, N. (2017). Choice of effective messaging protocols for iot systems : Mqtt, coap, amqp and http. In *2017 IEEE International Systems Engineering Symposium (ISSE)*, pages 1–7.

[Nam Bui and Jung, 2017] Nam Bui, K.-H. and Jung, J. J. (2017). Internet of agents framework for connected vehicles : A case study on distributed traffic control system. *Journal of Parallel and Distributed Computing*.

[Nauman et al., 2020] Nauman, A., Qadri, Y. A., Amjad, M., Zikria, Y. B., Afzal, M. K., and Kim, S. W. (2020). Multimedia internet of things : A comprehensive survey. *IEEE Access*, 8 :8202–8250.

[Nazir Raja et al., 2008] Nazir Raja, M. A., Farooq Ahmad, H., Suguri, H., Bloodsworth, P., and Khalid, N. (2008). Soa compliant fipa agent communication language. In *2008 First International Conference on the Applications of Digital Information and Web Technologies (ICADIWT)*, pages 470–477.

[Nguyen-Duc et al., 2003] Nguyen-Duc, M., Briot, J., and Drogoul, A. (2003). An application of multi-agent coordination techniques in air traffic management. In *IEEE/WIC International Conference on Intelligent Agent Technology, 2003. IAT 2003.*, pages 622–625.

[Nurmi, 2007] Nurmi, J. (2007). *Processor design : System-on-chip computing for ASIDs and FPGAs*. Springer.

[NXP, 2019] NXP (2019). Sc18is602b i2c-bus to spi bridge. Consulté le 09/11/2019.

[Othman et al., 2016] Othman, A., Zargayouna, M., Scémama, G., and Zeddini, B. (2016). Effets de l'information temps-réel des voyageurs : une simulation multi-agent.

[Pariès, 2013] Pariès, J. (2013). Resilience engineering in practice : A guidebook. chapter Lessons from the Hudson. ASHGATE.

[Picard, 2018] Picard, G. (2018). Optimisation sous contraintes distribuée : une introduction au domaine. In *26èmes Journées Francophones sur les Systèmes Multi-Agents (JFSMA 2018)*, pages 43–52, Métabief, France. Cépaduès.

[Pico-Valencia and Holgado-Terriza, 2018] Pico-Valencia, P. and Holgado-Terriza, J. (2018). Agentification of the internet of things : A systematic literature review. *International Journal of Distributed Sensor Networks*, 14.

[Pimentel et al., 2008] Pimentel, A. D., Stefanov, T., Nikolov, H., Thompson, M., Polstra, S., and Deprettere, E. F. (2008). Tool integration and interoperability challenges of a system-level design flow : A case study. In Bereković, M., Dimopoulos, N., and Wong, S., editors, *Embedded Computer Systems : Architectures, Modeling, and Simulation*, pages 167–176, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Poslad, 2007] Poslad, S. (2007). Specifying protocols for multi-agent systems interaction. *ACM Trans. Auton. Adapt. Syst.*, 2(4).

[Postel, 1981] Postel, J. (1981). Internet control message protocol. STD 5, RFC Editor. `http://www.rfc-editor.org/rfc/rfc792.txt`.

[Pujolle, 2006a] Pujolle, G. (2006a). Les réseaux. chapter Partie X, Le contrôle et la gestion, pages 737–866. Eyrolles, cinquième edition.

[Pujolle, 2006b] Pujolle, G. (2006b). Les réseaux. chapter Partie II, L'architecture en couche, pages 37–246. Eyrolles, cinquième edition.

[Quy Tran et al., 2018] Quy Tran, H., Van Phan, C., and Vien, Q.-T. (2018). An overview of 5g technologies. In Arya, K. V., Bhadoria, R. S., and Chaudhari, N. S., editors, *Emerging Wireless Communication and Network Technologies : Principle, Paradigm and Performance*, pages 59–80. Springer Singapore, Singapore.

[Rapicault, 2002] Rapicault, P. (2002). *Modèles et techniques pour spécifier, développer et utiliser un framework : une approche par méta-modélisation.* Theses, Université Nice Sophia Antipolis.

[Reggiani et al., 2002] Reggiani, A., De Graaff, T., and Nijkamp, P. (2002). Resilience : An evolutionary approach to spatial economic systems. *Networks and Spatial Economics*, 2(2) :211–229.

[Renault and Carlier, 2016] Renault, V. and Carlier, F. (2016). Triskell3S, une plate-forme embarquée multi-agents pour les IoT-a. In *Journées Francophones sur les Systèmes Multi-Agents (JFSMA 2016)*, pages 181–190, Rouen, Saint-Martin-du-Vivier, France. Fabien Michel and Julien Saunier.

[Renault et al., 2017] Renault, V., Carlier, F., and Schmitt, A. (2017). Framework sma pour visualisation multi-écrans. In *Journée Interaction Homme-Machine et Intelligence Artificielle*, Université Pierre et Marie-Curie, Paris, France.

[Reza Naji et al., 2004] Reza Naji, H., Wells, B. E., and Etzkorn, L. (2004). Creating an adaptive embedded system by applying multi-agent techniques to reconfigurable hardware. *Future Generation Computer Systems*, 20(6) :1055–1081. Computational science of lattice Boltzmann modelling.

[Ribeiro et al., 2008] Ribeiro, L., Barata, J., and Mendes, P. (2008). Mas and soa : Complementary automation paradigms. In Azevedo, A., editor, *Innovation in Manufacturing Networks*, pages 259–268, Boston, MA. Springer US.

[Richardson et al., 2011] Richardson, T., Levine, J., and Ltd., R. (2011). The remote framebuffer protocol. RFC 6143, RFC Editor. http ://www.rfc-editor.org/rfc/rfc6143.txt.

[Rocha et al., 2014] Rocha, A., Di Orio, G., Barata, J., Antzoulatos, N., Castro, E., Scrimieri, D., Ratchev, S., and Ribeiro, L. (2014). An agent based framework to support plug and produce. In *2014 12th IEEE International Conference on Industrial Informatics (INDIN)*, pages 504–510.

[Sabt et al., 2015] Sabt, M., Achemlal, M., and Bouabdallah, A. (2015). Trusted execution environment : What it is, and what it is not. In *2015 IEEE Trustcom/BigDataSE/ISPA*, volume 1, pages 57–64.

[Salminen et al., 2007] Salminen, E., Kulmala, A., and Hamalainen, T. D. (2007). On network-on-chip comparison. In *10th Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD 2007)*, pages 503–510.

[Sandstrom et al., 2003] Sandstrom, T., Henze, C., and Levit, C. (2003). The hyperwall. In *Proceedings International Conference on Coordinated and Multiple Views in Exploratory Visualization - CMV 2003 -*, pages 124–133.

[Savaglio et al., 2020] Savaglio, C., Ganzha, M., Paprzycki, M., Bădică, C., Ivanović, M., and Fortino, G. (2020). Agent-based internet of things : State-of-the-art and research challenges. *Future Generation Computer Systems*, 102 :1038–1053.

[Scheutz et al., 2005] Scheutz, M., Madey, G., and Boyd, S. (2005). tmans - the multi-scale agent-based networked simulation for the study of multi-scale, multi-level biological and social phenomena. In *Proceedings of Spring Simulation Multiconference (SMC 05)*, Agent-Directed Simulation Symposium, San Diego.

[Schmitt et al., 2018] Schmitt, A., Carlier, F., and Renault, V. (2018). Dynamic bridge generation for iot data exchange via the mqtt protocol. In *The 9th International Conference on Ambient Systems, Networks and Technologies (ANT 2018)*, volume 130, pages 90–97, Porto, Portugal.

[Schmitt et al., 2019] Schmitt, A., Carlier, F., and Renault, V. (2019). Data exchange with the mqtt protocol : Dynamic bridge approach. In *2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*, pages 1–5, Kuala Lumpur, Malaisie.

[Sebastian et al., 2018] Sebastian, A., Sivagurunathan, S., and Muthu Ganeshan, V. (2018). Iot challenges in data and citizen-centric smart city governance. In Mahmood, Z., editor, *Smart Cities : Development and Governance Frameworks*, chapter 6, pages 127–151. Springer International Publishing, Cham.

[Serpanos and Wolf, 2018] Serpanos, D. and Wolf, M. (2018). The iot landscape. In *Internet-of-Things (IoT) Systems : Architectures, Algorithms, Methodologies*, pages 1–6. Springer International Publishing, Cham.

[Shehory and Sturm, 2014] Shehory, O. and Sturm, A. (2014). Multi-agent systems : A software architecture viewpoint. In Shehory, Onnand Sturm, A., editor, *Agent-Oriented Software Engineering : Reflections on Architectures, Methodologies, Languages, and Frameworks*, pages 57–78. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Singh and Chopra, 2017] Singh, M. P. and Chopra, A. K. (2017). The internet of things and multiagent systems : Decentralized intelligence in distributed computing. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pages 1738–1747.

[Stratulat et al., 2009] Stratulat, T., Ferber, J., and Tranier, J. (2009). MASQ - Towards an Integral Approach to Agent-Based Interaction. In *AAMAS'09 : The Eighth International Conference on Autonomous Agents and MultiAgent Systems*, page 8, Budapest, Hungary.

[Sándor et al., 2015] Sándor, H., Genge, B., and Sebestyén-Pál, G. (2015). Resilience in the internet of things : The software defined networking approach. In *2015 IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)*, pages 545–552.

[Takabatake, 2011] Takabatake, T. (2011). Simulations of noc topologies for generalized hierarchical completely-connected networks. In *6th International Workshop on Reconfigurable Communication-Centric Systems-on-Chip (ReCoSoC)*, pages 1–5.

[Tan, 2018] Tan, R. (2018). Fitness app polar revealed not only where u.s. military personnel worked, but where they lived. *The Washinton Post*.

[Thirunavukkarasu et al., 2005] Thirunavukkarasu, S., Blair, G., and Coulson, G. (2005). Green : A configurable and re-configurable publish-subscribe middleware for pervasive computing. In Meersman, Tari, R., and Zahir, editors, *On the Move to Meaningful Internet Systems 2005 : CoopIS, DOA, and ODBASE : OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2005, Agia Napa, Cyprus, October 31 - November 4, 2005, Proceedings, Part I*, pages 732–749. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Trivedi et al., 2018] Trivedi, D., Khade, A., Jain, K., and Ruchira, J. (2018). Spi to i2c protocol conversion using verilog. In *Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, pages 1–4.

[Van Moergestel et al., 2016] Van Moergestel, L., Berg, M., Knol, M., Paauw, R., Voorst, K., Puik, E., Telgen, D., and Meyer, J.-j. (2016). Internet of smart things, a study on embedding agents and information as a service. *ICAART Proceedings*, 1 :102–109.

[Vasileios et al., 2015] Vasileios, K., Periklis, C., Francisco, V.-G., and Alonso-Zarate, J. (2015). A Survey on Application Layer Protocols for the Internet of Things. *Transaction on IoT and Cloud Computing (TICC), 2015*, 1(1).

[Vincenzo et al., 2006] Vincenzo, N., Concetto, S., and Corrado, S. (2006). Software agents for autonomous robots : the eurobot 2006 experience. In *Proceedings of the 7th WOA 2006 Workshop, From Objects to Agents*.

[Vinoski, 2006] Vinoski, S. (2006). Advanced message queuing protocol. *IEEE Internet Computing*, 10(06) :87–89.

[Vlacheas et al., 2013] Vlacheas, P., Giaffreda, R., Stavroulaki, V., Kelaidonis, D., Foteinos, V., Poulios, G., Demestichas, P., Somov, A., Biswas, A. R., and Moessner, K. (2013). Enabling smart cities through a cognitive management framework for the internet of things. *IEEE Communications Magazine*, 51(6) :102–111.

[Vural et al., 2017] Vural, H., Koyuncu, M., and Guney, S. (2017). A systematic literature review on microservices. In Gervasi, O., Murgante, B., Misra, S., Borruso, G., Torre, C. M., Rocha, A. M. A., Taniar, D., Apduhan, B. O., Stankova, E., and Cuzzocrea, A., editors, *Computational Science and Its Applications – ICCSA 2017*, pages 203–217, Cham. Springer International Publishing.

[Ware, 2012] Ware, C. (2012). *Information visualization : perception for design*. Elsevier, third edition.

[Weiser, 1999] Weiser, M. (1999). The computer for the 21st century. *SIGMOBILE Mob. Comput. Commun. Rev.*, 3(3) :3–11.

[White, 2011] White, E. (2011). *Making Embedded Systems : Design Patterns for Great Software*. O'Reilly Media, Inc.

[Wolf, 2019] Wolf, M. (2019). Chapter 2 - standard interfaces. In Wolf, M., editor, *Embedded System Interfacing*, pages 23–40. Morgan Kaufmann.

[Wu et al., 2018] Wu, C., Liu, Z., Zhang, D., Yoshinaga, T., and Ji, Y. (2018). Spatial intelligence toward trustworthy vehicular iot. *IEEE Communications Magazine*, 56(10) :22–27.

[Yazdanipour et al., 2012] Yazdanipour, M., Mahmoudi, D., Yazdanipour, A., Yazdanipour, M., and Mehdipour, A. (2012). Comprehensive review and selection criteria for virtual network computing technology. In *2012 Ninth International Conference on Wireless and Optical Communications Networks (WOCN)*, pages 1–5.

[Yichuan and Zhaofeng, 2011] Yichuan, J. and Zhaofeng, L. (2011). Locality-sensitive task allocation and load balancing in networked multiagent systems : Talent versus centrality. *Journal of Parallel and Distributed Computing*, 71(6) :822–836. Special Issue on Cloud Computing.

[Yu et al., 2013] Yu, H., Shen, Z., and Leung, C. (2013). From internet of things to internet of agents. In *Proceedings of the 2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*, GREENCOM-ITHINGS-CPSCOM '13, pages 1054–1057, Washington, DC, USA. IEEE Computer Society.

[Zikopoulos et al., 2012] Zikopoulos, P. C., Eaton, C., deRoos, D., Deutsch, T., and Lapis, G. (2012). *Understanding Big Data : Analytics for Enterprise Class Hadoop and Streaming Data*. McGraw-Hill Osborne Media.

[Zimmermann, 1980] Zimmermann, H. (1980). Osi reference model - the iso model of architecture for open systems interconnection. *IEEE Transactions on Communications*, 28(4) :425–432.

Liste des figures

Annexes

Chapitre A

Bus de communication

Cette annexe a pour but de présenter un comparatif des principaux bus de communication rencontrés dans le domaine des systèmes embarqués. Deux type de bus sont observés. Le premier (I2C et SPI) permet une communication intra-système, alors que ceux du second type (RS-232, USB, CAN et Ethernet), se concentrent sur la communication inter-systèmes.

Critères	**I2C**	**SPI**	**RS-232**	**USB**	**CAN**	**Ethernet**
Portée	<2 m	<3 m	15 à 30 m	5 à 10 m	30 à 5000 m	>100 m
Débit	de 400 kbit/s à 3,4 Mbit/s	>10 Mbit/s	1 Mbit/s	v2.0 : 48 Mbit/s v3.X : >5 Gbit/s	1 Mbit/s	de 10 Mbit/s à 10 Gbit/s
Topologie	Half-duplex	Full-duplex	Full-duplex	v2.0 : Half-duplex v3.X : Full-duplex	Half-duplex	Full-duplex
Nombre de périphériques max.	128	Non limitant	1	1 + sous éléments	v2.0A : 2048 v2.0B : 2^{29}	Non limitant

Tableau A.1 – Comparaison des bus de communication dans le domaine des systèmes embarqués et des objets connectés

Chapitre B

Technologies de communication sans fil

Critères	NFC	Bluetooth	ZigBee	Wifi	4G	5G	LPWAN
Portée	0.1 - 0.2 m	200 m	1500 m	50-200 m	5 Km	5 Km	3-50 Km
Débit	0.4 Mbit/s	2 Mbit/s	250 Kbit/s	300 Mbit/s	1 Gbit/s	10 Gbit/s	100 Kbit/s
Fréquence	13.56 Mhz	2.4 Ghz	868 Mhz 2.4 Ghz	2.4 Ghz 5 Ghz	de 800 Mhz à 2,6 Ghz	de 600 Mhz à 28 Ghz	868 Mhz

Tableau B.1 – Comparaison des technologies de communication sans fil pour le domaine de l'objet connecté

Cette liste de technologies présentées ici, ne se veut pas exhaustive. Elle détaille les communications radio les plus rencontrées dans le domaine de l'objet connecté.

NFC [Coskun et al., 2013] : Réseau à courte portée permettant l'échange de données entre des matériels. Une communication doit s'effectuer par un couplage inductif entre un matériel émetteur et un matériel récepteur. L'échange de données est opérationnel entre deux appareils compatibles séparés de quelques centimètres avec une fréquence de fonctionnement de 13,56 MHz. La bande passante est faible (0,02-0,4 Mb/s), mais la sécurité est forte due à la proximité nécessaire aux échanges d'informations. [Kennedy and Hunt, 2008].

BLUETOOTH [Collotta et al., 2018] : Utilisé pour la transmission de données entre deux ou un nombres illimités de matériels. Il n'a pas de limite en matière de transfert d'information, elles peuvent être des données de capteurs, des photos, des vidéos ou des flux musicaux. Le Bluetooth existe depuis 1994 et se décline avec plusieurs versions, dont la dernière et la 5 (2016). Il propose une portée de 200 mètres en environnement extérieur, une faible consommation et un débit jusqu'à 2Mb/s. Le Bluetooth semble devenir l'une des principales technologies utilisées pour la communication entre objets connectés. [Chang, 2014].

ZIGBEE [Khanh Tuan, 2006] : Spécification pour des communications à faible consommation énergétique et faible coût de déploiement. Il est basé sur le standard IEEE 802.15.4 pour la communication en réseau à rayonnement personnel (Personnal Area Network, PAN). Zigbee propose des transferts de données de l'ordre de 250 kb/s avec une portée de plusieurs centaines de mètres en environnement extérieur. Cette technologie opère sur la fréquence de 868 MHz en Europe et 2.4 GHz partout dans le monde.

WIFI [Du and Swamy, 2010] : Réseau sans fil standardisé sous le nom IEEE 802.11, introduit en 1997, dispose de plusieurs versions en lien avec son évolution : a/b/g/n. IEEE 802.11n propose depuis 2009 la mise en place d'un réseau LAN capable d'échanges avec des débits supérieurs à 300 Mb/s sur les fréquences 2.4 GHz et 5 GHz. Il est le réseau le plus utilisé par les ordinateurs personnels grand public, propose de nombreux protocoles de sécurisation du flux, une qualité de service (QoS) pour une portée de signal entre 50 et 200 m. Néanmoins, le Wifi consomme 10 fois plus que des technologies comme le Bluetooth ou le Zigbee se trouvant bien souvent implanté seulement dans des installations fixes d'objets connectés [Lee et al., 2007].

Réseau cellulaire [Du and Swamy, 2010] [Quy Tran et al., 2018] : Domaine de la téléphonie mobile avec un premier déploiement dans les années 1980 sous le nom de 1G. C'est en 2009 qu'est lancée la 4G LTE en Norvège puis en Suède et enfin le reste du monde à partir de 2012. La 4G propose des débits conséquent de l'ordre de 1 à 1,5 Gb/s sur une porté de plusieurs kilomètres relayé par des bases partout sur les territoires.

Enfin, la 5G est actuellement en développement et il est prévu qu'elle soit déployée commercialement à l'horizon 2021-2022 afin de fournir une plus grande fiabilité et une plus grande densité d'utilisateurs pour le haut débit mobile. Parmi toutes les avancées que propose la 5G, le standard promet des débits beaucoup plus importants que la 4G, soit environ 10 Gb/s dans les deux sens de communication et des latences de l'ordre de la milliseconde. L'économie d'énergie est l'un des aspects far du développement de la 5G qui souffre depuis les générations précédentes d'une consommation électrique encore trop importante pour le monde de l'objet connecté. L'un des axes proposés est la mise au point de communications *Device-to-Device (D2D)* pour permettre aux matériels de communiquer entre eux sur des distances plus courtes, en s'affranchissant des relais de communication.

LPWAN [Mekki et al., 2019] : Low-Power Wide-Area Network, est un ensemble de technologies de communication essentiellement adapté aux objets connectés et à leurs mobilités. Ils proposent de longue distance de communication 10-40 km en zones rurales et 1-5 km en zones urbaines. Les LPWAN consomment très peu d'énergie et permettent le développement d'équipements autonomes en énergie pour une durée de 10 ans sans rechargement. Le coût des solutions matérielles et d'utilisation des réseaux est très faible et permet le déploiement de très nombreux objets. Parmi les nombreux LPWAN existants, les trois réseaux les plus développés en Europe sont actuellement SigFox, LoRa et NB-IoT. Ces réseaux utilisent principalement les bandes de fréquences disponibles en 868 MHz et permettent des transferts de données de l'ordre de 50-200 kb/s.

Chapitre C

Protocoles de communication rencontrés dans le domaine de l'IoT

Hyper Text Transfer Protocol (HTTP) [Grigorik, 2013]

Le protocole le plus utilisé pour des échanges M2M est aujourd'hui le HTTP dédié aux transferts de médias pour le *World Wide Web* et créé dans les années 1990. Il est notamment standardisé par *Internet Engineering Task Force* en 1997 dans la RFC2068. HTTP repose sur les communications de type requête/réponse et utilise l'Universal Ressource Identifier (URI). Ce protocole est basé sur du texte et ne définit pas la taille de l'en-tête ni du contenu, cela dépend du serveur web utilisé. HTTP utilise TCP comme protocole de transport et TLS/SSL pour la sécurité.

Il apparaît dans l'étude [Naik, 2017] que le HTTP n'est pas un protocole adapté aux échanges pour l'IoT. En effet, le poids des messages orientés texte et les en-têtes des messages sont particulièrement lourds et demande une bande passante importante. De plus, la redondance d'information engendre une consommation importante des périphériques. Enfin, le HTTP ne propose aucune qualité de service en dehors des éléments proposés par le protocole TCP. Néanmoins, l'autorité étant en charge du protocole HTTP a proposé le CoAP bénéficiant d'une base d'interaction et une ontologie de messages adaptée au domaine de l'IoT.

Constrained Application Protocol (CoAP)

Le CoAP est un protocole léger standardisé par l'IETF en décembre 2009. Il est développé pour être interopérable avec le HTTP, au travers d'un serveur de passerelle, et ajusté aux problématiques de l'IoT. CoAP intègre aussi un mécanisme d'URI en mode requête/réponse, mais propose aussi la souscription et la publication d'informations sur des rubriques formatées par un ensemble d'URI. CoAP propose des échanges binaires avec un en-tête fixe de 4 octets et un contenu de message de courte taille dépendant des technologies de serveurs. Le protocole de transport est l'UDP, la communication n'est donc pas dans un mode connecté. Deux modes de communication sont possibles avec CoAP, un premier *confirmable* et un second *non-confirmable*. Le premier apporte deux niveaux de qualité de service et se sert des dépassements de temps de réponse pour détecter une perte de message. Le second n'apporte aucune assurance de délivrance des messages.

De parts l'utilisation du protocole UDP et de la petite taille des paquets, le CoAP est particulièrement adapté aux échanges entre objets connectés ne nécessitant pas une sûreté importante de leurs transmissions. Le protocole UDP offre une solution pour des transmissions à faible coût énergétique et faible bande passante. Néanmoins, le mode non connecté ouvre de nombreux problèmes de qualité de service étant donné qu'il n'est pas prévu de réponse lors d'une communication. Le CoAP sera principalement choisi dans le cas d'objets connectés proposant régulièrement leurs informations et qui ne sont pas sensibles à quelques pertes de messages.

Extensible Messaging and Presence Protocol (XMPP) [Bendel et al., 2013]

XMPP est un protocole d'application conçu à l'origine comme un protocole de communication pour les applications de messagerie ouverte. Il a ensuite été standardisé par l'IETF en 2004 puis mis à jour en 2011. Largement utilisé dans les applications de messagerie populaires comme Google Talk, il repose sur XML comme langage de représentation des données et met en place le mécanisme de publication/souscription. Compte tenu de ces capacités, le protocole a fait son chemin hors du monde de la messagerie et est également utilisé comme protocole de signalisation pour les systèmes de communication, les jeux, le transfert de fichiers, les réseaux sociaux, les réseaux intelligents et les applications IoT.

XMPP, implémente le protocole TCP comme solution de transport de l'information et dispose donc d'un mode connecté, mais ne propose pas de qualité de service autre que celle proposée par le TCP. Le XMPP n'est pas dit low-power car il nécessite une connexion constante pour pouvoir échanger des données binaires. De plus, le formatage XML des données apporte une surcharge des paquets et donc une bande

passante légèrement plus importante que d'autres protocoles ne nécessitant pas ce formatage.

Advanced Message Queuing Protocol (AMQP) [Vinoski, 2006]
AMQP est un protocole centré sur les messages qui a émergé du secteur financier en 2003 dans le but de libérer les utilisateurs des systèmes de messagerie propriétaires et non interopérables. Ce protocole supporte les modes de communication requête/réponse et publication/souscription. AMQP échange des messages sous différentes formes : directement, par broadcast, par rubriques ou en fonction de l'en-tête du message. C'est un protocole avec des échanges binaires disposant d'un en-tête fixe de 8 octets et sans de limite de taille pour son contenu. Le protocole de transport est le TCP et dispose de deux niveaux de qualité de service. L'utilisation de TLS/SSL pour le chiffrage des trames est la méthode de sécurisation. Chaque trame contient son numéro de canal et les trames sont précédées de leur taille pour permettre au récepteur de les lire efficacement.

Message Queuing Telemetry Transport (MQTT)
MQTT est un protocole M2M reposant sur le mode publication/souscription standardisé par le consortium OASIS en 1999. Il est proposé par la société IBM avec le but de développer une solution d'échange d'informations entre des capteurs sur un pipeline de plusieurs centaines de kilomètres. Les capteurs devaient relayer l'information à un central de mesure et avaient une forte contrainte d'autonomie énergétique. La communication était le principal défi en matière de communication. IBM se voit alors dans l'obligation de développer un protocole léger avec un coût de communication le plus faible possible. Le MQTT répond aux exigences initiales du projet pétrolier et propose des échanges avec un en-tête de 2 octets et un contenu n'excédant pas 256 Mo. En plus d'un mode de transport connecté avec TCP, MQTT dispose de trois niveau de qualités de service. La sécurité des trames est assurée par un chiffrage TLS/SSL et il est possible de limiter l'utilisation de certaines rubriques à des utilisateurs. Le nom d'utilisateur ainsi que son mot de passe sont inclus directement dans l'en-tête du protocole.

UNIVERSITE BRETAGNE LOIRE / EDUCATION LANGAGES, INTERACTIONS COGNITION, CLINIQUE

Le Mans Université

Titre : De l'IoT à l'IoT-a : une approche pour des communications dynamiques

Mots clés : IoT, communication, Systèmes Multi-agents, Systèmes embarqués, hétérogénéité, interactions dynamiques.

Résumé : L'essor des objets connectés (Internet of Things, IoT) depuis ces dernières années change profondément nos liens avec la technologie. Les industries connaissent de nouveaux modèles technologiques et économiques pendant que les foyers s'ouvrent à des usages inédits. L'IoT est vu comme une nouvelle opportunité de marché et l'approche adoptée pour la gestion de ces objets repose principalement sur la centralisation des traitements et des données dans des fermes de serveurs. Pourtant de nombreux travaux montrent les atouts offerts par la décentralisation dans ce domaine : meilleure tolérance aux pannes, mobilité accrue, gain en évolutivité, ou encore déploiement facilité. Nos travaux portent sur l'analyse des choix d'organisations afin que les objets connectés puissent porter leurs données et leurs traitements de façon décentralisée. Le paradigme multi-agents offre de multiples leviers facilitant la réponse à cette approche. Il décrit le comportement d'entités autonomes agissant dans un environnement commun. Pour cela, nous proposons une approche permettant d'adapter des agents à la complexité des architectures matérielles. L'objectif est de considérer les objets comme des « Internet of Things - agent », IoT-a. Chaque élément progresse de façon autonome et dispose des moyens suffisants pour interagir avec d'autres afin de créer de nouveaux comportements. Nous nous concentrons sur les problématiques de communication, avec des systèmes embarqués hétérogènes, en présentant un modèle de communication dynamique. Il permet le partage actif d'informations à différents niveaux applicatifs et matériels entre les agents. Nous définissons les notions de centralisation locale et décentralisation collective de constellations d'IoT-a, permettant l'évolution autonome des objets agents dans un environnement proactif. Nous expérimentons notre proposition au travers de la mise en œuvre réelle d'un ensemble d'IoT-a, formant un mur d'écrans, capable de répondre à plusieurs types d'applications de façon décentralisée. Pour sa validation et son évaluation, un scénario se focalise sur l'interopérabilité du système. Le retrait et l'ajout d'écrans, à la volée, durant la diffusion de visuels permet d'adapter les flux vidéos à la surface d'écrans disponible. Plusieurs prototypes d'outils de visualisation dynamique ont été proposés dans le cadre d'un partenariat de thèse CIFRE avec l'entreprise SARP-VEOLIA.