



HAL
open science

Towards interoperable IOT systems with a constraint-aware semantic web of things

Nicolas Seydoux

► **To cite this version:**

Nicolas Seydoux. Towards interoperable IOT systems with a constraint-aware semantic web of things. Web. INSA Toulouse, 2018. English. NNT: . tel-02093952v1

HAL Id: tel-02093952

<https://laas.hal.science/tel-02093952v1>

Submitted on 9 Apr 2019 (v1), last revised 20 Aug 2020 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Université Fédérale



Toulouse Midi-Pyrénées

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ FÉDÉRALE TOULOUSE MIDI-PYRÉNÉES

Délivré par :

l'Institut National des Sciences Appliquées de Toulouse (INSA de Toulouse)

Présentée et soutenue le 16/11/2018 par :

NICOLAS SEYDOUX

**TOWARDS INTEROPERABLE IOT SYSTEMS WITH A
CONSTRAINT-AWARE SEMANTIC WEB OF THINGS**

JURY

NAZIM AGOULMINE	Professor	Rapporteur
MATHIEU D'AQUIN	Professor	Rapporteur
RAUL GARCÍA CASTRO	Associate professor	Membre du jury
MARIA MALESHKOVA	Associate professor	Membre du jury
THIERRY MONTEIL	Professor	Membre du jury
CATHERINE ROUSSEY	Research associate	Membre du jury
KHALIL DRIRA	Research Director	Directeur de thèse
NATHALIE HERNANDEZ	Associate professor	Directrice de thèse

École doctorale et spécialité :

MITT : Domaine STIC : Intelligence Artificielle

Unité de Recherche :

*Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS),
Institut de Recherche en Informatique de Toulouse (IRIT)*

Directeur(s) de Thèse :

Khalil DRIRA et Nathalie HERNANDEZ

Rapporteurs :

Nazim AGOULMINE et Mathieu D'AQUIN

Acknowledgments

It will never be stressed enough to aspirant PhD student how research is a subtle mix of collective and individual work. Each researcher has “his” or “her” domain of predilection, and the hope of many is to be acknowledged by their peers as an expert of their field. However, no scientist works alone. From summer schools to conferences, one meets a research community, passionate about the same issues. At the lab, one meets coworkers, sharing the same struggles. Outside the lab, one has family and friends, who, no matter how remote one’s work feel from their lives, provide affection and support. Without all this network of people, no research is possible, and the collective is what allows an individual to thrive. For these reasons, I would like to express my profound gratitude to all the people who brought me where I am.

For their continuous support, for all the knowledge they shared with me about my research topic in general, and about Research (capital R intended) in general, for all the opportunities they offered me, for their time and their dedication, for never failing to challenge me, I would like to thank my advisors, Nathalie, Khalil and Thierry.

For accepting to review my thesis, I would like to thank Mathieu D’Aquin and Nazim Agoulmine. I would also like to thank Raúl García Castro, Maria Maleshkova and Catherine Roussey, who accepted to be on my jury.

For their everyday support, their wit and gossips, for all the time we spent together working, pretending to work, accepting not to be working, and drinking coffee, I would like to thank my fellow coworkers. At LAAS, there with me from start to finish, always down for board games, Tic&Tac, and Chloé for coming up with such a brilliant nickname. Rémy and Gilles, for their outstanding patience when I discourse about indie videogames. Santi, for putting up with my constant wriggling and for always offering mate. Han, for setting up a biscuit-bringing tradition in the office. Imane, for teaching me what “saha” means and when to use it properly. Quentin, for his fierce fight with six-legged invaders. Inès, for her communicative cheerfulness. Nicola, for his heated footballistic arguments. At IRIT, Fabien, for the whiteboard conversations, the beerstorming sessions and the unfading will to call me chicken. Élodie, for always reminding us that the Onador is a lunch option, and for reminding me that everything is political. Caroline, for sharing my love of comic books and showing how good vegetarian cuisine can be.

More generally, I would like to thank all the members of both SARA and MELODI teams, who answered my questions when I asked and provided advice when I needed. I would also like to thank the members of IDRA, for expending my vision to research of all domains.

I would like to thank Christelle Ecrepont, for her will to always share her expertise, and for enduring my above average capability to bring a server down. I would like to thank Aurélie Clodic for bringing robots into the game. I would also like to thank all the interns whom I worked with, and who helped me develop the LAAS open data: Joris, Bastien, Camille, Jérémy, and Juan. I would like to

thank Caroline and Justine as well, without whom I would have never attended any conference and who remained friendly and helpful despite our best effort to be an unmanageable bunch.

I would like to thank the members of the OEG lab at UPM, who welcomed me warmly and offered me new insightful perspectives on my work, as well as a collection of 1 cent coins from the coffee machine I cannot get rid of.

I would like to thank Anne Hernandez, who worked really hard to make it seem that I could write in english properly. All the reviews of my paper noticing that it was well-written are for her.

I would like to thank my friends, for enjoying to ask me how my research was going even if they knew they should not ask such a question to a PhD student, for listening to my rants about failed experiments, and above all for keeping inviting me over no matter how many times they heard “I cannot make it, I’m researching”. I would like to thank my parents, my sister and my brother, for no matter how little they understood what an ontology was always asking how was my work, for not complaining that I prepared classes or wrote papers during Christmas holidays, for giving me values I am proud to stand for, for supporting me from long before I began to work for a PhD, and for being likely to continue to do so afterwards.

Finally and above all, I would like to thank the person who heard the most about my work, my successes and my failures, who has been there for me all along, and to whom my work is dedicated. For all the “I’m leaving the lab” messages that turned out to be an hour early, and for all the “I’m just wrapping this up” that carried on really late. For all that, everything in our past and everything to come, Alexia, you have my everlasting love.

People need not only to obtain things, they need above all the freedom to make things among which they can live, to give shape to them according to their own tastes, and to put them to use in caring for and about others.

*Ivan Illich - **Tools for conviviality***

Résumé en français

Introduction

Les objets connectés représentent une nouvelle étape dans la miniaturisation et la diffusion dans l'environnement des dispositifs informatiques. Des capteurs et des actionneurs sont disséminés afin de permettre l'élaboration de nouveaux scénarios pour une agriculture de précision, un habitat automatisé, ou encore une ville dite intelligente. La diversité des cas d'utilisation entraîne une diversité de besoins, laquelle induit une hétérogénéité tant matérielle que logicielle parmi les objets connectés. Cette hétérogénéité est la source de l'un des verrous qui pèse sur le développement des objets connectés : le manque d'interopérabilité. Des objets sont dits interopérables s'ils peuvent échanger des services ou des informations.

Selon le point de vue de la conception centrée sur l'utilisateur [Abrams 2004], la capacité de l'utilisateur à contrôler l'exécution d'une tâche effectuée par un système est primordiale. Une première étape dans l'ouverture de ce contrôle réside dans la liberté de choisir les éléments qui composent le système déployé, plutôt que d'être contraint par un écosystème limité. C'est là que les limitations imposées par le manque d'interopérabilité commencent.

Afin de garantir l'interopérabilité, des standards ont été développés dans le domaine de l'internet des objets. De plus, afin d'offrir une interopérabilité la plus profonde possible, ces standards ont été complétés par l'utilisation des techniques et des principes du Web Sémantique (WS). L'interaction entre le domaine de l'IoT et le domaine du WS a amené l'émergence d'un nouveau domaine, le Semantic Web of Things (SWoT). Cependant, déployer les technologies du Web Sémantique dans des réseaux dans lesquels sont présents des objets très contraints n'est pas trivial. En effet, les représentations de connaissances et les principes qui sous-tendent ces technologies tendent à être consommateurs de ressources, ce qui est incompatible avec la sobriété énergétique et calculatoire imposée par la nature des objets connectés.

La problématique de cette thèse est donc double : il s'agira de discuter les intérêts en terme d'interopérabilité du développement du SWoT, tout en proposant des solutions permettant d'adapter les technologies du Web Sémantique aux contraintes de l'IoT. La suite de ce résumé présente le contenu des chapitres de cette thèse dans laquelle mes principales contributions sont décrites. Tout d'abord, le Chapitre §3 détaille le rôle du Web Sémantique en tant que support de l'interopérabilité, avec en particulier la proposition de IoT-O, une ontologie pour l'IoT désignée comme contribution I.A. L'utilisation de IoT-O est illustrée dans la contribution I.B à travers semIoTics, un système autonome de contrôle d'un habitat connecté. Le Chapitre §4 propose un état de l'art analysant la place du Fog computing (une technique de calcul distribué) dans le déploiement du SWoT. Cette analyse s'appuie sur un patron architectural de référence pour les réseaux d'objets

implémentant les principes du SWoT: le Cloud-Fog-Device. Dans le Chapitre §5, la nature répartie des réseaux d'objets connectés est mise à contribution pour décentraliser le raisonnement, à travers une approche générique de distribution dynamique de raisonnement à base de règles, appelée EDR. EDR est la contribution II.A de cette thèse. Elle est complétée par $EDR_{\mathcal{T}}$, une version raffinée de EDR implémentant une stratégie particulière de gestion des règles. Les performances de $EDR_{\mathcal{T}}$ sont détaillées dans le Chapitre §6.

Le SWoT, porteur d'interopérabilité

Que ce soit par choix d'un modèle économique, ou contraint par des considérations techniques, beaucoup de systèmes d'objets connectés ont été développées par leurs concepteurs dans un modèle d'intégration verticale, dit "en silo". Une telle approche implique que la couche applicative, le protocole de communication et la couche matérielle sont des systèmes fermés, avec lesquels il n'est pas possible d'interagir facilement depuis un système tiers.

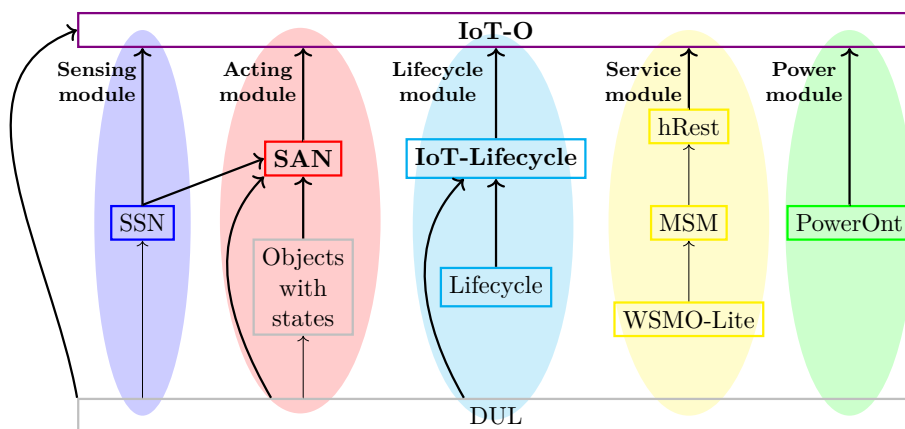
Pour pallier ce manque d'interopérabilité, il est courant d'avoir recours aux standards, tels que oneM2M ou le Web of Things du W3C. Ces standards ont ceci de particulier qu'ils considèrent non seulement l'interopérabilité syntactique, en harmonisant les schémas représentant les informations, mais aussi l'interopérabilité sémantique, en promouvant des vocabulaires et techniques du Web Sémantique.

En particulier, le Web Sémantique supporte l'interopérabilité en proposant des vocabulaires formels décrivant un domaine de connaissance donné, que l'on appelle ontologies [Gruber 1991]. Il existe de nombreuses ontologies pour caractériser le domaine de l'IoT, et la première partie de notre travail a consisté à lister ces ontologies pour identifier celles qui satisfont un ensemble de besoins que nous identifions comme des critères de qualité, inspirés par la méthode NeOn [del Carmen Suarez de Figueroa Baonza 2010]. Nous distinguons deux types de critères :

- Des critères **conceptuels**, c'est à dire les domaines couverts par l'ontologie. L'IoT étant un domaine étalé, plusieurs sous-domaines y sont rattachés : Capteurs et Observations, Actionneurs et Actions, Objets et Agents virtuels, Services, Gestion de l'énergie, et enfin Cycle de vie.
- Des critères **fonctionnels**, déterminant des caractéristiques techniques de l'ontologie. Ces critères cristallisent un certain nombre de bonnes pratiques visant à faciliter la réutilisation des ontologies d'un projet à l'autre, les rendant ainsi interopérables. L'accessibilité en ligne, la modularité [Aquin 2012] ou l'appui sur des patrons de conception [Gangemi 2005] font partie des critères identifiés.

Face à l'absence d'ontologies qui couvre l'ensemble des critères identifiés, nous proposons **IoT-O, une ontologie modulaire coeur de domaine pour l'IoT**. La description de IoT-O a été publiée dans [Seydoux 2016b]. Cependant, pour ne

Figure 1: Dépendances des modules de IoT-O



pas tomber dans un travers que nous avons observé dans le domaine, IoT-O réutilise des ontologies déjà existantes, de manière à ne pas redéfinir de concepts. Chacun des domaines identifiés dans les critères conceptuels est couvert par un module de IoT-O, et l'ontologie IoT-O en tant que telle est le module de plus haut niveau qui établit des liens entre les autres. Les modules de IoT-O, ainsi que les ontologies qui les composent, sont représentés dans la Figure 1.

IoT-O est par exemple utilisée par semIoTics, un logiciel de gestion autonome déployé pour automatiser le contrôle des objets dans un appartement connecté. semIoTics implémente la boucle MAPE-K [Kephart 2003], et s'appuie sur une base de connaissance dans laquelle l'appartement et ses caractéristiques, mais aussi les objets qu'il contient et les services que ces derniers implémentent sont décrits avec les différents modules de IoT-O. La mise en place de semIoTics est aussi rendue possible par l'utilisation d'OM2M¹, un logiciel libre implémentant le standard oneM2M. OM2M permet d'assurer l'interopérabilité syntactique vers les différents objets déployés dans l'appartement. Une question se pose cependant : dans le cas de semIoTics, les décisions sur les actions à effectuer sont prises sur un serveur disposant d'importantes capacités de calcul, et ces décisions sont converties en actions atomiques transmises directement aux actionneurs concernés. On observe là une séparation nette entre la partie du réseau dans laquelle les informations ont été sémantisées, et la partie dans laquelle les objets sont trop contraints pour manipuler ces représentations plus complexes. Dans la suite de cette thèse, des approches distribuées sont considérées afin de rapprocher au maximum cette séparation entre données brutes et informations sémantisées des objets connectés.

¹<http://www.eclipse.org/om2m/>

Quel est le rôle du Fog computing dans le domaine du SWoT ?

Pour analyser comment est répartie la charge de calcul imposée par l'utilisation des technologies et des formalismes du Web Sémantique dans le domaine du SWoT, identifier un patron architectural récurrent dans les déploiement SWoT est une première étape. On trouve d'un côté de ce patron les serveurs puissants et accessibles de manière globale à travers le Web (aussi appelés serveurs du Cloud), et de l'autre les objets connectés contraints, avec leurs protocoles de communication propres. Entre les deux, la communication n'est pas forcément directe : des dispositifs sont souvent utilisés pour assurer l'interface entre les technologies Web côté serveur, et les technologies spécifiques côté objets. Ces dispositifs sont en général appelés passerelles, et on en trouve dans de nombreuses architectures de la littérature telles que décrites dans [Su 2018], [Ben-Alaya 2015], [Zanella 2014] ou [Liu 2015]. Or, si l'on se réfère à la définition données par l'Open Fog Consortium², le Fog est une architecture distribuant services et ressources entre le Cloud et les objets. Par cette définition, le Fog est un composant intrinsèque du SWoT, d'où notre proposition d'un patron architectural, le Cloud-Fog-Device, qui capture les trois niveaux identifiés.

En partant de ce constat, il est intéressant d'étudier le rôle du Fog computing dans l'intégration des technologies du Web Sémantique dans les réseaux IoT. C'est pourquoi le Chapitre §4 est centré sur un état de l'art dans lequel des fonctions récurrentes remplies par les technologies du Web Sémantique dans le SWoT sont identifiées. Le déploiement de ces technologies est situé dans le patron Cloud-Fog-Device, pour distinguer deux cas de figures :

- Soit les noeuds du Fog se voient délégué des fonctionnalités sémantiques, auquel cas on peut parler de "Semantic Fog Computing" : le Fog joue un rôle passif dans l'obtention de l'interopérabilité sémantique.
- Soit les noeuds Fog n'ont aucune fonctionnalité en lien avec les technologies du Web Sémantique, auquel cas on parle de "Semantic Cloud Computing" : le Fog a un rôle d'interopérabilité technique ou syntactique, mais il est passif pour l'interopérabilité sémantique.

Cet état de l'art permet de souligner l'intérêt d'une approche distribuée en mettant en avant les caractéristiques des déploiements appuyés sur un Fog actif dans le déploiement des technologies sémantiques pour le SWoT. En particulier, le besoin de scalabilité est mis en avant, ainsi que l'augmentation de la responsivité rendue possible par le raisonnement plus local. C'est pourquoi, dans le chapitre suivant, une approche décentralisée est proposée, pour proposer un SWoT qui s'adapte aux contraintes de l'IoT.

²<http://openfogconsortium.org/>

Adapter les principes du SWoT aux contraintes de l'IoT

Dans le Chapitre §5, nous proposons la seconde contribution théorique et technique de cette thèse. Afin de bénéficier des avantages des solutions de SWoT distribuées, et pour permettre de marier les capacités offertes par le Web Sémantique avec les contraintes de l'IoT, nous proposons EDR (contribution II.A), une approche générique de distribution dynamique de raisonnement à base de règles. EDR repose sur l'architecture hiérarchique capturée par le patron Cloud-Fog-Device. C'est une approche basée sur un vocabulaire dédié qui permet la propagation de règles SHACL modulaires en contrôlant le comportement des noeuds du réseau. EDR est décorrélée de la stratégie qui guide la propagation de règles vers des noeuds en particuliers, c'est pourquoi EDR est une approche dite générique, qui peut être raffinée en implémentant une stratégie particulière.

Les règles propagées dans EDR sont des règles de production qui permettent de capturer la logique métier d'une application. Au lieu de fournir à l'application un flux de données pour qu'elle les traite avec ses règles, EDR vise à ce que l'application soumette ses règles au réseau, que celles-ci soient propagées, et que leurs résultats soient transmis directement à l'application par le noeud qui les a obtenus. Afin d'être compatibles avec l'approche proposée, les règles doivent se composer de plusieurs modules :

- Un module "cœur", qui contient la partie métier de la règle.
- Un module d'activation, qui permet à un noeud de déterminer s'il peut appliquer la règle
- Un module de transmission, qui permet à un noeud de déterminer auquel de ses voisins il peut transmettre la règle
- Un module notification, qui permet à un noeud de déterminer à qui envoyer les résultats de la règle quand celle-ci est appliquée

Les modules d'activation, de transmission et de notification sont tous liés à la stratégie de propagation des règles. Les décisions qu'ils permettent s'appuient sur la base de connaissance de chaque noeud : les décisions sont prises localement. Pour rendre ce mécanisme possible, les noeuds échangent des informations pour se représenter les capacités de leurs voisins et l'état de leur environnement.

Afin de démontrer la faisabilité d'EDR, et pour montrer l'intérêt de cette approche, nous proposons en tant que contribution II.B EDR \mathcal{T} , un raffinement d'EDR implémentant une stratégie qui vise à propager les règles le plus bas possible dans le réseau. EDR \mathcal{T} est basée sur les types de données observées par les capteurs, et consommées par les règles qui sont propagées. Ainsi, on cherchera à ne pas propager une règle consommant un type de donnée δ dans un sous-arbre du réseau dans lequel un tel type de donnée n'est pas produit.

Les performances d'EDR \mathcal{T} sont évaluées dans le Chapitre §6 dans un scénario d'usine connectée. On mesure le délai entre la collecte d'observations par les cap-

teurs, et le moment où l'application reçoit les déductions obtenues par la consommation de ces observations par les règles. L'impact de deux paramètres sur ce délai est mesuré :

- Le nombre de nœuds dans le réseau, qui détermine la capacité de passage à l'échelle,
- La "décentralisation" possible du traitement. Pour faire varier ce paramètre, les capteurs produisant les données sont placés plus ou moins profondément dans l'arbre représentant le réseau. Si tous les capteurs produisant une donnée sont situés proches de la racine du réseau, répartis entre un nombre limité de nœuds, les règles consommant ce type de donnée ne pourront pas être réparties plus profond. À l'inverse, une donnée produite à une grande profondeur dans l'arbre permet un étalement plus large des raisonnements à base de règle.

Les mesures obtenues sur le modèle simulé de l'usine montrent une capacité bien supérieure des solutions décentralisées comparées aux solutions centralisées au passage à l'échelle. De plus, quand les performances restent constantes dans une solution centralisée face à la répartition des capteurs dans le réseau, une plus forte répartition est bien corrélée avec une augmentation de performances de l'approche décentralisée que nous proposons.

Conclusion

Au cours des différents chapitres de cette thèse, différents aspects du domaine du SWoT sont abordés. Le rôle des principes et des technologies du Web Sémantique dans le développement de l'interopérabilité sont mis en avant dans le Chapitre 3. Après une analyse sommaire du rôle des standards pour faire face à l'hétérogénéité du domaine, les principales ontologies pour l'IoT sont présentées. Pour respecter des critères de qualité que nous identifions, nous proposons IoT-O, la contribution I.A de cette thèse. IoT-O est une ontologie modulaire, cœur de domaine pour l'IoT, et son utilisation est illustrée dans la contribution I.B à travers semIoTics, une application de contrôle autonome pour un appartement connecté. L'architecture mise en place dans semIoTics amène à une réflexion dans le Chapitre §4 sur le rôle du traitement décentralisé des technologies du Web Sémantique, en particulier illustré par le Semantic Fog Computing. Un état de l'art des approches du domaine est proposé, pour identifier les points forts et les éléments à développer dans l'intégration de technologies du Web Sémantique dans les réseaux IoT. Dans le Chapitre §5, le paradigme du Semantic Fog Computing est mis en oeuvre dans une approche dynamique de distribution des raisonnements à base de règle, appelée EDR, constituant la contribution II.A de cette thèse. EDR est ensuite raffinée par l'implémentation d'une stratégie de propagation de règles pour obtenir $EDR_{\mathcal{T}}$, la contribution II.B. $EDR_{\mathcal{T}}$ est évaluée dans le Chapitre §6 à travers un scénario d'usine connectée.

Le design centré utilisateur a été utilisé comme argument pour souligner l'importance de l'interopérabilité, mais il est aussi un élément important pour mettre en avant le traitement décentralisé. En effet, permettre de traiter des données sensibles collectées par des objets placés dans la sphère privée dans un contexte local, plutôt que de concentrer les données sur un serveur distant offre de meilleures garanties pour le respect de la vie privée. C'est pourquoi, dans les futurs travaux poursuivant ceux amorcés dans cette thèse, une nouvelle extension d'EDR sera développée, prenant en compte non plus seulement les types de données produites, mais aussi les critères de vie privée. En combinant EDR et semIoTics, nous comptons aussi concevoir un système autonome qui puisse dynamiquement se reconfigurer pour s'adapter à la topologie du réseau dans lequel il est déployé.

Contents

1	Introduction	1
1.1	Interoperability and scalability issues in the IoT	1
1.2	Enabling user centricity through the SWoT	3
2	From the IoT to the SWoT: context and background	7
2.1	Illustrative smart home example	8
2.2	The IoT, a heterogeneous technological domain	8
2.2.1	Definition	8
2.2.2	Support technologies	9
2.3	The Semantic Web, a machine-understandable knowledge space . . .	12
2.3.1	Definition	12
2.3.2	Formalisms and technologies	13
2.4	Merging the Semantic Web into the IoT: the SWoT	16
2.4.1	The WoT, putting the IoT on the Web	16
2.4.2	From the WoT to the SWoT	17
2.5	SWoT deployments architecture	18
2.5.1	Abstracting devices and services with nodes	18
2.5.2	Identifying three node classes	19
2.5.3	Characterizing the identified node types based on the complementary Cloud and Fog computing paradigms	21
2.5.4	SWoT reference deployment architecture	23
2.6	Conclusion	24
3	Achieving semantic interoperability with the SWoT	25
3.1	State of the art	26
3.1.1	Refining the notion of interoperability	26
3.1.2	Toward semantic standards for the IoT	27
3.1.3	The ontologies of the SWoT	30
3.2	Contribution I.A: IoT-O, an ontology for the IoT	33
3.2.1	Design requirements for IoT ontologies	33
3.2.2	IoT-O, a modular core-domain IoT ontology	43
3.3	IoT-O in use	46
3.3.1	Motivating use cases around the ADREAM smart building . .	46
3.3.2	Smart building use case: Making data reusable with OPA . .	48
3.3.3	Data federation use case: Integration into FIESTA-IoT . . .	51
3.3.4	Contribution I.B: User-centric smart home use case and autonomous control with semIoTics	53
3.4	Towards semantic interoperability for constrained devices	60
3.4.1	Enrichment and Lowering: IoT content management functionalities	61

3.4.2	A pivot-tree based approach to mapping reversal-based knowledge lowering	63
3.5	Conclusion	65
4	Semantic Cloud and Fog computing for the SWoT	67
4.1	Survey context and key terms	69
4.1.1	Survey methodology	69
4.1.2	Identifying SWoT functions	70
4.1.3	Related work	71
4.2	SWoT functions and semantic Fog computing	73
4.2.1	Content value creation functions	73
4.2.2	Node interworking functions	79
4.2.3	Node-content dependency functions	87
4.3	Identifying trends in SWoT processes	90
4.3.1	Analysis of the surveyed contributions	90
4.3.2	Envisioning future functions	93
4.3.3	Adapting Semantic Web technologies to constraints of the IoT domain	96
4.3.4	Making semantic Fog computing transparent	98
4.4	Conclusion	99
5	Decentralized SWoT: an adaptation to IoT constraints	101
5.1	Desirable characteristics for the proposed solution	102
5.1.1	Reasoning decentralization requirement	102
5.1.2	Applicative logic modularization requirement	103
5.1.3	Dynamism requirements	103
5.2	Related work for rules deployment in SWoT architectures	104
5.2.1	Rules for the SWoT	104
5.2.2	Centralizing rule processing on Cloud nodes	105
5.2.3	Distributing rule processing on Fog nodes	107
5.3	Extending the smart building use case	108
5.4	Contribution II.A: EDR, approach for distributed reasoning	110
5.4.1	Assumptions on the underlying architecture	110
5.4.2	Overview of the EDR approach	110
5.4.3	A deployment mechanism based on a dedicated vocabulary	112
5.4.4	Rule representation and deployment	117
5.5	Contribution II.B: Refining EDR with $EDR_{\mathcal{T}}$	122
5.5.1	Implementing a deployment strategy based on property types with $EDR_{\mathcal{T}}$	122
5.5.2	Node characteristics at stake in $EDR_{\mathcal{T}}$	124
5.5.3	Implementation of $EDR_{\mathcal{T}}$ in rule modules	128
5.5.4	Unraveling the main steps of $EDR_{\mathcal{T}}$	130
5.6	Conclusion	135

6	Experimentations	137
6.1	Deductions delivery mechanisms	138
6.2	Experimental setup and implementation	141
6.2.1	Hardware setup	141
6.2.2	Software setup	141
6.2.3	Measured results	142
6.3	Initial implementation	143
6.3.1	Smart building use case details	143
6.3.2	Impact of distribution on responsiveness	144
6.3.3	Scalability of the proposed approach	147
6.4	EDR _T implementation in a smart factory use case	148
6.4.1	Use case details	149
6.4.2	Impact of distribution on responsiveness	151
6.4.3	Scalability of the proposed approach	154
6.5	Conclusion	157
7	Conclusion and future work	161
7.1	Conclusion	161
7.2	Future work	163
7.2.1	Short-term work: Extending the EDR ecosystem	163
7.2.2	Medium-term work: Bringing semIoTics and EDR together	166
7.2.3	Long-term work: Supporting natural-language interaction	167
A	Appendix	169
A.1	Namespaces and prefixes	169
A.2	Example EDR rule	170
A.3	Acronyms	176
	176
A.4	Image credits	178
A.5	Miscellaneous remarks	178
	Bibliography	179

Introduction

Contents

1.1	Interoperability and scalability issues in the IoT	1
1.2	Enabling user centricity through the SWoT	3

1.1 Interoperability and scalability issues in the IoT

The “Things” of the Internet of Things (IoT) are now part of everyday life for many. To support this claim, one might consider the increasing number of smart cities including but not limited to Dublin (IE)¹, Santander (ES)², Milton Keynes (UK), San Francisco (US), New York (US), Jaipur (IN)³ or Yokohama (JA). All these municipalities are deploying IoT technologies to monitor city facilities such as street lightning, public transportation as well as environmental factors such as air quality, in order to offer innovative services to citizens.

The growing integration of connected Things to human activities has an impact on a wide scope of application domains, such as environmental metering with sensor networks, transportation, home automation, e-health, agriculture, manufacturing, or smart user space (*e.g.*, shopping malls, airports, parkings). In the context of the IoT domain, the word “smart” is used to refer to devices whose core functionalities have been extended based on connection and computing capabilities: smartphones, smart watches, or smart fridges for instance. By extension, domains where such connected devices are integrated are also referred to as “smart”, *e.g.*, smart cities, smart homes or smart agriculture. Systems qualified as “smart” are thus meant to communicate with other “smart” devices in order to provide innovative services to their users depending on his/her requirements. When user requirements are driving the conception of a system, principles of **user-centered design** apply [Abramson 2004].

One of the core principles of user-centered design is to “**empower the user to control the task**”. In the case of IoT, such control starts with the installation of devices chosen directly by the user to match his/her needs. However, the customizability of IoT-based applications is restricted by the approach of the industry so far, and has been oriented toward **vertical silos** [Desai 2015]. Proprietary systems are designed with a specific purpose, and on top of the devices the vendor

¹<http://smartdublin.ie/>

²<http://www.smartsantander.eu/>

³<http://smartcities.gov.in/content/>

also distributes the application serving the purpose. Siloed development may be an business model, chosen in order to build a closed ecosystem to keep customers captive, but it is also driven by technological constraints. Developing a closed system is easier, and enables the implementation of optimizations dedicated specifically to the use cases intended by the manufacturer. This approach cannot match the diversity of possible scenarios driven by user requirements, and raises **interoperability issues**. Interoperability is the ability of systems to work together, and can be achieved in roughly two ways:

- either the interworked systems are natively compatible (*e.g.*, by vendor-specific design, or through standardization), or
- a third-party acts as an interoperability provider.

Designing devices as black boxes hinders the development of third-party interoperability solutions, which is why vertical silos promote closed ecosystems: they favor vendor-specific interoperability. Such restrictive vision limits the diversity of the potential applications, since deployments are locked by vendor design. Users should be allowed to combine their connected devices in a **personalized fashion**, and application developers should be able to deploy generic applications that adapt to the available devices, which is totally opposed to vertical integration. Customizing one's IoT system requires devices to understand each other: **Things must be interoperable**. The extreme diversity of technologies involved in IoT deployments [Cabé 2018], as well as the multiplicity of IoT standards, including but not limited to oneM2M⁴, OIC⁵, and LWM2M⁶, makes IoT interoperability a non-trivial issue. Moreover, standards do not solve interoperability issues entirely, and do not support the development of smart applications as depicted in the original vision of [Berners-Lee 2001], where intelligent agents seamlessly interact with devices and information. Beyond enabling the communication between entities, **understanding must be achieved**.

The heterogeneity of the IoT systems brings another issue: management complexity. As stated in [Zanella 2014], [Barnaghi 2012] or [Foteinos 2013], the heterogeneity of an IoT system makes it hard to manage, especially on a large scale. The more interactions there are, and the more technologies involved, the harder and the costlier human management can become. Heterogeneous devices require different actions to obtain a similar result, which opposes another principle of user-centered design: “follow natural mappings between intentions and the required actions” [Abrás 2004]. Interoperability should therefore be used to give systems the ability to self-manage and self-configure, based on a shared understanding of **human-understandable high-level goals**.

Disseminating myriads of devices in the environment also raises obvious **scalability issues**. It is estimated that by 2020, **30 billion devices**⁷ [moz 2018] will be

⁴<http://www.onem2m.org/>

⁵<https://openconnectivity.org/>

⁶<http://www.openmobilealliance.org>

⁷<https://internethealthreport.org/2018/spotlight-securing-the-internet-of-things/>

connected by Machine-to-Machine (M2M) technologies, compared to the 0.9 billion connected in 2009 [Rivera 2013]. The exponential growth of the number of connected devices is correlated with a dramatic increase in the volume of data they collect and exchange. Therefore, **future-proof IoT devices and data management solutions must be designed for scalability** as well as interoperability.

Early studies for IoT data management, as in [Zhao 2010], are mainly based on centralized approaches: data is collected and processed by a unique entity, usually a powerful remote Cloud-hosted server. Such a design choice is motivated by the intrinsic constraints of IoT devices: the swarm of sensors and actuators on which the IoT system is based has a limited processing power, being more dedicated to the interaction with the physical world (i.e. sensing or acting). Therefore, resource-consuming decision-making mechanisms based on large-scale data analysis cannot be deployed on peripheral devices, and they are centralized on powerful machines. Cloud-based system design supports a smoother user experience by providing an interface reachable from any device and by hiding the complexity of the underlying infrastructure, while providing developers with on-demand provisioning and maintainability. However, centralized architectures have flaws when used for to massively distributed systems such as for IoT systems deployment. The drawbacks include response time, and the introduction of a single point of failure. In such a centralized architecture, the Cloud server may become a bottleneck for communication and a threat for privacy. Storing and processing a large data volume in a central place induces delay [Shi 2016] and may degrade the quality of service for IoT applications. This may hamper the development of a variety of applications and inhibit the implementation of time-critical IoT applications. Therefore, relying on a solely Cloud-based infrastructure is not satisfactory. The characteristics of Cloud architectures must be combined with a complementary paradigm appropriate for distributed systems. In Cloud computing, scalability is envisioned in terms of horizontal expansion (more machines) or vertical expansion (increase of machine capabilities), but structurally data still needs to be concentrated in a central place before being processed. An alternative approach is proposed by Fog computing, by taking advantage of the reduced processing power of equipments available close to IoT devices, in order to decentralize processing and to bring small pieces of computation executed locally. Two aspects of scalability are therefore considered, with different implications for the designed solutions. In order to enable emergent decentralized behaviors, an understanding of both its own components and the policies it should enforce, are necessary to the system. Moreover, the interoperability solutions should be aware of the resource-constrained environment in which they are deployed.

1.2 Enabling user centricity through the SWoT

The will to produce machine-understandable knowledge has been associated to the idea of devices able to communicate directly with each other in order to extend

their functionalities for a long time. In his article proposing a reference definition for ontologies in computer science, [Gruber 1991] uses the formal description of electromechanical devices as an example of an ontology developed in a pilot project. Even if the connected nature of these devices is not discussed, and was probably not part of the work produced at that time, the necessity to produce rich, meaningful device descriptions is still a challenge today.

In the scenario depicted by [Berners-Lee 2001], the foundational article for the Semantic Web (SW), devices discover each other based on their descriptions. In this vision, the extended capabilities of software agents and physical devices help to seamlessly provide complex services to human users. Important aspects of the scenario revolve around a user-centric approach:

- interoperability among devices and services
- the dynamic discovery process, adaptative to spatio-temporal constraints as well as to user preferences.

All of these concerns are at the core of the IoT domain in general, as coined in [Ashton 2009a].

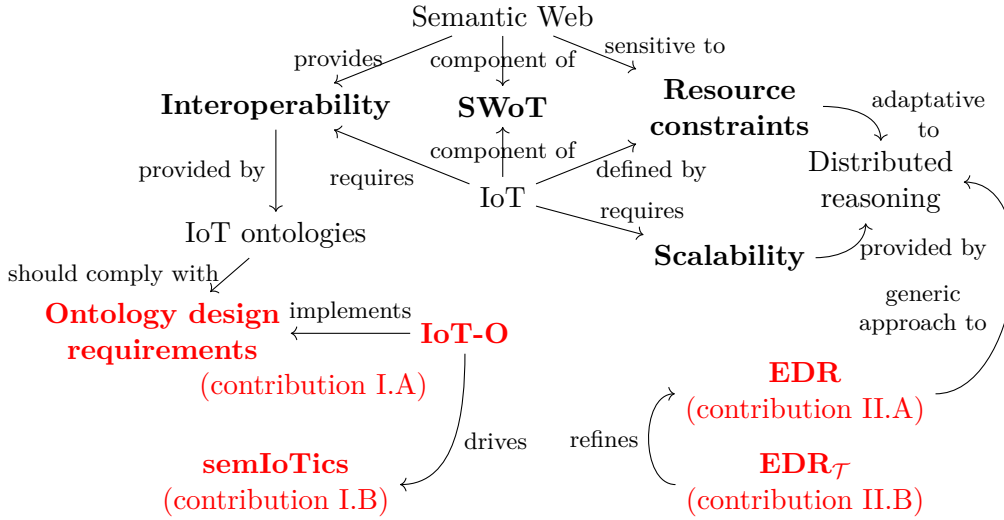
As [Corcho 2010] and [Murdock 2016] point out, **SW principles and technologies can provide solutions to the interoperability issues the IoT domain is facing**. The use of dereferencable vocabularies such as ontologies enables the capture of metadata in a machine-understandable way supporting a richer M2M communication. Many recent research contributions from the SW community introduce SW capabilities (rich content description, reasoning...) into the Web Of Things (WoT) in order to develop the Semantic Web Of Things, an evolution of the WoT where the IoT domain is extended with SW principles and technologies.

There is a natural dependency between the IoT and the SW domains: since IoT devices communicate directly among themselves, in an M2M fashion, no human agent can intervene in order to provide context or translation between two devices. Therefore, it is necessary, in order to enable the deployment of ubiquitous autonomous smart systems, to support the mutual understanding of devices. Achieving global interoperability tends to enable the communication from anything to anything, not limited to physical devices. This vision is referred to by Cisco as the Internet of Everything (IoE) [Evans 2012], where **people, data and devices are interconnected**. By supporting the representation of **complex knowledge and reasoning processes**, the SW also paves the way towards self-managing systems where reconfiguration actions are based on logical inference.

However, SW technologies are resource-consuming, while IoT systems are characterized by resource-constrained devices. Therefore, **the challenge in the development of the Semantic Web Of Things (SWoT) is to support interoperability while adapting to the constraints of IoT systems**.

The work presented in this thesis proposes contributions in the context of the SWoT, and aims at elaborating approaches to tackle this issue. An overview of the contributions is provided in Fig. 1.1. For the sake of clarity, and in order

Figure 1.1: Overview of problematics and contributions



to support the technical description of contributions proposed in this dissertation, Chapter §2 provides background information about the IoT and the SW domains. For both, a contextual and technical overview is provided, leading to introducing the emergence of the SWoT from the convergence of the IoT and the SW principles and technologies. An architectural pattern supporting SWoT deployments is also described.

In Chapter §3, the first contributions of this thesis are presented, focusing on interoperability in the SWoT. Interoperability being a core concern of the SW, and the drive for the emergence of the SWoT, it is necessary to define exactly the aspects of interoperability that are in the scope of the present work. The role of SWoT standards is briefly discussed, before considering how ontologies can support interoperability in the SWoT. The multiplicity of ontologies dedicated to the IoT led us to propose a set of requirements to measure the quality of IoT ontologies. We instantiated these requirements with IoT-O, a core-domain modular IoT ontology, referred to as **contribution I.A**. The role of IoT-O in the support of interoperability is discussed in three use cases centered around a smart building:

- an open data,
- a federated data hub,
- a home automation use case.

Self-management of SWoT systems is addressed in the latter, by proposing **technical contribution I.B**, namely semIoTics, an autonomic computing system.

Since the considered problematic is twofold, first focusing on interoperability and then on IoT systems scalability and computational power constraints, the description of related work is distributed in the appropriate chapters. However, a

complete chapter is dedicated to surveying the inter-dependence of interoperability and IoT system constraints in the SWoT. In Chapter §4, the decentralization of the SW technological stack is studied in a state-of-the-art of SWoT deployments. Recurring functions of SW technologies in IoT architectures are identified, and they are used to characterize the role of Fog-enabled architectures for supporting the development of a decentralized SWoT.

The survey proposed in Chapter §4 builds the context for the second contribution of this thesis, presented in Chapter §5. In order to adapt the SWoT solutions to IoT systems constraints, both in terms of processing power and scalability, we propose a generic approach to dynamically distributed rule-based reasoning, where Cloud and Fog nodes collaborate to achieve an emergent distribution of processing, called Emergent Distributed Reasoning (EDR), and referred to as **contribution II.A**. Being a generic approach, EDR is agnostic to application-level requirements. Therefore, **technical contribution II.B** is a refinement of EDR, called $\text{EDR}_{\mathcal{T}}$, that implements a deployment strategy aiming at bringing rule computation as close as possible to sensors producing data. Such strategy is devised to fulfill a delay reduction requirement, from the collect of observations to the delivery of deduction to the application. The performances of $\text{EDR}_{\mathcal{T}}$ are studied in two use cases detailed in Chapter §6, dedicated to home automation and smart manufacturing. Finally, limits of the current contributions and perspectives for future work are considered in Chapter §7, the concluding chapter of the present thesis.

From the IoT to the SWoT: context and background

Contents

2.1	Illustrative smart home example	8
2.2	The IoT, a heterogeneous technological domain	8
2.2.1	Definition	8
2.2.2	Support technologies	9
2.3	The Semantic Web, a machine-understandable knowledge space	12
2.3.1	Definition	12
2.3.2	Formalisms and technologies	13
2.4	Merging the Semantic Web into the IoT: the SWoT	16
2.4.1	The WoT, putting the IoT on the Web	16
2.4.2	From the WoT to the SWoT	17
2.5	SWoT deployments architecture	18
2.5.1	Abstracting devices and services with nodes	18
2.5.2	Identifying three node classes	19
2.5.3	Characterizing the identified node types based on the complementary Cloud and Fog computing paradigms	21
2.5.4	SWoT reference deployment architecture	23
2.6	Conclusion	24

The contributions proposed in the present thesis are at the interface between the IoT and the SW. This chapter aims at giving an overview of both these technological domains, as well as to motivate their convergence towards the SWoT, a composite domain where SW principles and technologies are used in an IoT setting.

A very brief illustrative use case is initially described, in order to be used in the descriptions provided all along the chapter. The IoT and the SW domains are then introduced individually, respectively in Section §2.2 and §2.3. For each of them, a definition is provided, as well as a technological landscape, before describing some challenges they face. Then, in Section §2.4 the SWoT is introduced, as well as the motivations for its emergence. Finally, the architecture supporting the SWoT is examined in order to identify an architectural pattern.

2.1 Illustrative smart home example

In order to support the description of the numerous technologies and design principles at stake in the technical background of the present work, a simple smart home use case is introduced here. This example is inspired by an actual experimental deployment in which experimentations of my work have been performed, as it is discussed in Section §3.3.4. The smart home we consider is equipped with several devices disseminated in the environment, such as temperature, presence or luminosity sensors, or light bulbs. These devices are provided by different vendors, and they are **not designed to be used together natively**. A software is installed on the house computer in order to read the state of sensors, and to control actuators. This software runs a Web server, and may be reached via a dedicated smart phone application. From his/her phone, the user is thus able to check the temperature in some rooms, as well as to remotely control lightning. The scenario motivating this example is kept to a minimum for the sake of simplicity, in order to focus on the technological stack supporting the use case.

The details of the technological stack deployed to connect the control platform to the end devices are given in the remainder of this chapter.

2.2 The IoT, a heterogeneous technological domain

The term IoT, as many words with marketing value, tends to be used with a wide range of meanings. After providing a definition of what IoT stands for in this thesis, representative IoT technologies are described, some of which being part of the illustrative smart home use case.

2.2.1 Definition

The term “Internet of Things”, coined in 1999 by Kevin Ashton (according to a later statement [Ashton 2009b]), initially referred to networks of Radio Frequency Identification (RFID) tags. The meaning of the word evolved with the emergence of active computing power ubiquitously deployed in connected devices. A broader definition of the IoT has been provided by the ITU in 2012 [itu 2012b], embracing the diversity of nature and purpose of the so-called Things: “[**The IoT is**] **a global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies**”. We adopt this definition in this thesis, and the term IoT thus refers to the area of technology and research enabling the deployment of Things networks. The same ITU report defines a Thing as “**an object of the physical world (physical things) or the information world (virtual things), which is capable of being identified and integrated into communication networks**”. The notion of Thing is thus not limited to devices such as temperature sensors or connected

light bulbs, it also includes services and elements of the environment about which characteristics may be collected, *e.g.*, a room or a package.

However, even if the technology of the IoT domain have radically transformed, the core idea introduced by [Ashton 2009b] is still relevant: **the Internet, and a fortiori the Web, is both fueled by human productions and targeted at human consumption**. It is in that regard that the IoT domain is a game changer: exchanged data is no longer solely made of abstract concepts shared between humans, but rather of observations of the physical world meant for M2M communication.

2.2.2 Support technologies

The notion of ubiquity associated to the IoT [Qu 2016] emerges from the deployment in the environment of constrained devices, and the term “IoT technologies” is in general used to refer to the dedicated technologies deployed to support their operation and communications.

In the remainder of this section, technologies representative of the constraints of the IoT domain are introduced. Causes for these constraints are discussed in Section §2.2.2.1. Different types of communication technologies, based on different physical channels, are introduced in Section §2.2.2.2, and transport protocols of higher level common in IoT use cases are described in Section §2.2.2.3.

2.2.2.1 Sources of IoT devices constraints

The constrained nature of IoT devices comes from the necessity for their large-scale deployment in the environment. They are not necessarily connected to regular power grids, and therefore might rely on an **internal battery**, or even on **external energy harvesting**. Such harvesting may be based on wireless technologies [Kamalinejad 2015], where the device consumes the power converted from an electromagnetic signal received by an embedded antenna. Another approach is to collect energy from a mechanical source [Gorlatova 2014], similarly to the principle of self-winding watch¹. Such a system is for instance integrated into the EnOcean devices²: the signal generated when a user presses a button is generated thanks to the mechanical energy issued from the button press. An EnOcean remote is available in our illustrative use case. The integration of IoT devices in the environment also requires to **reduce their size**, which is another source of constraint. Of course, when deploying devices at a very large scale, the individual cost of each device has to be minimized. Since each IoT device is meant to perform a few simple tasks, granting these devices with large computational capabilities, and thus raising their cost,

¹https://en.wikipedia.org/wiki/Automatic_watch

²<https://www.enocean.com/>

2.2.2.2 Communication technologies

Multiple communication technologies support the IoT, and this section only gives a very broad overview of the main ones.

Proprietary technologies Proprietary technologies are many in the IoT domain, the variety of use cases and application domains creating a wide range of requirements.

- EnOcean, that has already been introduced, is an example of wireless proprietary technology.
- Phidget³ is a wired proprietary protocol based on USB communication. In our example, a temperature and a luminosity sensor are connected via this technology.
- Z-Wave is a short-range mesh network wireless communication technology based on a proprietary radio technology. The presence sensor of our use case communicates over Z-Wave.

Short-range technologies The concentration of devices in a limited geographical space, potentially indoors, allows to use telecommunication technologies only able to reach a short range. If necessary, multiple devices communicating locally at a short range can create a mesh covering a wide area.

- Bluetooth Low Energy (BLE) is an extension of the Bluetooth communication technology designed to have a much lower power consumption. BLE is however based on the same paradigm as Bluetooth, and only star topologies are allowed, with a central master and some peripheral slaves.
- Zigbee is a radio protocol developed by the Zigbee Alliance⁴. Contrary to BLE, Zigbee devices may be organized in a mesh. The main characteristics and use cases for Zigbee are quite similar to Z-wave. However, since Zigbee is an open standard, more manufacturers can produce Zigbee devices. This creates a more diverse ecosystem, but generates interoperability issues among devices that are supposed to be based on the same technology. The connected light bulb installed for the use case communicates over Zigbee.
- 6LowPan is an acronym for “IPv6 over Low-Power Wireless Personal Area Networks”, proposed in IETF⁵ RFC 4944⁶. Deploying an IP network over low-power devices enables the creation of a mesh network at the packet level (based on the OSI layered model⁷). BLE and Zigbee are Personal Area Networks technologies that may support 6LowPan networks.

³<https://www.phidgets.com/>

⁴<http://www.zigbee.org>

⁵<https://ietf.org/>

⁶<https://tools.ietf.org/html/rfc4944>

⁷<https://www.iso.org/standard/20269.html>

Long-range technologies In order to be able to implement some use cases such as environment monitoring or agriculture, IoT devices must be deployed over large areas, potentially not covered by traditional communication networks. Some technologies have been developed to provide ad-hoc networks that allow long-range and low-power communication.

- SigFox⁸ is both a network operator and a communication technology deployed by said operator. SigFox devices communicate with SigFox gateways, that are connected to the Internet. Messages produced by SigFox devices are therefore stored on servers to be accessible via a Web interface from the client side.
- LoRa is a communication technology that is supported by the LoRa alliance⁹, and contrary to SigFox it is not tied to an operator: anyone may deploy an ad-hoc LoRa network. The network topology enabled by LoRa is however quite similar to SigFox: devices communicate over LoRa with gateways that are connected to “traditional” networks, and make the messages available to the user on dedicated servers. When a LoRa device wakes up to send a message, it is briefly possible to send a message to it, enabling bi-directional communication.

2.2.2.3 Transport protocols

HTTP/CoAP HTTP¹⁰ is the protocol at the core of the Web. However, HTTP is based on TCP¹¹, requiring a permanent connection between the communicating entities during the communication. Establishing such connection is costly, and HTTP is therefore not adapted to all IoT architectures, where more lightweight protocols might be preferred. That is why Constrained Application Protocol (CoAP)¹², based on UDP¹³, has been introduced. CoAP is a protocol especially designed for constrained applications, with reduced headers and limited packet body. UDP being a datagram-based protocol, the establishment of a connection is not necessary before exchanging messages. CoAP mimics the verbs of HTTP, such as GET or POST, and adds a new verb, OBSERVE, to enable notification of the client when a resource is changed.

The notion of REST services is usually associated to the HTTP protocol: a Web server exposes an HTTP interface which is not meant to be accessed by a Web browser but rather by a REST client. The Constrained RESTful Environments (CoRE) Link Format¹⁴ enables the deployment of a REST architecture for constrained devices. CoRE is based on CoAP.

⁸<https://www.sigfox.com>

⁹<https://www.lora-alliance.org/>

¹⁰<https://tools.ietf.org/html/rfc2616>

¹¹<https://tools.ietf.org/html/rfc793>

¹²<http://coap.technology/>

¹³<https://www.ietf.org/rfc/rfc768>

¹⁴<https://tools.ietf.org/html/rfc6690>

MQTT Message Queue Telemetry Transport (MQTT)¹⁵ is a publish-subscribe protocol standardized by the OASIS consortium¹⁶. Messages are published to a broker in topics, and subscribers to said topics are notified on publications. In order to enable the notification, a connection must be established between the client and the broker: that is why MQTT is based on TCP.

2.3 The Semantic Web, a machine-understandable knowledge space

The Semantic Web is a branch of AI focusing on knowledge representation and manipulation, merged with the formalisms of Linked Data (LD). After providing a definition of the Semantic Web encompassing its foundational principles, its main formalisms and technologies are described, as they are used in the remainder of this work.

2.3.1 Definition

The SW has been initially proposed in [Berners-Lee 2001] by Sir Tim Berners-Lee, who received in 2016 the Turing Prize for inventing the Web¹⁷. The proposition of the SW is rooted in the limitations of the “regular” Web, with an observation somewhat similar to the vision of [Ashton 2009b] for the IoT domain that may be summarized as follows: the Web is built by humans for humans, and it is hard to understand for machines. The purpose of the SW is to leverage techniques from Artificial Intelligence (AI) domains such as logic, knowledge representation or knowledge engineering in order to foster the creation of **machine-understandable content** while keeping its **human-understandable aspect**.

The SW is based on the foundational principles of the Web, and more specifically of what is called LD¹⁸:

- Entities are identified by dereferencable International Resource Identifier (IRI)
- When its IRI is dereferenced, the representation of an entity is based on World Wide Web Consortium (W3C) standards
- Entities are connected together by crawlable links

The main difference with legacy Web resources is that both entities and links are typed in the SW, supporting the understanding by machines. In order to type resources and links, as well as to describe their properties, dedicated vocabularies are used. Such vocabularies are called **ontologies**, as they capture a representation of the world. The notion of ontology precedes the emergence of the SW, as it has

¹⁵<http://mqtt.org>

¹⁶<http://oasis-open.org>

¹⁷<https://www.acm.org/media-center/2017/april/turing-award-2016>

¹⁸<https://www.w3.org/DesignIssues/LinkedData.html>

2.3. The Semantic Web, a machine-understandable knowledge space 13

been defined by [Gruber 1991]. The association of an ontology and the information it describes is referred to as a knowledge graph, or as a Knowledge base (KB), term that will be used from now on in this thesis.

2.3.2 Formalisms and technologies

SW technologies are standardized by the W3C¹⁹, a standard consortium founded by Tim Berners-Lee in order to ensure the interoperability of Web-related technologies, such as XML and its derivatives (especially HTML), CSS or SVG, but also accessibility of Web content, internationalization, or authentication. In the remaining of this section, some technologies standardized by the W3C and related to the SW are introduced.

2.3.2.1 Representing graphs in RDF

SW resources, their relations and their characteristics can be represented in the form of a graph. Ressource Description Framework (RDF)²⁰ is a language proposed by the W3C to represent such graphs, based on the notion of triple.

A triple is constituted of a subject, a property and an object. The subject is the resource about which the triple expresses a characteristic, the property is the identifier of said characteristic, and the object is the value of the property for the subject. The subject and the property are two resources identified on the graph with an IRI (in the most cases), and the object can either be another individual with an IRI or a literal (*e.g.*, a string, a integer, or a timestamp) expressing a value for the property.

For instance, let us assume that the temperature sensor deployed in the use case is identified with the IRI *ex:MyTemperatureSensor*²¹, the living room with the IRI *ex:MyLivingRoom*, and the location property expressing that something is located somewhere is expressed with the IRI *ex:isLocated*. Therefore, the fact that the temperature sensor is located in the living room may be expressed with the triple $\langle ex:MyTemperatureSensor, ex:isLocated, ex:MyLivingRoom \rangle$ in a KB. In this case, both subject and object are individuals with an IRI. Assuming that the property *ex:hasSerialNumber* captures the serial number of a device, then the fact that the same sensor has the serial number 0042 is represented with the triple $\langle ex:MyTemperatureSensor, ex:hasSerialNumber, 0042 \rangle$. This last triplet is an example where the object does not have an IRI, and captures a literal value characterizing the subject.

2.3.2.2 Describing ontologies with RDFS and OWL

RDF is a language enabling the description of resources, but its semantics is limited. In order to gain the expressiveness necessary to represent ontologies, additional

¹⁹<https://www.w3.org>

²⁰<https://www.w3.org/RDF/>

²¹Namespaces are provided in Appendix §A.1

schema must be added on top on RDF.

RDFS²² is a vocabulary supporting the description of both taxonomic and non-taxonomic relationship between classes. Ontologies expressed solely in RDFS are considered lightweight ontologies, as opposed to heavyweight ontologies expressed in Web Ontology Language (OWL)²³.

OWL enables to use an extended set of logical axioms in ontologies description, enriching the expressiveness of the obtained model. These axioms may express characteristics of properties, such as their reflexivity, specific relations among individuals such as equivalence or disjointness, or characteristics for a property for a specific individual, such as its cardinality. Let us briefly introduce the Sensor, Observation, Sample, and Actuator (SOSA)²⁴ ontology, proposed by the W3C, which defines a vocabulary to describe IoT devices. SOSA defines OWL classes to classify concepts, in particular *sosa:Sensor*, which represents any entity capturing a phenomenon into a virtual representation. The temperature sensor of the use case belongs to this definition, and therefore the following triplet is added th the KB: *<ex:MyTemperatureSensor,rdf:type,sosa:Sensor>*.

Some RDF triples are used to define the ontology, by describing the vocabulary and its properties. Such triples in a KB are referred to as T-box, since they structure a terminology. Expressing that *sosa:Sensor* is a class with the triplet *<sosa:Sensor,rdf:type,owl:Class>* is part of the T-box declaration. Other RDF triples are used to describe data with the previously described vocabulary, and constitute what is referred to as the A-box, for assertions. The triplet *<ex:MyTemperatureSensor,rdf:type,sosa:Sensor>* is an element of the A-box.

2.3.2.3 Querying a knowledge base with SPARQL

RDF, RDFS and OWL enable the description of KB as graphs. In order to retrieve knowledge from a KB, the querying language must therefore enable the description of graph patterns. To this intent, the W3C proposed SPARQL²⁵, or SPARQL Protocol and RDF Query Language²⁶. SPARQL defines a query language initially designed to enable the retrieval of knowledge from a KB, and extended to enable insertion, removal or actuation.

SPARQL also defines a protocol, built over HTTP verb and properties, to communicate with a service deployed on top of a KB to submit SPARQL queries. Such service, enabling Web access to an RDF KB, is referred to as a SPARQL endpoint.

2.3.2.4 Reasoning about ontologies with reasoners

The role of reasoners: The introduction of formalism with RDFS and OWL in ontologies enables their processing with a reasoning engine, or reasoner. A reasoner

²²<http://www.w3.org/2000/01/rdf-schema>

²³<http://www.w3.org/2002/07/owl>

²⁴<http://www.w3.org/ns/sosa/>

²⁵<https://www.w3.org/TR/sparql11-overview/>

²⁶Recursive acronyms are a refined form of computer scientist humor

2.3. The Semantic Web, a machine-understandable knowledge space 15

is a piece of software that infers new knowledge from a KB, based on the data it contains and how said data is described with ontologies. Reasoners are implementations of the theoretical logic embedded withing languages such as RDFS and OWL.

Expressing reasoning axioms with rules: Complementary to domain knowledge representation through ontologies, logical rules can be seen as a paradigm for knowledge modeling dedicated to specific usages. Logical rules are purely used for deduction: if their preconditions are true, the engine deduces their postconditions. Different formalisms are available to represent logical rules, such as SWRL²⁷ and SPIN²⁸. Both these languages have been submitted to the W3C, but were not adopted as recommendations, contrary to RIF [Kifer 2013]. RIF documents can be used to represent inference rules over RDF graphs, and can themselves be serialized in RDF. Therefore, it is possible to associate an IRI to a RIF rule on a resource graph, and to establish link between rules, making RIF a format compliant with the requirements of Linked Rules. SHACL²⁹ and its extension³⁰ are the latest W3C recommendations for rules representation. SHACL aims to represent constraints on an RDF graph, called “shapes”, as well as deduction rules. SHACL rules, similarly to SPIN, can be based on SPARQL: it is possible to express a production rule in SHACL as a SPARQL CONSTRUCT query.

2.3.2.5 Building links among ontologies

Ontologies being themselves Web resources, they abide to the LD principles, and in particular interlinking. Two complementary types of interlinking are defined here in order to be reused later in Chapter §3.

Ontology reuse: When building an ontology, the terms previously defined by another ontology may be reused and extended. The *owl:import* enables to specify that the vocabulary defined by an ontology should be examined in the context provided by a previous ontology. In our use case, the *sosa:Sensor* class might be extended by a *ex:TemperatureSensor* new class, which would be more specific. Such extension is expressed with the triple $\langle ex:TemperatureSensor, rdfs:subClassOf, sosa:Sensor \rangle$. Ontology reuse is typically performed at conception time.

Ontology alignment: An ontology is primarily defined by the domain it covers. However, it is possible that multiple ontologies define concepts from overlapping domains. In such case, it is desirable to express links between these two ontologies, even after they both have been published. To do so, triples can be added th the KB,

²⁷<https://www.w3.org/Submission/SWRL/>

²⁸<https://www.w3.org/Submission/spin-modeling/>

²⁹<https://www.w3.org/TR/shacl/>

³⁰<https://www.w3.org/TR/shacl-af/>

unilaterally or bilaterally, to express relations among terms such as subsumption or equivalence.

2.4 Merging the Semantic Web into the IoT: the SWoT

When the SW principles and technologies are used in an IoT context, the resulting hybrid domain is called SWoT. The emergence of the SWoT is presented here progressively: first, Web technologies have been used in order to unify data and Thing accessibility through Web protocols. Then, the SW principles have been introduced, leading to the SWoT.

2.4.1 The WoT, putting the IoT on the Web

In order to bridge the gap between applications and IoT devices on a technical level, Web principles and technologies have been used. Around 2008, the notion of WoT was introduced in several studies such as [Stirbu 2008] or [Guinard 2009], later defined by [itu 2012a] as “A way to realize the IoT where (physical and virtual) things are connected and controlled through the World Wide Web”. On top of the heterogeneous IoT communication network, the WoT provides a unified access to both data and Things identified with IRI through Web protocols.

Indeed, even in the small scale use case that we described, four communication technologies are deployed, namely EnOcean, Z-Wave, Phidget and Zigbee. Requiring the end application to be able to communicate over these four technologies, three of which being proprietary, some of them on different physical supports (wired or wireless), is impossible. That is why we propose an HTTP communication between the user’s smart phone and the home computer.

The WoT is standardized by a dedicated W3C working group³¹, using the [itu 2012a] definition as a reference. The deployment of the WoT is supported by Web protocols such as HTTP and CoAP. Both these protocols enable content negotiation, allowing a client to query the server for a specific representation of a given resource. The WoT also uses technologies associated to the programmable Web in general, and to Web services in particular, such as REST architectures, and the CoRE Link Format. In particular, CoRE provides a default URL to expose links and enable Web browsing for constrained devices.

Things of the WoT are identified with an IRI, but the target device might not be able to communicate over HTTP. In this case, a proxy interfacing the WoT on top of the underlying IoT network is necessary to map HTTP to an ad-hoc IoT protocol. Applications communicate with Web servers, but these servers are usually not directly connected to IoT devices: dedicated gateways are deployed to ensure the communication. This deployment pattern is discussed in more details in Section §2.5.4. Depending on the technological stack implemented, the separation between

³¹<https://www.w3.org/WoT/WG/>

the IoT part and the WoT part of a deployment, *i.e.* the protocol bridging from/to HTTP, may vary.

For instance, in our example, multiple gateways are deployed to ensure the communication between the end devices and the server. A Raspberry Pi³² equipped with an EnOcean and a Z-Wave dongle is connected over WiFi to the home computer. Therefore, the HTTP request from the smartphone is routed by the server towards the Raspberry Pi, where it is mapped to the appropriate target technology. Similarly, a proprietary bridge implements a Web server mapping HTTP requests towards the ZigBee lamp. In the case of the Phidget devices, a Phidget driver is directly installed on the home computer, where the devices are connected.

2.4.2 From the WoT to the SWoT

Relying on Web technologies to expose devices and IoT data to applications brings interoperability at the technical level: applications can access representations of the devices over protocols they understand, such as HTTP. However, the notion of interoperability (refined in Chapter §3) is richer than just being able to communicate over the same protocol. Once a message is exchanged, it has little value if its content cannot be understood by its recipient. That is why in parallel to the WoT, the SWoT was developed [Scioscia 2009, Pfisterer 2011] in order to achieve semantic interoperability in IoT networks. The notion of **SWoT** covers the **integration of SW principles and technologies in the IoT**, which is often but not necessarily conjoined with the WoT. A sensor network where observations are enriched with an ontology, but where resources are not accessible through HTTP, would be an example of a SWoT instantiation not based on the WoT. It is however an exception, and in general the SWoT is built on top of the WoT: it is an assumption made for the remaining of this work.

The convergence of the IoT and the SW domains is supported by their common motivation: **producing content with machines, meant for machine consumption**. By nature, IoT systems are based on M2M communication, which is at the core of the SW. That is why fundamental components of the SW, that is to say linked data principles, formal vocabularies and deductions mechanisms are integrated into the IoT to form the SWoT. Ontologies are used in the SWoT systems to describe both **Things** of IoT systems, the **environment** in which they are deployed and the **information** they manage (observations and actuations). The potential applications enabled by the interoperability, dereferencability and formal properties of the SW technologies for the IoT in the SWoT are many. A survey of the recurring practices in the SWoT is proposed in Chapter §4.

The dereferencability of the SW vocabularies enable the discovery of descriptions at runtime, which is required by the dynamic nature of IoT networks. The mobility of devices and users makes the assumption of a static known environment *a priori* impossible in general. That is why producing machine-understandable content and services descriptions supports the realization of a global IoT network. The formal

³²<https://www.raspberrypi.org/>

nature of SW vocabularies enables the development of reasoning applications adding value to the information manipulated by IoT devices by leveraging its context, and thus access aspects of this information that might be implicit in the first place.

2.5 SWoT deployments architecture

From the definitions provided of IoT and WoT, it is clear that SWoT deployments are supported by multiple machines which capabilities are highly heterogeneous, from small IoT devices to powerful servers. The purpose of this section is to identify some defining characteristics of these multiple machines. First, a unifying abstraction is proposed in Section §2.5.1 to ease further discussion. This abstraction is then used in Section §2.5.2 to identify more homogeneous classes of nodes, which are aligned to identified concepts in Section §2.5.3.

2.5.1 Abstracting devices and services with nodes

In the definition we adopted for IoT, the notion of Thing refers indifferently to both to physical devices, such as sensors or actuators, and to virtual entities such as services. This choice sheds light on the similarity of the roles devices and services have, especially in the WoT. A service can be seen as the exposed endpoint of a device, and a device can be seen as the physical implementation of a service, depending on the chosen perspective. However, the word “Thing” conveys an ambiguity, among its multiple definitions³³:

- “Some entity, object, or creature that is not [...] specifically designated or precisely described”
- “A material object without life or consciousness; an inanimate object”

The first definition is the meaning we intend, because the second is more restrictive as it only pertains to physical entities and excludes services or virtual representations. For disambiguation purpose, we introduce here the notion of **node, an abstraction covering both device and service** concepts. Essentially, **a node is an active entity that can be addressed on the network**. An **active entity** is able to send and/or receive requests, which is the case for both devices and services. This definition extends to virtual representations, where a third party acts on behalf of an entity that is logically defined, such as the composition of two services. An entity is considered active as opposed to a passive entity which would have a URI, but cannot answer to a request, such as a concept in a KB. The notion of node is already present in architectures described in [Ben-Alaya 2015] or in [Perera 2014a]. Such abstraction allows to refer not only to the physical device, but also potentially to its virtual representation. Virtual objects are important components of the SWoT [Nitti 2016], and the IoT design pattern Device Shadow [Reinfurt 2016] is designed for capturing such virtual representations.

³³<http://www.dictionary.com/browse/thing>

Being an abstraction between device and service, the notion of node is also a tool to analyze contributions from different research perspectives. For instance, devices characteristics are used for self-configuration in [Chatzigiannakis 2012], while services descriptions are used for automatic composition in [Han 2012] or [Compton 2009b]. Analyzed from the node perspective, contributions focused on services can be applied to devices, and reciprocally.

Moreover, the notion of node we propose is not limited to IoT devices, and it is intended to refer to any type of connected entity supporting a SWoT deployment. In the next section §2.5.2, this generality of the proposed “node” concept is used to propose an operational classification of the machines and devices supporting the SWoT.

2.5.2 Identifying three node classes

We saw that **not all nodes of an IoT network are equivalent**: some devices are very constrained, whereas the servers providing analytic capabilities are powerful machines. In our literature search, three homogeneous classes of nodes are identified, namely “powerful nodes”, “middle nodes”, and “constrained nodes”. The distinction between these classes is based on a clusterization driven by characteristics summarized in Tab. 2.1.

The core characteristic of a node is its **processing power**, *i.e.* its ability to apply treatments of varying complexity to content. The processing power also determines the ability of the node to process content of a varying expressiveness, from the very simple agreed-upon byte array to the much more complex KB instantiations. The higher a node’s processing power is, the more expressive content it can handle, and the more complex operations it can achieve. Nodes are also characterized by their **memory**, *i.e.* the quantity of information they can hold at a given time, and **storage** capability. Storage is the available space giving access to persistent content. The notion of IoT node is inseparable from the notion of connectivity, and a node can also be classified according to its **communication capabilities**. These capabilities include the protocols it supports, its general availability on the network, and its bandwidth. Nodes also differ by the nature of their energy source: while some nodes are attached to traditional power grids, other nodes, deployed in the field, are reliant on batteries, or on renewable energy sources like solar panel, or energy harvesting.

The definitions for these three clusters are loose enough to be projected onto similar notions presented in the literature:

- **Powerful nodes** are named “weakly constrained nodes” in [Zanella 2014], Infrastructure Node in [Ben-Alaya 2015], and are referred to as Cloud in [Liu 2015, Szilagyi 2016]. In this category of nodes, we classify remote or local Cloud servers as well as powerful devices compared to other nodes of the deployment, potentially including standard laptops and domain-specific mobile robots or machines with powerful computation capabilities embedded.

Table 2.1: Node characteristics

	Energy supply	Processing power	Communication capabilities	Memory	Storage
Powerful node	Traditional power grids	High to very high	Web and internet protocols	High to very high (Several Go)	Large to very large (internal HD to disk bay)
Middle node	Mixed	Medium	Extended : IoT, Web and internet protocols [Desai 2015]	Medium to low (Up to 4Go)	Medium to limited (SD card to flash)
Constrained node	Often limited: battery, renewable source	Very limited, often μ controler	Constrained, Ad-hoc, potentially short range (BLE, Z-Wave) [Desai 2015] [Zanella 2014]	Very low (Under 500Ko)	Limited (Flash memory to none)

They have high processing power, extended communication capabilities, and large storage capabilities. For instance, in [Le-Phuoc 2016] or [Su 2018], powerful nodes are servers which characteristics are provided in details, in charge of performing demanding operations on large data quantities. The home computer can be considered a powerful node in our use case.

- **Middle nodes** are very often referred to in the literature as **gateway** ([Compton 2009a], [Ben-Alaya 2015], [Desai 2015]), because they are bridges between powerful nodes and more constrained devices. middle nodes are usually dedicated to content transformation and protocol bridging: in [Barnaghi 2009], fog nodes are presented as intelligent nodes where content can be converted from its raw representation to a richer one. [Desai 2015] proposes an architecture where **the gateway is both a technical and semantic interoperability provider between the IoT and the SWoT nodes**, performing both protocol proxying and semantic annotation. In [Nikoli 2011] and [Zanella 2014], fog nodes are proxies for wireless devices networks. They are nodes where content gathered by sensors or sensing services is collected, transformed, and redistributed. The raspberry Pi and the proprietary bridge are middle nodes in the smart home use case.
- **Constrained nodes** typically have very limited power source, processing and communication capabilities, and little to no storage capabilities. These are by definition present in every IoT architecture, and in direct contact with the physical world. In some studies such as [Vlacheas 2013], these nodes are not directly present, their representation is wrapped by a gateway at the middle node level. Constrained nodes were mainly sensors in early studies such as [Compton 2009a], and evolved toward diverse nodes including actuators, displays and composite devices in more recent work such as [Ben-Alaya 2015].

The end devices of the use case are constrained nodes.

2.5.3 Characterizing the identified node types based on the complementary Cloud and Fog computing paradigms

So far, the classes of nodes we defined have been identified with ad-hoc names. The criteria considered in this classification have been discussed in other fields of the literature, in particular Cloud and Fog computing, defined thereafter. Relating these domains to the node classes we defined would allow to replace ad-hoc concepts with elements shared in a wider community, and would enable interrelation.

2.5.3.1 The Cloud computing paradigm

Managing powerful servers with little concern about their computing resources is one of the main characteristics of Cloud computing, a concept which definitive definition has been provided by the NIST in [Mell 2011]. According to the NIST, “Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction”. Cloud resources are characterized by their **network accessibility** and their **elasticity**, making them an interesting relay for SWoT applications. Data is collected at a large scale by Things of IoT systems, and then concentrated on Cloud nodes where it is processed, stored, and accessed by applications.

However, SWoT architectures are characterized by the presence of constrained nodes, which limited communicating capabilities might prevent from accessing remote Cloud servers. These node being disseminated in the environment, processing the collected data in a remote Cloud architecture might also introduce a delay due to the necessary communication.

2.5.3.2 The Fog computing paradigm

In order to complement the Cloud characteristics with features suitable to IoT constraints, the Fog computing paradigm was proposed in 2012 [Bonomi 2012]. The authors introduce Fog computing as an approach for using processing power and storage capabilities located **at the edge of the network**, in between Cloud nodes and IoT devices. Fog computing is defined by the Open Fog Consortium³⁴ as a “system-level horizontal architecture that distributes resources and services [...] anywhere along the continuum from Cloud to Things”. **Fog nodes are massively distributed, heterogeneous, and they provide limited processing power between IoT devices and Cloud nodes.** With this definition, standard IoT gateways connecting Things to Cloud nodes are part of the Fog architecture. Since the SWoT technologies and principles have been initially deployed in Cloud architectures, based on data collected by Things, there necessarily is a layer to connect

³⁴<http://openfogconsortium.org/>

the ad-hoc IoT networks to Cloud nodes. Therefore, **Fog nodes intrinsically part of the SWoT architecture.**

Fog computing is a paradigm which characteristics differ fundamentally from Cloud computing.

- Fog devices are characterized by their **proximity with IoT devices**, bringing computing capabilities closer to data sources. Therefore, by design, Fog computing tackles the trombone effect introducing delay when a piece of data is produced by an IoT device, is processed in a remote Cloud node, and triggers an action that is sent back to the IoT device.
- A distributed approach based on the Fog computing provides resilience by removing a single point of failure, and by distributing some computing power in a myriad of devices.
- Fog computing also supports the scalability of Cloud architectures to face the expansion of IoT networks [Dastjerdi 2016].
- In the same paper, the authors discuss the support for user mobility by the Fog computing, which is especially adapted to IoT use cases where devices are disseminated over a wide geographic area, where both users and devices are mobile.

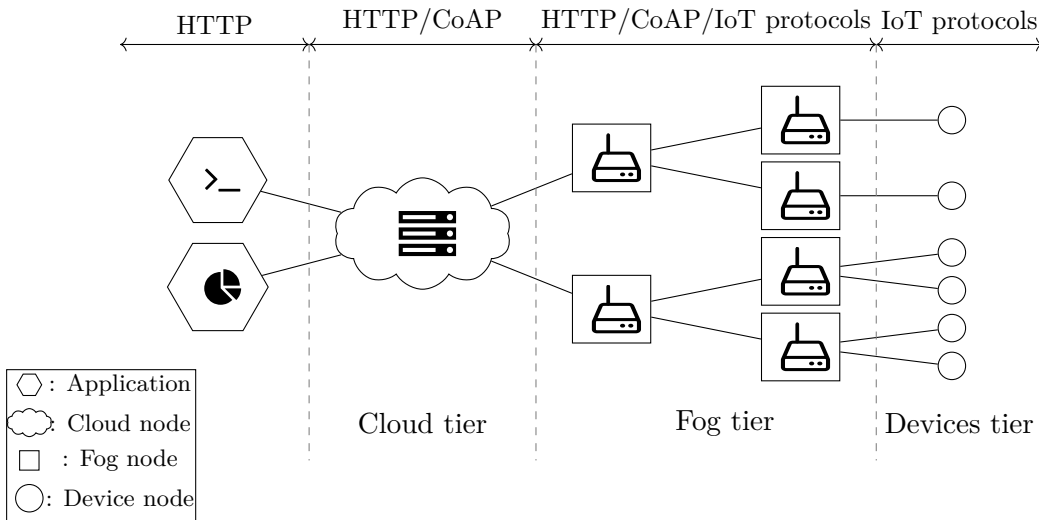
Offloading computation from the Cloud node to devices at the edge of the network is a very active research domain, with many interconnected concepts, such as Mobile Edge Computing (MEC), Cloudlets, or Mist computing [Patel 2017]. However, the Fog computing is not introduced as a paradigm meant to replace Cloud computing: its limited computing capabilities, as well as the locality of the scale of its deployments, are not suited to support Cloud computing use cases. Cloud and Fog computing are two complementary approaches, and their combination is seen as a solution to provide an architecture supporting the deployment of complex IoT and SWoT applications [Sahni 2017].

2.5.3.3 Redefining node classes

The node classes defined previously in Section §2.5.2 have properties that can be related to the Cloud and Fog paradigms.

- The computational power, the large storage capacities and the online availability of the **powerful nodes** are similar to **Cloud** architectures.
- The large communication capabilities and medium processing power associated to the **middle nodes** are parallel to the definition we provided for **Fog** architectures
- The **constrained nodes** have capabilities that are typical of IoT deployments, and are therefore referred to as **Devices**.

Figure 2.1: Typical three-tiers SWoT architectural pattern



These three classes of nodes are used in the following section §2.5.4 to define a deployment architecture.

2.5.4 SWoT reference deployment architecture

With the adoption of the Cloud, Fog and devices concepts to refer to the classes of nodes identified in SWoT deployments, the architectural pattern briefly introduced in Section §2.4.1 can be refined. The separation between servers, gateways and devices constitutes a **three-tier architectural pattern: Cloud-Fog-Devices**, depicted in Fig. 2.1. In such pattern, clients access services exposed by a Cloud node over Web protocols, connected to IoT devices by Fog nodes acting as gateways. The Cloud-Fog-Devices pattern is recurring in the literature, even though the terminology is not always the same. Examples of similar architectures include [Xu 2016] with the Cloud-Edge-Beneath architecture, [Su 2018], [Ben-Alaya 2015], [Zanella 2014] or [Liu 2015]. The architecture of the proposed use case is a direct implementation of this pattern.

Cloud and Fog nodes usually communicate over HTTP, but other protocols, such as CoAP or MQTT, may also be used depending on the use case and the deployment. An IoT design pattern capturing the role of the gateway as a facade for multiple communication technologies is proposed in [Reinfurt 2016]. The communication among Fog nodes is not necessarily homogeneous, since they may have very different communication capabilities from one another. At the lowest level, the communication is constrained by the capability of devices nodes: Fog nodes must implement the protocol enabled for the IoT devices they have to be connected with.

How SW technologies and principles are precisely deployed within the Cloud-Fog-Devices pattern is discussed in a survey presented in Chapter §4.

2.6 Conclusion

This chapter aimed at giving an overview of the technological landscape in which the work of this thesis has been executed. Being at the interface between two technological fields, a broad spectrum of concepts is covered.

First, the IoT paradigm has been introduced as the **networking of physical and virtual Things**. One of the characteristic features of IoT architectures is the presence of constrained devices, disseminated into the environment. The pervasiveness of IoT devices leads to energy constraints, as well as to the necessity for energetically sober wireless communication.

Second, the SW principles and technologies have been briefly described. **Design principles for Linked Data** are combined with descriptions based on **ontologies** in order to create **interconnected knowledge bases**. These KB, by describing information with formal vocabularies, enable the inference of new information based on logical deductions. Such inference process is performed by reasoners based on vocabularies such as RDFS and OWL, embedding logical axioms. In order to allow custom reasoning processes, ad-hoc logical axioms can be expressed based on rules.

On top of these two domains, the SWoT has been introduced as the convergence of the IoT and the SW. Broader access to IoT networks is granted by enabling Web endpoints over IoT infrastructures, leading to the emergence of the WoT. Interoperability issues in the IoT require to use machine-understandable knowledge representations, which are provided by the SW.

A typical SWoT architecture has then been captured into a three-tier architectural pattern. The top two tiers have characteristics related to the Cloud and Fog paradigms, which is why the **identified architectural pattern is referred to as Cloud-Fog-Device**. This pattern enables a full-stack interoperability by weaving IoT and Web technologies in a structure fostering SW technologies and principles.

However, achieving interoperability in the IoT domain by deploying such SW techniques and principles is a challenging task. In the next chapter, the notion of interoperability is defined in more details. How ontologies contribute to interoperability is discussed, by considering the question: **“What is a good IoT ontology?”**

Achieving semantic interoperability with the SWoT

Contents

3.1 State of the art	26
3.1.1 Refining the notion of interoperability	26
3.1.2 Toward semantic standards for the IoT	27
3.1.3 The ontologies of the SWoT	30
3.2 Contribution I.A: IoT-O, an ontology for the IoT	33
3.2.1 Design requirements for IoT ontologies	33
3.2.2 IoT-O, a modular core-domain IoT ontology	43
3.3 IoT-O in use	46
3.3.1 Motivating use cases around the ADREAM smart building	46
3.3.2 Smart building use case: Making data reusable with OPA	48
3.3.3 Data federation use case: Integration into FIESTA-IoT	51
3.3.4 Contribution I.B: User-centric smart home use case and autonomous control with semIoTics	53
3.4 Towards semantic interoperability for constrained devices	60
3.4.1 Enrichment and Lowering: IoT content management functionalities	61
3.4.2 A pivot-tree based approach to mapping reversal-based knowledge lowering	63
3.5 Conclusion	65

Interoperability is a core issue in the IoT. It drives the convergence of the SW and the IoT toward the SWoT. Standards are interoperability enablers in all technological domains, and two standards are particularly interested in the SWoT, namely oneM2M¹ and the W3C WoT². Moreover, multiple ontologies have been proposed to model the IoT domain. This chapter introduces the first scientific and technical contributions of this thesis:

¹<http://www.onem2m.org/>

²<https://www.w3.org/WoT/WG/>

- **Contribution I.A is the proposition of IoT-O, a modular core domain IoT ontology compliant with IoT ontologies design requirements we defined.** The scientific value of the contribution comes from the design process of IoT-O, motivated by the need for interoperability. Ontology design requirements dedicated to ontologies are used to assess ontologies of the state of the art, before being implemented in our contribution. IoT-O reuses reference ontologies to cover identified IoT sub-domains, and explicit alignments to other core domain ontologies are made available to support its own reusability.
- Usage of IoT-O is then described through three use cases, where different applications of interoperability are considered. In particular, IoT-O is used for the **autonomic control of a smart home by semIoTics, representing a technical contribution numbered I.B.**

Contributions I.A and I.B have been published in [Seydoux 2016b].

The notion of interoperability is discussed in Section §3.1, with a particular focus on the role of standards for interoperability in technical domains. Existing ontologies are also identified, and evaluated against quality criteria defined in Section §3.2. Contribution I.A is detailed in this section. Section §3.3 describes three use cases in which IoT-O is used to provide interoperability. In particular, contribution I.B is discussed. Finally, Section §3.4 pivots the issue from interoperability in general to interoperability specifically directed towards constrained devices. The content of this section, which introduces issues discussed in the remainder of this thesis, has been published in [Seydoux 2016c].

3.1 State of the art

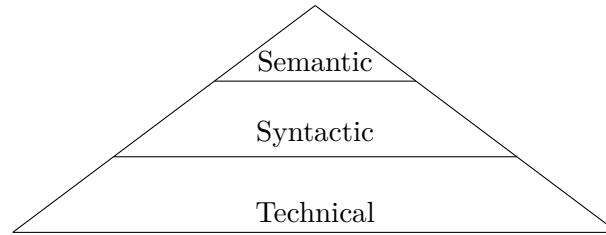
Our first contributions being dedicated to semantic interoperability in the IoT domain, it is necessary to define interoperability in the scope of our work. Standards relevant to this interoperability type are then introduced, before presenting IoT ontologies.

3.1.1 Refining the notion of interoperability

Interoperability is a notion that can be declined at multiple levels, each one dependent on the previous [Gyrard 2015], represented on Fig. 3.1.

- **Technical interoperability** requires system to communicate using the same channel and protocol, *e.g.*, Bluetooth, WiFi or Ethernet. Systems achieving technical interoperability are able to exchange messages.
- **Syntactic interoperability** requires system to share a common message format, *e.g.*, JSON or XML. Systems achieving syntactic interoperability are able to parse the message they exchange.

Figure 3.1: Interoperability levels defined by [Gyrard 2015]



- Finally, **semantic interoperability** requires systems to agree on the meaning of the terms used in the exchanged messages. Systems achieving semantic interoperability are able to understand the parsed message.

The level of abstraction of the interaction between systems increases with interoperability types. Technical interoperability is achieved at the physical level, and in the lower layers of the OSI model (ISO/IEC 7498-1). Syntactic interoperability is achieved at the protocol level: it is based on the compliance of exchanged messages with predefined schema. The schema is not necessarily available to the machine, and its internal logic is not necessarily explicit and machine-understandable. Finally, semantic interoperability is achieved at the conceptual level: the agreement does not target the serialization of the exchanged message, but rather the meaning it conveys.

3.1.2 Toward semantic standards for the IoT

Interoperability is an issue in many technological domains where several stakeholders have to interact with each other. Agreeing on standards is a response to face interoperability issues. Standards are typically providers of syntactical interoperability: they define data schema that must be respected by the implementations. However, such interoperability has some limits, and in the domain of the IoT, some standards now offer a layer of semantic interoperability as well. oneM2M and W3C's WoT are two such standards, described in the remainder of this §.

3.1.2.1 oneM2M

oneM2M is an international consortium of Standard Developing Organization (SDO), and authors an eponymous standard dedicated to the IoT. The consortium is composed of several working groups, each dedicated to an aspect of the standardized features, *e.g.*, architecture, protocol or security. In particular, a group is dedicated to Management, Abstraction, Semantics (MAS). The oneM2M standard aims to offer a unified horizontal RESTful interface to heterogeneous IoT deployments. To do so, it proposes a resource-oriented approach, where resources are organized in a hierarchical architecture. Common services, such as authentication, discovery or notification are defined within the standard, and they are implemented at different level in host platforms, called nodes. An Infrastructure Node (IN) is the root

node of a deployment, while Middle Nodes (MN) are intermediary nodes meant to be deployed on gateways between IN and devices. The oneM2M standard describes resources with predefined attributes, along with a protocol instantiating CRUD operations to interact with said resources. The abstract oneM2M protocol is bound with different existing protocols, such as HTTP, CoAP or MQTT. Syntactical interoperability is ensured by dedicated components, namely Interworking Proxy Entities (IPE). An IPE is an adapter for a specific technology, and makes a translation between messages compliant with the oneM2M protocol and the target technology. However, these mechanism only guarantee **syntactic interoperability**. Two system implementing the oneM2M standard will be able to successfully exchange messages down to the Content Instance granularity, which is the smallest unit of data storage in oneM2M. The data format stored inside the Content Instance is not standardized, and requires an *a priori* understanding between the systems to be understood.

The standard oneM2M architecture can be directly projected on the three-tiered SWoT architectural pattern introduced in Section §2.5.4 (seen on Fig. 2.1): the IN is the server, MN are gateways, and devices connect to these MN. Moreover, MN may implement a Web protocol, being thus part of the WoT, and they bridge with dedicated IoT technologies implemented by the devices that are not part of the standard.

In order to support a deeper interoperability, **oneM2M integrated semantic features** from its second release in 2017. Dedicated resources, namely Semantic Descriptors, can be attached to another resource to provide an RDF/XML description of its content. An ontology, the oneM2M Base Ontology (BO)³ is defined in the standard, in order to provide a reference vocabulary for resource annotations. These descriptors can be queried using the SPARQL language in order to enable a more expressive discovery. It is interesting to note that the gateway tier is potentially part of the SWoT deployment: implementing the semantic functionalities of the standard is not mandatory, but it is possible for both IN and MN. As we will discuss in Chapter §4, deploying the SW stack in the gateway tier is challenging.

Eclipse OM2M, a oneM2M open implementation Eclipse OM2M⁴ is a free implementation of the oneM2M standard, developed primarily at LAAS-CNRS. It offers a standardized horizontal REST interface over a potentially heterogeneous devices network. In order to ensure its extensibility and to ease the integration of new technologies, OM2M is modular: the platform internally uses the standard oneM2M protocol, which is technology-independent, and dedicated modules bind this core protocol to specific technologies. OM2M supports the use case introduced in Section §3.3.4. The public version of OM2M has been extended with the semantic functionalities of the standard in this thesis.

³http://www.onem2m.org/ontology/Base_Ontology/

⁴<http://eclipse.org/om2m>

3.1.2.2 W3C WoT

The W3C is an SDO dedicated to standardizing the Web, by promoting standards such as XML, HTTP or SOAP (for Web services). Moreover, the W3C also proposes Semantic Web standards, such as RDF or SPARQL. The W3C is composed on working groups, each focused on a Web standard, *e.g.*, CSS, SVG or Web payments. One of these working groups is dedicated to Web of Things since December 2017. The WoT working group⁵ proposes the WoT standard, composed of several components. The WoT is composed of an overall architecture, a Thing Description vocabulary, a scripting API, and a binding to other existing protocols. The WoT architecture is composed of building blocks that can be applied at the three levels of the Cloud-Fog-Device architectural pattern we proposed in Section §2.5.4. At the time of writing, the proposals of the working group are still a work in progress, and not yet official recommendations.

The standard seeks to enable both syntactic and semantic interoperabilities. The WoT standard is built around the notion of Thing, which is the equivalent of what we called “node” in Section §2.5.1: an entity, either physical or virtual, identified on the network. A WoT Thing must have a Thing Description attached, describing the characteristics of the Thing. The Thing Description is composed of four parts:

- Metadata giving general information about the Thing, such as its identifier, manufacturer, or location. The vocabularies used to provide this information are not in the scope of the WoT standard.
- Interaction information, describing the endpoint of the Thing and the interaction mechanisms it supports, *i.e.* request-response or publish-subscribe. **Describing what observations may be read and what actions may be triggered enables semantic interoperability.**
- Communication information, describing the supported protocols. Binding templates are blueprints of communication information that are can be instantiated in Thing Descriptions, and describe how to communicate with other standards, such as oneM2M. This description is **dedicated to technical interoperability.**
- Security information, enabling the management of access rights and the negotiation required to secure the communication channel.

The building blocks of the WoT architecture, deployed on a Thing and described by a Thing Description, constitute a Servient, both a server and a client. In order to embed some application logic in Things, servient may implement the WoT scripting API, which enable the development of scripts driving the interaction of the Thing with its surroundings. In order to enable Thing discovery, both for Web clients and servients, Thing Descriptions are stored in Thing Directories, aligned with CoRE

⁵<https://www.w3.org/WoT/WG/>

Resource Directories⁶. In addition of the CoRE Resources lookup methods, Thing Directories may implement SPARQL endpoints.

3.1.2.3 Interoperability beyond the standards

Other IoT standards exist, such as LWM2M⁷ or OSGi⁸. To the best of our knowledge, none of these standards consider semantic interoperability, which is why they are not discussed in this thesis. The emergence of standards considering semantic interoperability is an indicator of the technological and industrial drive behind it. Members of both the oneM2M MAS group (of which I am part) and the WoT group came together to write a whitepaper on semantic interoperability for the IoT [Murdock 2016]. The authors of this work identify semantic interoperability as an important value-enabler for the IoT, and promote good practices in the design and usage of ontologies for the IoT. The first step toward semantic interoperability is the use of ontologies, which are the focus of the remainder of this section.

3.1.3 The ontologies of the SWoT

Ontologies are structured knowledge models providing a machine-understandable vocabulary to describe a certain domain. The heterogeneity of IoT application domains leads to a multiplicity of ontologies enabling the SWoT. However, semantic interoperability is based on the use of either the same **reused ontologies**, or on the use of **ontologies aligned with each other**. That is why some reference ontologies emerged.

3.1.3.1 Proposing semantic models

Since the inception of the Semantic Web [Berners-Lee 2001], the notion of ontology has been associated with its principles and technologies. Ontologies for the SW are expected to be compliant with the LD principles introduced in Section §2.3.1: IRI-based identification, inter-linking, and usage of W3C standards. In the literature, many papers propose ontologies that fail to meet these expectations, by representing ad-hoc models such as [Avancha 2004], [Jurdak 2004], [Russomanno 2005], or in more recent work such as [Li 2015], [Hussein 2016], and [Wang 2017]. In these contributions, only a graphical representation of the ontologies is provided. In papers such as [Nachabe 2015], [Kibria 2015], [Pease 2017], the ontology is also graphically represented, and its actual implementation using Semantic Web technologies (such as OWL) is discussed, but the implementation is not available online. In both these cases, the data model is not accessible, which makes it impossible to reuse and extend as is.

The Linked Open Vocabularies (LOV)⁹ is a vocabulary portal built to foster

⁶<https://tools.ietf.org/html/draft-ietf-core-resource-directory-11>

⁷<http://openmobilealliance.org/release/LightweightM2M/>

⁸<https://www.osgi.org/about-us/working-groups/internet-of-things/>

⁹<https://lov.linkeddata.es/dataset/lov/>

reusability of ontologies by enforcing good practices and by enabling the search of existing ontologies [Vandenbussche 2017]. Some drastic criteria being prerequisites to reference vocabularies on the LOV, many ontologies proposed in papers from an IoT background failed to be shared this way, hampering their reusability. In order to enable sharing ontologies dedicated to the IoT and related domains, especially when they are not qualified for the LOV, [Gyrard 2015] proposed the LOV4IoT¹⁰. The LOV4IoT is a portal classifying ontologies into application domains, and providing a quality evaluation ontology according to the LOV best practices. Papers are qualified into six quality levels:

1. The ontology is not and will never be available online
2. The ontology is not online yet
3. The ontology is being made available online (ongoing work)
4. The ontology is available online, but does not implement LOV best practices
5. The ontology initially did not comply with LOV requirements, but now does
6. The ontology was initially made available compliant with LOV requirements

The LOV4IoT is built in order to be an incentive for improving the quality of IoT ontologies, that is why the categories 5 and 6 are separated. At the time of writing, the LOV4IoT references 450 ontologies, 31 of which are compliant with LOV best practices. A small subset of these ontologies is actually being reused by other ontologies and among other projects, these ontologies are described in Section §3.1.3.2.

3.1.3.2 Reference ontologies

SSN and SOSA: The first version of the Semantic Sensor Network ontology¹¹ (SSN) was proposed by the W3C in 2011 [Lefort 2011] as a synthesis of pre-existing data models. This ontology emerges as a de-facto standard in the SWoT, used in around a quarter of the SWoT papers referenced in this work. SSN is an ontology dedicated to the modeling of sensors and observations. Contextual information, such as time and location, as well as application-specific knowledge, are purposely left out of the ontology, as they are meant to be imported from other dedicated namespaces. [Wang 2015a] proposes a vision of the evolution of the IoT toward the SWoT. In the timeline drawn by the authors, the predominance of sensor networks in early stages of the IoT appears clearly. That is why the SSN ontology only focuses on sensors, instead of devices in general (including actuators or computational devices). In order to overcome this limitation, as well as unrelated technical issues, SSN was

¹⁰<https://lov4iot.appspot.com/?p=ontologies>

¹¹<http://purl.oclc.org/NET/ssnx/ssn>

redesigned in 2017, and split into a core module called SOSA¹² (Sensor, Observation, Sample, and Actuator) and an extension mapping to the former SSN terminology¹³.

The WoT ontology: The WoT ontology¹⁴¹⁵ is specified in the Thing Description proposed by the WoT working group. It defines terms associated with the WoT architecture, and focuses purely on interaction with the devices, leaving out elements present in SSN such as physical characteristics of the sensor or its deployment. The WoT group having a Web-oriented approach, elements from the Web domain such as authentication or crawlable links are represented in this ontology.

The oneM2M Base Ontology: As part of the oneM2M standard, the oneM2M BO¹⁶ is documented in Technical Specification TS-0012¹⁷. It is a core-domain ontology: it only provides the minimal set of concepts enabling the alignment of standard concepts with specific ontologies referenced by standard implementations. The BO defines high-level concepts such as Device, Service or Variable, which are related to concepts presented in the specification of the oneM2M standard architecture.

SAREF and its extensions: SAREF¹⁸ (Smart Appliances REference) is an ontology initially dedicated to the management of energy and services in smart homes, supported by the European Commission and adopted by ETSI as a technical specification¹⁹. Its construction was based on the semantization of pre-existing data models²⁰, and driven by the interaction with domain experts. Its scope has been broadened since [Daniele 2016], in order to propose extensions²¹ to new domains, namely environment²², smart buildings²³ and energy²⁴. A reference alignment between SAREF and the oneM2M base ontology is also available²⁵

IoT-Lite and M3 taxonomy: IoT-Lite²⁶ is a lightweight core-domain IoT ontology [Bermudez-Edo 2017]. It provides an extension of the first version of the SSN ontology with high-level definition for terms such as Actuating Device or Service. The simplicity and the versatility of the IoT-Lite core lead to its adoption

¹²<http://www.w3.org/ns/sosa/>

¹³<http://www.w3.org/ns/ssn/>

¹⁴<http://iot.linkeddata.es/def/wot>

¹⁵<http://w3c.github.io/wot/w3c-wot-td-ontology.owl>

¹⁶https://git.onem2m.org/MAS/BaseOntology/raw/master/base_ontology.owl

¹⁷<http://www.onem2m.org/technical/published-drafts>

¹⁸<https://w3id.org/saref>

¹⁹https://docbox.etsi.org/Workshop/2017/201710_IoTWEEK/WORKSHOP/S02_SEMANTIC_INTEROP/TNO_DANIELE.pdf

²⁰<https://sites.google.com/site/smartappliancesproject/ontologies>

²¹<http://saref.linkeddata.es/>

²²<https://w3id.org/def/saref4envi>

²³<https://w3id.org/def/saref4bldg>

²⁴<https://w3id.org/saref4ener>

²⁵https://git.onem2m.org/MAS/BaseOntology/blob/b9843e0c078888af915bcd823ff92d11f078ebb/Example_usage_of_the_Base_Ontology_-_combinig_SAREF_and_BO/BO_SAREF.owl

²⁶<http://iot.ee.surrey.ac.uk/fiware/ontologies/iot-lite>

in two European projects, FIWARE²⁷ and FIESTA-IoT²⁸. In order to support a unified description of heterogeneous devices and domains of interest, the FIESTA-IoT project promotes an extension of IoT-Lite, the M3-Lite taxonomy²⁹, and uses IoT-Lite as the core module of the FIESTA-IoT ontology³⁰.

iot.schema.org: schema.org is a a lightweight ontology proposing a wide variety of concepts relevant to resources commonly published online, such as blogs, online stores or online multimedia platforms. It has been extended with domain-specific vocabularies, such as automotive³¹ or bibliographic³², both based on W3C working groups. In particular, an IoT extension³³ has been proposed, loosely related to the W3C WoT working group³⁴. A description of the terms of the vocabulary is available online³⁵, and the machine-understandable representation of the vocabulary is available on a GitHub repository³⁶.

All the ontologies presented in this section propose a different model for the IoT, with its own assumptions and design choices. The multiplication of IoT ontologies, if they are not related to each other, does not support semantic interoperability. That is why one of the good practices of semantically enabled applications design is reusing existing ontologies. This leads to the necessity of identifying quality criteria for assessing ontologies in the IoT domain.

3.2 Contribution I.A: IoT-O, an ontology for the IoT

This section is dedicated to IoT-O³⁷, a modular core-domain IoT ontology that we proposed based on a previous work presented in [Ben-Alaya 2015]. The motivation for the proposition of IoT-O came from the evaluation of existing ontologies based on a set of quality criteria proposed in Section §3.2.1. Details about IoT-O are provided in Section §3.2.2, and finally applications enabled by the use of IoT-O are introduced in Section §3.3. This work has been published in [Seydoux 2016b].

3.2.1 Design requirements for IoT ontologies

Ontology engineering is a research domain on its own, with dedicated tools and methodologies. The design of IoT-O is compliant with the NeOn methodology, presented in [del Carmen Suarez de Figueroa Baonza 2010]. The purpose of this

²⁷<https://www.fiware.org/>

²⁸<http://fiesta-iot.eu/>

²⁹<http://purl.org/iot/vocab/m3-lite>

³⁰<http://purl.org/iot/ontology/fiesta-iot>

³¹<https://auto.schema.org/>

³²<https://bib.schema.org/>

³³<https://iot.schema.org/>

³⁴<https://github.com/schemaorg/schemaorg/issues/1272>

³⁵<http://iotschema.org/docs/full.html>

³⁶<https://github.com/iot-schema-collab/iotschema>

³⁷<https://www.irit.fr/recherches/MELODI/ontologies/IoT-O>

section is to **identify design requirements for a core-domain IoT ontology**, which is the first step of the NeOn process. Two types of requirement can be distinguished: **conceptual requirements**, regarding the concepts that should be present in the ontology (detailed in Section §3.2.1.2), and **functional requirements**, regarding the ontology structure and design principles (detailed in Section §3.2.1.1).

These requirements are used to analyze existing IoT ontologies. In addition to reference ontologies introduced in Section §3.1.3.2, ontologies with a similar scope as the one we are designing are included in the comparison. These ontologies are retrieved from the IoT and WoT sections of the LOV4IoT. Additional ontologies include:

- Hypercat³⁸, supporting the *BT Hypercat Data Hub* [Tachmazidis 2017], aimed at federating data from multiple sources.
- OpenIoT³⁹, integrated to the homonym middleware infrastructure platform⁴⁰
- MOFI⁴¹, dedicated to the description of resources within federated infrastructures [Al-Hazmi 2015]
- IoT.est⁴², used for the geospatial indexing of IoT nodes [Wang 2015b]
- iot-ontology⁴³ a generic IoT ontology used for IoT node description to enable automated deployment [Kotis 2012a]
- IoT-S⁴⁴, extending SSN with service descriptions
- SA⁴⁵, developed in the context of work related to service-oriented middleware for the IoT [Hachem 2014]
- Spitfire⁴⁶, developed to describe SWoT systems [Pfisterer 2011]
- STN⁴⁷, describing Socio-technical networks, *i.e.* network of people and IoT nodes
- SWOT-O⁴⁸, used for annotation by [Wu 2017]
- SemIoT⁴⁹, part of the SemIoT project⁵⁰ dedicated to IoT interoperability

³⁸<https://portal.bt-hypercat.com/ontologies/hypercat>

³⁹<http://sensormeasurement.appspot.com/ont/sensor/openIoT.owl>

⁴⁰<https://github.com/OpenIoTOrg/openiot>

⁴¹<https://github.com/alhazmi/mofi-ontology>

⁴²<http://personal.ee.surrey.ac.uk/Personal/P.Barnaghi/ontology/IoT.est.owl>

⁴³<http://ai-group.ds.unipi.gr/kotis/ontologies/IoT-ontology>

⁴⁴<http://personal.ee.surrey.ac.uk/Personal/P.Barnaghi/ontology/OWL-IoT-S.owl>

⁴⁵http://sensormeasurement.appspot.com/ont/sensor/hachem_onto.owl

⁴⁶<http://sensormeasurement.appspot.com/ont/sensor/spitfire.owl>

⁴⁷<https://w3id.org/stn/core>

⁴⁸<https://github.com/minelabwot/SWoT/blob/master/swot-o.owl>

⁴⁹<http://w3id.org/semiot/ontologies/semiot>

⁵⁰<http://semiot.ru>

The design of IoT-O being driven by the identified requirements, it is included in the evaluation for comparison purpose. Ontologies related to specific domains impacted by IoT (domotics, agriculture, smart cities...) are out of the scope of this study. We also did not consider ontologies upon which SSN was built, as they are focused on observation and measurement, which is a subset of the core domain we want to address.

3.2.1.1 Enforcing ontology design good practices with functional requirements

Functional requirements description: These requirements capture ontology design guidelines and general SW good practices in a domain-agnostic fashion.

Reusability: One of the most important aspects of an ontology in such a broad domain as IoT is reusability. If an ontology is ad-hoc to a project, the work done in its definition will not benefit further projects. It is a critical issue that can be solved by different, non-mutually exclusive approaches:

- **FR1:** the ontology is **compliant with the LOV requirements**. In order to implement its cataloging and search features, the LOV imposes requirements to ontologies features in its registry. Such requirements include metadata on the ontology, the availability of the ontology online (**FR1.1**), and the enablement of content negotiation at the ontology IRI. The complete compliance is achieved when the ontology is registered on the LOD (**FR1.2**) The intent is to improve visibility and reusability of ontologies, for both human and machines.
- **FR2:** The ontology is **modular**. As stated in [Aquin 2012], designing ontologies in separated modules makes them easier to maintain, reuse and extend. IoT applications being related to many various domains, capturing every aspect of the knowledge at stake in the same monolithic ontology is neither easy, nor desirable. The interdisciplinarity of IoT application design is another argument in favor of modular design: once domain experts produced an application-agnostic domain-specific ontological module, it can be reused as is in multiple applications, benefiting future developments. Modular ontologies can be combined together according to specific needs, which is a more scalable approach. An overview of a simple modular ontology is proposed in [Sheth 2008], where a meteorology use case is captured by different ontology modules: sensor, weather, time, and geospatial concepts are clustered in separate module composed for the application. Building a network of ontologies is also part of the NeOn methodology proposed by [del Carmen Suarez de Figueroa Baonza 2010].
- **FR3:** The ontology is based on **Ontology Design Patterns**. Ontology Design Pattern (ODP)s were introduced in [Gangemi 2005]. ODPs are similar to design patterns as they are used in software engineering: they capture

application-agnostic structures, with identified characteristics, providing a solution to a known recurrent issue. Designing ontologies based on ODPs increases reusability and their potential for alignment [Scharffe 2008]. Similarly to ontology modules, ODPs are reusable by nature. In order to ease sharing of ODPs, initiatives such as the ODP portal⁵¹ propose common repositories of patterns. ODPs are published accompanied with a textual description and a reusable skeleton OWL file, to ease their integration into new projects. ODPs capture modeling efforts: using them is a way to capitalize on previous work, and to take advantage of the maturity of the SW domain compared to the IoT domain.

- **FR4:** The ontology is **aligned to upper ontologies**. Upper-level ontologies define abstract concepts in a horizontal manner. Such definitions are too broad to be included as-is in an application, but their purpose is rather to be broad enough in order to cover concepts from different vertical domains. Therefore, two ontologies dedicated to different domains may be aligned to the same upper-level ontology, easing their common reuse in an application. Top-level ontologies may also be used to design ODP: there is for instance a close relationship between the ODP portal and DUL⁵², a top-level ontology: some of the proposed ODP are based on DUL. This ontology has been explicitly used to describe design patterns [Gangemi 2005]. Upper ontologies achieving high levels of formalism, they may appear overzealous for simple use cases, as it is pointed out by the designers of SOSA⁵³. In this case, modularization is also a technique enabling to harness complexity: the alignments to upper ontologies may be contained, so that they are only imported on-demand, and they are not considered by default.
- **FR5:** the ontology **reuses existing resources**. When designing a new ontology, reusing existing resources helps avoiding redefinition, and prevents from having to align a posteriori the redefined concepts to the existing sources in order to provide interoperability. Identifying existing resources, and reusing them, is also part of the NeOn process, as well as a guideline identified by [Bermudez-Edo 2017]. Avoiding redefinition of every aspect of a model from scratch allows to focus on the novelty of one's contribution, while ensuring interoperability by design with other applications based on similar resources. Popular resources emerge from such a process of collective reuse: since interoperability aims at having heterogeneous systems able to communicate, a network effect promotes resources that are already shared by a large number of participants. Such an effect is observed in Section §3.2.1.1. Reusable resources include, in addition to regular ontologies, ontological modules, ODP and top-level ontologies.

⁵¹<http://ontologydesignpatterns.org>

⁵²<http://www.ontologydesignpatterns.org/ont/dul/DUL.owl>

⁵³<https://www.w3.org/TR/vocab-ssn/#Modularization>

Table 3.1: Coverage of the functional requirements

	FR1.1 Online	FR1.2 LOV- compliant	FR2 Modular	FR3 ODP	FR4 Upper ontologies	FR5 External resources
SSN (v1)	**	**	*	**	**	*
SOSA	**	**	**	(**)	**	(**)
WoT	**	*				
oneM2M	**					
SAREF	**	**	**			*
IoT-Lite	**	**	*			**
FIESTA-IoT	**	*	**		**	**
iot.schema.org	*		**			*
Hypercat	**					
Spitfire	*	**			**	**
OpenIoT	*				**	**
MOFI	*	*	**			
IoT.est	**				**	**
iot-ontology	*				**	**
IoT-S	**		*	(**)	(**)	**
SA	*		*	(**)	**	**
STN	**	*	**			
SWOT-O	**	*				
SemIoT	**	*	**		**	**
IoT-O	**	**	**	(**)	**	**

Level of formalism: To use the full advantages of the semantic description of devices and data, the ontology used should enable reasoning and inference. However, for real-world applications, the model should also be decidable, and within a reasonable time. This requirement *de facto* excludes OWL-full models. **All surveyed ontologies are expressed in OWL-DL**, with the exception of iot.schema.org.

Functional requirements coverage An assessment of existing IoT ontologies regarding the presence of key concepts is summarized in Tab. 3.1.

- *: The requirement is partially covered:
 - FR1.1: The ontology is online, but not under its own namespace
 - FR1.2: Some LOV requirements are met but the ontology is not referenced in the catalog
 - FR2: The structure of the ontology is modular, but all modules are in the same file under the same IRI
 - FR3: A peripheral portion of the ontology is structured by an ODP
 - FR4: A peripheral portion of the ontology is aligned to an upper ontology
 - FR5: The ontology is inspired from existing ontologies, and the alignment is straightforward, but not explicit

- **: The requirement is completely covered
- (*): Conditions for partial requirement coverage are met in imported modules
- (**): Conditions for requirement coverage are met in imported modules

Results displayed in Tab. 3.1 show that SW best practices for reusability are not always followed: some ontologies are not available online under their own IRI (they are fostered by initiatives such as the LOV4IoT), and the majority of surveyed ontologies are not available on the LOV. Few ontologies we listed are reused by others as external resources: only SSN, SOSA, IoT-Lite and Spitfire are reused. Especially, SSN emerges as a *de facto* standard, being reused by 11 other ontologies among the ones we listed. SOSA being the new version of SSN, and being as its predecessor supported by the W3C, its adoption is likely to grow in the future. The large adoption of SSN is likely to be the cause of the large adoption of DUL as an upper ontology: SSN is aligned with DUL, and DUL is reused by 7 surveyed ontologies. Two ontologies (namely SA and *iot-ontology*) are aligned to SWEET⁵⁴, another top-level ontology. Smart Appliance REference (SAREF) has a particular approach to external resources reuse: it redefines the concepts present in multiple ontologies, and proposes alignments in an external, textual document. Design patterns have only been used in ontologies importing SSN.

The limited reuse of ontologies shows a lack of federating ontologies, apart from SSN. SSN being a modular ontology compliant with SW best practices, it is possible to say that these guidelines favor reuse. Moreover, the support of a standard organism may give leverage to an ontology. Initiatives emerged for instance to align the *oneM2M* ontology and SAREF, the latter being adopted as a standard by the ETSI⁵⁵.

It should also be noted that SSN is not *stricto sensu* an IoT ontology, but rather a sensor network ontology, which is a subset of the IoT. Therefore, the discrepancy of reuse between SSN and “proper” IoT ontologies can also be explained by the necessity to model the sensor domain for an IoT ontology (as it has been presented in Section §3.2.1.2), as opposed to the different modeling possible from one IoT ontology to the other. The ontologies we surveyed reused individual components modeling part of their competency domain, rather than ontologies proposing another model of the same domain. For instance, OWL-S, a service ontology is reused by IoT-S and IoT.est. Such reuse practice, of modules modeling a subset of the concepts required in the ontology that one is building, is also compliant with the FR2 requirement.

In order to cover both the identified conceptual and functional requirements, IoT-O is based on modules, each covering a conceptual requirement. Most of these modules are based on existing ontologies, described in Section §3.2.2.1.

⁵⁴<http://sweet.jpl.nasa.gov>

⁵⁵<https://portal.etsi.org/STF/STFs/STFHomePages/STF534>

3.2.1.2 Identifying IoT core concepts with conceptual requirements

Conceptual requirements definition Conceptual requirements come from an analysis of the IoT domain, driven by smart building use cases introduced in Section §3.3, but not limited to them. The use cases are not seen as an end *per se*, but rather as an instantiation of the general domain of the IoT. Non-semantic sources, such as the oneM2M standard and interaction with IoT experts, have been used as an input in the ontology construction process. Pre-existing ontologies, described previously in Section §3.1.3.2 and at the beginning of the present section have been taken into consideration. Ontologies that have not been made available online, and that are therefore not considered in the previous list, have been studied as well, from publications such as [Compton 2009b], [Nachabe 2015] or [Ben-Alaya 2015]. Finally, usage and remarks presented in publications such as [Barnaghi 2009], [Hachem 2011] or [Barnaghi 2012] have also been considered in the requirements construction.

To be reusable in a wide scope of projects, a core-domain IoT ontology should contain a set of key concepts. These are representative of IoT systems **with no regard to the application domain**. This approach facilitates the merging of data collected in different domains for horizontal applications, and allows the ontology to be an extensible core-domain ontology. We distinguish namely:

- **CR1: "Device" (CR1.1) and "software agent" (CR1.2)** constitute the two basic components of an IoT system, composed of both physical and virtual elements. The devices can be of two principle types, not mutually exclusive, listed below. The notion of software agent enables the description of logical modules integrated into IoT systems, such as a data processing algorithm or a remote data repository.
- **CR2: "Sensor"(CR2.1) are devices acquiring data, and "observation"(CR2.2)** describe the acquisition context and the data collected by the system. These concepts capture the perception of the evolutions of its own environment by the system.
- **CR3: "Actuator"(CR3.1) are the devices that enable the system to act on the physical world, and "action"(CR3.2)** represents what they can perform. These concepts capture the knowledge the system has on its own abilities to impact its environment, and to make it evolve.
- **CR4: "Service"**: By many aspects, the IoT and the programmable Web are very close. Connected devices can be seen as service providers and consumers, and by specifying a notion of service, every aspect of an IoT system can be represented.
- **CR5: "Energy"**: In the paradigm of pervasive computing, many distributed Things perform computations. Most of these Things being physical devices, a complete modeling of the system will include a description of their energy consumption. Energy management is a crucial topic in IoT systems.

Table 3.2: Coverage of the conceptual requirements

	CR1.1 Device	CR1.2 Software agent	CR2.1 Sensor	CR2.2 Observation	CR3.1 Actuator	C3.2 Action	CR4 Service	CR5 Energy	CR6 Lifecycle
SSN (v1)	**		**	**				*	*
SOSA			**	**	**	**			
WoT	*	*		**		**	**		*
oneM2M	**						**		
SAREF	**	**	*		*		**	**	*
IoT-Lite	(**)		(**)		*		**		
FIESTA-IoT	(**)		(**)		(*)		(**)		
iot.schema.org	*		*	**	*	**			
Hypercat			*	*					
Spitfire	(*)		*	(*)	*			**	
OpenIoT	(**)		(**)				(**)	(**)	*
MOFI				**			*		*
IoT.est	(*)		(*)	(*)	*		(**)	*	(*)
iot-ontology	(**)	**	(*)	(*)	*	*	**	(*)	(*)
IoT-S	(*)		(*)	(*)			(**)	(*)	(*)
SA	(**)		(**)	(**)	*		*	(**)	(**)
STN	*	*							
SWOT-O	(**)		(**)	(**)	(**)	(**)	(**)	*	
SemIoT	(*)								
IoT-O	(**)	*	(**)	(**)	(**)	(**)	(**)	(**)	(**)

- **CR6: "Lifecycle"**: Be it data, devices or services, IoT components are all included in different scales of lifecycles, because the IoT is by nature dynamic. Devices are switched on and off, services are deployed or updated, pieces of data become outdated... The evolution through a set of discrete states representing a lifecycle is an important concept for IoT systems.

Conceptual requirements coverage The assessment of existing IoT ontologies regarding the presence of key concepts is summarized in Tab. 3.2.

- *: The concept is represented superficially. It is present in the ontology, with few properties and a coarse-grained specialization.
- **: The concept is described in the ontology in details.
- (*): The concept is imported as-is from a remote ontology.
- (**): The concept is imported from a remote ontology, and its definition is enriched.

Sufficient coverage for CR1.1, CR2, CR5: The concepts identified in CR1.1, CR2 and CR5, namely Devices, Sensors/Observations, and Energy are already well-defined in reference ontologies. According to the reuse requirement FR5, there is no need to redefine such concepts for IoT-O. In particular, Semantic Sensor Network (SSN) is a widely used W3C recommended ontology for sensors and observations. It covers CR2 about sensors and observations, and it is also compliant

with FR1, FR3 and FR4, which makes it a fitting module to be **reused in IoT-O as is**. The notion of **energy consumption dedicated to the IoT is specified in PowerOnt**, an ontology referenced by SAREF.

Ad-hoc definitions for CR4 and CR6: Even if concepts capture in CR4 and CR6, namely Service and Lifecycle, have some coverage in the existing IoT ontologies, more detailed definitions exist in ontologies that are not specific to the IoT. To define the notion of service, IoT-O imports Minimal Service Model (MSM), a lightweight service ontology which is generic enough to represent both REST and WSDL services, as opposed to OWL-S⁵⁶, for instance, dedicated to WSDL-based services. hRest⁵⁷, proposed in [Kopecký 2008], is an extension of WSMO-Lite⁵⁸, upon which MSM is based. hRest is dedicated to the description of REST Web services, and it is integrated to IoT-O regarding the predominance of the REST model in the IoT domain. The concepts of lifecycle are described using Lifecycle⁵⁹, a lightweight vocabulary defining state machines discovered using the Watson semantic search engine⁶⁰. We extended Lifecycle in the IoT-lifecycle⁶¹ ontology with some classes and properties specific to the IoT, while taking advantage of the generic initial definition.

Lack of coverage for CR3: We can observe that some of the IoT ontologies cover most of the key concepts but none covers them all. Moreover, the different concepts are not represented with the same level of expressiveness. In *iot-ontology* or SAREF for instance, key concepts such as Actuator or Action are present, but their representation is limited. In these vocabularies, an actuator is defined as a device that modifies a property. This is less expressive than what can be expressed for a sensor with SSN which proposes a deep modeling of the sensors and the property they observe, but also of the relations between the sensors and their observations, and of the observations themselves. In eDIANA⁶², an ontology referenced by SAREF, some specializations of actuator are given, but the mappings from these specializations to the *saref:Actuator* concept are not available directly. This analysis highlights the fact that, before the publication of SOSA in 2017, no ontology provided satisfactory definitions for Actuators and Actions. In order to satisfy this requirement for IoT-O, while maintaining a modular design, **we propose SAN, an actuator ontology imported as a module of IoT-O**.

Semantic Actuator Network (SAN)⁶³ is intended to describe actuators the way SSN describes sensors. Actuators are devices that transform an input signal into a physical output, making them the exact opposite of sensors. SAN is built around

⁵⁶<https://www.w3.org/Submission/OWL-S/>

⁵⁷<http://www.wsmo.org/ns/hrests/>

⁵⁸<http://www.wsmo.org/ns/wsmo-lite/>

⁵⁹<http://vocab.org/lifecycle/schema>

⁶⁰<http://watson.kmi.open.ac.uk/WatsonWUI/index.html>

⁶¹<https://www.irit.fr/recherches/MELODI/ontologies/IoT-Lifecycle>

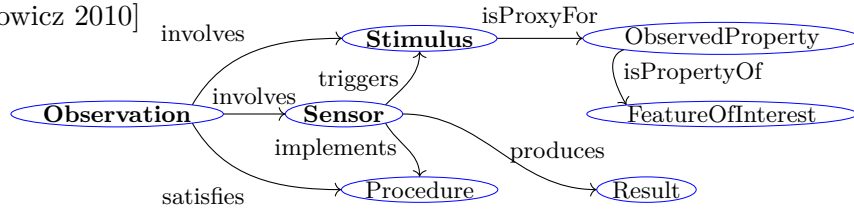
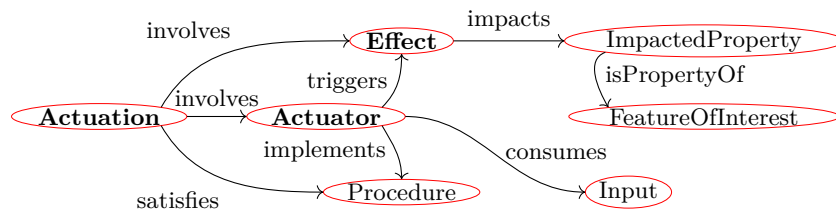
⁶²<https://sites.google.com/site/smartappliancesproject/ontologies/ediana-ontology>

⁶³<https://www.irit.fr/recherches/MELODI/ontologies/SAN>

Figure 3.2: SSO and AAE design patterns, structuring respectively SSN and SAN

SSO design pattern

[Janowicz 2010]

**AAE design pattern**

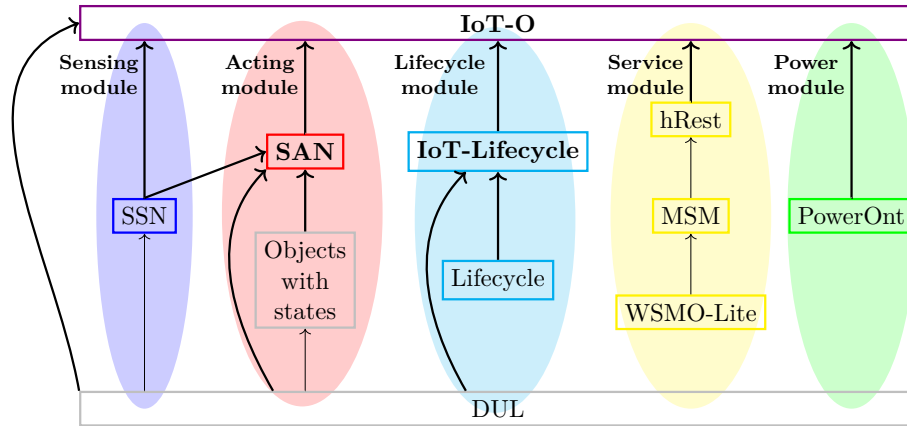
Action-Actuator-Effect (AAE)⁶⁴, a design pattern we propose, inspired from the Stimulus Sensor Observation (SSO) design pattern described in [Janowicz 2010]. Fig. 3.2 shows a representation of both the AAE and the SSO design patterns. SSN models the state of the world through stimuli converted by sensors into abstract observations, making the system able to be aware of the evolution of its environment. SAN is complementary: it models the transformation of abstract actuations by actuators into real-world effects, leading to the representation of the evolution the system brings into its environment. This complementarity enables the representation of an IoT system completely integrated into its environment, both able to be aware of its state and to modify it. Actuators are stateful devices: depending on their current state, they do not necessarily allow the same set of operations. For instance, a device might be turned off, offering a unique possible operation, the activation. In order to capture this statefulness, a dedicated pre-existing ontology designed pattern is integrated into SAN, Object With States (OWS)⁶⁵.

SAN was built with the same functional requirements as SSN, and it aims at covering CR3. It is intended to be used as an ontological module, it is aligned with an upper ontology (namely, DUL), and it reuses some concepts from SSN (such as *ssn:Device*). SAN is also available on the LOV. Further details of the main classes of SAN are provided in Section §3.3.4.3, in a detailed use case. Since its inception, SAN has been reused by [Wu 2017], which supports our claim that the functional requirements we identified support reusability.

⁶⁴<http://ontologydesignpatterns.org/wiki/Submissions:Actuation-Actuator-Effect>

⁶⁵http://ontologydesignpatterns.org/wiki/Submissions:Object_with_states

Figure 3.3: IoT-O's dependency graph



3.2.2 IoT-O, a modular core-domain IoT ontology

IoT-O, the core-ontology we propose is composed of several modules, organized as shown on Fig. 3.3. The names of the newly created resources are in red and highlighted, the names of the reengineered resources are underlined, and the arrows show dependencies. Solid arrows represent imports, and dashed arrows the reuse of concepts without import.

When we initially built it and put it online, IoT-O was referenced on the LOV4IoT, and thanks to the advice of Amelie Gyrard, we updated IoT-O and SAN to make them available on the LOV⁶⁶⁷.

3.2.2.1 The modules of IoT-O

An overview of the different modules of IoT-O is given on Fig. 3.3, which provides a dependency graph among ontological modules. Ontologies which name are in bold were built during the creation of IoT-O, and arrows in bold represent dependencies that were added in the same process, either by import or alignment. For instance, SSN and OWS already imported DUL, so no modifications were necessary to make them compliant with requirement FR4. In the case of the service module however, none of the ontologies were aligned to DUL. Such alignment is performed in IoT-O, but in order to avoid introducing side-effect inconsistencies in third-part ontologies that might import IoT-O in a context where its modules are already used, the concepts from modules are not directly aligned to the concepts from DUL. Instead, a new concept is created in IoT-O as a subclass of the two concepts we want to align. These new concepts are then used in the remainder of IoT-O instead of the concepts of the module from which they inherit, as it is shown on Fig. 3.1. The *imoto:Service* class is defined as a subclass of both *msm:Service*, *wsmo:Service* and *dul:FormalEntity*, instead of classifying *msm:Service* as a subclass of *wsmo:-*

⁶⁶<https://lov.linkeddata.es/dataset/lov/vocabs/ioto>

⁶⁷<https://lov.linkeddata.es/dataset/lov/vocabs/SAN>

Listing 3.1: Aligning Service definitions in IoT-O

```

ioto:Service a owl:Class;
  rdfs:subClassOf dul:FormalEntity, wsmo:Service, msm:Service.

ioto:hasService a owl:ObjectProperty;
  rdfs:domain dul:entiry;
  rdfs:range ioto:Service.

```

Service and *wsmo:Service* as a subclass of *dul:FormalEntity* directly. *ioto:Service* is subsequently used in the definition of properties such as *ioto:hasService*.

- The **Sensing module** describes the input data. Its main classes come from SSN: *ssn:Sensor* and *ssn:Observation*. *ssn:Device* and its characteristics (*ssn:OperatingRange*, *ssn:Deployment...*) provide a generic device description.
- The **Acting module** describes how the system can interact with the physical world. Its main classes come from SAN: *san:Actuator* and *san:Actuation*. It also reuses SSN classes that are not specific to sensing, such as *ssn:Device*.
- The **Lifecycle module** models state machines to specify system life cycles and device usage. Its main classes are *lc:State* and *lc:Transition*, and it is integrated into IoT-O with the Objects with States (ows)⁶⁸ ontology design pattern.
- The **Service module** represents Web service interfaces. Its main classes come from MSM: *msm:Service* and *msm:Operation*. Services produce and consume *msm:Message*, and RESTful services can be described with hRest.
- **Energy module**: IoT-O's energy module is defined by PowerOnt. It provides the *poweront:PowerConsumption* class, and a set of properties to express power consumption profiles for appliances.

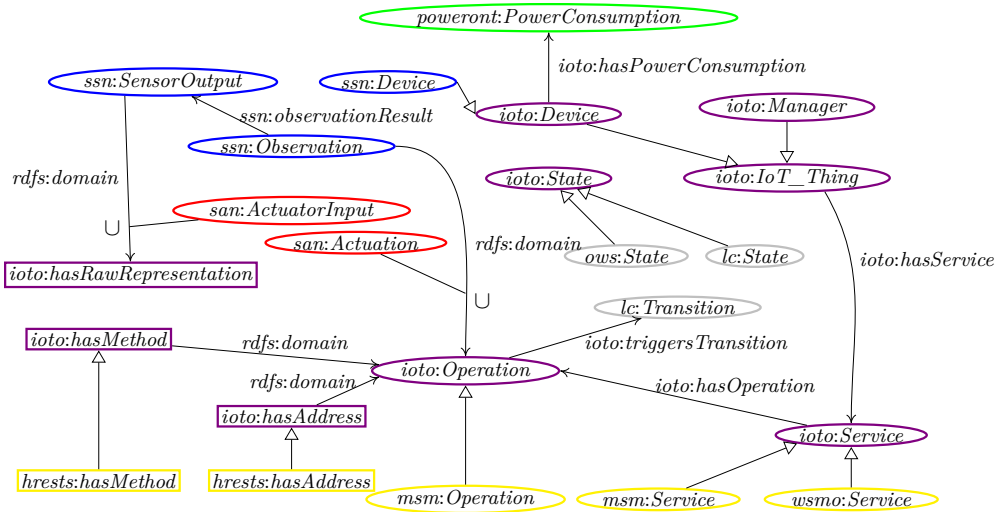
3.2.2.2 The core of IoT-O

IoT-O⁶⁹, prefixed *ioto:*, is both the name of the ontology and of the core module. It gives a conceptualization of the IoT domain, independent of the application, providing classes and relationships to **link the underlying modules**. Since many concepts are already defined in the modules, IoT-O's core is limited: it defines 14 classes (out of 1126 including all modules), 18 object properties (out of 249) and 4 data properties (out of 78). IoT-O key class is *ioto:IoT_Thing*, which can be either an *ssn:Device* or an *ioto:SoftwareAgent*. The power consumption of a *ssn:Device* is associated to *lifecycle:State* and *poweront:PowerConsumption*. *ioto:IoT_Thing*

⁶⁸<http://delicias.dia.fi.upm.es/ontologies/ObjectWithStates.owl>

⁶⁹<http://www.irit.fr/recherches/MELODI/ontologies/IoT-O>

Figure 3.4: Sample of IoT-O's core



is a provider of *msm:Service*, and an *msm:Operation* can have an *ioto:ImpactOn-Property* on an *ssn:Property*, **linking abstract services to the physical world through devices**.

As a core domain ontology, IoT-O is meant to be extended regarding specific applicative needs and real-life devices and services. This design, inspired by SSN, makes IoT-O independent of the application. Changes and corrections to the ontologies may be submitted as issues on the ontology public git repository⁷⁰.

3.2.2.3 Evolution of IoT-O

Context of publication In order to understand the motivation behind the proposition of IoT-O, it is important to consider the context of the contribution. The W3C WoT interest group was launched at the beginning of 2015⁷¹, and the semantic features of oneM2M were part of a 2017 release Previous work, since [Sheth 2008] in 2008 up to the proposition of SSN in 2011, has been mostly centered around the Semantic Sensor Web, and not the SWoT. When IoT-O was initially incepted in 2015 [Ben-Alaya 2015], and proposed in 2016 [Seydoux 2016b], to the best of our knowledge no other ontology was covering the requirements we presented in Section §3.2.1. Ever since, ontologies covering at least partly similar conceptual requirements, such as WoT, or SOSA, have been developed. The emergence of new standards and ontologies during my PhD has been both motivating, as it proved the liveliness of the research domain, and challenging. In order to avoid being a hurdle for semantic interoperability, IoT-O has been opportunistically aligned with other ontologies, as it is shown in the use cases in Section §3.3 or in the next paragraph.

⁷⁰https://framagit.org/IRIT_UT2J/ontologies

⁷¹<https://www.w3.org/blog/wotig/2015/01/20/launching-the-web-of-things-interest-group/>

Alignment with SOSA: In the process of adopting ontologies as recommendations, the W3C studies the support of said ontology in external resources. In order to maintain IoT-O, and to support the adoption of SOSA, which represents an evolution from SSN regarding conceptual requirements, we proposed a version of IoT-O aligned to SOSA⁷², which is bound to eventually become the reference version of IoT-O. This alignment also impacts SAN, since some of the concepts newly integrated in SOSA, namely its actuation module (*e.g.*, *sosa:Actuator* or *sosa:Actuation*), are related to CR3, a requirement covered by SAN.

For instance, *san:Acting* is a subclass of *sosa:Procedure*, and *san:Actuation* is a subclass of *sosa:Actuation*. In IoT-O, the service module is connected to the sensing and the acting modules by the *imoto:isGroundedBy* predicate. In the alignment to SOSA, the domain of *imoto:isGroundedBy* is a union of *sosa:Observation* and *san:Actuation*.

3.3 IoT-O in use

As a core-domain ontology, IoT-O is designed to be extended with application-specific modules. In this section, three use cases are presented, in order to show how IoT-O can support the development of interoperable IoT applications. The context of these use cases is presented in Section §3.3.1, and the use cases are respectively presented in Section §3.3.2, Section §3.3.3 and Section §3.3.4.

3.3.1 Motivating use cases around the ADREAM smart building

IoT technologies can have a direct impact on the everyday life of citizens, since it connects their physical environment to virtual applications. That is especially relevant in the case of domotics and smart buildings, where the home can be equipped with multiple low-power devices to provide new services. At LAAS-CNRS, the ADREAM project⁷³ aims at conducting research on smart buildings thanks to an instrumented, energy-positive building shown on Fig. 3.5.

Technical characteristics

- Area: 1,700 m²
- Technical facilities: 500 m²
- Office: 700 m²
- Photovoltaic: 100 kWp
- Solar panels area: 720 m²

Chronology

- Start of construction: June 2010
- Delivery of the building, installation of platforms: December 2011
- Host the 1st project: January 2012

The building is meant to have as little energy footprint as possible and is thus equipped with large sets of solar panels. Its heating and air conditioning systems

⁷²<https://www.irit.fr/recherches/MELODI/ontologies/IoT-0/sosa>

⁷³<https://www.laas.fr/public/en/adream>

Figure 3.5: Aerial photography of the ADREAM building



are also energy-optimized, with the use of natural ventilation, heat pumps and a ground-coupled heat exchanger. ADREAM is a living lab providing a horizontal platform to foster research projects, either focused on one aspect of the building or cross-domain to promote collaboration among research teams. Two aspects of ADREAM are relevant to our use cases:

- **The smart building itself**, equipped with more than 6500 sensing devices, producing up to 450,000 measures a day. Data is collected in four sub-systems:
 - **Lightning**, monitoring the luminosity in the building and the power consumption of lamps, as well as controlling the lightning dynamically.
 - **HVAC**, managing temperature and air flows
 - **Energy**, monitoring energy consumption of all the appliances in the building
 - **Photovoltaic**, measuring the power produced by the solar panels as well as the environmental conditions, especially weather.

The complete building is featured in an open data use case described in Section §3.3.2, extended in a data producers federation project described in Section §3.3.3.

- Inside the building, there is a **mock-up apartment** equipped with commercial devices from various vendors. Deployed devices include sensors (temperature, luminosity, humidity, pressure), actuators (fan, space heater, multiple lamps), which communicate using different technologies (phidget, Ethernet, zigbee) with gateways connected to a server. This apartment is a shared experimentation space among multiple research teams, and it is featured in an autonomic control use case described in Section §3.3.4.

3.3.2 Smart building use case: Making data reusable with OPA

The Open Platform for ADREAM (OPA) is an open data project supporting the dissemination of the data collected in the ADREAM building. Data is made accessible via the OPA portal in two forms: raw CSV, and enriched RDF. After motivating the creation of the platform in Section §3.3.2.1, the details of the creation of the KB supporting the data enrichment process is detailed in Section §3.3.2.2. The mechanism enabling the enrichment itself is described in Section §3.3.2.3.

3.3.2.1 Use case motivation

The purpose of this use case is to add value to the data collected by the building sensors, and to ease its reuse. A legacy system has been initially integrated in the building at construction time, to collect data and feed it to a Building Automation System (BAS) enabling visualization and remote control. Data is continuously stored in a first protected Relational Database (RDB), which is for technical reasons cloned daily into a second RDB where it can be queried. Observations collected in ADREAM are used for interdisciplinary research, such as energy management, green IT or thermics. They include weather observations, indoor temperature and luminosity, power production and consumption, as well as user inputs such as temperature request on thermostats. In order to enable the exploitation of this data in research projects, two issues are at stake: **accessibility**, and **reusability**.

Data is initially accessed through a Web form, with a limited expressiveness: a user may access data either from a certain sub-system (*e.g.*, Luminosity or Power consumption) over a specified time interval, or from a set of sensor manually described. Only specifying the sub-system has a very coarse granularity, but individually specifying sensors requires an expert knowledge on the building technical details in order to consume its data. Enhancing accessibility requires to allow a more flexible querying method, with a finer granularity yet not requiring an expert knowledge. The reusability issue is caused by the nature of the deployment: the system has been installed over a large period of time, by different vendors, leading to a heterogeneity of data models and naming conventions. Understanding the data without the engineer in charge of the building is very challenging.

In order to solve the accessibility issue, an open data was deployed, with a data portal providing multiple filters to access data at a finer granularity. Sensors have been organized in smaller consistent sets, used to cluster data into datasets. Two granularities have been identified from interaction with final users: the subsystem (as it was previously used), and inside each subsystem functional unit of devices, *e.g.*, in the Photovoltaic system two datasets are created, one for each type of current inverter. Datasets are also created for different time frames: day, month, year. The new portal allows to perform the same queries as the previous, but with a more intuitive interface and extra criteria. Data is also cleansed before being published on the open data. However, increasing accessibility does not improve reusability, and the data is still hard to understand for a non-expert. That is why a knowledge base has been built to describe the building, in order to provide a rich

description of the sensors and their organization. This knowledge base is used to transform the original CSV data files into RDF, enabling a more expressive querying and promoting the publication of data according to the principles of LD in order to increase its reusability [Christian Bizer 2009].

In order to give incentive to users to use enriched data, value must be added to the RDF files to compensate for the extra effort, compared to opening a CSV in a regular spreadsheet software. That is why generated observations are linked to the sensors that produced them, making it easier to locate each data point geographically in the building. Units have been unified, and explicitly attached to sensors. The expressiveness of queries enabled by the language, combined to complementary information provided about the system, are the added value of the enrichment. The construction of the description of the building and the system it contains is detailed thereafter.

3.3.2.2 Building the knowledge base

The knowledge base describing the ADREAM building is modular, in order to ease the separate development of the two use cases previously introduced. The central module, ADREAM-Core (prefixed *adr*:⁷⁴) is an extension of IoT-O defining classes required for the description of the building. It reuses concepts from taxonomies such as M3-Lite, generic for the IoT or Dogont⁷⁵, dedicated to the smart home. Types for devices deployed both in the building and in the apartment are described in this module, as well as generic concepts such as the observed properties (*e.g.*, *adr:Presence* or *adr:WindDirection*, aligned with the M3 taxonomy) or some indoor location elements (*e.g.*, *adr:hasIndoorLocation* or *adr:Apartment*). This module has been entirely created by hand, that is why it only presents a reduced number of features of the building. The apartment module, ADREAM-Apartment (prefixed *adra*:⁷⁶), has also been created manually, and it is described in the dedicated Section §3.3.4. Finally, the last module of the knowledge base, ADREAM-Building (prefixed *adrb*:⁷⁷), and the largest one, is the description of the living lab building itself. Due to the large number of devices in the building, a manual construction is not possible. An automated instantiation pipeline has been developed.

The instantiation process starts from the gathering of devices information stored in the legacy RDB. For each device, the following information is available:

- An identifier, built on a schema that differs from one vendor to the other. Devices identifiers are built based on a BAS naming rule system similar to the point name detailed in [Butler 2010]. Points are designed to provide The points for each subsystem, and within one subsystem, are not consistent with each other, nor are they described precisely in a documentation. An example of BAS naming rule is provided in Fig. 3.6: the device identifier provides

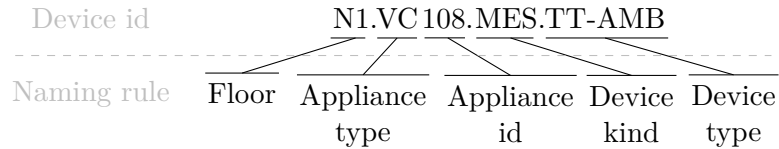
⁷⁴<https://w3id.org/laas-iot/adream>

⁷⁵<http://elite.polito.it/ontologies/dogont/dogont.html>

⁷⁶<https://w3id.org/laas-iot/adream-apartment>

⁷⁷<https://w3id.org/laas-iot/adream-building>

Figure 3.6: Example of BAS naming rule



information about the appliance the device is related to, about the kind of device (MES denotes a sensor), and about the device type (TT_AMB denotes an ambient temperature sensor).

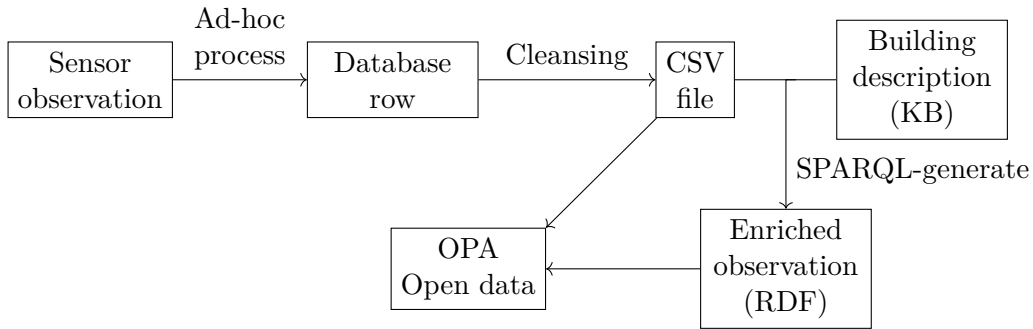
- The subsystem to which it belongs.
- Optionally, a descriptive sentence in free text, usually used to provide the sensor location.
- Optionally, the unit for sensor observations, with some inconsistencies: the same unit may be represented with different abbreviations.

A strong collaboration with the engineer in charge of the building has been necessary. She shared her knowledge of the naming systems for instance, in order to enable the exploitation of the BAS naming rules to instantiate individuals based on their identifiers. The logic she described has been formalized in a Java program that applies a series of rules in order to represent explicitly the characteristics of devices embedded into the naming system. Intermediary individuals are generated in the process, representing elements shared by multiple devices such as floors, rooms and even the appliances to which devices are connected. For instance, no lamp is represented as a standalone individual in the RDB, however pairs of devices, luminosity sensors and power consumption sensors, are associated to the same appliance. Therefore, the lamp is represented by an individual in the generated KB, and the attachment of the two sensors onto the appliance is represented with the *dul:AssociatedWith* object property. The obtained individuals are linked to classes from the common module *adr:*, from IoT-O, and from reused ontologies.

3.3.2.3 Using the knowledge base for enrichment

The cycle of data production by the ADREAM building has already been briefly introduced: data is collected continuously, and made available by batch once a day. This raw data is cleansed and stored as CSV files in the open data portal. The enrichment process takes these CSV files as input in order to generate the equivalent RDF graphs. The enrichment is performed based on a query using SPARQL-Generate, a SPARQL extension proposed by [Lefrançois 2017]. SPARQL-Generate enables CONSTRUCT-like queries with binding from heterogeneous formats, and not necessarily RDF.

Figure 3.7: Data enrichment and publication process



The daily conversion from CSV to RDF, representing approximately 450000 data points, lasts a hundred second on average, with daily variations due to server load and the number of observations.

The choice to make data available as RDF dumps instead of a SPARQL endpoint was made for a scalability motivation: a large quantity of data being uploaded each day, the size of the SPARQL endpoint would have grown permanently, reducing performances or queries. With RDF dumps, user may collect the data they are interested spread over several RDF files, and build an adapted SPARQL endpoint on their end. Moreover, RDF files can be compressed with a great efficiency to reduce the footprint of the datasets on the disk.

3.3.3 Data federation use case: Integration into FIESTA-IoT

3.3.3.1 Use case motivation

FIESTA-IoT⁷⁸ is a European project aiming at proposing a data federation among heterogeneous testbeds. The FIESTA-IoT platform is designed for end-user to run experiments on data collected in different IoT networks, such as smart cities or smart building. Testbeds are completely independent from each other, and they are based on very different IoT technologies according to their own requirements. In order to hide the underlying heterogeneity of data models, specific to each deployment, the platform requires testbed providers to annotate their data with a common vocabulary, the FIESTA-IoT ontology. This ontology is also used by end users in order to describe their experiments with SPARQL queries over the homogeneous model. The scalability issue that prevented us from deploying a SPARQL endpoint to share the ADREAM data is managed on the FIESTA-IoT side. Moreover, integrating the ADREAM data in the FIESTA-IoT federation makes it available to a larger audience, due to a network effect: by only using the FIESTA-IoT ontology, users have access to multiple datasets outside ADREAM.

The limit between IoT and WoT, already discussed in Section §2.4.1, can be seen clearly in the FIESTA architecture. The IoT part of the network is completely

⁷⁸fiesta-iot.eu

managed on the testbed side, where the technologies and design choices are made in complete independence. Each testbed implements a WoT interface on top of its local IoT deployment, and the FIESTA platform collects data from this WoT interface. The platform implements the SWoT technological stack, and exposes data to final users as SWoT resources.

3.3.3.2 Alignment with the FIESTA-IoT ontology

In the case of the ADREAM building, the data to be pushed to the FIESTA-IoT platform is already enriched with our own vocabulary. Starting from scratch by enriching the raw data would be possible, but it would not take advantage of the modeling effort that had already been put in the construction of the OPA platform. We also wanted to propose an approach that would be reusable by other partners in the project in a similar situation. By design, the FIESTA-IoT platform prevents a query-rewriting approach. Data must be uploaded to the platform annotated with the shared vocabulary: user requests cannot be redirected to a testbed specific endpoint where the query could be rewritten. The transformation must be centered on data, which leads to redundancy, since data will be available both in our testbed and in the FIESTA-IoT platform, but ensures compliance with the FIESTA-IoT platform requirements, since data is validated at each upload. Therefore, we selected an alignment-based approach to transform the annotated data from ADREAM-Building to FIESTA-IoT, initially proposed in [Euzenat 2008].

The alignments between the source and the target ontologies are described in EDOAL^{79,80}. ADREAM-Building and FIESTA-IoT are based on the same reference ontologies such as SSN and DUL, therefore their structure is rather close, and only simple alignments were necessary, as opposed to complex alignments [Thiéblin 2018]. To promote the reusability of our approach, a GUI tool has been developed in order to ease EDOAL alignments writing, for both simple and complex alignments. The tool is available online⁸¹ as free software.

The EDOAL alignments are then used to generate SPARQL CONSTRUCT queries: the WHERE clause captures elements of the source vocabulary, and the CONSTRUCT clause generates the equivalent elements described with the target vocabulary according to the alignments. Alignments are used once at conception time to generate the queries, and said queries are then used daily for data transformation. The EDOAL alignments are also an output of this work, and represent a resource *per se*, since they are not specific to the generation of request and provide interoperability between two existing ontologies. It is worth noting that the creation of the alignment was possible without *a priori* intervention of the expert, who validated it once written. Alignments have been built mostly by an intern, who did not take part in the initial creation of ADREAM-Building KB, which tends to support our claim that the description with a rich structured vocabulary increases

⁷⁹<http://alignapi.gforge.inria.fr/edoal.html>

⁸⁰<https://w3id.org/laas-iot/alignment/fiesta.edoal>

⁸¹https://framagit.org/IRIT_UT2J/ontology-tools-sandbox

reusability of data.

Integrating the ADREAM building to the FIESTA-IoT federation increased visibility of its data, since it has been used by end-users of the FIESTA-IoT platform that discovered the testbed through the federation. It is also a demonstrator of the reusability of data enabled by the SW principles and technologies.

3.3.4 Contribution I.B: User-centric smart home use case and autonomous control with semIoTics

3.3.4.1 Use case motivation

In this use case, the mock-up apartment models a smart home equipped for assisted living for an elderly. Simple scenarios are enabled, such as lightning automation, as well as advice regarding weather conditions, for instance hydration when hot. However, small highly distributed devices usually have a limited processing power, which restrict the range of applications they can support. More complex agents can interact with these devices to collect their data and perform advanced processing to provide a higher level service. In our use case, the complex agent is a PR2 robot, shown on Fig. 3.8. It is present in the house, and performs tasks such as helping the person in case of fall, moving heavy objects, pushing a wheelchair, fetching objects and bringing medications. Some of these tasks require the robot to know **where the person is** in the apartment. To have this information, the robot can move around the apartment, scan it with its embedded cameras, and through image processing figure out where the person is. However, it requires the robot either to follow the person around all the time, or to scan the apartment completely each time it has to find the person. To make the robot more acceptable to the person, the house can be equipped with an IoT system, collecting information useful to the robot, such as information given by **presence sensors**. Moreover, the connected devices provide new functionalities to the robot, by either extending its perceptions or its capabilities. Thanks to connected sensors, the robot can access ubiquitous observations of different properties of the apartment, and thanks to connected actuators he can easily interact with light bulbs or fans. Our use case is composed of two scenarios: the robot must bring pills at fixed hours to the person using the presence sensors to locate her, and the robot must control the temperature in the apartment using temperature sensors and connected fans to improve the comfort of the person. The conditions for one's comfort should be based on preferences expressed by oneself, hence the user-centricity of the proposed scenario.

In this use case, both syntactic and semantic interoperability are required, among the devices and between the devices and the robot. Syntactic interoperability is ensured using OM2M, on top of which another platform, semIoTics⁸², enriches the collected data with semantic descriptions, and makes them available to

⁸²“Semiotics is the study of meaning-making”, from <https://en.wikipedia.org/wiki/Semiotics>

Figure 3.8: PR2, the companion robot



the robot through a REST interface. SemIoTics is driven by a KB capturing knowledge about the devices of the system represented according to our core-domain IoT ontology, and about the environment shared by the robot and the devices (here, the apartment). It is a Java software I developed to showcase the role of SW technologies in IoT data management, based on Apache Jena. The robot itself is also a semantically enabled agent, it uses a "common sense" ontology and a KB to reason about its 3D environment, as described in [Lemaignan 2012]. The knowledge specific to the robot relies on ontologies out of the scope of this thesis, but any ontology can be included in its KB. In particular, the robot's KB has been loaded with IoT-O and its use case-specific extensions.

That is why the knowledge described needed for this use-case is implemented in a dedicated KB using IoT-O, ADREAM-Robot⁸³: the ontology is shared by the robot and semIoTics, and each system has its own KB. The synchronization between the KB of the different agents is out of the scope of this thesis. For instance, it could be used to support an air quality monitoring system in a smart city, by describing the sensors that collect the data and the services the citizens can subscribe to. The usage of IoT-O and its module in the use case is double: it is used to model the observations about modifications of the apartment, allowing the robot to keep an up-to-date representation of its environment, but also to model the changes the robot wants to make into the apartment through its actions and through the connected devices.

3.3.4.2 The MAPE-K loop and the knowledge at stake

One of the principles of user centricity, briefly discussed in introduction, is to establish intuitive connections between the intent of the user and the required actions [Abrams 2004]. The heterogeneity and complexity of IoT systems often require mul-

⁸³<https://www.irit.fr/recherches/MELODI/ontologies/Adream-Robot>

multiple low-level actions to achieve a global behavior, which is why self-management capabilities are desirable to support user-centered design in such complex systems. From this observation, including but not limited to the IoT, [Kephart 2003] proposes a new paradigm, namely **autonomic computing**, aimed at enabling self-management to harness complexity. The self-management abilities of an autonomous system are refined in four aspects: self-configuration, self-optimization, self-healing, and self-protection. The purpose of autonomic computing is to consider **the controlling agent and the controlled entity as one system**: actions of the agent have repercussions on the entity, that will be observed by the agent that will act on the entity in reaction, and so on. The autonomous system behavior is driven by introspective knowledge, and by high-level policies defined by human operators. These policies are then broken down by the system, based on its knowledge of its own components, in order to convert high-level instructions into a set of low-level, implementable commands. Therefore, **autonomic computing requires a shared understanding of high-level goals** among the components of the system, as well as their **semantic interoperability to support introspection**.

To enable self-management, the system must have the following characteristics:

- It is able to maintain a **self-representation**.
- It has the ability to **break down high-level policies** into components that can be compared to its introspective representation.
- It is capable of **taking actions in order to change its own state**, in order to be compliant with the goals set by high-level policies.

The MAPE-K loop is a classic control structure in autonomic computing (see Fig. 3.9), separated in four steps implementing the characteristics described above: Monitoring, Analysis, Planning and Execution. The K stands for Knowledge, because the behavior of the autonomic agent at each step of the loop is guided by a KB, in the general meaning of the term (including but not restricted to the W3C's formalisms of knowledge representation).

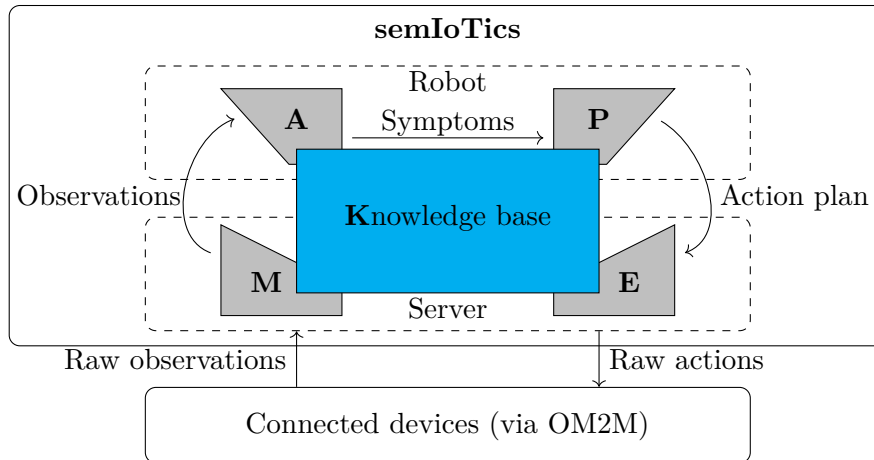
3.3.4.3 Implementing the MAPE-K loop with semIoTics

We propose semIoTics, contribution I.B of this thesis, as an implementation of the MAPE-K loop to drive this use case. semIoTics is deployed over two execution platforms:

- a computer acting as a server, performing the Monitoring and the Execution steps when connected devices are involved, and
- the robot, performing the Analysis and the Planning

semIoTics performs the complete cycle of IoT data as described in [Sun 2014]: observations from the physical-world are collected, they are converted as events in

Figure 3.9: Representation of the MAPE-K loop instantiation by semIoTics



the “cyber world”, which are used by a decision system to generate commands that have effects back in the physical world.

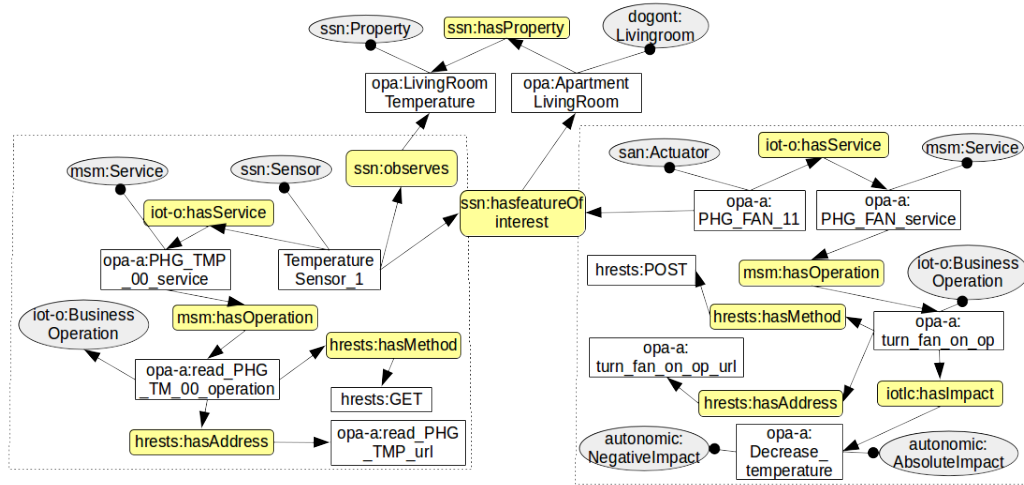
The process described in Fig. 3.9 structures the use case: data is first gathered by the sensors, and enriched by semIoTics on the server. Enriched observations are processed on the robot, where the embedded semIoTics instance decides to perform actions represented as enriched actuations. These actuations are sent back to the server, which translates them into raw commands for the actuators to perform.

The robot and the server have distinct KB, one hosted on each machine. However, the two systems exchange knowledge freely through a REST interface, which is why a unique KB is on Fig. 3.9. Consistency issues are not considered in this use case, as only one smart agent acts in the system. Both KB are loaded with knowledge necessary to describe the complete use case. In particular, ADREAM-Apartment is a KB extending ADREAM-Core, dedicated to the mock-up apartment. In complement to IoT-O, the Dogont⁸⁴ ontology is used to describe the apartment and the location of devices inside it. Dogont is an ontology identified in the SAREF project, imported by Poweront, that we aligned to IoT-O to integrate it to the use case. No new concepts are defined in ADREAM-Apartment, only individuals describing the three rooms and some of the dozen of devices monitoring their properties. The specificity of the apartment is that, thanks to OM2M, devices are associated to services enabling direct interaction. Therefore, the description of devices, in addition to properties defined by SSN or SAN identical to elements generated for ADREAM-Building, include REST service descriptions. An extract of ADREAM-Apartment, dedicated to the management of the temperature in the apartment, is provided on Fig. 3.10.

At the core of control mechanism shown on Fig. 3.10, the feature of interest is *adr:ApartmentLivingRoom*, and its property with which the system interacts, here *adr:LivingRoomTemperature*. The property is measured by a sensor, and impacted

⁸⁴<http://elite.polito.it/ontologies/dogont/dogont.html>

Figure 3.10: Temperature management in ADREAM-Apartment



by two actuators, one of which is not represented on the figure for space consideration. All the devices can be accessed through a REST service described by the *msm:Service* class, constituted of some *msm:Operation* instances. Said operations are associated to impacts they have on the property, specified with a vocabulary dedicated to autonomic computing, Autonomic, prefixed *auto*:⁸⁵. This module allows the modeling of the relation between actions and their consequences, in order to enable planing by the autonomic system. Moreover, it defines concepts enabling the definition of high-level policies over properties, such as upper or lower bounds to be enforced by the system. For instance, an operation supposed to decrease the value of a property is denoted by the concept *auto:NegativeImpact*. The orchestrations of the operations is represented as a state machine based on the Lifecycle module of IoT-O.

Overall, an interesting aspect of the use case is the decoupling it shows between the IoT devices from one side, and their abstract representation manipulated by the agent on the other. The technical and syntactic heterogeneity are hidden, and semantic interoperability is enabled by the combination of the standard platform and its extension with semantic functionalities. Similarly, the rules expressed by users as constraints on properties are independent from the underlying devices, and they are dynamically resolved in order to be adaptive to the dynamism of the IoT network.

In the remainder of this section, the implementation of each step of the MAPE-K loop by semIoTics is detailed, with a particular interest in the knowledge manipulated.

Monitoring, where raw sensor data become meaningful observations: The first step of the MAPE-K loop is the monitoring of the controlled system. In the

⁸⁵<http://www.irit.fr/recherches/MELODI/ontologies/Autonomic>

apartment, sensors produce data reflecting their observations. This data is enriched to become a reusable piece of knowledge. Enrichment of sensor data is performed using the SSN ontology, which is in the Sensing module of IoT-O. Each *ssn:Sensor* has an *ssn:Observation* stream composed of *ssn:SensorOutput* whose value is described by *ssn:ObservationValue*. For provenance purposes, a *ssn:SensorOutput* can be linked to its original representation (before enrichment) with the *iot-o:hasRawRepresentation* data property. The sensor’s characteristics (*ssn:MeasurementProperty*, the *ssn:Property* of the *ssn:FeatureOfInterest* it observes) are used to enrich the observation as well. IoT-O and SSN are generic ontologies, so they might need to be extended with application-specific modules to be fully functional. Such extension is proposed in the ADREAM-Robot module⁸⁶. The *ssn:Observation* allows the representation of a characteristic of the environment at a given point in time. The temporality of the sensor measures (and of actuators actions) is represented by a *san:hasDateTime* relations with a *time:Instant*, itself characterized by an *iot-o:hasTimestamp* data property. All the observations related to the same point in time are connected to the same individual, allowing the agent to have a timed representation of its environment and of its evolution.

In our use case, presence and temperature sensors produce raw observations in the form of XML documents standardized according to the oneM2M Content Instance resource type. The enrichment process requires an approach specific to the data, either by writing a dedicated enrichment script, or by using semantic mappings embedded in the data as in [Le-Phuoc 2011], where raw data is stored in relational databases and the database schema is mapped to an ontology for enrichment. SemioTics uses a dedicated enrichment script that could in the future be extended by producing annotated data.

The presence observation indicates the position of the person in the apartment, and the temperature observation represent the temperature of the air at a given point in space and time, both in the form of *ssn:ObservationValue* instances. This enriched information is accessed by the robot through semIoTics’ REST interface, and it is used to update the robot’s representation of the world. This view of the world is stored in the robots KB, and used as a context in the Analysis step.

Analysis, aggregation of observations in abstract symptoms: In the Analysis step, semIoTics processes the representation of the world built in the monitoring step to determine high-level symptoms that need to be addressed by actions.

For the medication scenario, the robot compares the present time to the time when the medication is due to generate the symptom “Medication must be delivered” if necessary.

For the temperature control scenario, user preferences are represented using the concepts defined in the Autonomic module. *ssn:Property* of the environment controlled by the robot within explicit boundaries expressed in the form of *auto:-PropertyConstraints* are classified as *auto:ConstrainedProperty*. In our use case,

⁸⁶<https://www.irit.fr/recherches/MELODI/ontologies/Adream-Robot>

the *adra:LivingRoomTemperature* has two constraints, instances of *auto:MaximumValue* (25°C) and *auto:MinimumValue* (19°C). The last *ssn:ObservationValue* of the *auto:ConstrainedProperty* is out of the bounds defined by the *auto:PropertyConstraint* (26°C instead of 25), so the temperature is classified by the reasoner as an *auto:OutOfBoundsProperty* thanks to custom rules.

Planning, where symptoms are used to create a plan: In the planing phase, the autonomic agent uses the inferred symptoms and policies defined by the user or by the administrator beforehand to define a series of actions that have to be implemented on the system.

In the medication scenario, *semIoTics* uses its representation of its environment to locate the person, as it is kept updated in the monitoring phase thanks to the knowledge produced by the sensors. The robot will plan a trajectory to fetch the medication and to reach the person. In this case, the representation of the trajectory itself is ad-hoc to the robot, and isn't linked to IoT-O or *semIoTics*: only the high-level indoor location is represented in the KB. The ontology is used to connect the robots internal representation of the world with the observations collected by the sensors and enriched by *semIoTics*, providing semantic interoperability between the robot and *semIoTics*. If the robot expresses its needs using the same ontology as *semIoTics*, or if their ontologies are aligned, it can seamlessly use elements measured by the sensors to plan its trajectory.

In the temperature control scenario, the description of the actions is performed using SAN, the actuator ontology that also describes the actuators in the system. The agent, with successive queries to the KB, will look for *san:Actuator* instances that *san:actsOn* the *auto:OutOfBoundsProperty*, and which *auto:ImpactOnProperty* is coherent with the symptom. In the example, since the temperature is too high, the *adra:PHG_FAN_11* (a connected fan) can be used, but also the *adra:SpaceHeater-01*, since its *adra:turnOff* operation has a *auto:NegativeImpact* on the temperature. The orchestration of these actions (if need be) are determined using the Lifecycle module of IoT-O, which represents the devices as state machines. *ssn:Device* (superclass of both *ssn:SensingDevice* and *san:ActuatingDevice* thanks to IoT-O) are objects that *ows:hasState exactly 1 ows:State*, because objects should only be in one state at a time. The *ows:State* is equivalent to the *lifecycle:State* (from the Lifecycle⁸⁷ vocabulary, extended by the IoT-Lifecycle⁸⁸ ontology), and *lifecycle:State* instances are connected by *lifecycle:Transition* instances. Thanks to this modeling based on state machines, stateful transitions, *i.e.* that are only available in certain states of the device, can be represented. Only *msm:Operation* instances that *iotl:isGroundedBy* a *san:Actuation* that *iotl:triggersTransition* a *lifecycle:Transition* that is a *lifecycle:possibleTransition* of the device current *lifecycle:State* can be called at a given time. For instance, the fan *adra:turn_off_operation* operation will only be available if the space heater is on. In our example it is off, so the agent plans to

⁸⁷<http://vocab.org/lifecycle/schema>

⁸⁸<http://www.irit.fr/recherches/MELODI/ontologies/IoT-Lifecycle>

turn on the fan and creates the corresponding *san:ActuationValue*. The selection of devices and their operations is driven by necessity (only the devices impacting the right property are selected), but it can also be driven by policies based on knowledge about the device intrinsic characteristics expressed with *san:ActuatingCapability*, composed of *san:ActuatingProperty* described in the actuator profile. It can for instance be used to minimize energy consumption (combined with the Energy module), or to optimize reaction time.

Execution, where the plan is converted into actions: In the execution step, the robot implements the planned actions.

For the medication scenario, the robot fetches the medication and brings it directly to the person, it does not have to search for her in the apartment. The MAPE-K loop can be repeated while the robot is moving to update the trajectory if the person moves in the house.

For the temperature control scenario, the robot transmits the *san:ActuationValue* that it wants the system to implement to *semIoTics* via a REST interface. *semIoTics* will handle the transformation of the knowledge into a representation that can be processed by the target device. This translation can be driven by the semantic description of *msm:Operations*, or dedicated annotations as in [Kopecký 2007], where XML schema are annotated for transformation from RDF to XML. *semIoTics* uses the semantic description of operations to perform lowering, and perspectives for this technique are presented in Section §3.4. This translation enables the interaction with low-level, constrained devices that are not able to process complex knowledge representations. At this point of the MAPE-K loop, actions have been implemented in the environment, and their effect is measured by the sensors, bringing the execution back to the monitoring step.

With these four steps, the autonomic agent performs a complete MAPE-K loop, from the collection of sensor observation to the reaction with actuator commands. The representation of both the physical environment and the device deployment enables a semantically-enabled agent (here the robot) to extend its capabilities of action and perception. *semIoTics*, thanks to OM2M and to IoT-O, addresses both technical and semantic interoperability in an IoT network. The use case focuses on home automation, but IoT-O and our approach are generic enough to be adapted to other domains. However, as it will be discussed in the next section, generating semantically rich actions representations is not enough to solve semantic interoperability issues in the IoT.

3.4 Towards semantic interoperability for constrained devices

Use cases introduced in Section §3.3 promote the semantic interoperability brought by the SWoT, as well as the potential for advanced applications it enables. In or-

der to be used in the Analytics step of the use case's MAPE-K loop, observations produced by sensor are transformed from their raw representation into a richer, semantically-enabled one. The same loop leads in the Planning step to the production of rich representation of actions. However, knowledge expressed in the W3C formalisms (RDF, OWL...) are heavier to process and exchange than models based on simpler technologies such as XML or JSON, or than ad-hoc data formats composed of small bytes sequences, commonly used in embedded softwares. Manipulating complex knowledge representations is antithetic with the deployment of very constrained devices that have a low processing power and a restricted bandwidth. These nodes cannot produce or consume enriched data, so making the system semantically enabled requires performing transformations:

- from raw to enriched data for sensor observations, and
- from enriched to raw data for actuator commands.

In order to face this issue, and to enable the Execution phase of the MAPE-K loop presented in Section §3.3.4, we propose a lowering method to capture rich knowledge into simpler data formats so that they can be processed by constrained devices, that has been published in [Seydoux 2016c]. This method is based on **reusing existing mappings** used for the generation of enriched data, instead of requiring dedicated mappings from enriched to raw data, thereby reducing the cost of the transformation.

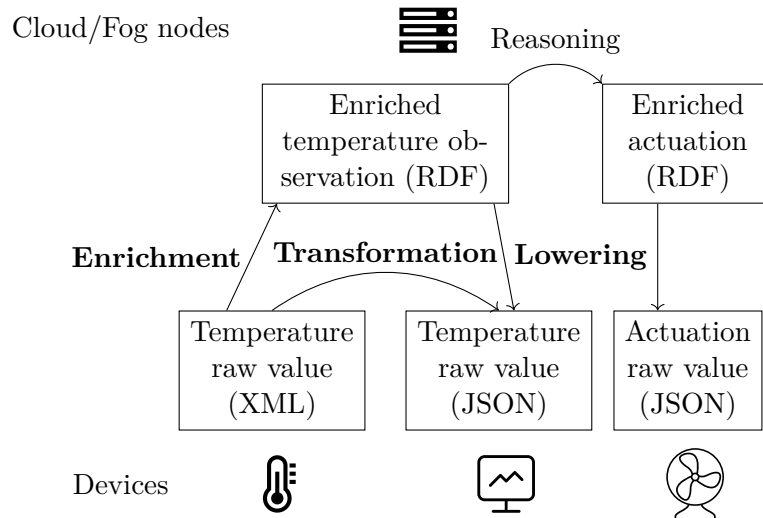
3.4.1 Enrichment and Lowering: IoT content management functionalities

Recurring patterns are observed in IoT content management. Such patterns, that we call functions, are the topic of interest of Chapter §4, the next chapter of this dissertation. However, two functions illustrating the transition from the first part of the dissertation, focused on interoperability, toward the second part, focusing on adapting the SWoT to IoT constraints, will be introduced in the present section. Namely, these two functions are 1. enrichment, the transformation from raw data to semantically enabled information, and 2. lowering, the inverse transformation.

Fig. 3.11 gives an illustration of data management issues in the IoT by a simple use case: data measured by a temperature sensor is displayed by an application using a different data format, and it is also used to control a fan.

Several approaches to **enrichment** exist, and will be detailed in the survey presented in Chapter §4. By definition, IoT nodes produce and consume structured data (raw data value on Fig. 3.11). The structure of such data can be represented by a schema, *i.e.* a syntactic data model. Therefore, a possible enrichment technique is to annotate the schema with semantic elements in order to generate rich representations from data compliant with the schema. Such schema annotations are referred to as mappings. [Sheth 2008] and [Ferdinand 2004] exploit the **implicit semantics** encoded in the **syntactic data model** by aligning it to ontologies with

Figure 3.11: Illustration of data enrichment and lowering



such mappings. To reduce the payload when transferring data, the schema can be accessed and manipulated only when necessary. Schemas are usually annotated by hand at design time: used ontologies are chosen in order to be able to describe the schema elements closely, leading to simple equivalence alignments.

Some existing approaches for data **lowering** (from a semantic representation to a syntactic one) also rely on dedicated mappings, as in [Kopecký 2007] or [Bischof 2012]. Transformation from RDF to other data formats is proposed in [Corby 2015] based on SPARQL queries. To bring interoperability to lower nodes, [Köpke 2010] proposes an ontology-driven **data transformation** approach, *i.e.* the presentation of a piece of data in a different target format (also illustrated on Fig. 3.11). We propose to use schema mappings to ontologies initially meant for data enrichment for knowledge lowering.

Our approach is therefore dependent on the existence of mappings from a schema to ontological elements. It is a reasonable assumption, since many standards, especially for the IoT, use normalized schema as syntactic interoperability providers:

- oneM2M⁸⁹ and HGI⁹⁰ with the SDT (Smart Device Template)
- OMA⁹¹ and the IPSO alliance⁹² with LightWeightM2M
- OCF⁹³ with OIC⁹⁴ and AllJoyn⁹⁵

⁸⁹<http://onem2m.org/>

⁹⁰<http://www.homegatewayinitiative.org>

⁹¹<http://openmobilealliance.org/>

⁹²<http://www.ipso-alliance.org/>

⁹³<http://openconnectivity.org>

⁹⁴<http://openconnectivity.org>

⁹⁵<https://openconnectivity.org/developer/reference-implementation/alljoyn>

Therefore, using schema mappings for enrichment or lowering is suitable for an IoT system, as it limits bandwidth consumption when exchanging data, integrates constrained devices, and brings semantic interoperability. However, manually aligning a schema to an ontology is a **time consuming process**, and it has to be done for both enrichment and lowering. Our hypothesis is that if a schema only aligned for upstream enrichment meets a set of requirements, the mappings can be used for knowledge lowering as well, cutting by half the work required to make constrained devices semantically enabled.

3.4.2 A pivot-tree based approach to mapping reversal-based knowledge lowering

Our approach to schema alignment reversing is divided into two steps: transformation of an RDF graph into an abstract pivot tree, and serialization of this pivot tree into a concrete syntax. The choice of an abstract step in the transformation process helps to decorrelate the approach from the source language. The pivot tree is a **language-agnostic** representation of the final tree (either XML, JSON or an IRI for REST interfaces) with no concrete syntax. It associates the knowledge expressed in the RDF graph (with its explicit semantics) to the tree-like structure of the targeted schema, so that its semantics is no longer implicit. Algorithm 1 shows the creation process of the abstract tree from the annotated schema and the target ontology.

Algorithm 1 Abstract tree creation

```

1: procedure BUILDABSTRACTTREE(Schema $s, Graph $g, Resource $individual)
   match is a function that finds the schema element corresponding to the
   root individual
2:   $schemaRoot ← match($s, $individual)
3:   build constructs an abstract node from an RDF node and a matching schema
   node
4:   $abstractRoot ← build($schemaRoot, $individual)
5:   for $property in $graph.triples($individual, $property, $object) do
6:     if $schemaRoot.hasAnnotatedDescendant($property) then
7:       if $property is ObjectProperty then
8:         $abstractNode ← buildAbstractTree($schemaRoot, $g, $object)
9:         $abstractRoot.buildObjProp($s, $g, $property, $abstractNode)
10:      end if
11:     if $property is DataProperty then
12:       $abstractRoot.buildDataProp($s, $g, $property, $object)
13:     end if
14:   end if
15: end for
16: end procedure

```

To be suitable for our approach, the annotated schema must be compliant with

these requirements:

- Design-time hypotheses
 - The data format must have a tree-like structure (like XML, JSON, RAML...)
 - The schema describing the data structure must be explicit
 - The schema must be aligned to semantic resources to allow data enrichment
 - All potential root elements should be mapped to a concept in the ontology
 - All meaningful inclusions must be annotated with relationships, as opposed to purely syntactic inclusions that do not need annotation. Syntactic inclusions do not carry data semantics, but rather structural information
 - Only inclusions leading to leaf elements should be mapped to RDF data properties
 - No ambiguity: when transforming the RDF graph into the abstract tree, every relation from a node in the graph should be associated to exactly one relation from the corresponding node in the schema, and each individual should be mappable to exactly one schema root node.
- Run time hypotheses
 - The required properties in the schema must be instantiated in the graph

3.4.2.1 Preliminary results

To evaluate our solution, we measured the compliance of existing schema with our requirements, and conducted a qualitative comparison with existing works presented in Tab. 3.3. Different sources of data schemes were used to assess the hypothesis for a reversible mapping: schemes provided as part of standards specifications (AllJoyn), and a sample of schemes from an online API repository⁹⁶. Overall, these schema met our requirements, or needed minimum backward-compatible transformation (such as the semantic annotation for enrichment).

Despite their runtime efficiency, transformation-only methods are not suitable because raw data is not enriched and cannot be reasoned upon. Our approach covers less schema than the two-way mapping approach because of the hypothesis that must be validated by the schema, but it is suitable for the schema at stake in the IoT as they are standardized, and requires less mapping work than any other approach.

⁹⁶<http://www.programmableweb.com>

Table 3.3: Qualitative comparison of approaches

	Supported scenario	Code base	Execution time	Generation required	Supports inference	Flexible	Schema type applicability	Mapping required
Direct schema transformation	Transformation	Small	Fast	No	No	No	Any to any	None
Schema transformation generation [Köpke 2010]	Transformation	Large	Fast	Yes	No	Partially	Annotated schema	Two-way
Two-way mapping [Bischof 2012]	Transformation and lowering	Large	Slow	No	Yes	Yes	Annotated schema	Two-way
Mapping reversal (our approach)	Transformation and lowering	Large	Slow	No	Yes	Yes	Restricted annotated schema	One way

3.5 Conclusion

This chapter was dedicated to the semantic interoperability fostered by the emergence of the SWoT.

After providing a definition for interoperability in order to identify scope of our work, we examined the standards that consider semantic interoperability. In particular, I contributed to the white paper [Murdock 2016] published conjointly by members of the W3C and the oneM2M consortium, in order to call for a deeper integration of SW technologies in both standards. In this joint white paper, interoperability issues are described as a major impediment for the development of the IoT, and SW principles and technologies are seen as interoperability providers. My participation to the oneM2M standard in the MAS group also led to my contribution to Eclipse OM2M, a free implementation of the standard. My contribution consisted in developing the semantic functionalities of the standard to OM2M, enabling SPARQL-based discovery of resources.

We identified the crucial roles of ontologies in achieving semantic interoperability. Ontologies for the IoT domain, and the failure to follow good practices such as concept reuse from one ontology to another limits the semantic interoperability benefits for the SWoT. Therefore, we proposed a set of requirements to assess the quality of IoT ontologies, and define guidelines for IoT ontology design. Two types of requirements have been distinguished: conceptual, *i.e.* what the ontology should talk about, and functional, *i.e.* how the ontology should talk about it. Major ontologies for the IoT have been assessed against these requirements, and since no ontology fulfilled all of them, we proposed our own modular core-domain IoT ontology, IoT-O, introduced in [Seydoux 2016b]. In order not to redefine concepts and contribute to the multiplication of IoT ontologies, IoT-O integrates existing ontologies in its different modules whenever it is possible. IoT-O has also been aligned with other core-domain ontologies, and in particular SOSA.

IoT-O has then been used to provide semantic interoperability in three use cases:

- IoT-O has been integrated to the OPA platform in order to promote the reusability of enriched data. Data collected by a smart building is enriched and made available in an open data, providing daily access to **over 283000**

enriched data points.

- The OPA open data has been integrated as a testbed of the FIESTA federation. This use case supports the semantic interoperability enabled by the principles and technologies of the SW. After their daily enrichment, observations made available in the open data are transformed thanks to alignments with the FIESTA-IoT vocabulary, and are pushed on the federated platform.
- Finally, a smart home automation use case has been discussed to show the potential for the integration of smart agents in SWoT-enabled environments. *semIoTics*, a system we implemented on top of OM2M, instantiates a MAPE-K loop to support knowledge-driven autonomic scenarios. In this use case, the autonomic agent manipulates virtual representations of the devices, which are used to impact their physical counterparts. This use case requires both syntactic interoperability, provided by the use of a standard platform, and semantic interoperability, ensured by *semIoTics* and the ontologies it uses. In order to facilitate the expression of their preferences to users, the constraints they impose on properties are independent from the underlying devices. These preferences are dynamically resolved by discovering available devices, in order to be adaptive to the dynamism of the IoT network. This use case has been used in [Seydoux 2016a] and [Aïssaoui 2016].

For this last use case, the communication between the applications and the devices is bidirectional: sensor observations are enriched with ontological elements to be consumed by applications, and the high-level representation of commands produced by applications are translated to be adapted to the target legacy device. This transformation back and forth between rich and raw representations is often based on manual schema annotations. The last section of the present chapter has been dedicated to early work published in [Seydoux 2016c], aiming at automating part of this transformation by reversing enrichment mapping under certain conditions.

The contribution introduced in this last section has shed light on an issue with the emergence of the SWoT: SW technologies require more resources than what is available in the end devices of IoT networks. In order to enable semantic interoperability by the integration of SW technologies into the IoT, the specific constraints of the IoT domain must be considered. That is why the next chapter is dedicated to surveying how the SW technological stack is deployed in IoT networks.

Supporting the SWoT with semantic Cloud and Fog computing: a survey

Contents

4.1	Survey context and key terms	69
4.1.1	Survey methodology	69
4.1.2	Identifying SWoT functions	70
4.1.3	Related work	71
4.2	SWoT functions and semantic Fog computing	73
4.2.1	Content value creation functions	73
4.2.2	Node interworking functions	79
4.2.3	Node-content dependency functions	87
4.3	Identifying trends in SWoT processes	90
4.3.1	Analysis of the surveyed contributions	90
4.3.2	Envisioning future functions	93
4.3.3	Adapting Semantic Web technologies to constraints of the IoT domain	96
4.3.4	Making semantic Fog computing transparent	98
4.4	Conclusion	99

Achieving semantic interoperability is a primary issue in the IoT, that leads to the emergence of the SWoT. However, the integration of SW technologies into IoT systems is quite challenging due to some characteristics and constraints of the IoT domain. While IoT architectures are by design based on **resource-constrained devices**, the SW technologies are **resource-consuming**. In order to tackle this core divergence, it is obvious that SW technologies cannot be deployed directly on IoT devices in the majority of cases. Therefore, **the SW stack has to be instantiated along the reference SWoT three-tier architectural pattern** characterized in Section §2.5.4.

The Cloud computing paradigm is well-established, and Cloud architectures support a majority of Web services used at a world-wide scale. However, as it has been described in Section §2.5.3, some characteristics that are desirable for

SWoT deployments are not compatible with the nature of Cloud architectures. In order to complement said Cloud architectures by instantiating these properties, the Fog computing paradigm has been proposed. Fog computing focuses on localizing processing power closer to IoT devices in distributed architecture, and its impact on the SWoT has been studied for instance in [Patel 2017].

This chapter aims at studying **the role of Fog computing in supporting the deployment of the SWoT** by surveying the literature. **The term “Fog computing” has been coined rather recently** [Bonomi 2012] compared to “Cloud computing” (1994¹), “IoT” (1999 according to [Ashton 2009b]) and “Semantic Web” ([Berners-Lee 2001]). However, with the vision of Fog nodes as the equipment connecting IoT devices to Cloud servers provided in Section §2.5.3, we characterized Fog nodes as an intrinsic component of the SWoT architectural pattern discussed in Section §2.5.4. In this Cloud-Fog-Device pattern, the role of Fog nodes is double:

- On the one hand, Fog nodes provide technical interoperability between Cloud nodes and devices. In this case, Fog nodes serve as gateways connecting WoT and IoT tiers, and do not implement any element of the SW stack, which is deployed on Cloud nodes. We refer to this deployment type as **semantic-agnostic Fog nodes**, and the processing based on SW technologies is qualified as **semantic Cloud computing**.
- On the other hand, the processing power provided by Fog computing may be used to support some elements of the SW stack. Fog nodes still provide communication capabilities between the devices and Cloud nodes, but they are also full-fledged components of the SWoT deployment. We refer to this kind of approach as based on **semantic-enabled Fog nodes**, enabling **semantic Fog computing**.

This distinction entails a question: how is the use of either semantic-enabled or semantic-agnostic Fog nodes impacting the SWoT domain ?

The purpose of this survey chapter is to provide a reading grid to: 1. classify the research contributions transforming the IoT into the SWoT domain, and 2. analyze the integration of semantic Fog computing in this evolution by comparing it with semantic Cloud computing.

The ambivalent role of Fog nodes in SWoT architectures is used to define precisely the research question addressed by the ensuing survey with a description of the systematic literature review methodology followed. Recurrent patterns in existing research are identified, and used in conjunction with the reference architecture to describe the role of Fog computing in the integration of the SW principles into IoT networks in Section §4.2. In Section §4.3, deductions are extracted from the survey, then we discuss future directions for the SWoT. Challenges the SW faces to be compliant with constraints of the IoT domain are pointed out, as well as recent contributions from the SW research community proposed to adapt to these

¹<https://www.wired.com/1994/04/general-magic/>

constraints. The complementarity of the Cloud and Fog paradigms in supporting the SWoT architecture is brought into focus.

4.1 Survey context and key terms

In order to discuss the relationship between Fog computing and the implementation of the SWoT technological stack, this section describes the context of the conducted survey. The literature review methodology followed is described in Section §4.1.1, and the elements of the reading grid used to classify contributions are presented in Section §4.1.2. Related surveys are presented in Section §4.1.3.

4.1.1 Survey methodology

This survey has been conducted according to the systematic literature search technique presented in [Kitchenham 2004]. Since we established that Fog nodes are part of the SWoT architectural pattern in Section §2.5.4, the question is whether:

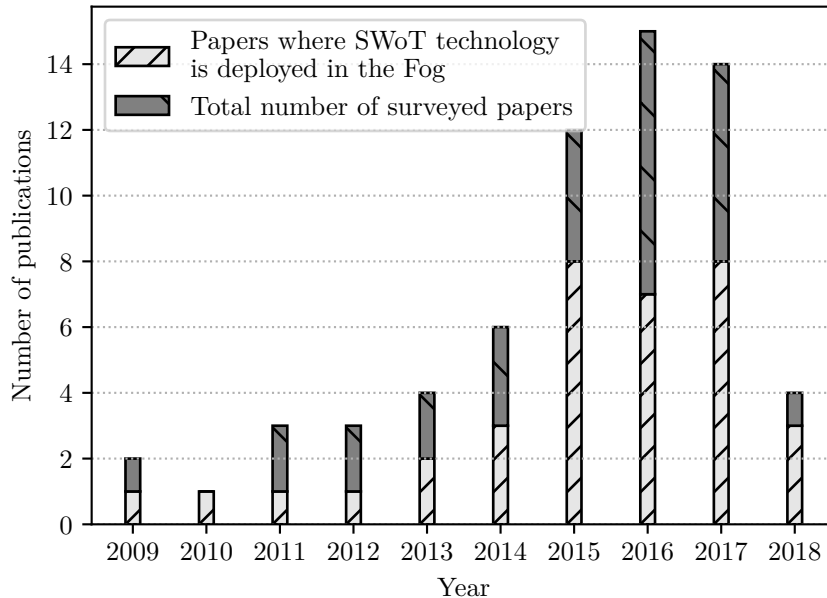
- SWoT technologies are only deployed in Cloud nodes, with semantic-agnostic Fog nodes being solely technical and syntactic interoperability providers, or
- if semantic-aware Fog nodes play an active role in the implementation of the SWoT technical stack to provide semantic interoperability.

We want to determine how semantic Fog computing supports SWoT architectures in the case where Fog nodes are semantically enabled, and the motivations for preferring semantic Cloud computing otherwise.

To answer this question, papers were collected from major digital libraries (namely IEEEExplore, ScienceDirect and ACM Digital Library) with the disjunction of keywords “SWoT”, “Semantic Web”, and “Internet of Things”. Both journal and conference papers were considered. Moreover, in order to focus on papers relevant to the research question, only papers in which the deployment architecture is clearly described were considered. Such a description can be done in a figure, by providing hardware capabilities of the deployment architecture, or by describing the capabilities of the deployment nodes. Contributions based on a purely functional architectures were excluded from the survey if the components were not situated on identified containers (machines classifiable as Cloud or Fog nodes). Since not all studies relevant to our survey explicitly referred to the equipment between IoT devices and remote Cloud servers as “Fog” nodes, but also as “gateways” for instance, the keywords “Fog” and “Cloud” were not used as filters. The relevance of architectures has been considered based on the characteristics described in the surveyed studies. No minimal date was considered, and the latest papers included were prior to July, 2018. Finally, 64 publications were considered for the survey, sorted by year of publication in Fig. 4.1.

Fig. 4.1 shows an increase of the number of papers relevant to our survey with time, with a proportional increase of the papers in which Fog nodes play an

Figure 4.1: Surveyed papers, per year



active role in SWoT functions. Overall, semantic capabilities are distributed on Fog nodes in 58 % of the studied papers. It should be noted that this rate is biased by the inclusion criteria for papers selected in this study: by default, papers where Fog nodes do not play an active role for semantic processing, and where the SW technological stack is only deployed in Cloud nodes do not provide details about their deployment architecture. The choice to exclude such papers from the survey was made because their analysis would have been entirely based on an assumption that is directly correlated with the research question we want to discuss. Therefore, the rate of papers both contributing to the SWoT and considering an active Fog node over all papers dedicated to the SWoT in general is likely to be much lower than it is in this survey. The purpose is rather to explore the role of Fog nodes in SWoT architectures rather than give a precise estimation of how many papers actively use Fog computing to support SWoT deployments, which is why the introduced bias is not an issue for our survey.

4.1.2 Identifying SWoT functions

When analyzing how the SW technologies are integrated in the IoT in surveyed SWoT publications, **different recurring functions have been identified, agnostic to the underlying technology stack and to the application domain.** The concept of function in this work is understood as an elementary processing unit that can be combined with others in order to achieve a feature of higher order.

Functions have been extracted from existing practices in a bottom-up approach.

Identifying these functions depicted in the surveyed publications enabled the organization of contributions with respect to one another, easing their comparison. This organization supports the capture of a structured landscape of the contributions of the SW to the IoT. Our objective is (i) to characterize these SWoT-enabling functions, (ii) to describe how they are instantiated in research contributions, and (iii) to analyze the impact of Fog computing on their implementations. SWoT functions capture both the usage of SW technologies for managing nodes themselves or their context, and the applicative content they are interested in. We identified three categories of SWoT functions:

- **Content value creation functions** are related to domain-specific activities, and focus on the **transformation** and **processing** of content relevant to an application. Content is used as a neutral term to refer to any element in the classification provided by [Rowley 2007], organized in the Data, Information, Knowledge and Wisdom (DIKW) hierarchy. Tab. 4.1 lists papers contributing to these functions and Section §4.2.1 describes said functions in detail.
- **Node interworking functions** are not specific to an application domain but common to the IoT domain. The development of such functions goes against vertical fracturation by focusing on the description of node characteristics, capabilities, and preferences. Node interworking functions allow nodes to be **aware** of their neighbors on the graph, and to offer a **homogeneous** self-representation. Tab. 4.2 references the papers contributing to these functions, and they are described in Section §4.2.2.
- **Node-Content dependency functions**, where the focus is not on content transformation but rather on facilitating access to content by leveraging node characteristics. Such functions, listed in Tab. 4.3, are described in Section §4.2.3.

4.1.3 Related work

Previous work has been done to survey the convergence between the IoT and the SW:

- Early work in the SWoT focused on semantic sensor networks. For instance, [Compton 2009a] surveys sensor ontologies and observation representations. The scope of this work is especially on sensor ontologies, and even if it proposes an overview of technologies enabling semantic sensor networks, it does not present specific applications relying on these ontologies. Similarly, [Szilagyi 2016] gives an overview of the SW stack applied to the IoT, and surveys IoT ontologies. It goes beyond semantic sensor networks, but is still limited to model analysis. We propose to focus on how the ontologies and the technologies of the SW are used to develop the SWoT, rather than on identifying exhaustively the models used.

- [Atzori 2010] is a survey of the IoT domain, proposing a definition for the notion of IoT and listing application domains and enabling technologies for the IoT. The IoT paradigm is described as the convergence of Internet technologies, electronic devices, and SW technologies. However, the paper itself does not cover how SW technologies are integrated into the WoT, while we intend to analyze in detail and compare different contributions to the SWoT.
- [Barnaghi 2012] studies the roles SW technologies can play in the IoT, as well as the challenges they represent. This paper identifies some functions similar to what is presented in the present thesis in Section §4.2. However, the deployments of these functions in a reference architecture is not studied, and the impact of technological constraints of the IoT on the SWoT are not described in depth, as they are not in the intended scope.
- [Jara 2014] gives an overview of the evolution from the IoT to the WoT and toward the SWoT. It is focused on the role of standards in interoperability, and the integration of SW technologies in standards. It also provides an overview of technologies involved in the IoT. This paper is oriented toward projects and industrial consortia, which is complementary to our study. We focus on the contributions of the SWoT to IoT issues, and only integrate standardization concerns when they are related to this domain.
- [Perera 2017] is a survey dedicated to Fog computing applied to the development of the smart city. Knowledge management capabilities of Fog nodes are analyzed, and some SW features such as semantic annotations are introduced, but the focus is primarily on the enablement of the smart city by Fog computing, rather than on the relationship between Fog nodes and SWoT architectures.
- [Sezer 2018] gives a detailed overview of context-awareness in the IoT, and on the methodologies used to achieve it. The authors focus more on techniques than on deployments: the role of Fog nodes is not considered. The survey gives a detailed analysis of the relationship between the IoT and the SW domains, but also with Big Data and machine learning, which is out of the scope of the present survey.

Overall, these surveys focus on the SWoT, and on the implication of the integration of SW principles and technologies in the IoT. To the best of our knowledge, the distinction between semantically-enabled and semantically-agnostic Fog nodes, and the respective roles of semantic Fog computing compared to semantic Cloud computing in supporting SWoT deployments has not been the topic of any survey.

4.2 Describing the role of semantic Fog computing in SWoT functions

In this section, the contributions of surveyed papers are described for each identified SWoT function, pointing out the value added by the integration of SW principles and technologies into the IoT. Particular attention is dedicated to the role of semantic Fog computing in the function instantiations. Each category of functions is separated into subcategories of general recurring patterns among functions, in order to cluster them into coherent sets.

4.2.1 Content value creation functions

Content value creation functions classification is summarized in Tab. 4.1. Two categories of content-related functions are distinguished, shown in the table headers: Data/Information (DI) transformation and Information/Knowledge (IK) transformation, each described in the remainder of this section.

4.2.1.1 DI transformation

These functions are dedicated to the transformation of content according to the DIKW hierarchy between the Data and the Information level. **The core meaning of the content is not changed** by these functions, but the expressivity of its representation varies.

Enrichment: The enrichment function is the transformation of content representation upward in the DIKW hierarchy. The description of data with meta-data to transform it into information is an enrichment: content is described with ontological entities to unambiguously define its meaning, capture its context and increase its reusability and its overall value to an application.

Based on semantic Cloud computing: In settings where Fog nodes are semantic-agnostic, content is forwarded to Cloud nodes by gateways that are in this case not semantically enabled. It is the approach chosen by [Compton 2009a] or by [Aïssaoui 2016]. In the latter, content collected from sensors is stored in a standardized structure on the gateway, but it is explicitly described with an ontology only when stored on Cloud nodes. In an industrial setting, [Wang 2018] chooses to separate

- "operational" data represented with the OPC-UA data model² on Fog nodes, close to the devices,
- data that is semantically enriched and processed by a Cloud node.

Based on semantic Fog computing: Directly creating semantically enriched content requires the devices to produce complex, structured content, which cannot

²<https://opcfoundation.org/about/opc-technologies/opc-ua/>

Table 4.1: Content value creation functions in the literature

Paper	Enrichment	Abstraction	Aggregation	Presentation.
[Su 2015]	F			
[Barnaghi 2009]	C			
[Ploennigs 2017]	F	C	C	C
[Sarkar 2015]		F		F
[Gyrard 2016]	F	F		
[Wang 2018]	C	C	C	
[Desai 2015]	F			
[Su 2018]		F		
[Kelaïdonis 2016]	F			
[Pease 2017]	C	C	C	C
[Le-Phuoc 2016]	C		C	C
[Kaed 2018b]	F			
[Hasemann 2012]	F			
[Howell 2017]		C	C	C
[D'Elia 2017]		C		
[Perera 2016]		C	C	C
[Singh 2017]		C	C	
[Ullah 2017]	C	C	C	
[Kharlamov 2016]			C	
[He 2012]		C		
[Dey 2017]	F			
[Aïssaoui 2016]	C			
[Al-Osta 2017]	F		F	
[Maarala 2017]		F		
[Nagib 2016]	C			
[Datta 2015]	F	F		
[Compton 2009a]	C			
[Costea 2016]	F	F		
[Han 2017]	F			
[Khan 2015]	F	C		
[Mathew 2014]	F			C
[Fensel 2017]	C	F	C	C
[Pfisterer 2011]	C			
[Gyrard 2017]		F		C
[Ara 2014]	C			
[Poslad 2015]	C	C	C	
[Kaed 2016]	F			
[Le-Phuoc 2012]	C			C
[Seydoux 2018b]		F		

be generalized to all resource-constrained devices as it requires producing and exchanging heavy documents over communication links that can be very restricted (LPWAN networks or CoAP for instance). To overcome this issue, [Barnaghi 2009] proposes to **resort to semantic Fog computing to perform the enrichment**: the sensors produce raw data, in any format, which is transported to a Fog node by a dedicated network, and only then does this Fog node use its own knowledge on the producing node to enrich the data. [Desai 2015] also proposes an enrichment based on semantic Fog computing, by introducing content annotation on a gateway with Semantic Gateway as a Service. Raw observations received by the gateway are described using reference ontologies (such as SSN) based on its local knowledge. In this approach, the gateway provides both technical and semantic interoperability, by bridging multiple protocols to the Web and by annotating content with both reference vocabularies (such as SSN) and domain-specific ontologies. Annotation is an enrichment technique in which metadata related to an ontology is added to non-semantic content to describe the information it conveys. Similarly, in [Mathew 2014], sensor data is enriched at the edge on the network by a Tiny Web Server which uses an ad-hoc vocabulary to generate RDF representations of the state of a parking spot. In [Al-Osta 2017], enrichment is also performed by semantic-enabled Fog nodes, and in order to limit resource consumption, raw data filtering is implemented prior annotation, and only raw data that qualified as relevant is enriched. Such pre-processing is an enabler for deploying the enrichment functionality among Fog nodes. [Mathew 2014] motivates the distribution of enrichment on Fog nodes by the scalability of the approach. Technical details necessary to enrich content are encapsulated in the local context in which they are relevant, and integrating heterogeneous technologies has no impact on the central Cloud node from a complexity standpoint: content is provided unified. From a more general point of view, some configuration is necessary to connect constrained devices to a gateway in order to use the right protocol and to decode the received content. Enriching content can therefore be added at this step of the content lifecycle with techniques necessitating a limited amount of resources, such as mappings in [Ferdinand 2004]. That is why semantic-enabled Fog nodes are seen as active components for enrichment in 54 % of the surveyed papers implementing this function, with an increasing trend in recent years.

Enrichment is one of the earliest functions that used SW technologies to contribute to the IoT with contributions such as [Sheth 2008]. This is explained by the predominance in early IoT work of sensor networks, a subset of IoT networks where all the devices are sensors. It is one of the most represented functions, with 28 of the surveyed publications (*i.e.* matching our inclusion criteria), 15 of which are dedicated to enrichment based on semantic Fog computing. [Khan 2015] compares different annotation strategies, on devices, Fog or Cloud nodes. For a small deployment, backing annotation with semantic Cloud computing is more efficient, but the authors defend the idea that when heterogeneity increases, performing enrichment based on semantic Fog computing might be preferable. Enrichment being a prerequisite to any semantically-enabled processing,

it is not surprising to find it in many publications. Moreover, contextualizing data using knowledge is a common practice in the SW domain, even outside the IoT.

Other DI functions, namely Lowering and Transformation, have been identified.

- Lowering is the function inverse to Enrichment, introduced with the mapping reversal approach in Section §3.4. Lowering consists of the transformation from a rich representation to a lower-level one (e.g., from RDF to a particular XML schema), depending on a target node's ability.
- Transformation is a function where representations from a model are transformed into another of the same complexity. Transformation approaches may for instance translate a KB from one ontology to another based on alignments.

However, there are too few contributions to these functions for us to make general observations as part of the survey. Some perspectives are provided in Section §4.3.2 to qualify these functions, and project them on future evolutions of the SWoT.

4.2.1.2 Information/Knowledge transformation

Processing content based on background knowledge is an important part of the IoT value creation. Content is leveraged in an application, and SW principles and technologies can be used in order to provide innovative services taking advantage of the Information and Knowledge levels of the DIKW pyramid.

Abstraction: Abstraction is a function sharing similarities with Complex Event Processing (CEP) [Robins 2010]: low level symptoms are extracted from content and correlated together in order to be transformed into a more abstract diagnosis. It can be based on reasoning, rules, pattern-recognition, etc. [Henson 2012] defines an abstraction as the “representation of an environment derived from sensor observation data”. Each contribution will not be presented in detail in this chapter, but they all have common characteristics: they consume enriched content, and infer new content described with a domain-specific vocabulary based on background knowledge.

Based on semantic Cloud computing: A representative example of abstraction supported by semantic Cloud computing is provided in [Khan 2015], where the impact of the location of a domain ontology and a reasoner is studied. In the approach centered on Cloud nodes, content collected by sensors is pushed to a remote server where it is processed according to domain knowledge in order to generate higher-level information through a deduction mechanism. In [Ploennigs 2017], abstraction is deployed to support cognitive behaviors, based on semantically enriched data, rule-based reasoning, and learning algorithms for pattern recognition and prediction. Reasoning is used to saturate the knowledge base by making explicit relations that were implicitly captured in the information in the KB, which can generate large quantities of triples. That is why reasoning and learning, which

is in general a resource-intensive computation, are deployed on Cloud nodes in this paper. [D’Elia 2017] proposes an extension to a work initially proposed in [Kiljander 2014] by enabling the registration of "persistent update" queries. These queries allow to capture rules in the form of SPARQL UPDATE queries to infer new knowledge when triples are added to the knowledge base. Ultimately, this component is meant to be deployed on Fog nodes, as the development of a mobile version by the authors suggests. However, the characteristics of the experimental deployment do not match the criteria we defined for Fog nodes, as they involve machines with large memories and high computing capacities. In an Industrial IoT (IIoT) use case, [Pease 2017] proposes an asset-tracking system where an application ontology is used to represent, in near-real time, the location of assets in a smart factory. Observations are combined with background knowledge in order to explicitly represent the position of an asset, producing high-value information from lower level observations. Presentation of the inferred position to users is also part of the requirements driving the use case described in [Pease 2017].

Based on semantic Fog computing: [Gyrard 2016] and [Gyrard 2017] propose a Cloud architecture to store and share rules, based on the principles of Linked Data applied to Rules: Sensor Linked Open Rules (S-LOR). These rules are used by gateways of mobile Fog nodes to make inferences when collecting data. Instead of a rule-based reasoning engine, [Costea 2016] proposes an approach based on streaming queries. The approaches benefit from both the global accessibility of Cloud nodes when sharing rules, and from the scalability of a distributed approach when processing content via semantic Fog computing. [Sarkar 2015] proposes the modeling of policies by ontologies in order to manage the services offered by diverse nodes, dynamically created according to context. The policies capture directives for decision making. In [Maarala 2017] and [Su 2018], the authors compare a reasoning process leading to content abstraction based on semantic Cloud and Fog computing. In the latter, GPS observations are also processed based on rules in order to infer abstractions about smart transportation. A single observation can be abstracted (*e.g.*, a velocity under 25Km/h is described as a low speed), as well as a conjunction of observations, potentially across time (the succession of two low speed observations, with opposite directions, indicate a U-turn). In [Seydoux 2018b], [Seydoux 2018a] and [Seydoux 2018c], we propose a distributed approach for rule-based reasoning, considering rules compliant with the principles of the S-LOR. Rules are used as modular applicative components that are dynamically distributed among Fog nodes in order to support the production of deductions in an abstraction functionality. Details about these approaches are provided in Chapter §5.

The production of abstract content is necessarily **driven by an applicative-level need**. A noticeable pattern in the surveyed semantic Fog computing of the abstraction function is the use of targeted, application specific deduction mechanisms, such as rules and streaming queries, rather than more generic reasoners. These technological choices enable a more modular reasoning, and are adapted to a resource-constrained environment.

Aggregation: Aggregation is a function where multiple instances of content are used in order to produce a new content instance of the same level in the DIKW hierarchy, as opposed to abstraction where the inferred content is of a different nature from the pieces of content used for inference. For instance, computing the average (or the maximum) of several values is aggregation, whereas using the same values to infer the occurrence of a meteorological event is abstraction. Aggregation can be used to expose a certain view over content, projected over temporal or spatial dimensions, as well as based on a more complex mathematical approach.

Based on semantic Cloud computing: [Poslad 2015] proposes an Early Warning System (EWS) architecture enabled with semantic functionalities. Content collected by sensors is channeled towards Cloud nodes, before being enriched and used in order to be abstracted into high-level events. Once the events are identified, the system takes operational decisions (where to send rescue, what places to evacuate...), and these operational decisions are implemented by Fog nodes. Aggregation is implemented in the form of filtering in [Poslad 2015]. Value is created with data fusion techniques, and some filtering is implemented before storage and processing: since content and metadata are stored separately in this paper, metadata is used to decide whether the content should be stored or not. In [Le-Phuoc 2016], aggregation of content is performed using operators from SPARQL, like COUNT, MIN, etc, from a customized extension of SPARQL adapted to streaming queries to perform aggregation over time or space. Content is streamed by Fog nodes to Cloud nodes, where the aggregation is performed. In this paper, aggregation is a driving mechanism for sensor mash-up. A similar approach mixing static and streaming content is proposed in [Kharlamov 2016], with the proposal of extending DL-Lite logic with aggregation operators. These operators are mapped to STARQL, a query language, and enable complex aggregation operations to be expressed directly in the queries over a combination of static and streaming KB. One of the flagship use cases of IoT-enabled appliances, be it for Smart home or Smart building, is energy saving. It is the use case motivating [Fensel 2017], where consumption information is collected from a smart plug monitoring a fridge. Measures are gathered and enriched locally by the users, to be published on a platform where they are compared to others on a large scale, after being aggregated in order to collect several metrics, *e.g.*, daily minimum, maximum, and average consumption. Aggregation is joined in this paper to abstraction: past data and background knowledge are used to make projections. Users are provided information about their appliance, allowing them to monitor its quality or to make purchases based on consumption information provided by others. Aggregation is performed on a large scale in [Ullah 2017]: e-health data collected about a user are enriched and processed on Cloud nodes using Big Data techniques and external KB. This system is deployed in order to deploy the decision-making process of the healthcare provider.

Based on semantic Fog computing: [Singh 2017] proposes a military use-case, where soldiers carry sensors whose enriched content is aggregated before being disseminated among Fog nodes. It is used by Cloud nodes to trigger alarms in risk-prone situations, in which case orders are dispatched to human units. Aggregation

techniques are not described in detail.

With the role of sink nodes played by gateways, filtering and aggregation is a function implemented in papers dedicated to Fog computing in general, but that are not focused on the SWoT, and are therefore not part of our survey, such as [Sheng 2015]. In the surveyed papers discussing aggregation functions, it is often implemented on large quantities of content, with potentially computation-intensive techniques, which explains the dominance of implementations based on semantic Cloud computing. A semantic Fog computing approach to aggregation would require focusing on less content, and to aggregate it with respect to the local context of the Fog node, rather than the large context presented in contributions such as [Le-Phuoc 2016] or [Fensel 2017].

Presentation: Presentation is the display of content dedicated to human consumption. When produced, SWoT content is typically a numerical value with some metadata, and presentation proposes a human-oriented interpretation of it instead of its raw form, for instance on a map or in the form of graphs: it is a user-facing function.

Based on semantic Cloud computing: [Le-Phuoc 2012] uses geographical metadata to display the location of content sources on a map, and to give access to these sources via a map interface. In [Le-Phuoc 2016] (an extension of [Le-Phuoc 2012]), content is aggregated prior to the visualization phase, in order to display refined content, like heat maps or graphs, complementary to the map information. When [Fensel 2017] provides classic charts to summarize user information, [Ploennigs 2017] uses an Augmented Reality (AR) system to provide visual information about appliances in a smart building. Enriched content is queried and displayed in order to have a projection of the appliance state for an operator.

In the surveyed papers implementing presentation functionalities, presented content is processed in order to provide a larger context to the consumer. Creating graphs summarizing historical content or enabling comparison with the behavior of other users, or projecting content onto a map, are different forms of global contextualization. Since semantic Fog computing processes data in a local context, it is not suitable for such a function, which is why it is only based on semantic Cloud computing in the surveyed papers.

4.2.2 Node interworking functions

In node-related functions, the messages exchanged between nodes do not focus on the applicative-specific content these nodes collect or process, but on the nodes themselves. Node-interworking functions classification is summarized in Tab. 4.2. They are separated in two sub-categories: functions dedicated to awareness of each other among nodes, and functions dedicated to node heterogeneity management.

Table 4.2: Node interworking functions in the literature

Paper	Abstraction	Composition	Configuration	Discovery	Selection.
[Ploennigs 2017]	F				
[Sarkar 2015]	F	F	F		
[Mayer 2013]					
[Kelaidonis 2016]	F	F			
[Kaed 2018b]	F		F		
[Nikoli 2011]	C				
[Wang 2017]	F			F	F
[Foteinos 2013]	C	C			
[Perera 2014b]	C				C
[Seydoux 2018b]				F	
[Kharlamov 2016]	C				
[Han 2014]		C	C		C
[Li 2015]				C	
[Mrissa 2015]	C	C			
[Hussein 2016]	C				C
[Dey 2017]	F				F
[Aissaoui 2016]	C		C		
[He 2012]			C		
[Lee 2016]	F		F	F	
[Compton 2009a]	C				
[Christophe 2011]	C				F
[Han 2017]	F				
[Khan 2015]	F				
[Vlacheas 2013]	C	C			C
[Mathew 2014]	F				
[Fensel 2017]	C				
[Pfisterer 2011]	C				C
[Kiljander 2014]			F	F	
[Gyrard 2017]	F				
[Ara 2014]	C	C		C	C
[Ruta 2016]				F	
[Poslad 2015]	C				
[Kaed 2016]	F			F	
[Bovet 2014]	F			F	
[Nachabe 2016]	C				
[Ben-Alaya 2015]					
[Perera 2016]	C	C		C	C
[Kibria 2015]	C	C		F	

4.2.2.1 Functions providing homogeneity among nodes

As stated in the introduction, heterogeneity at both content and node level, and the interoperability issues it brings, is one of the leading reasons for the introduction of SW technologies into the IoT. The three functions dedicated to heterogeneity management are focused on the harnessing of this diversity.

Node abstraction: Abstraction is the representation of a node by a virtual entity, usually described with an ontology in the case of the SWoT. In a context of heterogeneous nodes, abstraction aims at focusing on the modeling of **generic node representations to describe their characteristics in a unified way**. It allows applications to deal with a set of homogeneous nodes, breaking the vertical silos between application domains [Nitti 2016].

Based on semantic Cloud computing: Early studies such as [Compton 2009a] focus on node representation (in this particular paper, sensor representation) with SW technologies. This work led to the creation of *ssn*³, an ontology largely adopted by the community and used in nearly one in three papers of the survey. Sensor descriptions are stored in a KB by Cloud nodes. In [Pfisterer 2011], the behavioral patterns of nodes are used in order to automatically attach a description to unclassified nodes. Similarity is computed between the output of the nodes already described in the Cloud-hosted KB and those of the newly introduced nodes, and the undescribed nodes are annotated using the description of the similar nodes. This approach suffers a cold start issue: when only a few sensors are annotated, the clustering is less efficient, and the system needs to grow in order to propose a wider variety of sensors and more representative clusters. [Vlacheas 2013] proposes the notion of Virtual Object (VO) to describe how nodes can be abstracted, and describes how this approach tackles heterogeneity issues. The authors then describe how abstracted nodes can be used in other functions, such as composition or selection. These contributions are described in the corresponding paragraphs below. In [Mrissa 2015], physical nodes are associated to avatars, virtual representations described in OWL. The authors identify requirements for a WoT platform, and show how their proposed avatar architecture meets these requirements, such as interoperability, reactivity, safety... Avatars are also given introspection capabilities, in order to support collaboration and service composition, described more specifically in the paragraphs associated to these functions. Avatars are managed by Cloud nodes.

Based on semantic Fog computing: [Khan 2015], [Ploennigs 2017], [Dey 2017] or [Kaed 2018b] all manipulate node abstractions on Fog nodes. In [Bovet 2014], node abstractions located on Fog nodes are manipulated through the CoAP protocol, in order to be supported by constrained nodes. The characteristics of a node are separated into static and dynamic properties. Static properties constitute the actual device description, such as the sensor type or manufacturer, while dynamic properties relate to the content it collects, *e.g.*, the most recent measure

³<http://purl.oclc.org/NET/ssnx/ssn>

value. A similar distinction is made in [Mathew 2014] between static (“preset” in this case) and dynamic characteristics, embedded on a Fog device. Nodes are listed in a catalogue based on an ontology in the semantic gateway framework proposed by [Lee 2016] in order to ease device management tasks from the gateway.

With content enrichment, node abstraction is the most represented functions, with 30 papers of this survey discussing it. The primary drive in the emergence of the SWoT is interoperability, and this claim is reflected by the predominance of content enrichment and node abstraction, which are directly dedicated to interoperability and solving heterogeneity issues for both content and nodes. An important aspect of node abstraction is that it is a preliminary step to other functions using SW principles to manipulate devices. Initially mainly based on semantic Cloud computing, its support by semantic Fog computing is increasing.

Composition: Composition is the function associating nodes between themselves in order to create new nodes, offering services previously unavailable on the network.

Based on semantic Cloud computing: The notion of VO presented in [Vlacheas 2013] and in [Foteinos 2013] is associated to the notion of Composite Virtual Object (CVO). The authors of these contributions describe a process to **build CVO on top of homogeneous VO**, themselves being abstractions for real-world objects. Based on a specification of applicative needs, CVO are dynamically created. In [Han 2014], node composition is performed using the semantic description of different services. A plan is then computed, and the services selected are called sequentially according to the plan. The process is dynamic, and the composite nodes described by the execution plan is not stored in the KB. In [Kibria 2015], the creation of CVO is guided through the use of templates. Once defined, the CVO offers a service, and its characteristics are used to retrieve the service at runtime.

Based on semantic Fog computing: Two surveyed papers consider semantic Fog computing to perform composition: [Sarkar 2015], and [Kelaidonis 2016]. None of them covered the technical details of the composition, but focus rather on its relevance to the proposed frameworks. The authors of [Sarkar 2015] give a high-level vision of semantically-enabled Fog nodes, capable of both abstracting and dynamically composing the devices they are connected to. These dynamic composite virtual objects are used to implement contextual services. In [Kelaidonis 2016], node composition is presented from a service point of view. It is based on the description of capabilities of virtualized devices, and enables the creation of complex virtual devices based on preexisting services.

Composition is a function dependent on node abstraction, because an abstracted node representation enables the creation of virtual composite nodes. That is why all 9 papers discussing the composition function also discuss abstraction: they first define how they abstract physical nodes, and then demonstrate how these abstractions can be manipulated separately from the nodes they initially represent. Only two of these papers perform such composition on semantically enabled Fog nodes, but we expect this function to be more widely supported by semantic Fog computing

in the future as explained in Section §4.3.2.4.

Configuration: Configuration is the function opposite to abstraction: generic representations are adapted to specific deployments, and physical nodes are configured in order to match their virtual representation. This function can be used either to change the configuration of a node, or to send a command to an actuator in order to update its state.

Based on semantic Cloud computing: [He 2012] proposes a Cloud-based home automation platform in which sensor observations are used with domain knowledge in order to infer whether or not a plant should be watered, and control a watering actuator accordingly. The authors chose an approach to content processing based on semantic Cloud computing in order to centralize all intelligence in Cloud nodes, therefore allowing the user to only deploy simple low-cost devices in his environment. [Han 2014] proposes a node composition function, and uses the description of the obtained composite node in order to build the service calls implementing the composite service. The configuration function, *i.e.* binding services to their abstract node representation and execution of said service, is performed on a Cloud-based platform. In [Aïssaoui 2016], a representation of the actions to be taken is inferred from the representation of the user requirements and from the observations of the environment. These actions are associated to service descriptions that are processed by a remote node to actually make the service calls.

Based on semantic Fog computing: The possibility to avoid the delay of a round trip from a gateway to Cloud nodes and back to the gateway in order to implement autonomic control of a device is a motivation to implement control on Fog nodes. Decisions are in this situation taken closer to devices, without the need to communicate with a remote server. [Kiljander 2014] proposes an architecture where smart agents consume semantic messages from a broker, and control actuators by changing their virtual representation. However, the paper does not provide further details on the implementation, but it is located in the Fog tier of the Cloud-Fog-Device pattern. The framework proposed by [Lee 2016] contains a rule system to support control of devices. Rules component are expressed according to the semantic description of nodes, enabling their control by the gateway. Rule-based reasoning distributed among semantic-enabled Fog nodes is also the approach proposed by [Kaed 2018b], the continuation of [Kaed 2016] presented in Section §4.2.3. Their contribution proposes a mix of SW technologies and scripting, where content is used to trigger Event-Condition-Action (ECA) rules. On match, the rule modifies the state of a node.

Configuration has become more popular with the evolution of sensor networks into device networks (containing actionable nodes), and the associated bidirectional communication with devices. It promotes the evolution of semantic Fog computing, where decisions based on sensor observations can be taken based on local context in order to trigger actions. Distributing, within the network, decision-making capabilities combining sensor observations and contextual knowledge as input, and

actuators control or device reconfiguration as output, enables the creation of distributed local autonomic systems. The contribution I.B of this thesis, previously published in [Seydoux 2016b], described in Chapter §3, discusses the role of SWoT technologies for autonomic system. The proposed approach is based on semantic Cloud computing, since semIoTics is deployed on a powerful computer, and the necessity for rapid reactions in the autonomic control advocates for a shift to semantic Fog computing.

4.2.2.2 Node awareness functions

In order to communicate, IoT nodes need to **be aware of the existence of each other**, and to have respective addresses to exchange messages. Furthermore, nodes may be only intermittently available, to save battery life for example, or due to failure, maintenance operations, etc. In this context, a dynamic awareness of a node's surroundings is required, and the following functions contribute to it.

Discovery: Discovery is the function in which descriptions of remote pairs are gathered by a node, potentially with respect to some criteria.

Based on semantic Cloud computing: Similarly to composition, discovery is dependent on node characteristics described in an abstraction process. Such description allows discovery by querying on certain characteristics of the node, as in [Ara 2014]. [Kiljander 2014] proposes a discovery for Fog nodes based on a semantic description issued to a central repository whose address is known *a priori* by the issuer of the discovery request. A resource discovery platform is proposed by [Perera 2016], in which the user is helped to explicit the nodes it requires, be it at the sensor or at the service level.

Based on semantic Fog computing: [Ruta 2016] proposes a discovery method based on google's Physical Web (PW)⁴ and extending it to a so-called Physical Semantic Web (PSW). Discovery of nearby nodes is enabled by the protocols and technologies of the PW, but the PSW extracts from the identification messages semantic annotations in order to decide whether the node is relevant or not to the client, running on a mobile device. [Lee 2016] proposes a query-based discovery similar to [Ara 2014], but the interrogated KB is hosted on Fog nodes. Authors of [Wang 2017] rely on the borderline notion of Edge Cloud, which is a Cloud node with some Fog node characteristics, such as proximity to devices. It is a notion also present in [Kelaidonis 2016], where no reference characteristics are provided either, but where the Edge Cloud has the same role of sink and processing node as a Fog node, and exposes its functionalities directly to applications in the same way as a Cloud node. Edge Cloud is a notion that is close to Cloudlets as defined in [Verbelen 2012]. The authors of [Wang 2017] adapt the SPARQL discovery queries by using a spatial index in order to limit computation.

In a typical hierarchical IoT architecture, nodes of a given level are connected to multiple nodes of an inferior level, and to a few nodes of superior level (often

⁴<http://google.github.io/physical-web/>

only one). Therefore, devices may be deployed with the hard-coded address of the middle node they communicate with, while in Fog or Cloud architectures nodes need to discover their underlying neighbors as they connect and disconnect. Discovery supports the deployment of dynamic infrastructures, which is both useful within the Fog tier and between the Cloud and the Fog tiers of the Cloud-Fg-Device pattern. Discovery function implementations supported by both semantic Cloud and Fog computing are therefore represented, with 8 studies based on semantic-enabled Fog nodes out of 11.

Exposition: Exposition is the function which is complementary to discovery, in which a node makes its own description available in order to be discoverable. The discovery target provides the information necessary for the discovery.

Based on semantic Cloud computing: The authors of [Ben-Alaya 2015] propose to enhance the registration mechanisms offered by the oneM2M standard to support a semantic description exposition. A device can register itself onto a Fog node if it knows its address, and the registration query contains a standard self-description of the device. The authors propose to include a semantic description, or a dereferencable URI, in the registration request. Since then, the oneM2M standard included a "semantic descriptor" resource, carrying an RDF/XML description of its parent node. It allows a node to expose its capabilities to its target when registering onto it. Another aspect of exposition is **proxying**: after discovering a set of devices, a Fog node can act as a proxy and perform exposition of said devices in their stead, as [Nikoli 2011] proposes. In the former, the authors propose an architecture where the exposition/discovery functions can be specific to a type of network or technology. This way, the function is both adapted to the constraints of the devices and understood by the Fog nodes, which performs an enrichment phase on the node metadata in order to redistribute them in a more generic format. In the proposed deployment, proxying is performed by a Cloud node, but it could be adapted for a Fog node as well.

Based on semantic Fog computing: [Christophe 2011] and [Mayer 2013] envision the embedding of self descriptions directly in devices. These descriptions can be queried through a REST interface. With such approach, discovery is only partly automated, since accessing the REST interface in the first place is not covered and supposes a minimal pre-existing discovery mechanism (such as a lookup directory). However, nodes provide their own semantic description, enabling an exposition towards other nodes of the same Fog architecture. In [Ruta 2016], exposition is supported by the PSW, already described for discovery.

Discovery and exposition are two interdependent functions, enabling the deployment and management of dynamic IoT networks. The rate of papers where semantic Fog computing supports the exposition function is similar to discovery (7 out of 10). The Fog infrastructure being by nature mobile and dynamic, it is supported by such functions which are necessary to the emergence of the "intelligent edge" proposed by [Patel 2017].

Selection: Selection is a function where a node decides which other node should perform a task. It is especially relevant when multiples nodes offer similar services yet have different capabilities, characteristics, costs, etc.

Based on semantic Cloud computing: [Vlacheas 2013] refers to the node selection function as the assessment of relevance metrics. The authors use abstracted nodes, or VO, and the semantic description of their characteristics (not precisely described in the paper), in order to perform a selection. The proximity criteria are defined dynamically depending on applicative requirements, whose expression is stored by Cloud nodes. This work focuses on the ability to perform a selection despite the heterogeneity of underlying objects thanks to the abstract representation of nodes. In [Perera 2014b], the authors identify two types of criteria for node selection: non-negotiable criteria, representing the ability of the node to provide a service, and negotiable requirements, over which selected nodes will be ranked. The filtering phase is performed using SPARQL queries representing user requirements, and the ranking phase is based on a multi-dimensional criteria aggregation. [Han 2014] describes a service selection function to create a composite service out of existing services. The actual node selection is performed at binding time, where service descriptions stored in a service cache by the middle nodes is used to associate the composite service with actual nodes. In [Fredj 2013], node selection is driven by a service search initiated by the user. This search starts from a top gateway which is the entry point of the user to a building network. Node selection is performed recursively in a gateway hierarchy, based on the semantic description of service clusters. At the lowest level of the hierarchy, the final gateway, actually connected to the lower nodes, returns the actual characteristics of the nodes matching the service request. Ranking criteria can also be based on user criteria, as in [Ara 2014] [Hussein 2016]. This approach is particularly suited for smart user spaces, such as shopping malls [Ara 2014] or airports [Hussein 2016], where multiple devices and services will be similar and user-facing.

Based on semantic Fog computing: [Christophe 2011] identify the node selection issue, especially for densely equipped environments, and envisions a solution based on semantic Fog computing, but their position is not refined with a precise contribution. In [Dey 2017], Fog nodes communicate with robots, and they use an ontology-based task representation in order to assign tasks. The capabilities of robots and nodes are used in order to select nodes able to perform the required tasks.

Node selection is driven by high-level policies (*e.g.*, energy saving or time efficiency), based on criteria to compare nodes, enabling the computation of a score and ranking. The use of SW technologies and principles are used to define these criteria. Selection is different from sheer discovery, because in the latter the function is driven by yes-or-no criteria, and the notion of comparison between nodes in this case is not considered. Depending on the complexity of the metrics chosen for comparison, on the expected volume of nodes to compare, and on the location of the SW stack deployment, selection is implemented by semantic Cloud or Fog computing. In the majority of the surveyed cases, **substantial resources are required** for

Table 4.3: Node-content dependency functions in the literature

Paper	Querying	Dissemination
[Su 2015]		F
[Boldt 2015]	F	
[Charpenay 2018]	F	
[Kelaidonis 2016]	F	
[Kaed 2018b]	F	
[Pease 2017]	C	
[Le-Phuoc 2016]	C	C
[Hasemann 2012]	F	
[Wang 2017]	F	
[Loseto 2016]	F	
[Howell 2017]	C	
[Ullah 2017]	C	
[Kharlamov 2016]	C	
[Dey 2017]		F
[Siow 2016]	F	
[Nagib 2016]	C	
[Ashraf 2010]		F
[Costea 2016]	F	
[Fredj 2013]		F
[Christophe 2011]		F
[Khan 2015]	C	
[Puustjärvi 2015]	C	
[Poslad 2015]		C
[Kaed 2016]	F	
[Bovet 2014]	F	
[Le-Phuoc 2012]	C	C
[Perera 2016]	C	

the computation, and a **global comparison** is preferred to a comparison of nodes in a context local to a Fog architecture. Semantic Cloud computing is therefore predominantly adopted for the selection function, with 7 studies out of 10.

4.2.3 Node-content dependency functions

In Node-content dependency functions, content itself is not transformed, but its management is still the primary focus, as opposed to node interworking functions. These contributions focus on how content is distributed across an IoT network (*e.g.*, routing), or accessed (*e.g.*, consumption). Content itself is not necessarily expressed using the SW technologies, but its management on the network is based on these technologies.

Querying: Querying is the function where an application explicitly accesses content on a remote node, in a **request/response manner**.

Based on semantic Cloud computing: In an IoT network, content might not necessarily be enriched and stored in a knowledge base, preventing it from being queried directly using SW technologies. However, mappings are possible from relational database (DB) schema to ontologies, enabling Ontology-Based Data Access (OBDA). [Kharlamov 2016] promotes such an approach, extended with aggregation functions, and deployed among Cloud nodes. In [Nagib 2016], content is also collected and stored in a Cloud DB, where it is accessed by clients via SPARQL using D2RQ mappings. [Poslad 2015] proposes a mixed approach, where part of the data is stored in a traditional SQL DB, but its associated metadata is stored as RDF for more expressive SPARQL queries. In other papers, where querying is not the core contribution, direct querying in SPARQL is also used after storing RDF content in a knowledge base, as it is proposed in [Pfisterer 2011] for instance.

Based on semantic Fog computing: [Siow 2016] proposes an OBDA approach, with transformation from SPARQL to SQL via RML, with and without SPARQL streaming extensions. This contribution is meant to be deployed on a Fog node: it takes advantage of the storage efficiency of relational DB compared to RDF, while still providing the expressiveness of SPARQL for querying. Efficiency is also a core motivation for the contribution in [Kaed 2016], where the authors propose a dedicated query engine, with an ad-hoc language and functionalities such as minimal inference. The engine is deployed on gateways, to give access to both the devices connected to it and the observations they collect. [Loseto 2016] proposes a different approach to content access, based on W3C's recommendation Linked Data Platform (LDP). However, LDP is mapped to HTTP, which is unadapted to very constrained nodes. That is why the authors propose an HTTP-CoAP mapping preserving functionalities of LDP, while enabling its deployment among Fog nodes. [Hasemann 2012] also relies on the CoAP protocol to give access to content by directly embedding an RDF store onto devices. The authors implement a specific access mechanism as part of the Wiselib they propose. [Boldt 2015] extends the Wiselib, by enabling SPARQL federated querying for network of devices, where a Fog base station exposes an endpoint and queries both devices and external knowledge bases. In [Charpenay 2018], access to content is also allowed down to the device level with the use of a binary serialization of JSON-LD and some developing features of JSON-LD, namely framing, that provides a querying method over JSON-LD data.

Querying is the third most represented function, because like enrichment and node abstraction, it is directly dedicated to interoperability: using the SW technologies and principles to access content hides schema heterogeneity and provides a meaningful querying vocabulary. Out of 21, 11 rely on semantic Fog computing. Content consumption can be offered as a service based on semantic Cloud computing for external applications, in which case the delay added by Cloud nodes is a small inconvenience compared to the unified interface it exposes. The content access paradigm has long been perceived as the channeling of all content into a

central point (DB or KB), which is well suited for semantic Cloud computing. The development of decentralized solutions and of query federation shows an alternative supported by semantic Fog computing, where smaller datasets can be accessed at a large horizontal scale.

Dissemination: Dissemination is the forwarding of a piece of content to remote nodes **in a push manner**. It can be driven by the interest of the receiver, or on applicative logic of the emitter. Routing is also a form of dissemination, where content packets are transmitted across the network from node to node based on a policy.

Based on semantic Cloud computing: In [Le-Phuoc 2012] and [Le-Phuoc 2016], streaming queries are used as expressions of interest in order to be notified when content is created. In this case, dissemination is performed toward applications. [Poslad 2015] uses dissemination as a mechanism to propagate emergency information as well as to make content available for data fusion. Content is initially gathered, stored and processed in Cloud nodes before being disseminated.

Based on semantic Fog computing: Routing and dissemination are also important features in moving, horizontal Fog architectures. [Ashraf 2010] proposes a distributed approach to routing, where each node computes the optimal route from itself to a destination broker. Each node knows the distance of its neighbors to the destination broker, and, to break ties, the semantic distance between the node profile and the profile of its neighbors is used. A node profile instantiates a dedicated ontology, and it is serialized into a binary string. Each node profile is unique, and used as an identifier. The evaluation function computes the energetic cost of the produced routing tree, and shows a clear **reduction of energy consumption**. In this approach, the SW principles are used to reduce energy consumption at runtime. However, the algorithm used to serialize the semantic description of the node output is not clearly described. The generated serializations of node descriptions are suitable for constrained nodes, but it is unclear whether they capture the full expressivity of the ontology language or not. In a different, hierarchical approach, [Fredj 2013] describes the construction of routing tables for service provisioning based on node clustering. Servicing nodes are described with an ontology, and they are clustered by the gateway they are connected to based on a semantic similarity measure. The obtained clusters are provided to the upper-level gateway, which uses the clusters from the nodes to build a routing table, and reproduces the process recursively. Semantic Fog computing plays an active role in the function, since even if the root of the hierarchy is a Cloud node, the intermediary nodes are distributed Fog nodes.

Routing policies are at the core of the internet, and not only of the IoT, but integration of SW technologies enables the definition of specific SWoT routing policies. In these approaches, the routes to forward content from one node to another can not only be computed based on a semantic description of nodes, but can also consider knowledge about the content itself in the routing process. Out of 8, 5 pro-

pose dissemination of content approaches based on semantic Fog computing. Two different cases emerge:

- Among Cloud nodes, dissemination is directed toward high-level applications, or disseminated content has been previously processed
- Among Fog nodes, dissemination is a function more directly dedicated to content transport based on contextual information about other nodes, and to knowledge sharing about content of a lower level in order to maintain a consistent context among Fog nodes

4.3 Identifying trends in SWoT processes

The landscape of the relationship between semantic Fog computing and SWoT architectures depicted in the previous section in Section §4.3.1 some insights on the trends observed in the surveyed publications. In the process of the survey, we identified some functions which seemed relevant to the evolution of the SWoT domain towards semantic Fog computing, but which were under-represented in the surveyed publications. It did not allow us to make generic statements as part of the previous survey, but these functions are presented in Section §4.3.2 as perspectives for future developments of the SWoT. The survey is focused on the integration of SW technologies in IoT architectures and Fog nodes, but there are also interesting evolutions of the SW in order to adapt to SWoT requirements. The purpose of Section §4.3.3 is to give an overview of these evolutions. Finally, a perspective on the complementarity of semantic Cloud and Fog computing in supporting SWoT deployment is provided in Section §4.3.4.

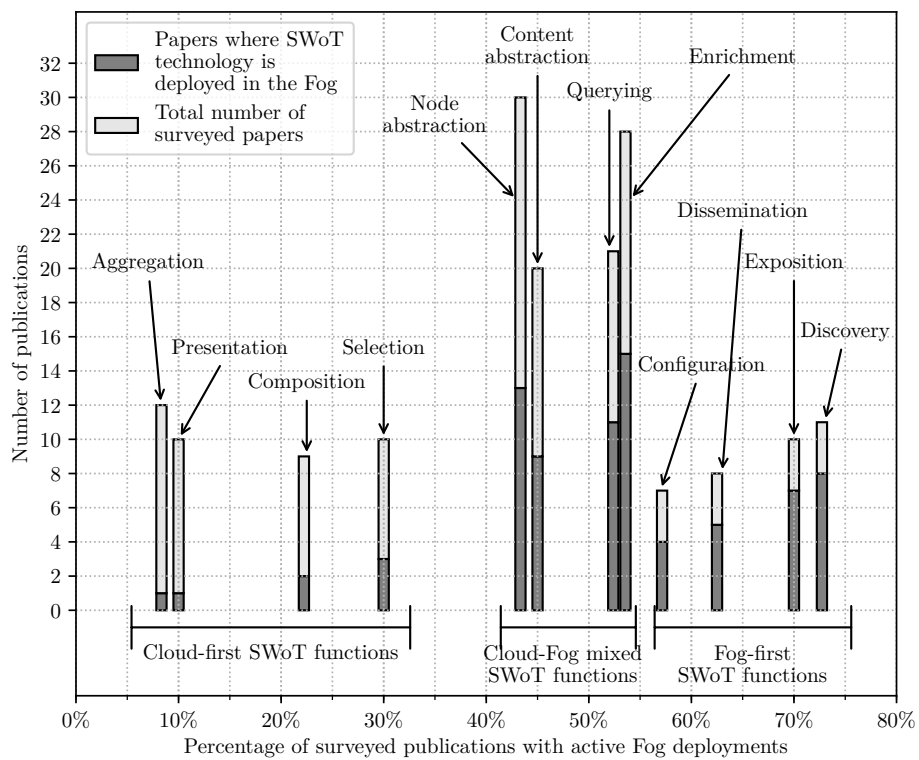
4.3.1 Analysis of the surveyed contributions

An overview of Tab. 4.2, and the analysis conducted in Section §4.2, leads to an initial answer to the research question: Fog nodes indeed actively support the deployment of SWoT functions. However, the simplicity of this answer does not cover the complexity of the relation between the Fog computing and SWoT domains.

Situating SWoT contributions in a reference 3-layered architecture shows how **nodes constrain the functions they support**: contributions dedicated to the same function but at different levels do not expect similar outcomes. This observation is an explanation for the predominance of solutions based on semantic Cloud computing in resource-intensive functions such as aggregation. On the other hand, functions that can be implemented with simple mechanisms requiring few resources, such as discovery or exposition, are easier to distribute on constrained nodes.

The notion of context scale is also a determining factor. Functions such as enrichment can be performed at a local scale, in a limited context, which is easy to achieve in Fog architectures, by definition more context-specific than Cloud architectures. Since Fog computing is based on highly distributed nodes, it is adapted to

Figure 4.2: Summary of identified functions, distributed per percentage of studies based on semantic Fog computing



parallelizable tasks, in which a local context is sufficient. On the other hand, functions such as presentation or composition are based in the surveyed contribution on a global context, centralized on a Cloud node. Aside from the resource point of view, distributing such global context among Fog nodes could introduce delays and consistency issues.

The identified functions have been defined as atomic units that can be combined together into functions of higher order. That is why papers introducing full platforms, such as [Kibria 2015], [Poslad 2015], [Perera 2016] or [Le-Phuoc 2016] are related to numerous functions: complex functionalities are achieved by building sequences of simpler functions. 9 papers share a common high-level function, namely analytics, a function where content is used to support decision-making and prediction. Decision support has two aspects: on the one hand, the supported decision-making process can be performed by human beings, and on the other hand it applies to autonomic systems. The former case is not specific to the IoT, but can be instantiated with a sensor network, and the system only provides guidance to a human decision maker. Analytics is implemented in [Fensel 2017], [Pease 2017], [Howell 2017], [Wang 2018] as a mix of content abstraction and aggregation, completed with presentation. Past content is aggregated with various mathematical approaches, and background knowledge is used in combination with aggregated content in order to achieve prediction, which is a form of abstraction. Both aggregated content and predictions are presented to the user in the form of graphs, maps, alerts, or dashboards, helping a human operator to capture a global understanding of the context. Some smart city scenarios use sensor networks and IoT deployments for decision support, *e.g.*, Dublin's Smart Energy Demand Analysis⁵. In this survey, all 9 papers implementing analytics does so on Cloud nodes. It is consistent with the considerable computing resources required to perform such a function, as well as with the globality of the contextualization of content preferred in decision-making that is better adapted to semantic Cloud computing. [Ullah 2017] provides a good illustration of the motivation for deploying analytics functionalities on Cloud nodes: the context necessary for a successful analysis is very large, and does not match the locality of decision-making enabled by Fog computing. [Ploennigs 2017] also considers analytics on Cloud nodes, but implements other functions (*e.g.*, enrichment) with semantic Fog computing, thus combining the characteristics of Fog and Cloud computing for complementary functions.

When composing functions to implement complex applications, a notion of lifecycle can be defined, where functions are chained in an order depending on the proposed application, and depend on each other. In this lifecycle, functions are executed either on Fog nodes or on Cloud nodes. The location of an individual function can also be explained by its dependency to other functions. For instance, if the enrichment of content is based on node abstractions that are only available on Cloud nodes, it is more likely that the enrichment function will also be implemented by semantic Cloud computing, even if the resources it requires would be adapted to

⁵<http://smardublin.ie/smartstories/spatial-energy-demand-analysis/>

a deployment among Fog nodes. These dependencies can be found in approaches promoting composition such as [Vlacheas 2013] or [Foteinos 2013]: the composition function is based on node abstractions built in the Cloud. These dependencies also explain the predominance of enrichment and node abstraction functions. Most of the identified functions depend on semantically described nodes and content, and enrichment or abstraction are prerequisite to advanced SWoT deployments. The increase of contributions based on semantic-aware Fog nodes for these preliminary functions in recent years enabled the upward trend of semantic Fog computing for functions such as content abstraction or node discovery. Of course, lifecycles can also be directed from the Cloud tier to the Fog tier of the Cloud-Fog-Device pattern as well, and complex deductions produced by Cloud nodes can be disseminated among Fog nodes.

4.3.2 Envisioning future functions

When proposing functions to describe contributions brought in the surveyed papers, we also identified functional units that provided functionalities relevant to SWoT deployments, but to the best of our knowledge they were not instantiated in the papers selected in the survey. The development of these functions is a next step in the evolution of the SWoT domain, as captured in [Wang 2015a]. The authors describe the initial notion of sensor devices, which evolved gradually into

- Sensor networks with the development of communications,
- Sensor Web with the concern of interactivity, accessibility and formats,
- WoT with the integration of smart Things
- finally Semantic sensor networks, fusion of the Semantic Web with Sensor Web.

The SWoT we depict is to the WoT what the Semantic sensor network is to the Sensor Web. The integration of actuators, and the possibility of communicating back and forth with constrained devices (instead of only consuming the data they collect), leads to the necessity for lowering, described in Section §4.3.2.1. The growing development of the SWoT also leads to the emergence of datasets, devices networks and applications using different ontologies, potentially to describe similar domains. Translating content from one ontology to the other is therefore an important interoperability-enabling function, that is only implemented in 1 contribution. We emphasize its potential in Section §4.3.2.2. Transforming content from one ontology to the other also raises consistency issues. Even more generally, consistency of the world representation in a dynamic SWoT system is an important issue, discussed in Section §4.3.2.3 Finally, the increase of publications basing basic functions on semantic Fog computing allows to expect the shift to semantic-enabled Fog nodes for some other functions, especially composition, as explained in Section §4.3.2.4.

4.3.2.1 Lowering

Lowering is the function exactly **opposed to enrichment**: it is the transformation of content from a semantically rich format to a less expressive, more constrained representation. It is a content value creation function of the DI transformation category. The need for such transformation arose from the integration not only of sensors, but also of actuators in the IoT. These devices are content consumers, but their constrained nature prevents them from being able to consume some types of content. Actions represented in rich formats in Cloud or Fog nodes need to be adapted to the target device so that despite their constraints, they can interpret these actions correctly and behave as intended. This transformation deprives the content of part of its expressivity and context, but the transformed, simpler content is interpreted in a known context, leading to a **trade-off for consistency**. Lowering requires the source node to have a representation of the capabilities and expectations of the remote target node. The source node in this function is by design more powerful than the destination node, and that is why the former should be aware of the restrictions of the latter. Lowering should not be confused with work such as [Maarala 2017], [Su 2018] or [Charpenay 2018], in which new formats are proposed to be supported in constrained environments. Even though these contributions are very valuable to the development of the SWoT, they *de facto* exclude legacy devices, that have no semantic capabilities whatsoever. The purpose of lowering is to produce low-level data, by taking advantage of the contextual interpretation performed by the legacy device.

Lowering is opposed in the literature to "lifting", a synonym for enrichment. [Kopecký 2007] describes SAWDSL, a language aiming at making it possible to lift XML to RDF and to lower RDF to XML thanks to XML schema annotations. The mapping reversal approach [Seydoux 2016c], introduced in Section §3.4, is an example of automatic lowering intended to support autonomic reasoning.

Overall, the transformation of content from a high-level representation to a form that can be processed by a constrained node is still a challenge for the SWoT. The lack of interest in this function partly comes from its contradiction with the usual practices in the SW community. Content is generally moved upward in the DIKW pyramid, because it gains value this way. However, the presence of **constrained content consumers** (actuator nodes) on the IoT, combined to the need for high-level content in more powerful nodes, makes lowering a function as necessary as enrichment for a complete SWoT deployment.

4.3.2.2 Translation

Translation is a function in which content enriched using an ontology is described with another one. It is a content value creation function of the DI transformation category. Translation enables interoperability between deployments based on different vocabularies. There are too few contributions to the translation function for us to make general observations as part of the survey.

[Howell 2017] is the only paper fitting the inclusion criteria in which translation is discussed. The authors refer to it as schema conversion, and implement such function in order to promote interoperability from their own domain ontology to SAREF, a reference smart building ontology. Mappings are expressed explicitly in the source vocabulary, and they are used in SPARQL CONSTRUCT queries to build an equivalent graph in the target vocabulary. This approach is deployed in a Cloud-based platform.

The ontologies of the SWoT are diverse, and only a few are reused by other [Seydoux 2016b]: the vertical fracturing between domains is not bridged, since domain-specific concepts are redefined disconnected from each other. Among the surveyed papers, SSN is the only widely reused ontology. In order to bring these ontologies together, alignments can be added, and the use of high-level ontologies (such as DUL⁶) or reference ontologies eases the integration of different ontologies in the same knowledge system. Due to the dynamicity of IoT, static manual alignments are not sufficient for a full semantic interoperability. A first contribution to translation in the SWoT was proposed by [Kotis 2012c] and [Kotis 2012b], that are not included in the survey because the nature of the proposed deployment is unclear. The authors however underline the importance of automating entity mapping to allow the dynamic deployment of heterogeneous devices. [Shvaiko 2013] lists ontology alignment tools, and identifies challenges, some of which are relevant to support the translation function by semantic-aware Fog nodes, such as matching efficiency or collaborative approaches. An example of such collaborative approach is provided by [Santos 2016], where correspondences are discovered through local exchange between nodes. This kind of automatic alignment technique is suited to a deployment among Fog nodes, even if not initially directed at the SWoT.

4.3.2.3 Consistency enforcement

Consistency enforcement is a content validity check function. It is a content value creation function of the IK transformation category. The notion of validity covers both the absence of contradictions among the facts expressed in the content, and the absence of wrong assertions (*e.g.*, sensor measures different from the reality of the physical world).

The only surveyed paper implementing this function is [Kibria 2015], where the consistency of inferred knowledge is validated using a reasoner. Both the inference and consistency enforcement are based on semantic Cloud computing. Other papers, such as [Charpenay 2015], discuss consistency enforcement techniques on SWoT content, but no details about the deployment are provided by the authors.

Consistency is not instantiated in detail in the contributions we surveyed, even if [Corcho 2010] identifies consistency as a challenge for the SWoT. Contradictions between several nodes can be dealt with in an aggregation function where data fusion techniques are applied, but consistency enforcement also includes the detection of logical issues in models and their instantiations. More generally, consistency

⁶<http://www.ontologydesignpatterns.org/ont/dul/DUL.owl>

enforcement is not a topic limited to the SWoT, and **generic consistency mechanisms could be applied to IoT datasets**. The integration of translation should be linked to a generalization of consistency enforcement: when manipulating different models, some local mappings might induce global inconsistencies that the system should be able to detect and prevent.

4.3.2.4 Composition

Increasing the availability of node abstractions and computing resources within Fog architectures gives potential to dynamically compose nodes in order to provide different services. The techniques that were initially developed for semantic Cloud computing that are presented in Section §4.2.2.1 could be adapted to semantic Fog computing, and techniques for service composition natively suitable to semantic-enabled Fog nodes such as [Martini 2015] could be adapted to the SWoT. Composing nodes in the Fog tier enriches local context, and exposes richer service to a Cloud node without requiring it to compute compositions, providing more scalability to large deployments.

4.3.3 Adapting Semantic Web technologies to constraints of the IoT domain

The issues tackled by emergence of the SWoT are issues affecting the development of the IoT and the deployment of functions within IoT architectures. These issues (*e.g.*, heterogeneity, lack of interoperability, content transformation) are recurrent concerns for the SW community, not necessarily related to the IoT or the WoT. However, the IoT also has intrinsic characteristics due to the constraints on its constituting nodes, the distributed nature of its deployments, and the dynamism of its topology. These constraints apply to any solution deployed in an IoT architecture. That is why **contributions between the SW and the IoT domains are not unidirectional**. The SW does contribute to the emergence of the SWoT domain by providing interoperability solutions to the IoT domain, but SW technologies and principles must also be adapted to meet IoT constraints in order to develop SWoT architectures.

One of the most important constraints in the IoT is the presence of physical nodes with limited capabilities, for both low-level devices and Fog nodes. The constraints on these nodes were presented in Section §2.5.2. Among the resources that can be limited for a node, we distinguished energy, processing power, communication channels, and memory. These resources are not independent, *e.g.*, the limitation of processing power or of time exchanging messages over the communication channel saves energy. In order to be suitable for an IoT network, a solution should dynamically adapt to the node running it, and the integration of SW principles and technologies must be thought differently on each level of the deployment architecture: a device does not have the capability to run the full SW stack. The sheer emergence of the functions identified in the literature is a first adaptation

to node constraints. Functions are spread across the IoT depending of the nodes able to support it. That is why resource-intensive functions such as analytics are performed by Cloud nodes, and simpler functions such as enrichment are deployed among Fog nodes.

Energy is a primary concern in the IoT domain for two reasons. On the one hand, the dissemination of nodes in the environment makes it hard to connect them to a power grid (especially in non-urban areas, such as fields or forests). Therefore, some nodes run on batteries, and their lifetime is directly related to their consumption of energy. On the other hand, the multiplication of nodes implies a multiplication of power consumers, and managing energy consumption on the scale of the node leads to energy saving on a more global scale. Using SW technologies to reduce the cost of content transport was already discussed in the description of the associated functions presented by [Ashraf 2010]. However, in this particular paper, SW technologies are not modified in order to comply with the constraints of an IoT network.

To show how SW technologies can be adapted to IoT network requirements, [Su 2015] compares the different serialization formats available for semantically rich data with respect to the size of the messages encoded in each format, as well as the number of CPU cycles required to produce these messages. The energy consumption associated to the creation, the transmission, the reception and the decoding of these messages is then compared for each format. In this paper, **two aspects of energy consumption** are pointed out: the **processing** required to handle the content, and the **communication** channels required to exchange it.

Reducing the cost of the communication can be achieved by using protocols adapted to the needs and constraints of the IoT, and by adapting existing platforms to these protocols. For instance, [Loseto 2016] proposes an extension of the LDP⁷ specification, a recommendation of the W3C. The W3C natively maps primitives of LDP to HTTP, and the authors of this paper propose a new mapping of LDP to CoAP, in order to make it suitable for constrained applications. CoAP is a protocol especially designed for constrained applications, with reduced headers and limited packet body, which has already been introduced in Chapter §2. Such an initiative allows IoT nodes to be connected to the Linked Open Data (LOD), and therefore to extend the WoT, while respecting the constraints of IoT nodes.

Distributing content via adapted protocol also requires said content to be stored on the device distributing it. Contributions such as [Bazoobandi 2015] or [Charpenay 2018] aim at allowing devices with limited memory to store semantically rich data. In the former, the authors propose a method to store compressed RDF data in memory while ensuring a certain level of efficiency regarding encoding and decoding of data. This paper is not specifically targeted to the IoT domain, but its contribution matches the requirements of this domain. Similarly, [Van Woensel 2018] proposes an algorithm for reasoning in memory-constrained environments which is not solely dedicated to the SWoT domain, but supports its

⁷<https://www.w3.org/TR/ldp/>

development. The ability to be able to access and modify the stored data efficiently is important in the case of streaming data, the authors point out. In the case of IoT deployments, and especially of sensor networks, the ability of storing and distributing up-to-date data is an important feature, and there must be a trade-off between memory optimization, and cost of encoding/decoding. [Hasemann 2012] also proposes a tuple store suitable for embedded systems, as well as a protocol-independent RDF broker, that can be mapped to CoAP for instance. The authors propose the adoption of a protocol stack adapted to constrained nodes in order to include them into the SWoT without the need for smart gateways acting as proxies.

4.3.4 Making semantic Fog computing transparent

With its IaaS, Paas, and SaaS services [Colombo-Mendoza 2012], the Cloud computing approach is based on easing access to resources by making the underlying complexity transparent. Cloud-based applications are widely popular nowadays, and Cloud providers such as AWS⁸, Google Cloud⁹ or Microsoft Azure¹⁰ offer transparent, instantaneous access to Cloud resources through APIs. Academic work has been pursued on Cloud architectures, to enable service orchestration [Bousselmi 2014] or application bursting [Charrada 2016] for instance, taking advantage of the simplicity of available resources.

IoT applications also benefited from the availability of large resources transparently accessible. IoT solutions targeted to end-users such as Apple Homekit¹¹ or Orange Homelive¹² are examples of IoT platforms transparently relying on Cloud nodes. Data collected from the local context of the user is stored and processed on third-party Cloud servers, providing a Web interface accessible from anywhere to the user. Privacy issues are not considered here, but rather user experience.

In its Cloud-based form supported by semantic-agnostic Fog nodes, the SWoT can be deployed in such Cloud architectures. However, we saw that deploying SWoT functionalities among semantic-enabled Fog nodes fostered scalability and reduction of application response time by bringing computation closer to data producers and consumers. If observation data is computed in an autonomic process to take action on local devices, performing the computation on a remote server induces a round trip from the local Fog node to Cloud nodes and back, leading to what is called a “trombone effect”¹³, or “tromboning”.

Considering that the **transparency of the infrastructure underlying the Cloud architecture** is a key element of its adoption, a next step in the integration of semantic-enabled Fog nodes in the SWoT architectural pattern is to **make semantic Fog computing transparent** as well. [Patel 2017] identifies gaps in

⁸<https://aws.amazon.com>

⁹<https://cloud.google.com/>

¹⁰<https://azure.microsoft.com>

¹¹<https://www.apple.com/ios/home/>

¹²<http://homelive.particule.orange.fr/>

¹³<https://www.networkworld.com/article/2224213/cisco-subnet/why-the-trombone-effect-is-problematic-for-enterprise-internet-access.html>

the current technologies constituting hurdles to the deployment of SWoT systems among Fog nodes. The heterogeneity of the Fog architecture is one of them, which provides motivation to try reusing the experience and competencies gathered in the Cloud computing community to hide the underlying Fog architecture complexity. Similarly to Cloud infrastructure, building affordable orchestration mechanisms for Fog infrastructure would support its active role in the deployment of the SWoT.

The transparency of semantic Fog computing is also a convergence point for a Cloud-Fog collaboration: rather than being seen as opposed, the Cloud and Fog paradigms should be seen as complementary approaches, each having specific characteristics and constraints. Cloud computing already provides an abstraction layer to cluster physical machines, therefore it is possible to envision the Cloud architecture as a wrapper for Fog nodes. In such a collaborative deployment, applications could communicate with Cloud nodes, where resource-intensive computations are located, along with large data storage. Part of the computation could also be distributed and processed by Fog computing, in order to be brought closer to IoT devices, in a dynamic architecture that is hidden from the end user. Such an approach would enable the deployment the SWoT technological stack while taking advantage of both Cloud and Fog paradigm characteristics.

4.4 Conclusion

In this chapter, a survey of the role of Fog nodes in supporting semantic processing in the deployment of the SWoT has been proposed. The Fog tier has previously been defined as an intrinsic component of the SWoT architecture in Section §2.5.3 with the Cloud-Fog-Device pattern. With Fog nodes providing a necessary interoperability layer connecting IoT devices to Cloud nodes, we surveyed in this chapter the type of interoperability supported in the Fog tier. Semantic interoperability may be enforced by semantic Cloud computing, in which case Fog nodes only provide technical and syntactical interoperability. However, part of the SW technical stack may be based on semantic Fog computing, in which case Fog nodes become semantic interoperability providers. In order to be able to assess the role of Fog nodes in the SWoT, we surveyed practices of the SWoT domain situated in the reference Cloud-Fog-Devices architectural pattern. Elementary functions have been identified through the projection of SWoT contributions on this architecture, in an extension of work initially published in [Seydoux 2017]. Contributions of the SW to the IoT domain have been classified with respect to these functions, with a particular focus on the support role played by the semantic Cloud and Fog computing. The complementarity of the Cloud and Fog paradigms has been confirmed by the trends observed among the surveyed papers.

- Cloud nodes provide considerable computing resources, as well as a global context due to their overview of IoT deployments.
- Fog nodes provide limited computing resources, that are disseminated in the

network and therefore closer to devices producing data. Since the Fog infrastructure is pervasive to the deployment, each Fog node only provides a local context for decision-making.

To complement this study, insights into the future of the SWoT has been presented, exposing future functions and evolutions of the SW developed to match constraints of the IoT. This twofold approach to a SWoT technical landscape showed the reciprocity of the convergence of the SW and the IoT domains:

- not only does the SW provide solutions to the interoperability and complexity issues of the IoT, but
- IoT constraints also challenge the SW principles and technologies to evolve.

The emergence of semantic Fog computing actively supports this convergence, providing promising solutions to the research challenges motivating the survey presented in this chapter.

The Fog paradigm promotes distributed approaches, by providing decentralized computing resources. We discussed the role of SW technologies and principles in support of interoperability in IoT architectures in the previous chapter §3. In order to overcome IoT constraints, we also identified the complementarity of the Cloud and Fog paradigms. Therefore, in the next chapter §5, the second set of contributions of this thesis is introduced, centered on the **Cloud-Fog collaboration** for semantic processing. Fog computing capabilities for scalability and Quality of Service (QoS) are leveraged with Cloud node capabilities for computation power and stability, as opposed to the volatility of a Fog infrastructure. These contributions intend to take full advantage of semantic Fog computing in order to support the deployment of more scalable SWoT architectures, adaptive to the dynamism of underlying IoT networks.

Decentralized SWoT: an adaptation to IoT constraints

Contents

5.1	Desirable characteristics for the proposed solution	102
5.1.1	Reasoning decentralization requirement	102
5.1.2	Applicative logic modularization requirement	103
5.1.3	Dynamism requirements	103
5.2	Related work for rules deployment in SWoT architectures .	104
5.2.1	Rules for the SWoT	104
5.2.2	Centralizing rule processing on Cloud nodes	105
5.2.3	Distributing rule processing on Fog nodes	107
5.3	Extending the smart building use case	108
5.4	Contribution II.A: EDR, approach for distributed reasoning	110
5.4.1	Assumptions on the underlying architecture	110
5.4.2	Overview of the EDR approach	110
5.4.3	A deployment mechanism based on a dedicated vocabulary .	112
5.4.4	Rule representation and deployment	117
5.5	Contribution II.B: Refining EDR with $EDR_{\mathcal{T}}$	122
5.5.1	Implementing a deployment strategy based on property types with $EDR_{\mathcal{T}}$	122
5.5.2	Node characteristics at stake in $EDR_{\mathcal{T}}$	124
5.5.3	Implementation of $EDR_{\mathcal{T}}$ in rule modules	128
5.5.4	Unraveling the main steps of $EDR_{\mathcal{T}}$	130
5.6	Conclusion	135

In the previous chapter, the support for SW technologies deployment in Fog nodes has been discussed. As explained, Fog nodes can be either passive technical interoperability providers between IoT devices and Cloud nodes, or active components where part of the SW stack is deployed. In the former case, all the computational load is dependent on Cloud computing, which by definition is remote from IoT devices, while in the latter it is possible to bring processing closer to data sources. Both Cloud and Fog computing have distinct desirable characteristics, that can be leveraged if used in complementarity to support IoT applications.

In the present chapter, two contributions fostering **Cloud-Fog cooperative semantic computing** are proposed:

- Contribution **II.A** is Emergent Distributed Reasoning, a **generic approach for dynamically distributed rule-based reasoning**. EDR is based on modular rules allowing the definition of deployment strategies fulfilling application-level quality-related requirements.
- An example of deployment strategy is implemented in $\text{EDR}_{\mathcal{T}}$, an approach refining EDR to support the **adaptative deployment of reasoning rules close to IoT devices**. $\text{EDR}_{\mathcal{T}}$, constituting contribution **II.B**, is dedicated to the reduction of the delivery delay for reasoning results.

The core characteristics of the contributions proposed in this chapter are described in Section §5.1. Existing distributed and rule-based reasoning approaches are studied in Section §5.2, with a particular attention to the characteristics previously introduced. Their shortcomings are detailed, so that we identify the limits we want to push with EDR. The detailed presentation of EDR is explained through a use case introduced in Section §5.3. The description of the EDR approach's core is provided in Section §5.4, and $\text{EDR}_{\mathcal{T}}$ is presented in Section §5.5.

5.1 Desirable characteristics for the proposed solution

As it has been discussed in Chapter §4, Cloud and Fog architectures play different role in the support of SWoT applications. To enable knowledge-driven applications, the efficiency of supporting reasoning by Fog computing must be considered. We define here the desirable characteristics constitutive of the solution we propose to support reasoning for SWoT applications.

5.1.1 Reasoning decentralization requirement

In order to ensure **scalability**, reasoning should be decentralized [Maarala 2017]: collecting all the data in a centralized place before processing it is a hurdle to scalability. IoT applications are deeply reliant on environmental observations, which values may vary unpredictably over time. In the case of centralized processing, all of the data captured by IoT nodes has to be forwarded towards the centralized, remote reasoning node. Such collect of data by a single node potentially generates an important stream of raw data in the network. Decentralizing reasoning allows data to be processed closer to where it has been initially collected, and thus to limit the resource consumption generated by its transit across the network.

Fog computing is a way of supporting the decentralized reasoning we want to achieve with EDR. The purpose of Fog-enabled architectures is to trade computational power for proximity with data sources, which is interesting for situations where increasing the proximity with data sources decreases the complexity of reasoning. When decentralizing processing, the individual computational load is reduced

for each node compared to a centralized approach, which can yield better performances [Su 2018]. However, the difference of computing power between Cloud and Fog nodes should not be neglected: Cloud architectures' intrinsic capabilities enable a resource upscale which is not possible for Fog architectures. Moreover, as it has been discussed in Section §4.3.4, Cloud infrastructures provide a stability that is complementary to the dynamic nature of Fog architectures. Therefore, we propose to leverage both the distributed nature of of Fog computing and the permanent, powerful nature of Cloud computing by adopting a mixed approach. Applications should only be required to communicate with the Cloud node to express their applicative logic, and such logic should be distributed among Fog nodes to process data closer to IoT devices.

5.1.2 Applicative logic modularization requirement

We determined that the reasoning supporting applications should be distributed. Inspired from the application bursting approach introduced in [Charrada 2016], we propose to consider modular applications to enable the distribution of some of their modules. Reasoning is oriented by knowledge representations, which are necessarily built subjectively for a specific purpose. Therefore, information is processed according to some application-level logic. In order to enable decentralization, this logic must be **explicitly represented and shared among nodes**. Rules are a common way to capture applicative logic: a rule is a self-contained representation of a logical process. Therefore, **EDR is a decentralized rule-based reasoning approach**.

Since our aim is to distribute rule among several nodes, **strategies must be developed in order to deploy computation on nodes**. Each node has intrinsic characteristics allowing them to process or not given rules successfully. We thus propose to base the strategies on node ad-hoc criteria depending on application requirements. Such characteristics will drive the placement of rules among nodes.

It is important to emphasize that deployment strategies are application-specific, since they are driven by application-level requirements. Therefore, focusing on a single deployment strategy only supports the deployment of applications sharing the same requirements. The purpose of our contribution is to foster multiple application types, and therefore to support different deployment strategies. A requirement for EDR is thus to be **agnostic to the deployment strategy**. Similarly to an abstract class in an object-oriented paradigm, EDR should define **generic functionalities** to support distributed reasoning, and must be instantiated with a deployment strategy to build a concrete approach.

5.1.3 Dynamism requirements

Since rules are elements necessarily dependent on applicative logic, we propose to embed the application-dependent deployment strategy within the rules. Each node handling a rule will thus be able to propagate it towards other Fog nodes according

to this embedded deployment strategy. The strategy considering characteristics of nodes in the deployment process, such characteristics should also be propagated among the nodes, in order for a given node to decide if the rule needs to be deployed. To this end, some node functionalities are designed to support information propagation among nodes, enforcing the decentralized nature of the EDR approach. Another requirement for EDR is that **each node should only depend on information that is available locally** when executing the deployment strategy. If a central controller was involved when a rule is propagated, then the burden of central computation would be shifted to a burden of centralized topology awareness, where a unique node would have a global perception of the topology. Such an oracle would be able to efficiently place rules according to the deployment strategy, but such an approach is not scalable. The dynamism of the topology would require a constant communication directed to a single point, and the oracle would need to have a complete representation of the topology, which could represent a large number of nodes.

The locality of decision-making requires nodes to exchange information continually to maintain a representation of their environment consistent with the evolving reality. EDR is meant to support deployment in Fog environment, in which resources are partially constrained. To limit resource consumption, **EDR should adopt an event-driven behavior**, in which messages are pushed from one node to another when an event occurs. Such behavior is opposed to poll-based or time-driven behaviors, where messages are exchanged constantly with no guarantee that a relevant information is conveyed. The event-driven nature of EDR enables the continuous adaptation of the rule deployment to the state of the topology.

The core of the EDR approach, implementing the characteristics identified in this section, is detailed in Section §5.4. Related work are studied in the next section §5.2, in order to compare the characteristics we want for EDR to the state of the art.

5.2 Related work for rules deployment in SWoT architectures

As the concern of the proposed approach is to deploy reasoning rules among Fog nodes to enable deducing application-dedicated information from IoT data, state-of-the-art work dealing with logical rules for the IoT, distributed reasoning and processing on constrained nodes is presented.

5.2.1 Rules for the SWoT

Rules are logical twofold elements, composed of preconditions and postconditions. Preconditions represent a state of the world such that the rule should be applied in order to generate its post conditions, which represent a new state of the world. In

our literature search, we identified two main types of rules associated to the SWoT [Boley 2007]:

- **Production rules**, or deduction rules, in which preconditions are expressed as a logical expression, and postconditions are new knowledge which are the logical consequence of preconditions. Schematically, production rules can be represented by IF-THEN structures.
- **ECA rules**, in which preconditions are the association of a logical expression and an event triggering its evaluation, and the postconditions are actions to be executed if the preconditions are matched. Such actions are not limited to knowledge inference: they can be instantiated by running a piece of code. Schematically, ECA rules can be represented by WHEN-IF-DO structures. For instance, IFTTT (If-This-Then-That)¹ is a Web service deployed on a Cloud node enabling the construction of ECA rules through a visual Web interface. The APIs of several IoT devices vendors², among other services such as social networks or popular websites, are available as triggers, inputs and outputs for IFTTT rules.

Production rules being explicit deduction representations, they have been considered in IoT networks to express and share the correlation between sensor observations and high-level symptoms since early work on the SWoT [Sheth 2008], implementing content abstraction functions as defined in Chapter §4. [Sezer 2018] lists numerous works using rules for context-awareness in the IoT.

With the goal of facilitating rule reuse, Linked Rules principles have been proposed [Khandelwal 2011]. They apply to rules the basic principles of Linked Open Data and Linked Open Vocabularies: rules are designated by dereferencable IRIs, expressed in W3C-compliant standards, and they can be linked to each other. Inspired from the Linked Rules, the Sensor-based Linked Open Rules (S-LOR)[Gyrard 2017] is dedicated to rules re-usability for deductions based on sensor observations. Production rules are a mechanism similar to CEP approaches (*c.f.* Section §4.2.1.2), used for instance in [Li 2010], but the rule representation shifts from an ad-hoc rule format in CEP to a unified format in the SWoT.

[Sun 2014] proposes a classification of production rules for the IoT, in order to identify recurring patterns. The authors distinguish rules enabling deductions from relations between nodes, and from relation between events (*i.e.* changes of the environment). In our contribution, we want to go further than this distinction by manipulating hybrid rules: their preconditions may both rely on conditions expressed on the nodes of the network, or on their environment.

5.2.2 Centralizing rule processing on Cloud nodes

In most existing approaches, *i.e.* [Li 2010], [Gyrard 2017] or [Xu 2017], production rules are handled by Cloud nodes. An example of IIoT use case enabled by

¹www.ifttt.com

²Such as Neato, iRobot, Samsung, WeMo, LG...

Cloud-based semantic rules processing is presented in [Wang 2018]. This paper proposes a self-configuring smart factory in which conveyors and machines produce data which is processed on a Cloud node where user rules are used to make reconfiguration decisions. Rules are expressed in SWRL. The same formalism is used in [Rodriguez 2010], where production rules are computed in a central Cloud node in order to dynamically reconfigure the communication network topology between devices and the Cloud node. The inferred deductions are converted into network reconfiguration actions by ad-hoc agents. A similar hybrid approach is used in [Evchina 2015]: rules are expressed as production rules, but their postconditions may include ad-hoc properties dedicated to the triggering of actions.

In [Kasnesis 2015], a multi-agent blackboard approach is chosen to dynamically manage rules in a smart home. Observations are published to a central node, the Domotic Status Board (DSB), where they are checked against rules in order to trigger inferences and reactions: the rules considered combine properties of production rules and ECA rules. Rules are expressed in the Jena formalism³, and an interface also allows users to control the system based on controlled grammar sentences. In this system, rules may be injected or deactivate at runtime. ECA rules are also used in a smart home use case in [Lillo 2015]: the authors propose an autonomic-like approach, where collected data is used to trigger actions of the system based on rules. The authors make a distinction between two types of actions stored in the KB. High-level actions, which are policies chosen by the user, and low-level actions, which are the actual implementations of the former, built by domain experts to hide the complexity of the system to the end-user. User preferences are expressed through a GUI, and converted from the GUI to KB individuals. During this conversion, appropriate low-level actions are selected to implement user-generated policies. The authors also introduce an ad-hoc approach to lowering (*c.f.* Section §4.3.2.1), with dedicated adapters transforming rich actuations representations into raw commands to be sent to actuators. The actual deployment topology is not presented, but the absence of any element indicating a distribution of the underlying platform leads to the conclusion that it is executed on a central node.

Production rules are used for context-awareness in a smart user space in [Hussein 2016]. Location information are combined to business knowledge, and to observations of the state of the user's environment, in order to make assumptions on the context. For instance, the following is a rule introduced by the authors: "IF the user is in an airport lounge with a low luminosity and the drapes closed THEN the user is sleeping". Such deduction is then used by context-aware services to adapt their behavior, materialized by ECA rules. Data required for the deductions are gathered into a central hub before being processed, and deductions are then sent to remote nodes.

An observation that was made in the previous chapter §4 regarding context scale is confirmed in the work that are described here. Rules are deported on Cloud nodes rather than executed in Fog nodes when used to achieve context-awareness,

³<https://jena.apache.org/documentation/inference/#rules>

such as in [Evchina 2015] or [Hussein 2016], in order to obtain a global execution context. However, in [Rodriguez 2010] for instance, some reconfigurations decisions could be taken only considering a local context. In this case, rules could be executed directly on Fog nodes.

5.2.3 Distributing rule processing on Fog nodes

The centralized architecture of the previously described papers raises multiple issues, such as the cost of semantic reasoning that increases rapidly with the size of the KB [Maarala 2017]. Fog computing offers a low-latency, resilient alternative for rule processing, even though the constrained nature of Fog nodes (compared to Cloud nodes) must be taken into account: processing power or bandwidth are critical resources. Centralization also requires all the content collected by IoT devices to be processed in the same place, while Fog computing makes computing power available closer to IoT devices. Therefore, Fog computing enables to process content with rules **where it is produced**, rather than requiring it to be transported to a remote node to be processed by Cloud computing. That is why rule placement in Fog architectures is a topic of interest for the SWoT

Most approaches for processing on constrained nodes focus on optimizations enabling such processing for a single node without considering the other. When considering a distributed execution composed of several Fog nodes, processing placement is not dynamic: all nodes execute the same rules, or each a predefined rule set statically assigned. For instance, even though it is not directly targeted at SWoT applications, the RETE algorithm proposed in [Van Woensel 2018] is dedicated to constrained nodes. RETE aims at reducing the memory requirements for production rules processing. This is a very interesting optimization, but it is dedicated to a single Fog node and does not consider distributed processing. [Desai 2015] shows how gateways are Fog nodes capable of enriching data: observations are initially produced by legacy devices in ad-hoc formats. It is the gateway, communicating with devices using protocols adapted to constrained environments, such as CoAP, that enriches the data before forwarding it towards a Cloud node. Therefore, observations are enriched on the edge of the network, and only the Fog nodes in direct contact with legacy devices have to perform data enrichment. [Lee 2016] or [Kaed 2018a] propose to execute ECA in Fog architectures, used to automate the response of the system to a stimulus. However, both authors only consider one gateway executing the rules, and the ad-hoc rule format is not suited for rule exchange. The contribution introduced in [Chatzigiannakis 2012] uses the same ECA model as IFTTT, but extends the rules expressiveness by using SW formalisms, namely SWRL and SPARQL. The authors use the Wiselib RDF provider [Hasemann 2012], as well as CoAP and 6LowPan communication, in order to enable semantic processing directly on constrained nodes. How rules are distributed in the network is not discussed by the authors.

Regarding processing distribution in existing work, the dynamic nature of IoT networks should be considered. The topology of a network evolves as devices con-









nect, disconnect, or move geographically. Therefore, a viable distribution of rules at a given moment is not guaranteed to remain optimal in the future, and **the distribution strategy should be adapted to the evolution of the network topology**. [Maarala 2017] does not detail the mobility strategy used for its mobile nodes, and each node applies all the rules regardless of their relevance to the content it aggregates. In [Su 2018], rule placement is static, in either Cloud or Fog nodes. [Taneja 2017] focuses on resource placement in a Fog-enabled IoT. The authors compute optimal deployment of application modules based on the representation of available resources in the Fog architecture compared to requirements expressed by applications. Module positions are static, and computed at the time of deployment. Rules are deployed on gateways in an IIoT context in [Kaed 2018b]. The rules themselves are not expressed using SW formalisms, but they are combined to a semantic engine proposed in [Kaed 2016] in order to consume enriched data. The placement of rule in the Fog architecture is not dynamic, however ad-hoc mechanisms enable rule update at runtime.

EDR differs from previous proposals by different aspects captured in the requirements described in Section §5.1:

- The genericity of the approach, enabling its adaptation to various application-level strategies.
- The locality of the knowledge involved in the rule deployment: each node only considers its own knowledge representation when propagating a rule.
- The dynamism of rule deployment in the SWoT system at runtime, constantly adapting to the state of the topology in an event-driven behavior.

5.3 Extending the smart building use case

The approach proposed in this chapter is applied on an extension of the smart building use cases introduced in Section §3.3.1. The ADREAM smart building is considered with a simplified architecture depicted by Fig. 5.1. Sensors in the building are represented with pictograms as follows:

- Anemometer: 
- Energy production: 
- Pyranometer: 
- Thermometer: 
- Presence sensor: 
- Luminosity sensor: 
- Thermostat: 
- Server load: 

The use-case building is instrumented with multiple sensors which observations are transported through gateways and M2M communication systems, constituting a Fog tier, to a Cloud node where the data is stored. Sensor observations are collected from the Cloud node to a control center on a dedicated computer or private

Figure 5.1: Reference architecture

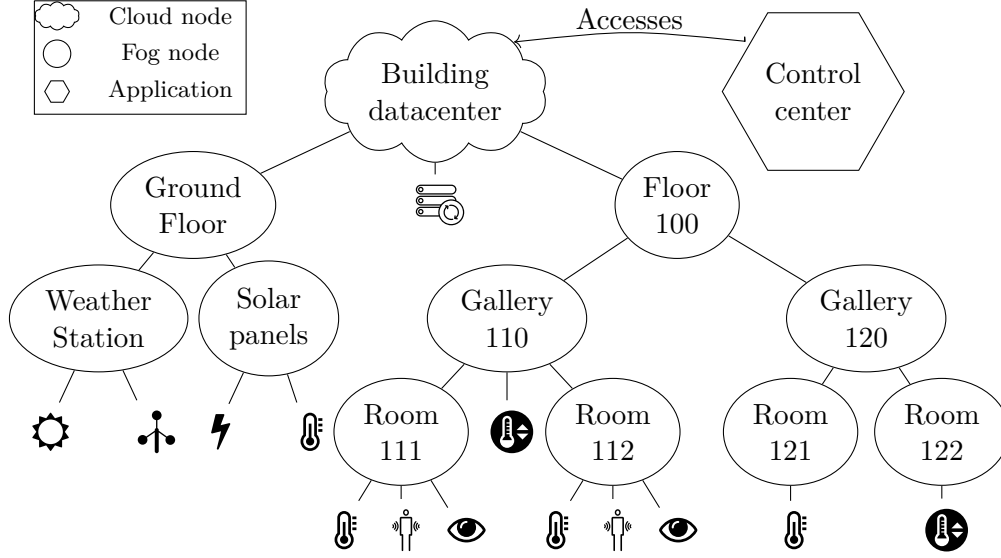


Table 5.1: Building monitoring rules

Rule ID	Rule core
$\mathbf{R}_{\text{Comfort}}$	$Location(?l) \wedge Temperature(?l, ?o_1) \wedge 19^\circ C < ?o_1 < 26^\circ C$ $\wedge Luminosity(?l, ?o_2) \wedge ?o_2 > 400L$ $\rightarrow Comfortable(?l)$
$\mathbf{R}_{\text{SolarProduction}}$	$OutsideLocation(?l) \wedge Temperature(?l, ?o_1) \wedge ?o_1 < 30^\circ C$ $\wedge Pyranometry(?l, ?o_2) \wedge ?o_2 > 430W/m^2$ $\wedge EnergyProduction(?p) < 15000W$ $\rightarrow AbnormalUnderProduction(?p)$
$\mathbf{R}_{\text{TemperatureGap}}$	$Location(?l) \wedge Temperature(?l, ?o_1)$ $\wedge TemperatureRequest(?l, ?o_2) \wedge ?o_1 - ?o_2 > 5^\circ C$ $\rightarrow TemperatureGap(?l)$

local Cloud servers where the BAS application is installed. This software displays collected observations to building administrators who monitor the activity of the different building systems. In order to make the notifications human understandable, they are processed based on logical rules deducing a higher-level information from a set of sensor observations. A set of illustrative rules is provided in Tab. 5.1.

Time is considered discrete, and cadenced by events such as sensor observations. Therefore, at any moment, a node holds the “most recent” representation of the state of features of interest it observes, from an event-driven point of view. For each property of said features of interest, only the last received observation is considered. Therefore, temporal considerations are out of the scope of the rules we manage.

The rules used by the administrators have different purposes, such as comfort or energy management. The diagnosis produced by the rules is also used to trigger

automatic actions taken by the BAS, which controls actuators such as lamps or cooling and heating systems. The fact that the building behavior is automated based on rules motivates their processing as early as possible.

5.4 Contribution II.A: EDR, a generic approach to dynamically distributed rule-based reasoning

In this section, EDR, a generic approach to dynamically distributed rule-based reasoning supported by semantic Fog computing, is introduced. It is the core contribution of this chapter, but since it is a generic approach, it is instantiated by a second contribution detailed in Section §5.5. EDR is based on architectural assumptions that are presented in Section §5.4.1. EDR's functional overview is depicted in Section §5.4.2, before presenting the vocabulary used to describe EDR core functionalities in Section §5.4.3. Modular rules are at the core of EDR, the formalisms used to represent them and the roles of their modules is described in Section §5.4.4. Once the main components of the EDR mechanisms are described, the relationship between EDR's characteristics and semantic Fog computing are discussed in Section §5.1.

5.4.1 Assumptions on the underlying architecture

EDR is based on the hypothesis of a **hierarchical network topology**: nodes are organized in a tree-like structure, and only communicate with neighboring nodes. This assumption is made because such topologies are frequent in IoT networks, represented in studies such as [Rodriguez 2010], [Zanella 2014], [Ben-Alaya 2015] (based on the oneM2M standard), [Szilagyí 2016], or [Su 2018]. Such hierarchical topology is related to the Cloud-Fog-Device architectural pattern: the tree root is a Cloud node, leaves are devices, and nodes in between are Fog nodes.

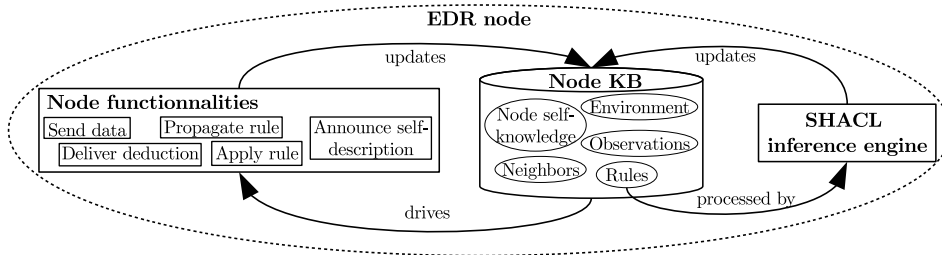
Applications are not deployed on a Cloud node belonging to the IoT topology: they are executed remotely on personal devices such as smartphones or laptops. **Rules represent applicative needs**: when deductions from sensor observations are required by an application, it injects the rule in the network in order to be provided directly with the deductions, instead of being forwarded raw data by the network and applying the rules itself.

It is assumed therefore that Fog nodes can communicate with applications directly. Rules are initially submitted by applications to the Cloud node, so it is the only node they know *a priori*. The Cloud infrastructure provides a unique permanent interface to the network, the dynamic Fog topology underneath is therefore transparent for applications.

5.4.2 Overview of the EDR approach

In order to ensure decentralization, the algorithm of the EDR approach is executed in parallel on each node able to perform reasoning in the topology, *i.e.* Cloud node

Figure 5.2: EDR node functional overview



and semantic-computing-enabled Fog nodes. EDR considers a step-by-step rule and data propagation: each node only communicates with its direct neighbors within the IoT network, *i.e.* its parent and children in the hierarchical topology. This design choice allows to reduce the nodes' knowledge of the topology to a limited subset of the complete deployment. Thus, consistency of the knowledge only has to be maintained with immediate neighbors, which limits required knowledge-related exchanges between nodes, and improves scalability. Due to the potential mobility and variable availability of Fog nodes, **EDR is meant to foster decision making in a local context for each node, leading at a large scale to the emergence of a desirable behavior.**

A parent node propagates a rule to its child if the parent considers that the child is empowered to apply the rule. This decision is made by the parent based on a **deployment strategy** embedded in the rule, as well as on the knowledge it has of its child. The deployment strategy captures the **criteria required for a node to process a rule**, and therefore characterizes if a child node is suitable to be forwarded said rule. In order to enable rule deployment, nodes exchange messages describing their capabilities, *e.g.*, their location, the type of data they observe, or the type of data they are interested in. When a node makes a new deduction based on a rule, it sends the result to all the nodes interested, including the application that submitted the rule.

The EDR approach itself is agnostic to the deployment strategy, which is defined by the rule implementer: that is why we qualify EDR as **generic**. The present section §5.4 is dedicated to the EDR approach, which defines the characteristics of a deployment strategy without implementing them. Such implementation is described with a refinement of EDR, $\text{EDR}_{\mathcal{T}}$, introduced in Section §5.5.

A functional representation of an EDR node is provided in Fig. 5.2: each node has a local KB, where knowledge necessary to the execution of EDR is stored. This knowledge is used to drive the basic functionalities of the node, and rules are used by the inference engine to update the KB.

Featured knowledge includes:

- the knowledge the node has of its own characteristics and capabilities,
- the knowledge it has about its neighbors,

- the knowledge it has about the static organization of the environment such as the geographic or indoor location, or the relationship between the surrounding elements,
- the value of the last observations depicting the current state of the dynamic features of the environment,
- the rules that it has received from either applications or other nodes.

This knowledge is used to control the behavior of the node, composed of simple functionalities. A node is able to:

- Send of a piece of data, typically a sensor observation, to a remote node,
- Propagate a rule to a remote node,
- Apply a rule on its knowledge base,
- Announce a description of its own capabilities to a remote node,
- Deliver a deduction obtained by processing a rule to a remote node,

How these node functionalities are related to the KB in the core EDR mechanism to enable the propagation of observations and rules is described in Section §5.4.3. The modular rule representation embedding the deployment strategy, and the updates of the KB they trigger, are detailed in Section §5.4.4.

5.4.3 A deployment mechanism based on a dedicated vocabulary

Nodes behavior is made on purpose quite simple, in order to decorrelate the rule-specific deployment strategy from the core algorithm on which EDR is based. Rule deployment strategies are dedicated to a particular purpose, *e.g.*, response time reduction or privacy enforcement, while EDR is generic. In order to support the genericity of EDR with a knowledge-driven method, nodes functionalities are based on a dedicated vocabulary, used to describe knowledge in the node's KB.

For instance, this vocabulary captures the hierarchical nature of the topology. The relation between a node n_p and its child n_c is expressed with the triplet $\langle n_p, lmu:hasDownstreamNode, n_c \rangle$ ⁴⁵, based on a nomenclature presented in [Seydoux 2017]. The inverse relation exists, to express the connection between a node n_c and its parent n_p : $\langle n_c, lmu:hasUpstreamNode, n_p \rangle$.

A description of all the functionalities of the nodes, and of the vocabulary that drives them, is provided in Section §5.4.3.1. Further details about the announcement functionality are provided in Section §5.4.3.2, especially with regard to the consumption of data. Finally, the scope of the announces is studied in Section §5.4.3.3.

⁴Namespaces are listed in Appendix §A.1

⁵Individuals such as n_p and n_c are identified with an IRI in the triplets

5.4.3.1 Basic node functionalities

Each functionality relies on dedicated triplets, and a node implements its behavior based on the description held in its KB. How this triplets are inferred from the deployment strategy is described in the next section §5.4.4. Before detailing how the strategy triggers nodes functionalities, let us examine the vocabulary describing said node functionalities.

Announce self-description: When a node connects, disconnects or changes capabilities, it notifies its neighbors of its self-representation. The neighbors of a node n are represented in its KB with the triples $\langle n, lmu:hasUpperNode, n_{parent} \rangle$ for the parent of n , and $\langle n, lmu:hasLowerNode, n_{child} \rangle$ for each child n_{child} of n . Since a notification is sent at each update of the state of the node, the perception of a node by its neighbors remains consistent with its evolution over time. Two mechanisms support this announce:

- a partial update, in which a node adds statements to its description already held by the target
- a complete update, in which the representation of the node is completely erased by the target before being updated.

These mechanisms allow to add information about a node by exchanging light messages containing partial representations, while enabling to remove outdated statements with the complete update. A particular node characteristic that is declared in the announce functionality is the type of data in which a node is interested, captured with the predicate *edr:isInterestedIn*, which is used in the data sending functionality. The announce functionality is extended by the mechanisms described in Section §5.4.3.2 to control which characteristics of the node are propagated, and the scope of this propagation in Section §5.4.3.3.

Apply rules: When a node n receives a new observation, either from its own sensors or lower nodes, n executes the rules r stored in its KB if the description of r contains $\langle r, edr:isRuleActive, true \rangle$.

Deliver deduction: If the processing of an observation with rule r by node n leads to a deduction δ , δ is sent to each node belonging to $\bigcup n_{consumer}$ where $\langle n_{consumer}, edr:consumesResult, r \rangle$ is in the KB of n . Especially, the application that submitted the rule r to the network is known as the rule originator o , and is represented by the triplet $\langle r, edr:ruleOriginatedFrom, o \rangle$. The originator of a rule is considered as a consumer of rule results, in order to enable deduction delivery to applications. The deduction delivery functionality is separated from the interest notification part of the announce functionality for flexibility. In Chapter §6, the deduction delivery mechanism is used to manipulate the communication flow, which would not be possible if it was not a functionality independent from the interest announce.

Send data: If the parent of node n (denoted n_p) has declared its interest for the type of the new observation α_t , the observation is forwarded toward n_p . Observations are exchanged lazily: if a node n receives an observation of type α_t , and knows no other node interest in such type, the observation is not forwarded. Such interest is represented in node n KB with the triplet $\langle n_p, \text{edr:isInterestedIn}, \alpha_t \rangle$. The notification of the interest is considered as a characteristic of the node, managed in the announce functionality.

Propagate rule: A node sends a rule to one of its neighbor if it considers that this neighbor is capable of applying the rule, such consideration being part of the rule deployment strategy. In the case where rule r should be propagated towards node n_{target} by n , the triplet $\langle r, \text{edr:transferableTo}, n_{target} \rangle$ is present in n 's KB.

5.4.3.2 Controlling the propagation of nodes characteristics

The EDR algorithm depends on the exchanges between neighboring nodes of their mutual descriptions. The announcement functionality is dedicated to the exchange of such descriptions. However, presupposing of the nodes characteristics relevant to any deployment strategy that will be implemented to refine EDR is not possible. In order to remain agnostic to the deployment strategy, EDR relies on a dedicated vocabulary used to describe which of each node's characteristics should be announced to its neighbors. A node has two types of neighbors: its parents, and its children, and since the parent is unique (according to our assumptions) while the children are potentially many, two approaches are devised.

Announcing characteristics to a node's parent: Let us consider a node n , with a characteristic represented by the property *hasCharacteristic* and captured in its knowledge base such that $\langle n, \text{hasCharacteristic}, \nu \rangle$, with ν either a literal or an individual denoting the value of the characteristic for n . When announcing its characteristics to its parent, n searches its KB for all the triples where it is the subject, and the predicate is a predicate types as *edr:ParentAnnouncedProperty*. If the property *hasCharacteristic* is such that $\langle \text{hasCharacteristic}, \text{rdf:type}, \text{edr:ParentAnnouncedProperty} \rangle$, then the triple $\langle n, \text{hasCharacteristic}, \nu \rangle$ is part of the self description sent by the node n to its parent because *hasCharacteristic* is considered a relevant characteristic of n .

Announcing characteristic to a node's children: The announce mechanism from parent to children is quite similar to the one from children to parent, with the exception that children may be many, whereas by assumption, a node only has one parent. Therefore, the class *edr:ChildrenAnnouncedProperty* has two subclasses to distinguish two possible cases:

- *edr:AllChildrenAnnouncedProperty* denotes a characteristic that is systematically announced to all the node's children.

- *edr:SomeChildrenAnnouncedProperty* denotes a characteristic that should only be announced to a subset of the node's children.

This distinction is made to give flexibility to the deployment strategy designers.

In the case of a characteristic captured by a predicate of type *edr:SomeChildrenAnnouncedProperty*, each child eligible to be proxied the new characteristic must be represented explicitly with the predicate *edr:announceTo*, which requires the reification of the announced characteristic. In order to be announced towards child node n_{child} , the triple $\langle n_{parent}, hasCharacteristic, \nu \rangle$ is transformed into the following reified statement: *statement rdf:subject n_{child} ; rdf:predicate c ; rdf:object ν ; edr:announceTo n_{child}* . The choice of the children to which the characteristic should be announced is application-specific, and is therefore part of the deployment strategy. As the rest of the deployment strategy, it is embedded in rules as it is described in Section §5.4.4.

The interest of a node for a type of data, denoted by the predicate *edr:isInterestedIn*, is managed as a node characteristic. Therefore, depending on the deployment strategy, the interest of nodes is classified as one of the subclasses of *edr:ChildrenAnnouncedProperty*. More details about this particular predicate is provided in Section §5.5, with the instantiation of a concrete deployment strategy.

5.4.3.3 Propagating knowledge beyond direct neighbors

The basic functionalities only enable the communication of a node with its direct neighbors in the hierarchy, either parents or children (with the exception of deduction delivery). This enforces the neighbor-to-neighbor nature of the propagation enabled by EDR. However, such design may hamper the propagation of rules, by preventing the diffusion of knowledge required by the deployment strategy to make decisions so as to where the rules should be placed. If the characteristics of a the child n_{child} of a node n makes it adequate to apply a rule which is held by the parent of n n_{parent} , but n cannot apply the rule, n_{parent} will not propagate the rule to n , preventing its eventual propagation to n_{child} . A complementary functionality is thus described by the EDR vocabulary to enable such diffusion of knowledge describing nodes capabilities: **proxying**.

The proxying mechanism implemented in EDR is inspired from [Nikoli 2011], where reasoning nodes act as proxy for the capabilities of legacy nodes unable to process enriched data. In EDR, each reasoning-enabled node has a similar role, and proxies capabilities of its neighbors. Such proxying is bidirectional: the capabilities of a nodes parent are proxied towards its children, and the other way around. Specifically, node n proxying its parents capabilities towards its children means that n announces these capabilities as its own to its children. An example of proxied node characteristics, detailed in Section §5.5.2.2, is the interest of a node for a data type, briefly introduced here for the sake of illustration. If a node n wants to be notified whenever a temperature observation is available, it notifies its children $n_{child} \in Lower(n)$ of such interest. If any child n_{child} collects temperature observations, it will forward such observation towards n . Moreover, each n_{child} will

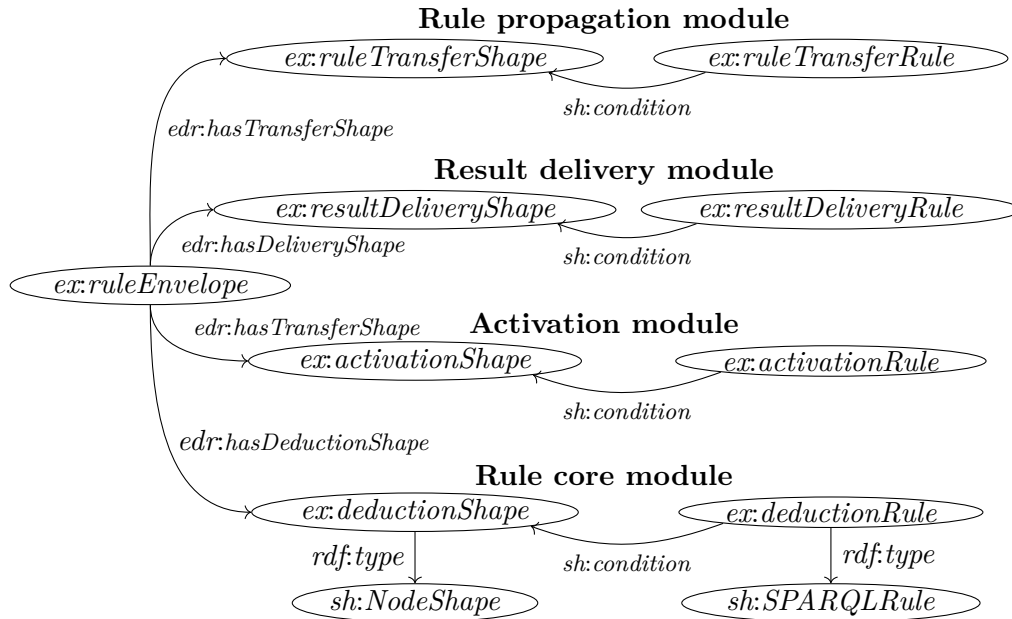
in turn notify that it is **itself** interested in temperature observations to any node $n'_{child} \in Lower(n_{child})$. Any node n'_{child} collecting a temperature observation will therefore send it to n_{child} , which will itself send such observation to n . The characteristic of the initial node n (here, the interest in temperature) has indeed been proxied to n'_{child} by n_{child} : n'_{child} only has knowledge of n_{child} , and communication is kept strictly between direct neighbors. To support this mechanism, two classes of properties are defined in the EDR vocabulary: *edr:ParentProxiedProperty*, and *edr:ChildrenProxiedProperty*.

Characteristics proxied from children to parent: Let us assume that node n has a child n_{child} , and that such child has a characteristic expressed by the triplet $\langle n_{child}, hasCharacteristic, \nu \rangle$, that should be proxied towards the parents of n . Such information about the predicate c is materialized by the triplet $\langle hasCharacteristic, rdf:type, edr:ParentProxiedProperty \rangle$. When receiving description of n_{child} , n checks for the presence of properties classified as *edr:ParentProxiedProperty*. Since *hasCharacteristic* is such a property, the node n updates its own representation towards its parents by sending the triple $\langle n, hasCharacteristic, \nu \rangle$, therefore proxying the capacity of n_{child} .

Characteristics proxied from parent to children: The proxying mechanism from parent to children is similar to the one from children to parent. Contrarily to the case of the announcement functionality, the multiplicity of children is not considered: all the children are proxied any received parent characteristic. Such policy is made necessary by the locality of decision-making enforced by EDR. On the one hand, a node n receiving a characteristic to proxy from its parent n_{parent} does not have the contextual knowledge that lead n_{parent} to announce this particular characteristic to n . On the other hand, the node n_{parent} does not have a detailed knowledge of the topology below its child n , and therefore cannot make any assumptions about to which children in particular n should proxy the characteristic of n_{parent} .

It is possible that the proxying mechanism and the announcement mechanism lead to conflicting behaviors. In particular, a node may have chosen not to announce a characteristic of its own to some of its children, but be required to proxy the same characteristic in the stead of one of its parent. In this case, the proxying mechanism supersedes the announcement mechanism, and any proxied characteristic is processed as a *edr:AllChildrenAnnouncedProperty*. For instance, if a node n did not announce its interest for a data type ρ_t to its child n_{child} , n will nonetheless announce such interest to n_{child} if the parent of n , n_{parent} , notifies n of its own interest for ρ_t , and requires n to proxy such interest.

Figure 5.3: Rule modules



5.4.4 Rule representation and deployment

5.4.4.1 Rule modular structure

EDR rules are composed of several modules, as it is represented on Fig. 5.3. Each of these modules enables some node functionalities:

- The Activation module triggers the rule application, the data consumption and the result delivery functionalities.
- The Deduction delivery module triggers the result delivery functionality
- The Rule transfer module triggers the rule forwarding functionality

Therefore, the intelligence regarding rule deployment is located in the rules, and not in the nodes. The behavior of the algorithm at a global scale can thus be parameterized at a fine granularity, for each rule. Rules are represented in SHACL, and the modules are based on SHACL advances functionality named “SHACL rules”. Each module is composed of two parts: a SHACL rule, that inserts deductions into the KB, and a SHACL shape that determines whether the rule is applied or not. An example rule is provided in Appendix §A.2, and the raw source is also available online for convenience⁶. In the remainder of this section, a generic description of these rule modules and their roles is given. An implementation is proposed in Section §5.5, where specific behaviors dedicated to a particular strategy are described.

⁶<https://w3id.org/laas-iot/edr/iiot/r1.ttl>

Listing 5.1: $r_{comfort}^{transfer}$ rule

```

CONSTRUCT {
  ex:officeLightComfortRule  edr:transferableTo $this.
  ex:officeLightComfortRule  edr:transferredFrom ?host.
} WHERE {
  $this lmu:hasUpstreamNode ?host.
  ?host a lmu:HostNode.
}

```

In order to associate all the modules to a rule represented as a single individual in a node's KB, we introduce the notion of **rule envelope** as a reification mechanism. The envelope of an EDR rule is an individual subject of triples which predicates are *edr:hasTransferShape*, *edr:hasApplyShape*, *edr:hasDeliveryShape* and *edr:hasDeductionShape*. The rule envelope is especially useful in the rule deployment process, when all the modules of a given rule must be collected for the rule to be propagated to a remote node. The envelope of rule r is denoted $r^{envelope}$.

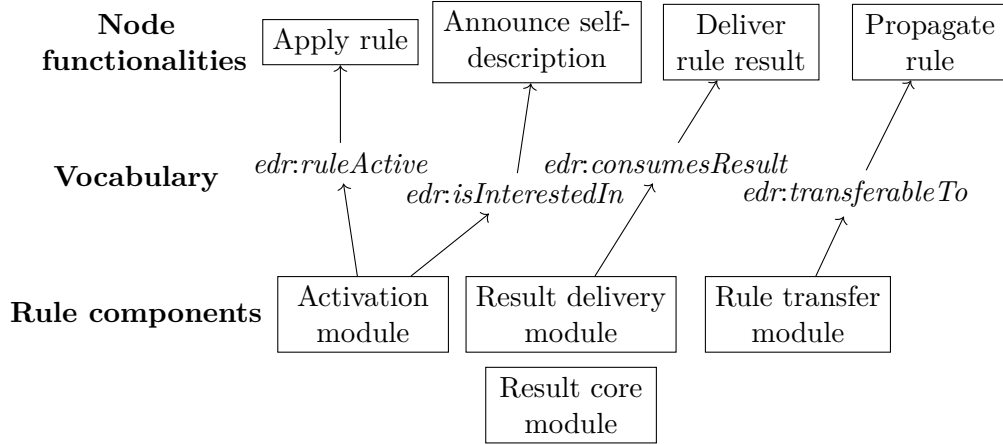
5.4.4.2 Rule modules

Core module The operational part of the rule, containing the application-dedicated inference, is referred to as the **rule core** module. The core module is based on a predicate logic rule used to deduce high-level information, similar to the rules introduced in the use case in Section §5.3. Let r^{core} be such a rule core module, noted as $r^{core} : \Gamma_1 \wedge \dots \wedge \Gamma_n \rightarrow \Delta_1 \wedge \dots \wedge \Delta_m$, where $\Gamma_1 \wedge \dots \wedge \Gamma_n$, designated as the **body** of r^{core} , is a conjunction of conditions and $\Delta_1 \wedge \dots \wedge \Delta_m$, designated as the **head** of r^{core} , is a conjunction of deductions. The rule core module only encompasses applicative deduction logic: it is unrelated to the deployment of the rule. This module is only evaluated when the rule has been declared active on a node in the deployment process, *i.e.* if the triple $\langle r, edr:isRuleActive, true \rangle$ is in the node's KB.

Rule transfer module The **rule transfer module** determines on which remote nodes the rule may be deployed, according to a rule-specific deployment strategy. This condition is expressed as a SPARQL query embedded in the SHACL rule being the conditional part of the rule transfer module. The deduction part of the module infers the triple $\langle r, edr:transferableTo, n' \rangle$, enabling the rule forwarding mechanism of the node (*c.f.* Section §5.4.3.1). An extract of the SHACL rule of the transfer module for rule $r_{comfort}$, denoted $r_{comfort}^{transfer}$, is provided in Lst. 5.1.

Rule activation module The **activation module** detects if the current node is suitable to apply the rule itself. If the conditional part of rule r activation module determines that the current node is suitable to apply r , the activation of rule r is made explicit by the triplet $\langle r, edr:isRuleActive, true \rangle$. In the case where

Figure 5.4: Relation between node functions and rules modules



some node characteristics are conditionally proxied towards children (*edr:SomeChildrenProxiedProperty*), the rule activation module may infer reified statements as described in Section §5.4.3.3. This case is illustrated in more details in Section §5.5.3. The activation module of a rule r is denoted $r^{activation}$.

Result delivery module The **result transfer module** enables the forwarding of deductions to other nodes that are not the originator of the rule, such as the parent n' of a node n if n' applies a rule r' that consumes the deductions made by a rule r applied by n . By default, the originator o of a rule r is assumed to be interested in the results of r , denoted with $\langle o, edr:consumesResult, r \rangle$. If a remote node n' is interested in the deductions made by rule r , the result transfer module infers that $\langle n', edr:consumesResult, r \rangle$. The result delivery module of a rule r is denoted $r^{delivery}$.

The relationship between node functionalities (represented in Fig. 5.2), the EDR vocabulary and the rule modules is shown in Fig. 5.4.

5.4.4.3 Dynamically managing modules activation

The rule core must be computed each time a new observation is received by the node, in order to check if new deductions may be inferred. However, it is worth noting that the other rule modules only need to be evaluated when the rule is received, or when the topology evolves, *e.g.*, with new productions by children, new consumptions by parents, or nodes connecting/disconnecting.

The SHACL standard is so that by default, when reasoning on a KB containing SHACL shapes and rules, all of them are considered. In order to reduce the computation load, and to only process rule modules when needed, a SHACL functionality is used: the reasoner does not consider shapes or rules r such that $\langle r, sh:deactivated, true \rangle$. The modules of a rule r are therefore only activated for a reasoning step when r is received, or when the topology evolves.

The appropriate modules, *i.e.* all except the core module, are classified as *edr:NodeSensitiveComponent* (as opposed to what would be a “Content sensitive component”). Therefore, a simple query is sufficient to activate or deactivate rule modules related to deployment, and to do so for all the rules stored in a node’s KB.

Deployment modules management is represented on Fig. 5.5, in an overview of the algorithm. When a rule is initially received, all of its modules are active. That is why no activation is required when receiving a new rule, (1) on Fig. 5.5. The rule deployment update, (3) on Fig. 5.5, is performed by the reasoner. Since no other rule deployment modules has been activated since the new rule has been received, and by default these modules are deactivated, only the deployment of the newly received rule is computed.

In the case where the node receives an information about a topology update, such as the connection or disconnection of a node or the change of capability of a known node, it is possible that the rule deployment should be updated accordingly. That is why, for all the rules stored in the node’s KB, the deployment modules are activated upon the reception of a topology update, as seen in (2) on Fig. 5.5. The received change is then integrated in the KB, and if necessary the new topology is propagated to parent nodes, before performing a reasoning step computing the deployment rule modules. If the placement rule needs to be updated due to the topology change, the new deployment is enforced by activating or propagating rules in compliance with the deductions and the EDR vocabulary, before deactivating the rules deployment modules (4) on Fig. 5.5.

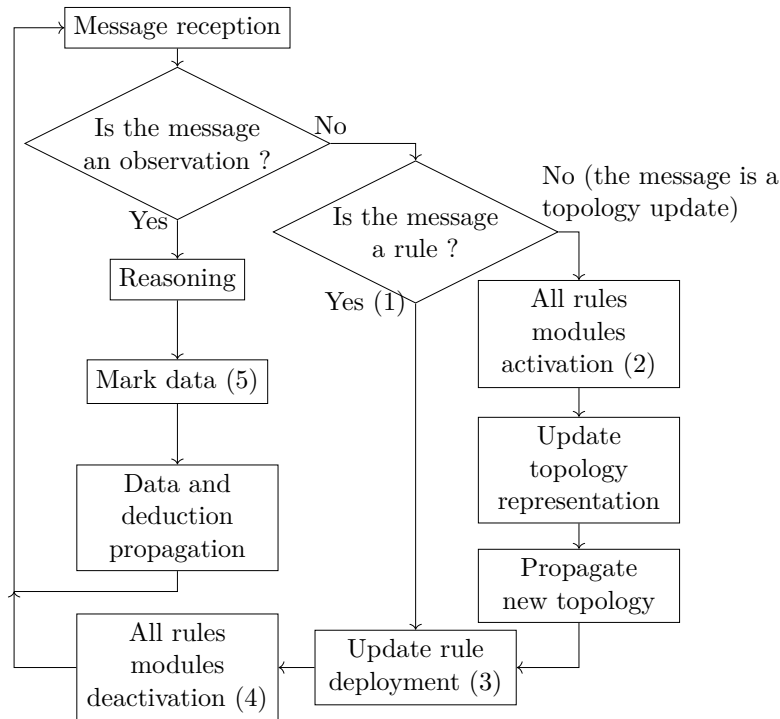
If the received message is an observation, no rule deployment update is required. The only active rule modules are the core modules for rule that the node should process, and they are used by the reasoner to test if new inferences are possible. The marking and propagation of deductions is discussed in Section §5.4.4.4.

5.4.4.4 Leveraging the unique identification of rules

EDR rules are compliant with the Linked Rules principles [Khandelwal 2011], and in particular they are uniquely identified by an IRI. The identification of rules being shared among all nodes, provenance can be traced for a given deduction. Two purposes have been identified for this traceability: the avoidance of redundant computation, and the update of rules at runtime.

Preventing redundant computation With the rules being uniquely identified among all nodes, it is possible to mark observations when they have been processed with a rule, successfully leading to a deduction or not. After an observation o has been involved in a reasoning step with rule r , a new triple is added to the observation description: $\langle o, \text{edr:usedForDeductionBy}, r \rangle$. This marking prevents an observation to be processed multiple times with the same rule when it is propagated from one node to another. Considering this marking or not is up to the rule implementors: for instance, the strategy presented in Section §5.5 takes it into account, so that each observation is at most processed once by each rule for performance issues.

Figure 5.5: EDR algorithmic overview



Depending on the propagation strategy, it may be necessary to process the same piece of data with the same rule in multiple contexts, in which case the marking may be ignored. The marking of observations with the *edr:usedForDeductionBy* property is shown on Fig. 5.5, (5).

If a rule is submitted by multiple applications to the topology, the uniqueness of the identifier also enables to avoid redundant processing. In a node's KB, each rule can be associated to several originators, indicating that the deduction should be sent to several applications. Expressed in an application-specific namespace, two identical rules would be applied twice, leading to a waste of resources.

Updating rules at runtime The use of a unique dereferencable identifier also allows to incrementally modify rules at runtime, so that the operation of the monitored system is not interrupted. Modifying rules allow applications to fine-tune their behavior according to a feedback loop that considers either previous responses to inputs, or external factors (*e.g.*, seasonal change, or regulation evolution). When a rule r is received by a node n , if r 's IRI is already known by n , all the triples describing the rule are compared to the triples stored in the node's KB.

If the newly received version of the rule is different from the version held by the node, then the rule representation is updated in the KB, and the rule is processed as if it were a new rule. However, it is possible that the new representation of the rule is no longer applicable by children of the current node, to which the former version of the rule had been previously propagated. In the regular EDR algorithm,

the rule would not be forwarded to such children, but in this case this is an issue: two different mutually exclusive versions of the rule are executed in the topology.

To tackle this issue, an object property is used: when a node transfers a rule r to its child n_{child} , it adds the triple $\langle r, edr:transferredTo, n_{child} \rangle$ to the rule description stored in its KB. When a node updates a rule representation, it transfers the new rule version towards the children which received the former version by searching for this property. If said children are not able to apply the new version of the rule (as determined by the application module of the rule), updating their rule representation enforces the consistency of the rule across the network. The same process is carried on recursively from parent to child node in order to ensure that all the nodes of the topology eventually have an up-to-date representation of the rule.

This approach however leaves a consistency issue unsolved: during the propagation of the new rule version, the two mutually exclusive versions of the same rule are both active. There is no guarantee that the latest version of the rule has been propagated successfully at any point in time after its injection in the network. A way to solve this issue is to attach a version number to the rule with the *owl:versioninfo* annotation property. This version information is then attached to deductions made with the rule, so that applications are aware of the version of the rule that lead to any deduction.

5.5 Contribution II.B: Refining EDR with $EDR_{\mathcal{T}}$

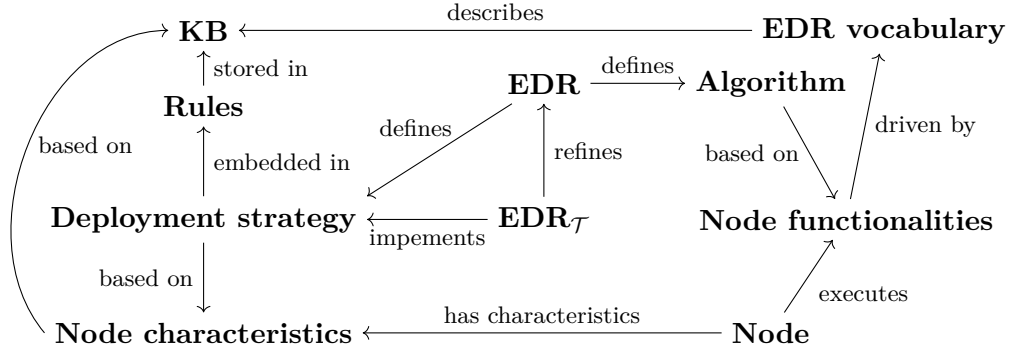
As it has been said in the previous section §5.4, EDR is a **generic** approach to rule deployment among semantic-enabled Fog nodes, agnostic to the criteria according to which rules are propagated in the topology. In order to demonstrate the applicability of EDR, the present section is dedicated to **$EDR_{\mathcal{T}}$, an approach refining EDR by implementing a deployment strategy**. For the sake of clarity, an overview of keywords is introduced in Fig. 5.6.

After introducing the $EDR_{\mathcal{T}}$ core principle in Section §5.5.1, the knowledge required by nodes executing $EDR_{\mathcal{T}}$ is described in Section §5.5.2. How $EDR_{\mathcal{T}}$ is implemented in rule modules is discussed in Section §5.5.3. The behavior of nodes executing $EDR_{\mathcal{T}}$ is detailed in Section §5.5.4, in order to capture the complete deployment process.

5.5.1 Implementing a deployment strategy based on property types with $EDR_{\mathcal{T}}$

The purpose of $EDR_{\mathcal{T}}$ is to bring rules as deep as possible in the topology, in order for them to be processed as soon as possible, while limiting unnecessary message exchanges. Therefore, $EDR_{\mathcal{T}}$ is meant to reduce the delay between the moment observations able to trigger a deduction by a rule are produced by devices, and the moment said deduction is received by the rule originator. Due to the assumed hierarchical nature of the network, the deeper a node is in the topology,

Figure 5.6: Key terms overview

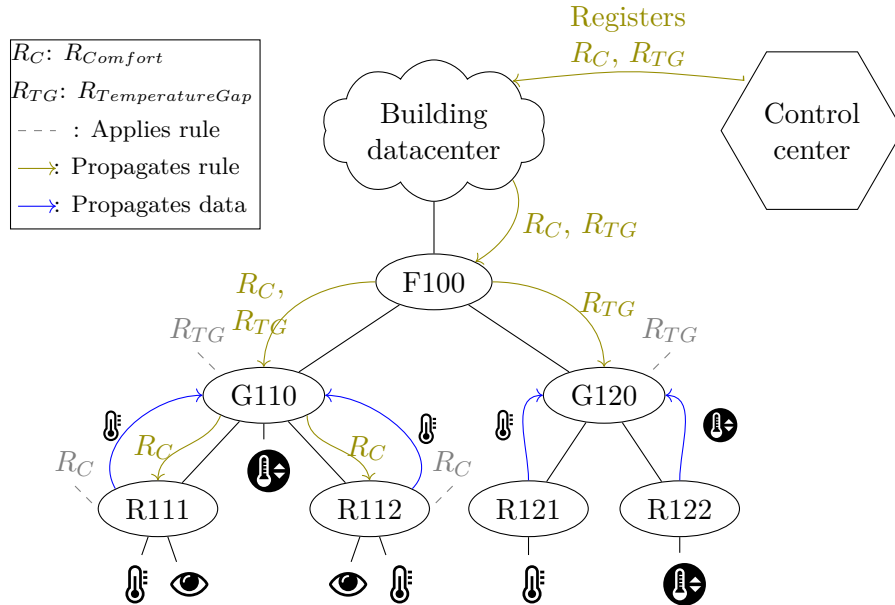


the fewer descendants it has. A node processing a rule deeper in the hierarchy will thus apply said rule less often, on a smaller KB, since it should receive less updates from its descendants. Since reasoning on a smaller KB yields better performances [Maarala 2017], propagating rules as deep as possible among reasoning nodes reduces computing complexity. Therefore, in EDR \mathcal{T} , a node receiving a rule propagates said rule to any of its children able to process it.

EDR \mathcal{T} implements a deployment strategy **driven by the types of properties produced by nodes**. These properties can be either environmental properties captured by sensor observations (*e.g.*, luminosity) or higher level properties deduced by other rules (*e.g.*, comfort). Nodes characteristics capturing these productions are exchanged between neighbors in order to identify the lowest possible node able to process the rule. These characteristics are captured in the rule modules to enable the deployment process. The conditional shape of rule modules is based on both **property types consumed by the rule** and **property types produced by neighboring nodes** to infer the node behavior.

To manipulate these property types in the following, the *body* and *head* notations introduced in Section §5.4.4.2 are extended. We introduce $body_t(r_x) = \{\gamma_1, \dots, \gamma_{m'}\}$ and $head_t(r_x) = \{\delta_1, \dots, \delta_{m'}\}$ where γ_i designates the property type of Γ_i , and δ_j the property type of the deduction Δ_j . It should be noted that not all Γ_i or Δ_j used in the rule are relevant to the EDR \mathcal{T} approach. Let us consider $R_{Comfort}$, an illustrative rule provided in Section §5.3 which body is reproduced here: $Location(l) \wedge Temperature(l, o_1) \wedge 19^\circ C < o_1 < 26^\circ C \wedge Luminosity(l, o_2) \wedge o_2 > 400L \rightarrow Comfortable(l)$. For this rule, $body_t(R_{Comfort}) = \{Luminosity, Temperature\}$, and $head_t(R_{Comfort}) = \{Comfortable\}$. *Location* is a property type that is not considered by the deployment strategy implemented by EDR \mathcal{T} .

The deployment of $R_{Comfort}$ and $R_{TemperatureGap}$ by EDR \mathcal{T} in an extract of the simulation topology is shown on Fig. 5.7. Both rules are submitted by the application to the Cloud node, and are deployed among Fog nodes. Nodes applying the rules (*e.g.*, R111 and R112 for $R_{Comfort}$) directly provide the control center with deductions, which is not represented on the figure for the sake of legibility.

Figure 5.7: Example of $EDR_{\mathcal{T}}$ deployments

5.5.2 Node characteristics at stake in $EDR_{\mathcal{T}}$

5.5.2.1 Node's knowledge on itself

A node n has in its KB information about the property types of the data it produces, denoted $own_productions(n)$. Data produced by node n is either collected by sensors to which n is directly connected, or obtained as deductions when n applies a rule. When a reasoning-enabled node is connected to a sensor, it enriches the raw observation, and propagates the enriched observation on the network, which ensures that the observation is only enriched once. In the topology displayed on Fig. 5.7, the Fog node R111 is connected to a luminosity sensor and enriches its production therefore $own_productions(R111) = \{Temperature, Luminosity\}$. An example of enriched observation is available online⁷. Observations and devices are described in each node's KB using the IoT-O ontology for our experiments (*c.f.* Chapter §6), but the proposed approach does not depend on the ontology used to describe data, as long as the same ontology is used to express the rules and their metadata. The production of observations by node n for a property type ρ_t is denoted $\langle n, edr:producesDataOn, \rho_t \rangle$.

5.5.2.2 Node's knowledge on the topology

A node n knows its parent in the network tree-like hierarchy, noted $Upper(n)$, and its children, referred as $Lower(n)$. On Fig. 5.7, $Lower(G110) = \{R111, R112\}$, and $Upper(G110) = \{F100\}$. The node communicates its characteristics to these

⁷https://w3id.org/laas-iot/rules/observations/enriched_data.ttl

neighbors to support the deployment strategy implemented by EDR_T. Such characteristics include the types of the data produced by the node, as well as the types of data consumed.

Announcing productions: The transmission of rules among nodes organized by EDR_T is driven by the knowledge each node has on the network around itself. Productions are propagated from children to parents, denoted by the triple $\langle \text{edr:producesDataOn}, \text{rdf:type}, \text{edr:ParentAnnouncedProperty} \rangle$. Therefore, when a child node connects to its parent, it includes the triplets denoting its productions in its self-description.

In order to enable the propagation of rules towards nodes that are not direct neighbors, the proxying mechanism introduced in Section §5.4.3.3 is implemented for property types productions: $\langle \text{edr:producesDataOn}, \text{rdf:type}, \text{edr:ParentProxiedProperty} \rangle$. This mechanism makes a node aware of the types of properties produced by any node below its lower nodes while communicating only with its lower nodes, therefore ensuring the locality of its decisions. To illustrate the proxying in more details, let us define $\text{productions}(n) = \text{own_productions}(n) \cup \text{productions}(\text{Lower}(n))$. Node n announces itself to its parent n_{parent} as a producer of $\rho_t^i, \forall \rho_t^i \in \text{productions}(n)$, ρ_t^i being the type of data produced by one of the sensors or lower nodes connected to n . For instance, on Fig. 5.7, $\text{productions}(G110) = \{\text{Temperature}, \text{Luminosity}, \text{Thermostat}\}$, with $\text{own_productions}(G110) = \{\text{Thermostat}\}$. If the parent node n_{parent} was not a producer of the property type ρ_t , it includes a new triplet in its KB $\langle n_{\text{parent}}, \text{edr:producesDataOn}, \rho_t \rangle$, and forwards this triplet to its own parent. If node n_{parent} was already a producer for ρ_t , its capabilities remain unchanged, and the information propagation stops.

Announcing consumptions: As it has been discussed in Section §5.4.3.1, in order to limit unnecessary exchanges, data is exchanged lazily based on the node consumption announcement functionality. A node n has to explicitly advertise its interest for a property type ρ_t to each node belonging to $\text{Lower}(n)$ in order to be notified when new observations are received or new deductions are made. In particular, a node is interested in a property type ρ_t when it is in charge of applying a rule whose body includes ρ_t . Identifying if $\rho_t \in \text{body}_t(r)$ is based on IRI comparisons. The interest of a node n for a property type ρ_t is represented in the KB by the triplet $\langle n_p, \text{edr:isInterestedIn}, \rho_t \rangle$, and $\langle \text{edr:isInterestedIn}, \text{rdf:type}, \text{edr:SomeChildrenAnnouncedProperty} \rangle$. Indeed, when a node applies a rule r and is thus interested in the properties $\rho_t \in \text{head}_t(r)$, it does not necessarily notify this interest to all of its children.

The interest of n for ρ_t is only announced to children of n that are producers of ρ_t . Moreover, if some nodes $n_{\text{child}}^i \in \text{Lower}(n)$ are able to apply the rule r themselves, node n will forward r to n_{child}^i , rather than notifying n_{child}^i of its interest. The details of the rule deployment strategy are provided in Section §5.5.3. In Fig. 5.7,

R111 announced to G110 that it produced *temperature*, and G110 notified R111 of its interest for *temperature* in order to receive observations.

Furthermore, nodes interests are proxied towards children: $\langle \text{edr:isInterestedIn}, \text{rdf:type}, \text{edr:ChildrenProxiedProperty} \rangle$. When a node n receives a message from its parent n_{parent} containing a triple $\langle n_{\text{parent}}, \text{edr:isInterestedIn}, \rho_t \rangle$, n announces to its children $n_{\text{child}} \in \text{Lower}(n)$ that $\langle n, \text{edr:isInterestedIn}, \rho_t \rangle$. Therefore, when one of the children produces a data of type ρ_t , n is notified, and itself propagates the received data to n_{parent} . The knowledge of nodes about their environment is thus limited to their neighborhood, enabling purely local decisions.

5.5.2.3 Exploiting the contextual locality of IoT data

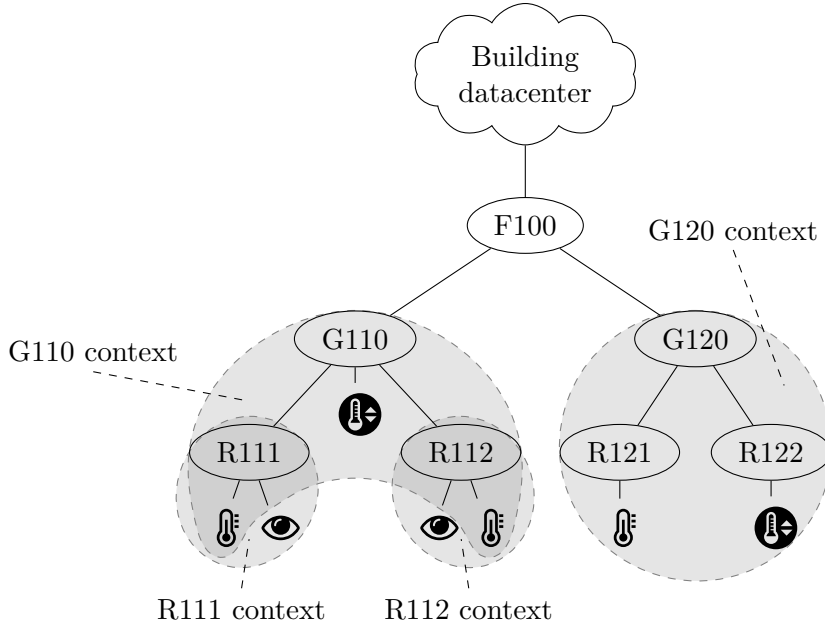
The rule deployment strategy supported by $\text{EDR}_{\mathcal{T}}$ is based on the assumption that the **correlation between pieces of data is embedded in the network topology**. IoT data is strongly bound to a spacio-temporal context [Perera 2014a], and the distribution of Fog nodes reflects the distribution of features observed by sensors. From this hypothesis, it can be inferred that the context of a node is a subset of the context of its parent. To illustrate this claim with R_{Comfort} previously introduced, it means that if it is possible to apply R_{Comfort} with luminosity and temperature observations collected by the same gateway, it is not necessary to compare the same luminosity observations with temperature observations collected elsewhere. IoT data being highly contextual, applications do not necessarily need to reason over a complete KB to get relevant results. EDR is therefore suitable for rules exploiting this context by correlating data sharing an identical context, *e.g.*, the correlation of temperature and luminosity in the context of a single room for R_{Comfort} .

The relation between the spatio-temporal context and the topology is represented in Fig. 5.8, where each gray area represents the context of a Fog node. The assumption we make entails that, since both R111 and R112 contexts contain enough information to process rule R_{Comfort} , the luminosity from R111 context and the temperature from R112 context will never be processed together by R_{Comfort} .

In the case of the G120 context, since neither R121 nor R122 produce the information necessary to process $R_{\text{TemperatureGap}}$, both nodes send their observations to G120. The fact that G120 is the parent of both R121 and R122 is considered a hint that the context of R122 is closer to the context of R121 than, for instance, to the context of R112. The proximity of context is associated to the distance of the closest common ancestor: R121 and R122 share a parent, while the closest common ancestor to R121 and R112 is F100, at a distance of 2 hops from both nodes. Since R121 and R122 are closer to each other than R122 and R112, there is a higher chance for the thermostat observation from R122 to lead to a deduction based on $R_{\text{TemperatureGap}}$ when processed with temperature from R121 rather than R112.

In a similar manner as context proximity, context inclusion is impacted by the hierarchy. A context A is considered included in a context B if the elements of context A are also available in context B. On Fig. 5.8, the G110 context partially

Figure 5.8: Illustration of observations spatio-temporal context



includes the R111 and R112 contexts, since only temperature values are propagated to G110, and not presence values. Since G110 applies $R_{TemperatureGap}$, R111 and R112 provide it with temperature observations, which it processes with its own thermostat value observations.

If, as in our case, the scope of rules is not broader than the context in which they are applied, applying rules deeper in the hierarchy does not impact the completeness of the result. However, if the rules are not adapted to the topology in which they are deployed with $\text{EDR}_{\mathcal{T}}$, some deductions will be inferred in a centralized approach that would be missed when data is processed in a decentralized manner. For instance, let us consider two sensors producing respectively observations of types α_1 and α_2 , connected to the same node n , and a rule r consuming α_1 and α_2 . $\text{EDR}_{\mathcal{T}}$ will eventually deploy r on n , and none of the observations of type α_1 and α_2 produced by n will be processed by r outside of the context of n . This is the intended behavior of $\text{EDR}_{\mathcal{T}}$, but it limits its applications so some types of rules, such as rules performing the aggregation of several values of the same type. For instance, a rule that sums electrical consumptions and compares the total to a fixed value cannot be executed successfully by $\text{EDR}_{\mathcal{T}}$, because its scope will be larger than the contexts in which it will be distributed, that is any node producing electrical consumption observations.

This behavior is adapted to rules supporting deductions for time-sensitive applications, which is the focus of the present contribution, and cannot be applied to aggregation rules, where time series or multiple instances of the same property types are considered. This choice is motivated by the assumption that aggregation rules are more likely to be used in applications supporting long-term reporting and

Listing 5.2: $r_{comfort}^{transfer}$ shape

```

SELECT $this
WHERE {
FILTER NOT EXISTS {
  $this a lmu:Node ;
  edr:producesDataOn adr:Temperature, adr:Luminosity ;
  lmu:hasUpstreamNode [a lmu:HostNode;].
  FILTER NOT EXISTS {
    {ex:officeLightComfortRule edr:transferredTo $this.}
    UNION
    {ex:officeLightComfortRule edr:transferableTo $this.}
  }
}
}
}

```

decision support, where the time constraint is not strong, and thus outside the scope of this contribution. Moreover, such aggregation rules might need to consider data in a global context, which is a purpose achieved by the deployment on Cloud nodes as discussed in Section §4.3. The EDR approach and its refinements (such as $EDR_{\mathcal{T}}$) do not aim at replacing semantic Cloud computing, but seek to complement its capabilities with semantic Fog computing. That is a second reason not to support aggregation rules.

To ensure decidability, only DL-safe rules are considered, and EDR is only suitable for stratified rule sets. Cyclic dependencies between rules are not resolved. When a node applies rule r , it is considered as producer of the $head_t(r)$, and this production information is used for the deployment of any rule r' such as $body_t(r') \cap head_t(r) \neq \emptyset$. However, a non stratified rule set where rules r and r' coexist such that $body_t(r') \subseteq head_t(r)$ and $body_t(r) \subseteq head_t(r')$ cannot be processed successfully by EDR, and neither r nor r' will be propagated or applied.

5.5.3 Implementation of $EDR_{\mathcal{T}}$ in rule modules

The behavior of a node implementing $EDR_{\mathcal{T}}$ is embedded in the modules of $EDR_{\mathcal{T}}$ -compliant rule. For now, these rules are built manually: the property types feature in the rule body and head of the rule are identified when the rule is written, and the modules are built accordingly. The knowledge required for the processing of each module is local to the node performing the reasoning process.

5.5.3.1 Rule Transfer module

The purpose of $EDR_{\mathcal{T}}$ is to **transfer each rule to the lowest possible node in the architecture**, to be applied as early as possible. The propagation of a rule r_x from node n to node n' is considered relevant if $n' \in Lower(n) \wedge body_t(r_x) \subseteq productions(n')$, which brings it closer to sensors. This condition is expressed in Lst. 5.2, an extract of the SHACL shape constituting $r_{comfort}^{transfer}$.

Since it is assumed that rules are initially submitted to the Cloud node, the neighbor-to-neighbor propagation is only considered downwards in the topology. Each node that handles the rule in the deployment process keeps its representation in its KB. Therefore, it is not necessary to re-propagate a rule upwards: if a node ceases to be able to apply a rule, the change should be considered by the activation module of the rule held by its ancestors, as it is detailed in Section §5.5.4.

Incrementally, the rule r will converge toward nodes such that, for any node n of them:

- n can no longer **propagate** r , i.e. $\forall n' \in Lower(n), body_t(r_x) \not\subseteq productions(n')$,
- n is able to **apply** r , i.e. $body_t(r_x) \subset productions(n)$.

These are the nodes able to apply the rule that are the closest to the original data producing: propagating the rule lower in the hierarchy is not necessary. Such a node is represented on Fig. 5.7 with gray dashes connected to $R_{Comfort}$.

5.5.3.2 Activation module

In order to apply a rule r , a node n must be the lowest common ancestor to the producers of property types in the rule body. Such node has a set \mathcal{P} of children (either sensors or other Fog nodes) partially producing the rule head. Individually, none of the children produce all the elements of the rule head, but combined, their productions enable the processing of the rule. It is characterized as such: $\exists \mathcal{P}$, such as $\forall n_c \in \mathcal{P}, \langle n, lmu:hasDownstreamNode, n_c \rangle$ and $\exists \{\rho_t, \rho'_t\} \subseteq body(r), \langle n_c, edr:producesDataOn, \rho_t \rangle$ and $\neg \exists \langle n_c, edr:producesDataOn, \rho'_t \rangle$, and $\forall \rho_t \in body(r), \exists n_c \in \mathcal{P}, \langle n_c, edr:producesDataOn, \rho_t \rangle$. Lst. 5.3 gives a SPARQL implementation of these conditions applied to $r_{comfort}^{activation}$.

If the conditional part of rule r activation module determines that the current node is suitable to apply r , some deductions are inferred. The activity of rule r is made explicit by the triplet $\langle r, edr:isRuleActive, true \rangle$, and the nodes $n' \in \mathcal{P}$ are identified as providers of the data type which r now consumes. The interest of n for the consumption of the nodes $n' \in \mathcal{P}$ is announced, as it is captured by the $\langle ?interest, edr:announceTo, ?partialDataProvider \rangle$ triple. The object of the interest, represented as a reified statement, will be bound to any partial production of the rule head by a child of n . The interest of the rule originator o is also denoted with $\langle o, edr:consumesResult, r \rangle$. These inferences enable both the **rule application** and the **rule result forwarding mechanisms** as described in Section §5.4.3. The SPARQL CONSTRUCT embedded in the SHACL rule for the $r_{comfort}^{activation}$ module is provided in Lst. 5.4. The focus of the SHACL shape, materialized by the `$this` variable, captures the IRI of the node applying the rule in its own KB. It is defined in the SHACL documentation as the only element shared natively between the SHACL conditional shape and the SHACL rule said shape conditions: the `$this` captures the node violating the shape defined in the condition. That is

Listing 5.3: $r_{comfort}^{activation}$ shape

```

SELECT $this
WHERE {
  FILTER NOT EXISTS {
    $this a lmu:HostNode.
    $this lmu:hasDownstreamNode ?temperatureProvider,
                                     ?luminosityProvider.
    ?temperatureProvider edr:producesDataOn adr:Temperature.
    ?luminosityProvider edr:producesDataOn adr:Luminosity.
    FILTER EXISTS {
      $this lmu:hasDownstreamNode ?lowerNode.
      FILTER(
        ?lowerNode = ?luminosityProvider
        || ?lowerNode = ?temperatureProvider
      )
      FILTER NOT EXISTS {
        ?lowerNode edr:producesDataOn adr:Temperature, adr:Luminosity.
      }
    }
  }
}

```

why some elements characterizing the child nodes of the current node need to be recaptured in the WHERE clause of the $r_{comfort}^{activation}$ rule, while the `$this` is already bound to the current node.

5.5.3.3 Result delivery module

In $EDR_{\mathcal{T}}$, the condition of the result delivery module checks if a node expressed interest for the type of deductions yielded by the rule. If there exists a triple $\langle n', edr:interestedIn, \rho_t \rangle$, with n' a remote node and ρ_t an element of the rule r 's head $head(r)$, then the result transfer module infers that $\langle n', edr:consumesResult, r \rangle$.

5.5.4 Unraveling the main steps of $EDR_{\mathcal{T}}$

Nodes executing the EDR algorithm maintain a coherent view of their neighborhood, and deploy rules with respect to this perception of their environment according to the strategy implemented by $EDR_{\mathcal{T}}$. The neighborhood of a node is modified when a new node connects or a known node disconnects, and when the productions or consumptions of a node are modified. The main events impacting the exchanges of a node with its neighbors are therefore: when its capabilities are changed (which includes startup and disconnection), when receiving a new rule, and when receiving a new piece of data. In the following, the behavior of $EDR_{\mathcal{T}}$ for each of these events is described to refine the high-level description given on Fig. 5.5.

When changing capability Sensors are the primary source of data for the network. The data they produce is collected by their reasoning-enabled parent. When

Listing 5.4: $r_{comfort}^{activation}$ rule

```

CONSTRUCT {
  $this edr:isInterestedIn adr:Luminosity, adr:Temperature.
  $this edr:producesDataOn ex:Symptom1.
  ?interest a rdf:Statement;
    rdf:subject $this;
    rdf:predicate edr:isInterestedIn;
    rdf:object ?partialProduction;
    edr:announceTo ?partialDataProvider.
  ex:R1 edr:isRuleActive "true"^^xsd:boolean.
  ?originator edr:consumesResult ex:R1.
} WHERE {
  $this a lmu:HostNode.
  {
    $this lmu:hasDownstreamNode ?partialDataProvider.
    ?partialDataProvider edr:producesDataOn ?partialProduction.
    FILTER NOT EXISTS {
      ?partialDataProvider edr:producesDataOn
        adr:Luminosity,
        adr:Temperature.
    }
  }
  UNION {
    ex:R1 edr:isRuleActivable "true"^^xsd:boolean.
  }
  ex:R1 edr:ruleOriginatedFrom ?originator.
  OPTIONAL{ex:R1 edr:isRuleActive "false"^^xsd:boolean.}
  BIND(STRAFTER(str(?partialProduction), "#") AS ?productionName)
  BIND(URI(CONCAT(str($this), ?productionName, "Interest"))
    AS ?interest)
}

```

semantic computing-enabled nodes start, they try to connect to their sensors children of which they have *a priori* knowledge. How nodes discover and gather information about sensors can be a process tightly related to the underlying technology, or hard-coded in the node KB.

Nodes connected to sensors announce the property types they produce to their parent node, according to the announcement functionality captured in the triple $\langle \text{edr:producesDataOn}, \text{rdf:type}, \text{edr:ParentAnnouncedProperty} \rangle$. As explained in Section §5.5.2.2, nodes propagate production information by proxying their children productions. Similarly, when a sensor or a lower node providing data of type ρ_i to node n disconnects, n announces its updated capabilities if they have been transformed, *i.e.* if the disconnected node was the sole producer of ρ_i .

In the case when the node already held some rules, their placement might need to be updated according to the new topology denoted by the received message. In order to adjust the rule deployment accordingly, rule modules dedicated to such deployment, namely application, transfer and delivery modules, are activated, processed in a reasoning step, before being deactivated again as detailed in Fig. 5.5. The deductions yielded by this reasoning step, based on the *edr* vocabulary, are used to control the node behavior as described previously. The use of these modules is similar when a new rule is received, as it is described in the next section. The propagation of $r_{TemperatureGap}$ in the illustrative deployment provided in Fig. 5.7 is represented as a sequence diagram on Fig. 5.9.

When receiving a rule When node n receives a new rule r , n evaluates whether it can apply r directly, and/or if it should propagate r to some of its children by performing a reasoning step with all modules of r activated. Based on the deductions produced by this reasoning step, some node functionalities are activated if necessary:

- If the rule r is applicable by the current node, the productions of n are updated by $r_{comfort}^{activation}$. n notifies its parent of its new productions, *i.e.* the head of r . Being able to produce the deductions of a rule is processed like a capability change, described in the previous section. If the applicability of rule r is enabled by the productions of some children of node n , the interest of n for their productions has been added in the KB, as well as the necessity for their notification of such interest. Node n thus notifies these children of its interest for these properties.
- The rule r is propagated to child nodes marked suitable by the rule transfer module. Local metadata is added to rule r in order to keep track of the lower nodes to which it has been transmitted with the predicate *edr:rule-TransmittedTo*. Such metadata is not added by the rule transfer module, but by the node after the completion of the propagation to the target.

When receiving new data The propagation of a new piece of data is represented as a sequence diagram on Fig. 5.10. Different kinds of data can be received by node

Figure 5.9: Propagation of $r_{TemperatureGap}$

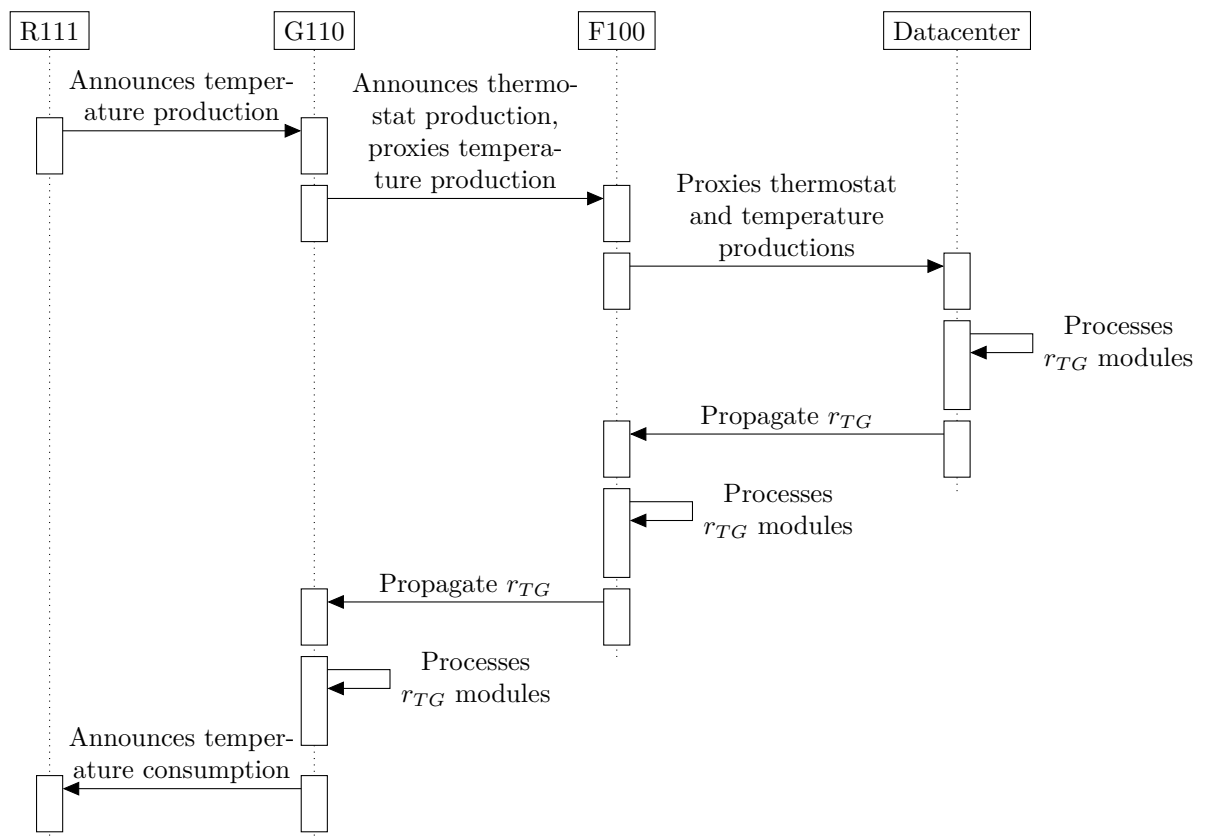
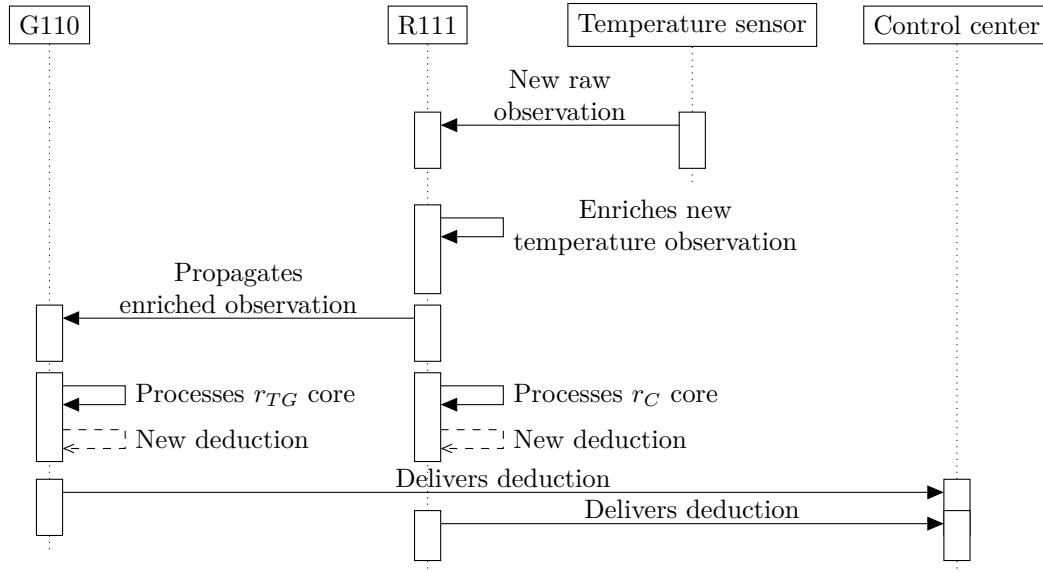


Figure 5.10: Propagation of a temperature observation



n :

- raw observations directly produced by a sensor connected to n , as for the observation sent by the sensor to R111 on Fig. 5.10
- enriched observation or deduction sent to n by node $n_c \in Lower(n)$, as for the observation sent by R111 to G110 on Fig. 5.10

If the received observation is raw, node n enriches it by annotating it with an ontology before its processing as a new enriched observation. If the piece of data is either an enriched observation or a deduction, it is directly integrated to its KB and processed.

The data, of property type ρ_i , is in the first place sent to $Upper(n)$ if it is a consumer of ρ_i . Then, node n checks if new deductions can be obtained by applying the rules it has marked up as active. When receiving new data, a node does not need to activate the rule modules for activation, transfer or delivery: only the core of the rule is relevant. If the rule body matches the KB of node n , and postconditions of type δ_j are deduced, these deductions are propagated to $Upper(n)$ if it is consumer of δ_j . Since rules are applied on the local KB of node n , there is no impact of data distribution on reasoning complexity. A new reasoning loop is simply applied each time new data is received. The deductions yielded by rule r are also **directly** sent to r 's originator(s). Therefore, applications are notified continuously by the nodes as those nodes apply the rules, instead of being notified by a restricted set of central nodes.

5.6 Conclusion

In this chapter, we proposed EDR, a generic approach for dynamically distributed rule-based reasoning, which exploits the properties of Fog architectures to achieve decentralization.

Many existing approaches to rule-based reasoning for the SWoT perform computation on Cloud nodes only, potentially leading to a centralized bottleneck, and by design creating network communication overhead. In order to tackle these issues, decentralized approaches have been proposed in the literature, taking advantage of the Fog computing paradigm. In such cases, computation is disseminated among Fog nodes in order to be brought closer to the IoT devices producing the data. However, these distributed reasoning approaches do not discuss rule placement. Some studies propose interesting optimizations, but they focus on one node, and do not consider the myriad of nodes constituting a Fog infrastructure. In other studies, the rule placement is static: it is either computed at design time, or all the nodes execute the same set of rules.

With the contributions described in this chapter, we aimed to address these shortcomings to enable cooperative semantic computing associating remote powerful nodes providing stability, and local limited and opportunistically available resources associated to semantic Cloud and Fog computing respectively. The elaborated solutions leverages the Cloud and Fog paradigms complementarity to implement an efficient deployment and propagation approach for data, rules and deductions respectively.

As contribution II.A we introduced **EDR, a generic approach to dynamically distributed rule-based reasoning**, based on modular SHACL rules. The execution by Fog nodes of **core EDR functionalities** is **controlled via a dedicated vocabulary** describing knowledge in each node's KB. This vocabulary is used by rule modules to **implement deployment strategies** enabling the propagation of rules neighbor-to-neighbor across the Fog tier of the Cloud-Fog-Device pattern. Rule **deployment strategies aim at optimizing rule placement for customizable criteria**, such as response time or energy consumption, based on the knowledge stored in each node's KB. Such knowledge include a description of its neighbors, the current state of the environment based on sensor observations, and background knowledge. Overall, EDR enables, in a purely **decentralized and emergent** manner, the **deployment of rule**, the **propagation of data** and the **delivery of deductions** inferred when applying the rules once they have been deployed. EDR has been accepted for publication in [Seydoux 2018c].

In order to enforce its genericity, EDR itself is made agnostic to individual deployment strategies. Therefore, it has to be refined by injecting **rules embedding their own deployment strategy**, selected according to application-level requirements. To this end, we proposed $\text{EDR}_{\mathcal{T}}$ (contribution II.B), an EDR refinement implementing a deployment strategy dedicated to **reducing delays** for transmitting deductions to applications. $\text{EDR}_{\mathcal{T}}$ aims at deploying rules on Fog nodes as close as possible to sensors, while avoiding unnecessary computation. Rules are thus

propagated toward sensors producing the **type of data** they consume, **as deep as possible** in the topology. The propagation stops when the rule is deployed on the Fog node being the closest common ancestor to these sensors in the topology. To enforce the locality of decisions, node capabilities are announced through the network thanks to a proxying mechanism, where data productions and consumptions are propagated. An early implementation of $\text{EDR}_{\mathcal{T}}$ has been proposed in [Seydoux 2018b], and a deeper description has been accepted for publication in [Seydoux 2018a].

The next chapter is dedicated to performances measurement in order to support our claims regarding the scalability of $\text{EDR}_{\mathcal{T}}$.

Experimentations

Contents

6.1	Deductions delivery mechanisms	138
6.2	Experimental setup and implementation	141
6.2.1	Hardware setup	141
6.2.2	Software setup	141
6.2.3	Measured results	142
6.3	Initial implementation	143
6.3.1	Smart building use case details	143
6.3.2	Impact of distribution on responsiveness	144
6.3.3	Scalability of the proposed approach	147
6.4	EDR_T implementation in a smart factory use case	148
6.4.1	Use case details	149
6.4.2	Impact of distribution on responsiveness	151
6.4.3	Scalability of the proposed approach	154
6.5	Conclusion	157

EDR being a generic approach, it cannot be subjected to a quantitative evaluation by itself: it must be refined by a concrete approach implementing a deployment strategy. Therefore, the evaluations presented in this section are dedicated to EDR_T, refining EDR with a a deployment strategy aiming at reducing the deduction delivery delay.

In order to compare the proposed contribution to baselines qualified by diverse properties, different delivery mechanisms are introduced in Section §6.1. By default, EDR_T delivers deductions directly to applications. The proposed alternative delivery mechanisms implement variations of this approach, by propagating deliveries differently across the network. A centralized deduction baseline is also introduced.

The setup in which the evaluations were performed is described in Section §6.2, along with the references to the code used for running the experiments. Two sets of experimentations are then introduced, assessing two different implementations of EDR_T, respectively in Section §6.3 and Section §6.4. These two groups of experimentations have been initially published in [Seydoux 2018b], and accepted for publication in [Seydoux 2018a] and [Seydoux 2018c].

6.1 Deductions delivery mechanisms

The purpose of the evaluations presented in this chapter is to compare the performances of centralized Cloud-based and decentralized Fog-based approaches to reasoning. The contribution we propose, $\text{EDR}_{\mathcal{T}}$, has been described in the previous chapter §5. It aims at distributing reasoning among Fog nodes in order to perform computation as close as possible to the sensors producing observations. The baseline to which $\text{EDR}_{\mathcal{T}}$ should be compared is a centralized approach, where raw data is sent up to a Cloud node to be processed by rules. Since the propagation of rules for semantic Fog computing is performed neighbor-to-neighbor, it seems logical that raw data is propagated in the same way back to the Cloud node. However, such comparison would be biased by the necessity for each piece of data to transit through multiple hops from Fog to Cloud nodes. In order to limit the impact of transfer time, and focus on processing time, new hypotheses are considered: in some configurations, Fog nodes will deliver deductions to Cloud nodes, instead of communicating directly with applications. Similarly, for centralized processing, Fog nodes should be able to deliver raw data to Cloud nodes, instead of an indirect propagation. These different configurations are referred to as “Deductions delivery mechanisms”.

Unlike rule deployment strategies, deductions delivery mechanisms are decorrelated from the rules: they are variations of the “Deduction delivery” functionality described in Section §5.4.3.1. Therefore, the propagation of rules, the deductions they yielded and data is described as intended according to ad-hoc strategies (here, $\text{EDR}_{\mathcal{T}}$) through the EDR vocabulary, but for experimental purpose this propagation can be altered at the node level, preventing rule deployment or rerouting deduction delivery. Five deduction delivery mechanisms are compared in our experiments:

- **Cloud-Indirect-Raw (CIR)** is the baseline approach: the rules are only kept in the top Cloud node, and raw observations are forwarded neighbor-to-neighbor from the nodes that collect them toward the central node. The Cloud then delivers deductions to applications. Applications are notified by the Cloud node, and not by Fog nodes, in all delivery mechanisms except the last one.
- **Cloud-Direct-Raw (CDR)** is also an approach where rules are not deployed, and only processed in the central Cloud node. In this configuration, the observation producers directly send raw observations to the Cloud node, where they are used for rule-based deductions. Such delivery mechanism enables to measure the impact of transfer time on deduction delay when centralizing raw data for processing. To implement this configuration, the interest proxying mechanism presented in Section §5.5.2.2 is altered. Nodes that are not the upper node in the hierarchy propagate the interests they receive without proxying them.
- **Cloud-Indirect-Processed (CIP)** is a hybrid delivery mechanism: rules

are deployed among Fog nodes according to $\text{EDR}_{\mathcal{T}}$, and deductions are propagated neighbor-to-neighbor towards the Cloud node before being delivered to applications. CIP mirrors the delivery mechanism of CIR, with a decentralized reasoning. The purpose of CIP is to measure the performance gain when distributing reasoning even when communication is only possible neighbor-to-neighbor in the Fog infrastructure. To modify the result delivery behavior, whenever a node propagates a rule, it declares itself as the originator of said rule instead of the previously registered originator. Processing rules based on semantic Fog computing means that the propagation of observations is limited to the Fog nodes applying rules consuming such observations, instead of going all the way up the Cloud node.

- **Cloud-Direct-Processed (CDP)** is another hybrid mechanism where rules are processed by Fog nodes, but deductions are delivered directly to the Cloud node instead of applications. It is the Cloud node that performs the delivery to applications. In this case, the purpose is to measure the impact of centralized delivery in a decentralized reasoning context. To implement CDP, when forwarding a rule it has received, the Cloud node declares itself as the originator instead of the application. Deductions can also be propagated among Fog nodes if a node explicitly expressed its interest.
- **Application-Direct-Processed (ADP)** is the purely decentralized strategy that we propose for $\text{EDR}_{\mathcal{T}}$, where rules are processed based on semantic Fog computing and deductions are delivered directly to applications that submitted the rules. In this case only, a deduction that has been inferred in the network will not be hosted by the Cloud node before being delivered.

The characteristics of the different delivery mechanisms are summarized in Tab. 6.1, where their important features are highlighted:

- whether rules are propagated among Fog nodes or not,
- whether deductions are propagated neighbor-to-neighbor or directly delivered,
- whether Fog nodes communicate with the Cloud node or directly with applications.

All these characteristics are illustrated in an example on Fig. 6.1, where the propagation of raw data and deductions according to the different delivery mechanisms is represented. In the case of deductions delivery, it is assumed for the sake of clarity that deductions are made in the lowest Fog nodes. The considered topology comes from the smart building use case described in Section §5.3, with a hierarchical deployment of nodes in a floor, galleries and rooms, respectively designated as FXXX, GXXX and RXXX on the figure. The manipulation of the EDR behavior by implementing different delivery mechanisms enables the comparison of centralized (CIR and CDR) and distributed approaches (CIP, CDP, ADP), and the comparison of approaches based on direct (CDP, CDR) and indirect (CIR, CIP) communication with the Cloud node.

Table 6.1: Delivery mechanisms summary

Approach	Rules propagation	Neighbor-to-Neighbor content delivery	Fog-App communication
CIR	✗	For data ✓	✗
CDR	✗	For data ✗	✗
CIP	✓	For deductions ✓	✗
CDP	✓	For deductions ✗	✗
ADP	✓	For deductions ✗	✓

Figure 6.1: Delivery mechanisms

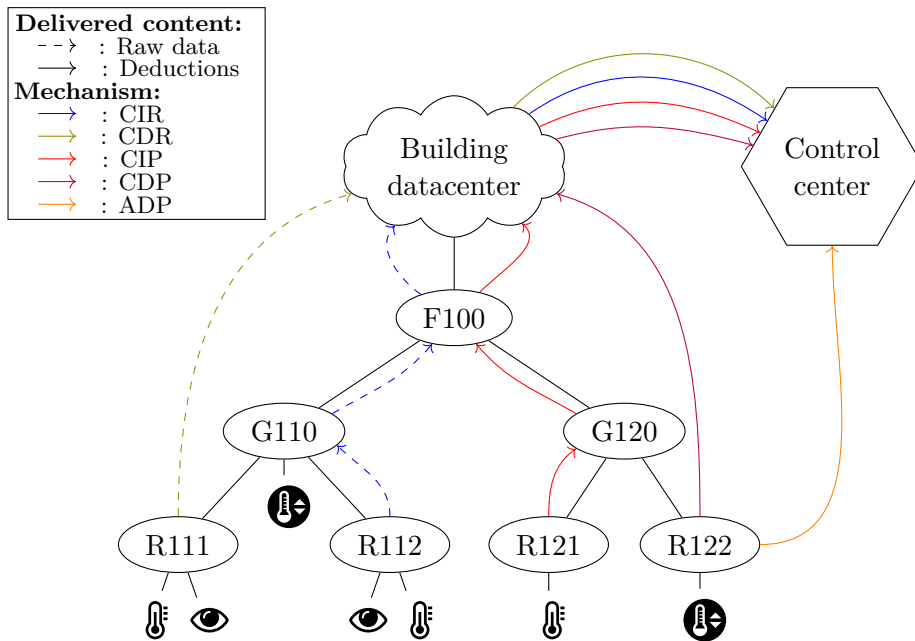


Table 6.2: Experimental setup

	RAM	Cores	CPU
Server	32GB	32	3.0GHz
Laptop	16GB	8	2.6GHz
RPi 3	1GB	4	1.4GHz
RPi 2	1GB	4	900MHz

6.2 Experimental setup and implementation

6.2.1 Hardware setup

In order to assess the distributed nature of the approach, and its suitability for constrained Fog nodes, the experimental setup includes a Raspberry Pi 2 and a Raspberry Pi 3, a laptop and a server, described in Tab. 6.2.

In order to measure the tradeoff between decentralization and the loss of computing power when reasoning on Fog nodes, experiments are run twice, in two different environments:

- In the first case, the complete topology is emulated on the same server, each node being run as an individual process. This environment is referred to as “**single-host execution**”. Such execution environment eases tests.
- In the second case, the topology is distributed across different machines listed on Tab. 6.2. This environment is referred to as “**multi-host execution**”. Such execution environment is more realistic than single-host execution, since it includes constrained nodes. However, the feasibility of large scale experimentations on such decentralized environments is limited, since it requires multiple machines. The necessity to run the experiments on multiple machines at the same time also creates technical issues making the testing process more complex.

6.2.2 Software setup

The topologies introduced in the use cases are simulated for the experimentations. Simulated nodes are organized in a tree-like hierarchy, with a Cloud node at the root, sensors at the leaves, and Fog nodes in between. Each sensor pushes a random observation to its parent every two seconds. Each physical machine running the simulation hosts multiple virtual nodes, composed of an HTTP server, a KB, a SPARQL engine, and a code base¹.

Experiments are run by simulating a building setup with sensors generating raw data. To enable the deployment on multiple machines, each node is implemented as a standalone Java process, and inter-process communication is performed over HTTP. To enable scalable experiments, sensors are implemented as multiple threads

¹The code is available at <https://framagit.org/nseydoux/edr>

of one process, otherwise the RAM overhead for having an HTTP stack deployed for each sensor prevents from deploying large topologies. Therefore, to enable replaying exactly the same sequence of observations, it would have been necessary to synchronize more than 400 threads since the order in which observations are received impacts the obtained result. We were not able to ensure such synchronization without reducing the rate at which observations are produced by sensors. That is why all the results were collected on simulated topologies.

6.2.3 Measured results

Two aspects of EDR have been evaluated:

- the validity of our hypothesis, namely that the distribution of rules increases responsiveness,
- the scalability of the proposed approach

To measure the responsiveness of applications enabled by EDR, the **delay between the moment observations are captured by sensors and the delivery of the deduction** these observation triggered is measured. Precisely, the delay for the processing of a rule is characterized as the time difference between the moment when the most recent data used in the body of the rule is produced, and the moment when the rule head is deduced. A dedicated timestamp is associated to each observation once it has been enriched, in order to avoid any impact of the enrichment process on the measure. For instance, if a luminosity observation observed at t_1 and a temperature observation observed at t_2 match $r_{comfort}$ and trigger a deduction that is delivered to the application at t_3 , the delivery delay for this particular deduction will be $t_3 - \max(t_1, t_2)$. The clock of all the machines used for the experiment are synchronized to a local server using Network Time Protocol (NTP)², in order to ensure a minimal time difference between the different nodes.

Experimental measures showed that, for each simulation, the number of deductions is consistent between centralized and distributed approaches: **there is no knowledge loss when applying EDR_T under our assumptions** of bound between the Fog topology and the correlation between data.

In order to analyze closely the cause for the increased delay, the journey of a message has been broken down in discrete timestamped events. The first event related to a message is its construction, either by enrichment of an observation or by achieving a deduction. In order to be propagated in the network, a message might be sent from a node n to another node n' , which is identified as two events: the sending from node n , and the reception by node n' .

Multiple hops are registered, from the first node responsible for the message creation toward any node that is interested in the message content for deduction. When a message is received by a node n , n starts a reasoning step where it tries to make new deductions based on the rules in its knowledge base. Events are logged

²<http://www.ntp.org/>

at the beginning and at the end of reasoning. In order to detail the delay for each deduction, the journey of the most recent observation leading to the deduction is reconstructed. This journey is built by identifying all consecutive events related to the piece of data leading to the deduction, from its initial enrichment to its processing leading to the deduction, and the delivery of said deduction to the application.

Three components of delay have been identified:

- **Transfer delays**, measured between the emission and the reception of a message. This delay is both impacted by the quality of the network link between two nodes, but also by the processing speed of the recipient: the transfer is considered completed when the recipient declares the reception at the software level, and it is not measured at the network layer. When the message is transferred through multiple hops, the delays are summed.
- **Reasoning delays**, measured between the beginning and the end of a reasoning step. Reasoning delays are summed if the same message is processed with different rules across the topology.
- **Idle delays**, measured between the reception of a message and its processing, or between the reasoning step and the propagation of deductions.

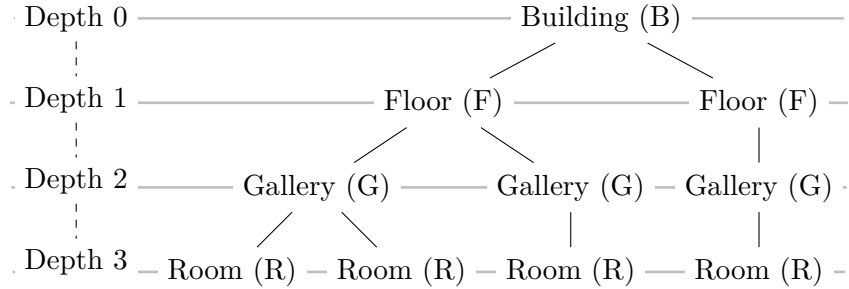
6.3 Initial implementation

The results presented in this section have been collected on an initial implementation of $\text{EDR}_{\mathcal{T}}$, initially introduced in [Seydoux 2018b] and detailed in [Seydoux 2018a]. This first implementation focuses on demonstrating the performance gain with $\text{EDR}_{\mathcal{T}}$ compared to a centralized baseline, and not on the genericity of EDR. Therefore, the propagation strategy was directly baked into the generic algorithm, and no distinction is made between EDR part and $\text{EDR}_{\mathcal{T}}$. The vocabulary used to manage the rules is directly attached to them in the form of metadata, instead of being embedded in SHACL modules. This non-genericity allows to consider simpler, non-modular rules. Moreover, since this initial implementation is a proof-of-concept for $\text{EDR}_{\mathcal{T}}$, only the extreme mechanisms, CIR and ADP, are compared. Due to the non-modularity of this initial work, implementing intermediate mechanisms would require more development. The results obtained with this first iteration were motivating for the second, more generic, implementation discussed in Section §6.4.

6.3.1 Smart building use case details

This use case is the implementation of the illustrative use case already introduced in Section §5.3. The reference architecture used for the simulation is depicted on Fig. 6.2. The root node at depth 0 is the Cloud server while other nodes are Fog nodes. Sensors, not represented on the figure, may be connected under any node.

Figure 6.2: Reference simulation architecture



In this architecture, some rules characterizing the behavior of the smart building are deployed, listed in Tab. 6.3. Rules R1 to R4 are inspired from actual rules implemented by the BAS deployed in ADREAM, and rules R5 to R14 are abstract rules, meant to introduce variability in the use case but not yielding deductions connected to an actual use case.

6.3.2 Impact of distribution on responsiveness

6.3.2.1 Experimental topologies

To measure how distribution impacts responsiveness, four topologies were distinguished, labeled d1 to d4 and further on simply denoted d^* . Each of these topologies is constituted of 47 identical nodes, and processes data according to four rules, r_1 to r_4 . The difference between the four d^* topologies is the location of sensors, as depicted in Fig. 6.3. Sensors producing data of the type γ_1 are directly attached to the top node in d1, while they are attached to its children in d2. Since $body_t(r_1) = \{\gamma_1, \gamma_4\}$, r_1 is applied at a maximum depth of 1 in d1, but is propagated to nodes of depth 2 in d2, hence a “more decentralized” execution is performed in d2 than in d1. Rule execution depths are given in Tab. 6.4: in d4, all sensors are connected to leaf nodes, and the distribution is maximal.

A factor to be considered is that this experiment is entirely run on the server which characteristics are given in Tab. 6.2: no constrained nodes are included in this evaluation. It would be wrong to assume that the decentralization towards nodes of lesser computing power has no impact on the results, that is why this question is addressed in Section §6.4.2.

6.3.2.2 Results

Fig. 6.4a and 6.4b summarize the aggregated delivery delay measures for the four rules applied in d^* topologies. The data shows that the delay remains stable for the four topologies with a centralized approach (Fig. 6.4a), which is our baseline. Since all computations are performed in the upper node in the centralized case, it is coherent that the distribution of sensor nodes in the network has little impact the delay. In any d^* distribution, when processing rules at depth 0 with a centralized

Table 6.3: Building management rules

Rule ID	Rule core
R1: Too hot	$Location(?l) \wedge Temperature(?l, ?o_1) \wedge ?o_1 > 25.0$ $\rightarrow HighTemperature(?l)$
R2: Comfortable room	$Location(?l) \wedge Temperature(?l, ?o_1) \wedge ?o_1 > 20.0$ $\wedge Luminosity(?l, ?o_2) \wedge ?o_2 > 150L \rightarrow ComfortablePlace(?l)$
R3: Uncomfortable room	$Location(?l) \wedge Temperature(?l, ?o_1) \wedge ?o_1 < 16.0$ $\wedge Luminosity(?l, ?o_2) \wedge ?o_2 < 120L \wedge Humidity(?l, ?o_3)$ $\wedge ?o_3 > 50.0 \rightarrow uncomfortablePlace(?l)$
R4: Costly temperature gap	$Location(?l) \wedge Temperature(?l, ?o_1)$ $\wedge TemperatureRequest(?l, ?o_2) \wedge PowerConsumption(?o_3)$ $\wedge (?o_2 - ?o_1) > 5 \wedge ?o_3 > 300.0 \rightarrow CostlyTemperatureGap(?l)$
R5	$Location(?l) \wedge Temperature(?l, ?o_1) \wedge ?o_1 \leq 25$ $\wedge Luminosity(?l, ?o_2) \wedge ?o_2 < 800 \rightarrow Symptom5(?l)$
R6	$Location(?l) \wedge Temperature(?l, ?o_1) \wedge ?o_1 \leq 25$ $\wedge Luminosity(?l, ?o_2) \wedge ?o_2 > 50 \wedge Noise(?l, ?o_3) \wedge ?o_3 > 30$ $\rightarrow Symptom6(?l)$
R7	$Location(?l) \wedge Temperature(?l, ?o_1) \wedge ?o_1 > 20$ $\wedge Luminosity(?l, ?o_2) \wedge ?o_2 > 150 \wedge Humidity(?l, ?o_3)$ $\wedge ?o_3 > 35 \rightarrow Symptom7(?l)$
R8	$Location(?l) \wedge Temperature(?l, ?o_1) \wedge ?o_1 \leq 18$ $\wedge Luminosity(?l, ?o_2) \wedge ?o_2 < 200 \wedge Noise(?l, ?o_3) \wedge ?o_3 > 30$ $\wedge Presence(?l, ?o_4) \wedge ?o_4 = True \rightarrow Symptom8(?l)$
R9	$Location(?l) \wedge Temperature(?l, ?o_1)$ $\wedge TemperatureRequest(?l, ?o_2) \wedge ?o_1 \leq ?o_2 \rightarrow Symptom9(?l)$
R10	$Location(?l) \wedge Temperature(?l, ?o_1)$ $\wedge TemperatureRequest(?l, ?o_2) \wedge ?o_1 \geq ?o_2$ $\wedge Humidity(?l, ?o_3) \wedge ?o_3 > 35 \rightarrow Symptom10(?l)$
R11	$Location(?l) \wedge Temperature(?l, ?o_1)$ $\wedge TemperatureRequest(?l, ?o_2) \wedge ?o_1 < ?o_2$ $\wedge PowerProduction(?l, ?o_3)$ $\wedge PowerRedistribution(?l, ?o_4) \wedge ?o_4 > ?o_3 \rightarrow Symptom11(?l)$
R12	$Location(?l) \wedge Presence(?l, ?o_1) \wedge ?o_1 = True$ $\wedge PowerProduction(?l, ?o_2) \wedge ?o_2 > 300 \rightarrow Symptom12(?l)$

Table 6.4: Depth of rule processing for d* topologies

	R1	R2	R3	R4
d1	1	2	3	4
d2	2	2	3	4
d3	3	3	3	4
d4	4	4	4	4

Figure 6.3: d^* topologies

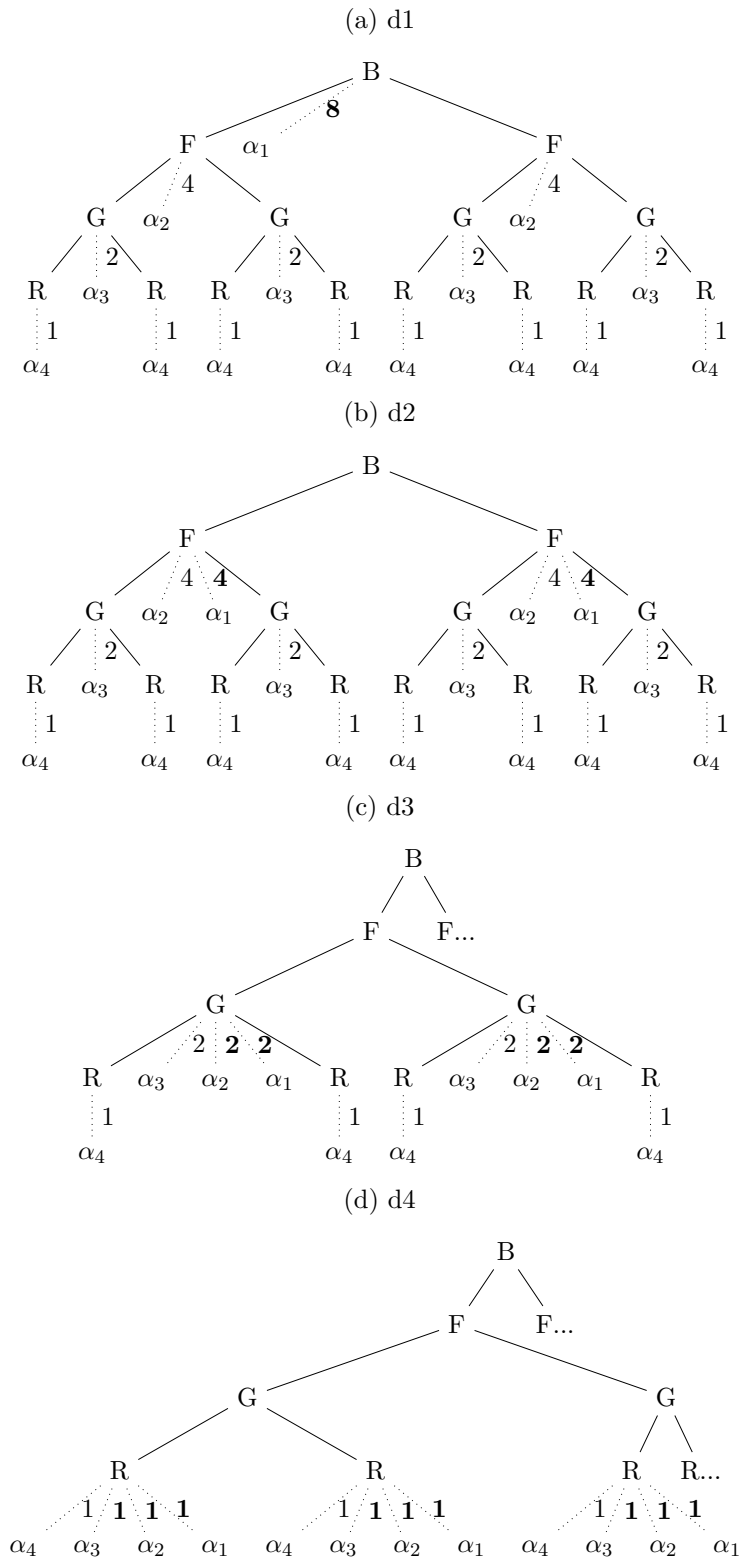
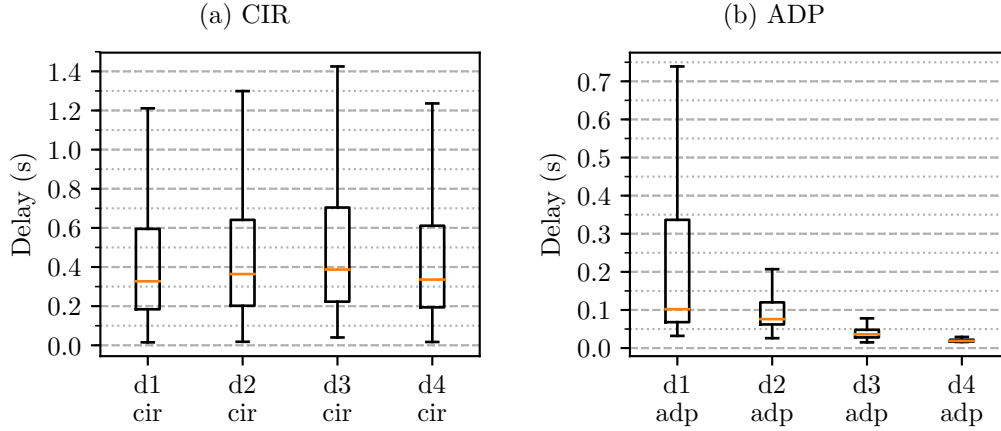


Figure 6.4: Impact of distribution on delays

Table 6.5: s^* topologies

Topology	s1	s2	s3	s4
Nodes	27	41	65	78

approach, at least one sensor is situated at depth 4. In d4, all sensors are situated at depth 4, but even in d1 for rule R1, a sensor is located at depth 4, maintaining a limiting factor for transfer time.

With the decentralized approach (Fig. 6.4b) based on EDR, as expected, the augmentation of the depth at which rules are applied is correlated with the reduction of the delivery delay. Due to the tree-like nature of the network, the deeper a rule can be processed, the more distributed its processing is. Our hypothesis that bringing rules closer to data-producing sensors reduces deduction delivery delay is therefore supported by experimental evidence. The decentralized approach also outperforms the centralized one.

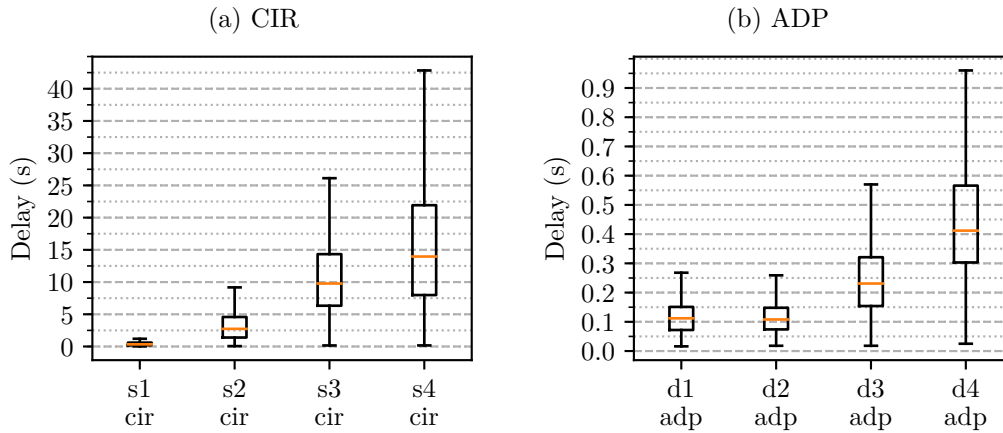
It is clear that not all rules can be processed at the very edge of the network, and that the case depicted in d4 is both purely theoretical and ideal for $EDR_{\mathcal{T}}$. This experiment is designed to show that the gain in quality of service is correlated with the possibility to distribute rules, even at a medium scale where the performance of a centralized approach are still viable compared to a decentralized one. As it is discussed in Section §6.3.3, centralized design leads to huge delay increases with scaling topologies.

6.3.3 Scalability of the proposed approach

6.3.3.1 Experimental topologies

The scalability of the proposed approach is assessed by measuring deduction delivery delay in four topologies, s1 to s4 (denoted s^*), with an increasing number of nodes stated in Tab. 6.5. Nine rules are deployed on s^* topologies. Compared to previous d^* topologies, the depth at which rules can be applied in s^* is constant,

Figure 6.5: Scalability measures



as sensors' depth is fixed. The number of nodes is increased by cloning branches in the topologies. This experiment has been run on a mixed setup: some Fog nodes are emulated by the server, and some other are actually deployed on Raspberry Pis.

6.3.3.2 Results

Fig. 6.5a shows how the delivery delay time increases with the number of nodes in the baseline approach. The median delivery delay increases progressively with the growth of the topology, up to values that are unacceptable for an application requiring responsiveness. The centralized approach is not scalable.

On Fig. 6.5b, the delivery delays measured when EDR is applied remains stable between s1 and s2, before increasing for d3 and d4. However, the observed increase is much less important than in the centralized delivery mechanism. The proposed decentralized mechanism, even if it creates a loss of QoS with the increase of the number of nodes, is much more scalable than its centralized counterpart.

These initial experimentations provide promising preliminary results while assessing the properties of $\text{EDR}_{\mathcal{T}}$ regarding scalability and rule distribution. However, as it has been stated at the beginning of the section, this first implementation was an initial design where EDR and $\text{EDR}_{\mathcal{T}}$ are not differentiated, and where rules are not expressed using the full expressivity of SHACL. This initial design was easier to design, and provided an interesting proof of concept, but does not support the genericity initially part of the EDR approach. In Section §6.4, the results of the proof of concept are confirmed with a complete EDR implementation.

6.4 $\text{EDR}_{\mathcal{T}}$ implementation in a smart factory use case

The results considered in this section have been obtained by executing the $\text{EDR}_{\mathcal{T}}$ approach as described in Chapter §5: it is based on modular SHACL rules, and

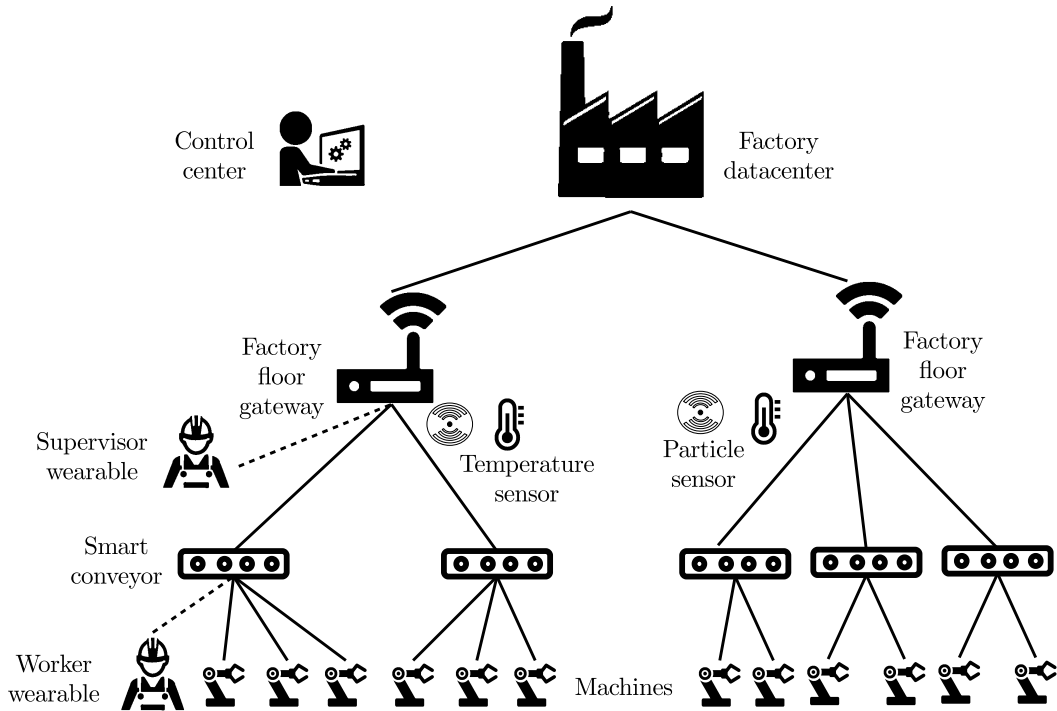


Figure 6.6: Fog-enabled smart factory

the propagation strategy is separate from the core algorithm. In order to show the versatility of the approach, a different use case is considered, as detailed in Section §6.4.1. Distribution and scalability are then discussed, respectively in Section §6.4.2 and §6.4.3.

6.4.1 Use case details

Let us consider a production plant divided into two floors, processing different kind of products. These floors are modular: the structure described thereafter is subject to change in order to adapt to new productions. Each floor is equipped with conveyor belts carrying products from machine to machine for transformation. Devices are organized hierarchically: machines are connected to conveyors that are connected to the floor gateway, that collects and delivers data to the factory datacenter. The factory is equipped with sensors in order to ensure the safety of workers: each floor is equipped with presence, luminosity particle and temperature sensors, and the workers are equipped with wearables that automatically communicate in BLE³ with nearby conveyors. Observations from the different sensors are used in order to identify potentially harmful situations, and then notify the control center, where actions can be taken remotely. Unsafe situations are described with deduction rules, based on the semantic description of observations and of the environment. Examples of rules include “the activation of a machine creating sparks in an at-

³https://en.wikipedia.org/wiki/Bluetooth_Low_Energy

Rule ID	Rule core
R1: Low Machine Visibility	$Location(?l) \wedge Presence(?l, ?o_1) \wedge ?o_1 = True$ $\wedge Luminosity(?l, ?o_2) \wedge ?o_2 < 300L \wedge Machine(?m)$ $\wedge Activity(?m, ?o_3) \wedge ?o_3 = True \wedge locatedIn(?m, ?l)$ $\rightarrow LowMachineVisibility(?m)$
R2: Low Conveyor Visibility	$Location(?l) \wedge Presence(?l, ?o_1) \wedge ?o_1 = True$ $\wedge Luminosity(?l, ?o_2) \wedge ?o_2 < 300L \wedge Conveyor(?c)$ $\wedge Activity(?c, ?o_3) \wedge ?o_3 = True \wedge locatedIn(?c, ?l)$ $\rightarrow LowConveyorVisibility(?c)$
R3: No supervision	$Location(?l) \wedge Presence(?l, ?o_1) \wedge ?o_1 = False$ $\wedge Conveyor(?c) \wedge Activity(?c, ?o_3) \wedge ?o_3 = True$ $\wedge locatedIn(?c, ?l) \wedge SupervisorPost(?s)$ $\wedge supervises(?s, ?c) \rightarrow NoSupervision(?c)$
R4: Fire hazard	$Location(?l) \wedge ParticleLevel(?l, ?o_1) \wedge ?o_1 > 25\%$ $\wedge SparkMachine(?m) \wedge Activity(?m, ?o_3) \wedge ?o_3 = True$ $\wedge locatedIn(?m, ?l) \rightarrow Firehazard(?m)$
R5: Cold chain broken	$Location(?l) \wedge Temperature(?l, ?o_1) \wedge ?o_1 > 6^\circ C$ $\wedge TemperatureSensitiveMachine(?m) \wedge Activity(?m, ?o_3)$ $\wedge ?o_3 = True \wedge locatedIn(?l, ?m) \rightarrow ColdChainBroken(?m)$
R6: Conveyor too fast	$Conveyor(?c) \wedge Machine(?m) \wedge onConveyor(?m, ?c)$ $\wedge MachineSpeed(?m, ?s_m) \wedge ConveyorSpeed(?c, ?s_c)$ $\wedge ?s_c > ?s_m \rightarrow ConveyorTooFast(?c)$
R7: Low quality product	$Machine(?m) \wedge ProductQuality(?m, ?o_1) \wedge ?o_1 < 98.5$ $\rightarrow LowQualityProduct(?m)$

Table 6.6: Safety and quality rules

mosphere loaded with particles creates a detonation hazard”, or “The presence of a worker near an operating machine in a low luminosity environment is a personal security hazard”. Some rules are also dedicated to quality insurance: sensors available in the factory, such as temperature sensors, or sensors integrated to machines and to the conveyor, enable the continuous control of production quality. Some operations are temperature-sensitive, and a quality insurance rule is “The detection of a temperature above a certain threshold is a break in the cold chain”. Adapting the speed of conveyors to the speed of machines is also part of quality enforcement. All the rules are summarized in Tab. 5.1, and their SHACL representation is available online⁴.

Safety and quality insurance are time-sensitive applications, which is why the processing of the rules should be as fast as possible. Moreover, the mobility of some sensors (*e.g.*, workers wearable), combined to the modularity of the factory floors, are suitable for a dynamic solution adaptative to their evolution over time.

⁴<https://w3id.org/laas-iot/edr/iiot/iiot.tar.gz>

Figure 6.7: Reference topology for d*

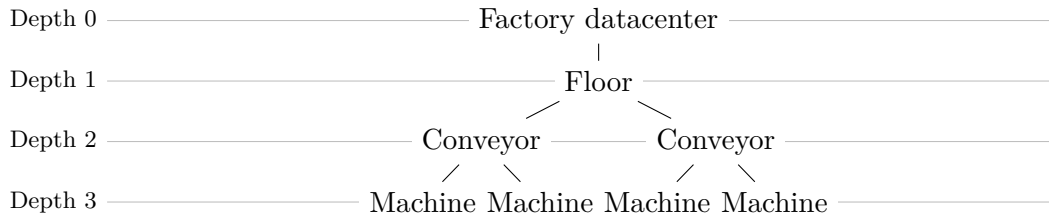


Table 6.7: Machines hosts for distribution experiments

Simulated node	Datacenter	Floor	Conveyor	Machine
Physical host	Server	Laptop	Raspberry Pi	Server

6.4.2 Impact of distribution on responsiveness

6.4.2.1 Simulation topology

The evaluation of the impact of distribution has been performed using the same approach as in Section §6.3.2: the same sensors are deployed from topology d'0 to d'4, but they are not situated at the same level, enabling the control of the level at which rules are processed. Sensors are situated in d'* topologies so that the rules are processed at the depths depicted in Tab. 6.8. The simulation topology is composed of 42 nodes in total (including sensors), hosted on the physical machines as detailed on Fig. 6.7. Fig. 6.8 shows results for centralized approaches, and Fig. 6.9 for distributed reasoning, both showing single-host and multi-host execution.

6.4.2.2 Results

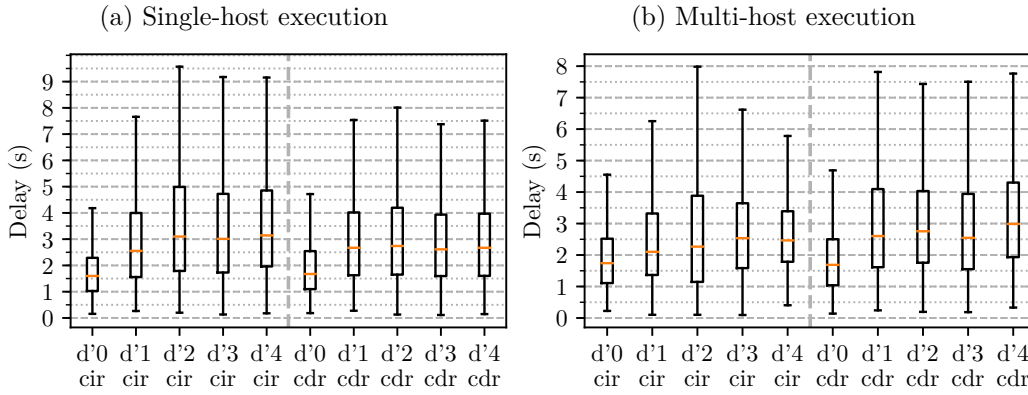
The trend observed in the initial implementation regarding the impact of distribution tend to be confirmed with the new experiments, but there are some differences.

With the centralized reasoning delivery mechanisms, there is little impact of the distribution on performances as seen on Fig. 6.8. The best performances are measured in the most centralized topology, d'0, when the sensors are directly connected to the reasoning node, thus minimizing the transit time, as it is shown on Fig. 6.8a and Fig. 6.8b. Moreover, for this completely centralized topology,

Table 6.8: Depth of rule processing for d'

	R1	R2	R3	R4	R5	R6	R7
d'0	0	0	0	0	0	0	0
d'1	0	1	0	1	1	0	0
d'2	1	1	0	1	1	0	0
d'3	1	1	0	3	3	1	3
d'4	3	2	2	3	3	2	3

Figure 6.8: Distribution experiments, centralized reasoning



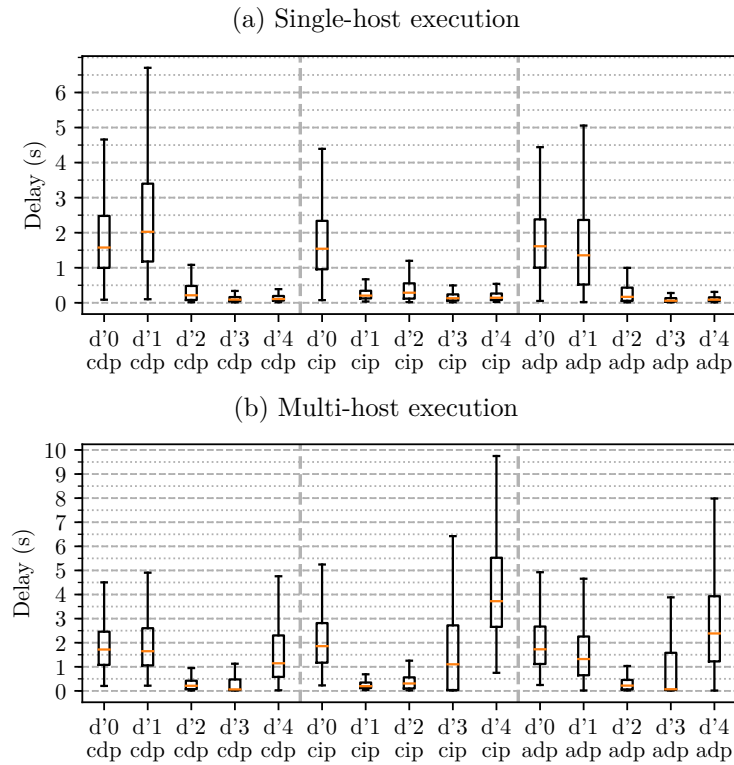
the delays measured with the decentralized delivery mechanisms (CDP, CIP, ADP) are comparable to the centralized ones (CIR, CDR), which is an expected result: since all the sensors are connected to a single node, there is no difference between rule deployments. It should also be noted that there are no significant differences between the centralized and decentralized executions. Since all reasoning, which is the most computing-intensive process of the simulation, is located in both cases on the most powerful node, it is also an observation consistent with our expectations.

For the decentralized delivery mechanisms, where rules are propagated into the network according to the $\text{EDR}_{\mathcal{T}}$ technique, the distribution has indeed an impact on deduction delivery delay, seen on Fig. 6.9. In the single-host execution environment (Fig. 6.9a), where all the nodes have comparable capabilities, there is a correlation between the depth at which rules can be executed (denoting a more important distribution of processing), and the delivery delay decreases. In this case, each node takes an increasing share of the reasoning in charge, leading to a relative decrease of the idle time compared to the reasoning time as seen on Fig. 6.10.

However, Fig. 6.9 shows a discrepancy between the simulation in a single-host and a multi-host environment, the latter actually including constrained nodes. For ADP and CIP on Fig. 6.9b, at the d'3 topology, the third and fourth quartiles show an increase in the delays. The median delay is compliant with the expected decreasing trend for ADP, but it begins increasing for CIP. For the d'4 topology on Fig. 6.9b, where the distribution is maximal, there is an important increase of delays for all decentralized delivery mechanisms, exceeding the delays measured even for d'0.

An explanation for this phenomenon is the saturation of the Fog node passed a certain work load, the tipping point being crossed around d'3. The progressive relative increase of the idle time when increasing distribution, seen when comparing d'3 and d'4 on Fig. 6.10 and Fig. 6.11, supports this hypothesis. To this regard, the $\text{EDR}_{\mathcal{T}}$ technique has a naive approach, where the capabilities of the Fog nodes are not considered in the deployment process. Moreover, the nature of SHACL rules requires the node to evaluate every active rules when a new observation is

Figure 6.9: Distribution experiments, distributed reasoning



received, event if the new information is not relevant for the new rule. This prevents optimizations that were possible in the initial implementation, where rules were individually processed: the generality of the EDR approach is traded for the efficiency of the implementation.

The technological choices made for the implementation of EDR \mathcal{T} are also factors to be considered in the observed results. Overall, EDR \mathcal{T} is still a proof of concept, and some choices in the implementation should be rethought for performance:

- The HTTP framework used (Jersey⁵) has been chosen for convenience for the flexibility of development it allows, but it adds a certain overhead in the memory print and execution time which is not negligible in a constrained environment.
- The SHACL engine used in our experimentations is described by its creators as "not really optimized for performance, just for correctness"⁶. It is possible that in the future, better performances will be reached by sheer improvement of the SHACL engine. This engine was chosen because, to the best of our knowledge, it was the only Jena-compatible SHACL implementation at the time of implementation.

⁵<https://jersey.github.io/>

⁶<https://github.com/TopQuadrant/shacl>

Figure 6.10: Distribution experiments delays breakout (single-host execution)

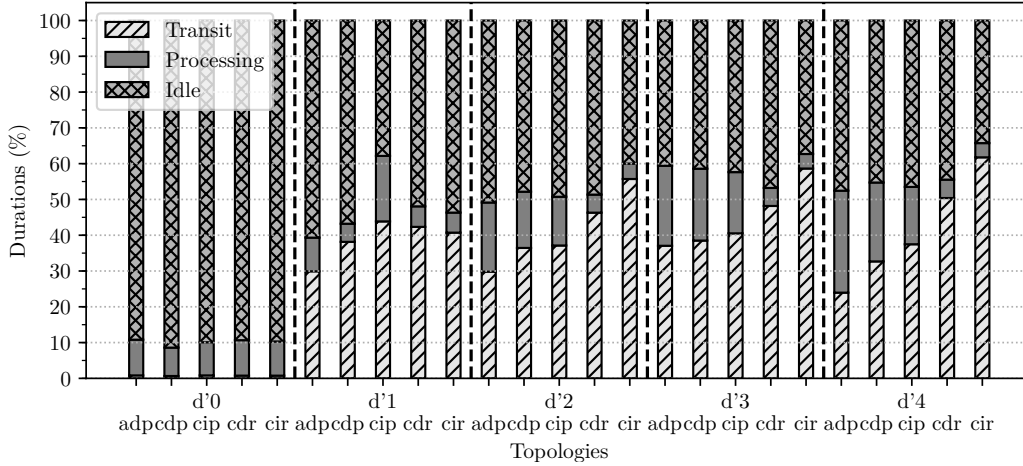
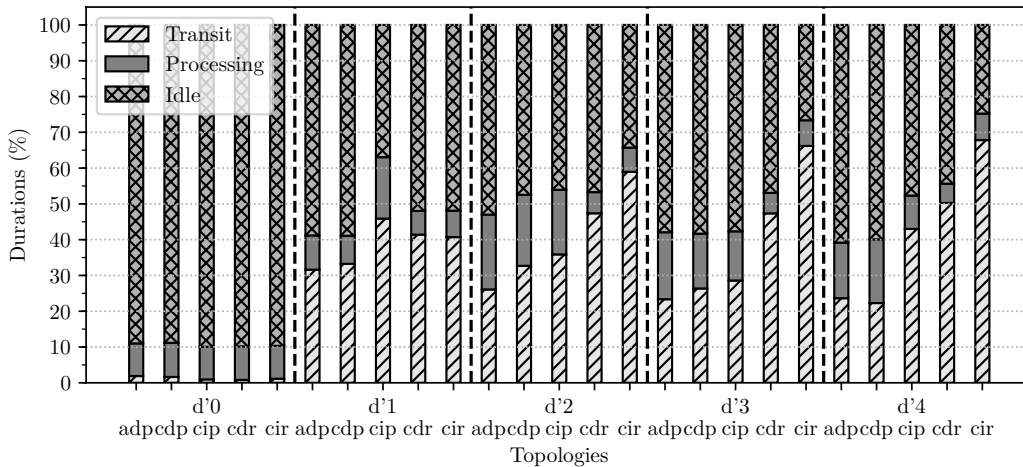


Figure 6.11: Distribution experiments delays breakout (multi-host execution)



- Knowledge is exchanged between nodes serialized in RDF Turtle. Other more compact RDF serializations exist [Su 2015], and switching to such a format would reduce the communication overhead when messages are exchanged.

6.4.3 Scalability of the proposed approach

6.4.3.1 Simulation topologies

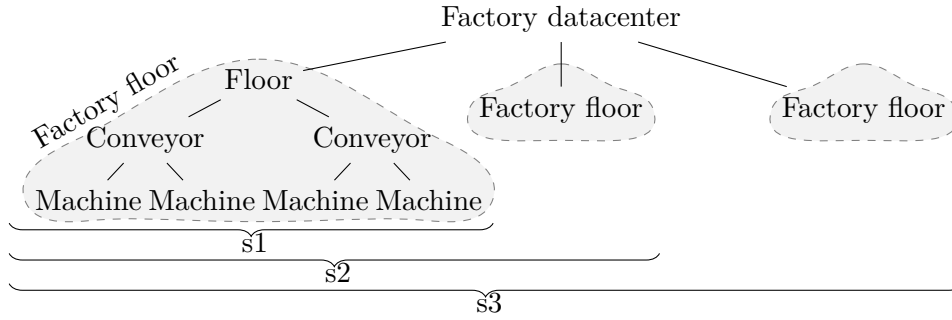
In order to assess the scalability of the proposed strategy for EDR, performances have been measured on three topologies, denoted s'1, s'2 and s'3⁷, and collectively as s'*, as represented on Fig. 6.12. All s'* topologies mimic the use case architecture presented in Fig. 6.6, with variations in the number of floors. A floor is constituted

⁷Topology representations are available at https://w3id.org/laas-iot/edr/iiot/scala_syndream/clone_f_<0,1,2>.ttl respectively

Table 6.9: s* topologies

Topology	s'1	s'2	s'3
Nodes	31	61	91

Figure 6.12: Simulation topology s'*



of two conveyors, each of which supports two machines, with sensors distributed as shown on a JSON blueprint provided online⁸, leading to a total of 30 nodes (including both reasoning nodes and sensors). The rules described in Section §6.4.1 are used. The number of nodes is increased by duplicating floors: s'0 has one, s'1 two, and s'2 three floors, for a total number of respectively 31, 61 and 91 nodes (as summarized on Tab. 6.9). Fig. 6.13 shows results for centralized approaches, and Fig. 6.14 for distributed reasoning, both showing single-host and multi-host execution.

6.4.3.2 Results

Due to scaling issues, results are separated in several figures:

- Results for centralized deduction delivery mechanisms (*i.e.* CIR and CDR) are shown on Fig. 6.13a for single-host execution, and on Fig. 6.13b for multi-host execution.
- Results for distributed deduction delivery mechanisms (*i.e.* CIP, CDP and ADP), are shown on Fig. 6.14a for single-host execution, and on Fig. 6.14b for multi-host execution.

The gain in scalability provided by the decentralized approaches appears in the results. In topology s'1, the discrepancy between delivery delay for distributed and

⁸https://w3id.org/laas-iot/edr/iiot/clone_f_0_blueprint.json

Table 6.10: Machines hosts for scalability experiments

Simulated node	Datacenter	Floor	Conveyor	Machine
Physical host	Server	Raspberry Pi	Server	Laptop

Figure 6.13: Scalability measures, centralized reasoning

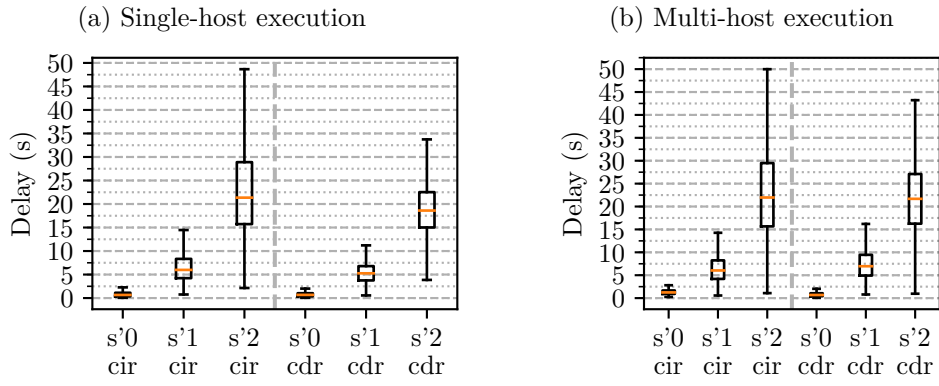
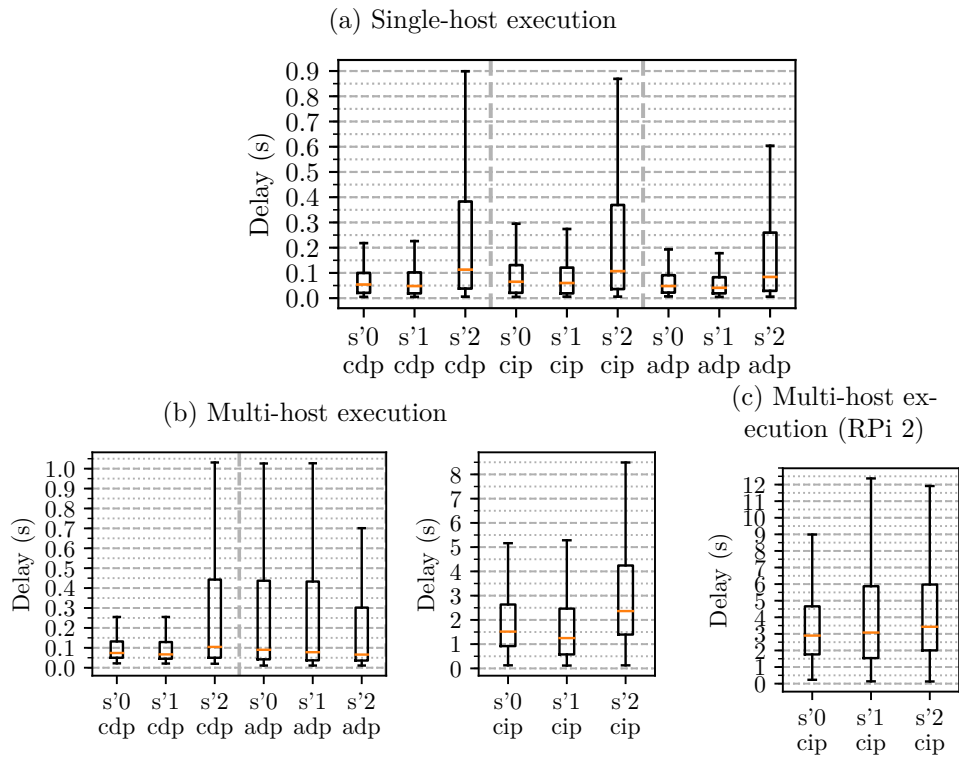


Figure 6.14: Scalability measures, decentralized reasoning



centralized reasoning approaches is reduced, especially in the single-host execution setting, with a median around 0.65s for CIR and CDR, and 0.065s for CDP, CIP and ADP.

However, in topologies s'2 and s'3, the gap between centralized and distributed approaches increases dramatically. The deduction time is multiplied by more than 20 from s'0 to s'2, while the relative share of reasoning time contributing to the delay decreases, as shown on Figs. 6.16 and 6.16. The transit times are the ones to increase relatively the most, which denotes a network overflow over a computing saturation on the centralized reasoning node.

An delay increase is also observed for distributed delivery strategies in the single-host execution environment, but it is much smaller, as seen on Fig. 6.14a. In the multi-host execution environment, there is a performance difference between direct and indirect delivery mechanisms. Even though overall the increase in the number of node has little impact on the measured delays, the delays measured in the CIP configurations are much longer than in CDP or ADP.

An explanation for this observation is the fact that, due to their location, the Raspberry Pis are a bottleneck for communication only in this configuration. In CIP, they must both forward observations and deductions towards a Cloud node, as well as performing reasoning, while they only have to process rules with the CDP and ADP strategies. This conclusion is also strengthened by the fact that, if the Raspberry Pis 3 are replaced by Raspberry Pis 2, which have a lower computing power, that same profile is observed, with longer delays, as seen on Fig. 6.14c for CIP for instance. On Fig. 6.16, among the three decentralized delivery mechanisms, CIP has the least important relative transfer time dedicated to reasoning. This is coherent with the fact that more deductions are forwarded by the constrained nodes rather than deduced directly by it, since it is at depth 1 in the topology, and it is only connected to few sensors compared to conveyor or machine nodes.

A trend that can be observed in the breakout is the increase of the share of transfer time in centralized strategies compared to decentralized ones. An explanation for this phenomenon is the saturation of the network link, combined to an overhead on the central node induced by the necessity to perform all the reasoning. The central node has less CPU time available to declare reception of messages, and therefore the time between the emission event and the reception event is increased. Overall, the limited increase of delays and the balance of the delays breakdown in the distributed settings support our claim that $EDR_{\mathcal{T}}$ is a scalable approach to rule-base reasoning based on semantic Fog computing.

6.5 Conclusion

The evaluations performed in this chapter, on two separate use cases, supported our claim that decentralizing computation is a scalable approach, and that $EDR_{\mathcal{T}}$ reduces delivery delay under some conditions. The first use case extended the smart building use case introduced in Chapter §3, and the second has been dedi-

Figure 6.15: Breakout of delays (normalized, single-host execution)

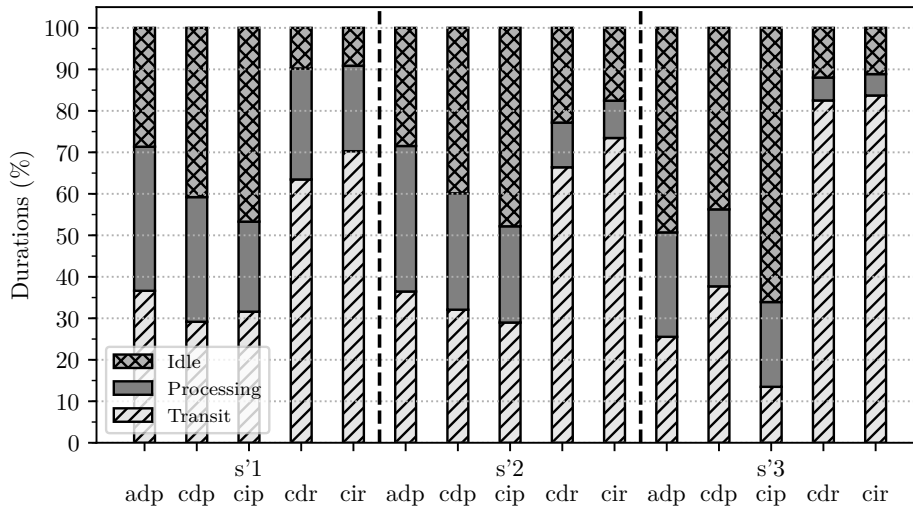
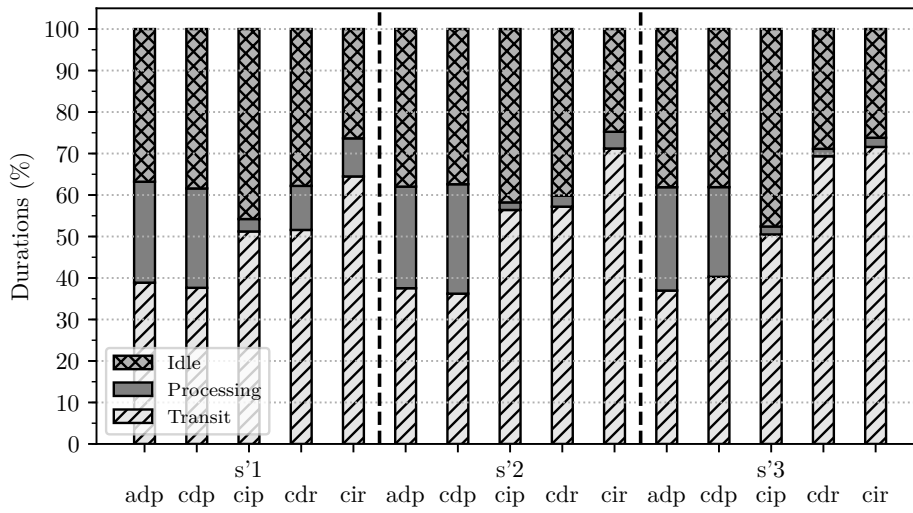


Figure 6.16: Breakout of delays (normalized, multi-host execution)



cated to a smart factory. Two implementations have been evaluated, one natively implementing $\text{EDR}_{\mathcal{T}}$, published in [Seydoux 2018b] and accepted for publication in [Seydoux 2018a], and the other deploying $\text{EDR}_{\mathcal{T}}$ as a refinement of EDR, accepted for publication in [Seydoux 2018c]. Considering these two implementations and comparing their specificities has given us some perspective on the tradeoffs in optimization and Fog computing.

On the one hand, the initial native approach has been easier to develop, and supported shorter rules, since they do not embed the deployment strategy. For instance, rule R1 in the building management use case is 54 lines long with the initial implementation, and 280 lines long with the EDR-based implementation. On the other hand, the second implementation, even though it requires more work from rules implementers, enables a finer control over rules propagation, with a strategy that can be selected at the scale of the rule. Rules are also more interoperable in the second case, since their processing is compliant with the SHACL standard.

In both use cases, decentralized delivery mechanisms have outperformed centralized ones: the degradation of QoS when the number of nodes increase is much slower when reasoning is performed in a distributed manner. Similarly, the enablement of a more widespread distribution of rules by a modification of the sensors deployment have not improved QoS with a centralized delivery mechanism. With a decentralized approach, such an increase of distribution has improved performances up to certain point, but when executed on constrained nodes, has eventually led to a QoS degradation.

This degradation is a drawback of $\text{EDR}_{\mathcal{T}}$ that have been revealed by the conducted experimentations. Not considering the capabilities of Fog nodes in the deployment process is a naive approach that reaches its limitations when overloading constrained nodes. Moreover, due to technical constraints, the experiments we conducted could not be performed at a large scale on constrained nodes. This introduces a bias in the measured results, since the simulated nodes are ran on machines much more powerful than the Fog nodes should be. We are aware of this bias, and the experiments are designed in such way that it has an impact as reduced as possible. For future experiments, we intend to set up a network of virtual machines, emulating the actual capabilities of physical nodes, rather than mere processes.

Initially, it was planned to reuse ADREAM data in the experiments, in order to test the approach on real-life data, and to ensure reproducibility and completeness of each approach. Using the same sample of observations would allow to compare the deductions, ensuring that they remain consistent from an approach to the other. However, we failed to reuse ADREAM data because of synchronization issues among the multiple simulated sensors. Reconsidering the simulation architecture to enable the reuse of actual data is also an improvement considered for future experiments.

The complementarity of Fog and Cloud paradigms is also supported by the results of our approach: there is an improvement of performances even in cases where deductions are forwarded to a Cloud node, and not directly to applications, compared to a centralized reasoning approach. Therefore, unloading the Cloud infrastructure by performing semantic Fog computing, while considering the Cloud

node both as a computation resource and as an stable Web endpoint for applications enables scalable deployments for the SWoT.

Conclusion and future work

Contents

7.1	Conclusion	161
7.2	Future work	163
7.2.1	Short-term work: Extending the EDR ecosystem	163
7.2.2	Medium-term work: Bringing semIoTics and EDR together	166
7.2.3	Long-term work: Supporting natural-language interaction	167

7.1 Conclusion

IoT devices are deployed pervasively in the environment to provide services to human users. To enforce compliance with principles of user-centric design, we determined the necessity for **interoperability among IoT devices**, and for the capacity of self-management of IoT systems. From this observation, the emergence of the SWoT domain is the logical culmination of the co-evolution of the IoT and SW domains. SW technologies and principles were defined in the first place to represent machine-understandable knowledge, used to provide M2M interoperability. Achieving such interoperability is a necessity for the expansion of IoT networks, since they are composed of deeply heterogeneous devices required to communicate with each other. However, the dynamism of IoT networks, and the inherent constraints of IoT devices represent challenges for the deployment of SW technologies in the IoT context. The work we propose in the present thesis addresses this twofold issue, in order to **achieve interoperability for IoT devices while adapting SWoT technologies to IoT constraints**.

First, we described how interoperability, and especially semantic interoperability, is achieved in the SWoT. Some industrial standards are moving towards semantic interoperability, but the main tools to achieve it are ontologies. In order to discuss the role of IoT ontologies as interoperability providers, two contributions have been described:

- We proposed ontology design requirements to characterize quality features for IoT ontologies, in order to support the adoption of good practices in the SWoT community. Existing IoT ontologies have been identified, described, and assessed based on these requirements. Since existing ontologies did not fulfill the proposed design requirements, we instantiated them by defining

IoT-O, a modular core-domain IoT ontology. **Contribution I.A of this thesis is the association of the IoT ontology design requirements and their instantiation by IoT-O.** To ensure a maximum interoperability, some IoT-O modules are based on existing ontologies, and IoT-O has been aligned to reference ontologies of the domain created after its publication. The description of IoT-O has been published in [Seydoux 2016b].

- IoT-O has been used as a semantic interoperability enabler in three use cases. The OPA platforms proposes an open data which is updated daily with observations collected in a smart building. These observations are enriched with knowledge described using extensions of IoT-O. The enriched observations are then transformed to be published in the FIESTA-IoT platform, a federated data hub deployed in a European project. The transformation making OPA observations compliant with the FIESTA-IoT platform are based on alignments between IoT-O and the FIESTA-IoT vocabulary. Finally, the role of **IoT-O and its extensions in driving the behavior of semIoTics, constituting contribution I.B of this thesis**, has been discussed. SemIoTics is an autonomic computing software used in a home automation use case, where syntactic and semantic interoperability are leveraged. SemIoTics was initially described in [Seydoux 2016a] and [Aïssaoui 2016].

Autonomic computing is a way to achieve self-managing IoT systems, but it requires constrained devices, in particular actuators, to adapt their behavior based on high-level decisions. We proposed initial work for supporting semantic interoperability towards constrained devices with the mapping-reversal approach, published in [Seydoux 2016c]. Such approach underlines the necessity to locate demanding computation on nodes with sufficient computing capabilities.

Indeed, SW technologies tend to be resource-consuming, which is not suitable for an IoT deployment built upon constrained devices. Therefore, the SW stack supporting SWoT applications has often been deployed on Cloud nodes, leading to a centralization of the computation. Such an approach impacts the scalability negatively, and degrades the QoS. That is why we considered the Fog computing paradigm to enable the distribution of processing, and allow scalable deployments. The technologically heterogeneous, and spatially spread nature of IoT systems requires the existence of a middle layer between IoT devices that collect data, and Cloud servers that process it. This layer can be used for Fog computing, and this thesis surveys the role of semantic Fog computing in SWoT architectures.

Surveying how semantic-enabled Fog nodes support the deployment of SWoT systems led to the consideration of the complementarity of Cloud and Fog computing. We proposed two contributions to leverage these paradigms:

- **EDR, a generic approach for dynamically distributed rule-based reasoning** in SWoT architectures has been introduced as **Contribution II.A.** EDR defines a propagation technique for rules and data, driven by a deployment strategy. Applicative-level requirements for rule propagation are

captured by the deployment strategy which is embedded into the rules. The core rule and data deployment technique is independent of the applicative-level requirements, making EDR generic. A description of EDR has been published in [Seydoux 2018c].

- **A refinement of EDR, called $\text{EDR}_{\mathcal{T}}$** , has then been described as **contribution II.B**. $\text{EDR}_{\mathcal{T}}$ implements a deployment strategy designed to reduce deduction delivery delay by bringing rule computation as close as possible to devices producing data. Two implementations of $\text{EDR}_{\mathcal{T}}$ have been evaluated in two use cases, a smart building and a smart factory. The results of these experimentations supported our claim that decentralization improves scalability, and therefore supports the deployment of the SWoT. $\text{EDR}_{\mathcal{T}}$ was initially incepted in [Seydoux 2018b], and its detailed description was provided in [Seydoux 2018a].

The purpose of the work presented in this thesis has been to support the development of semantic interoperability solutions in the domain of the IoT by promoting the emergence of SWoT technologies adapted to IoT constraints. By studying how SW principles and technologies provide interoperability, and by considering a reasoning approach both scalable and dynamically distributed on semantic-aware Fog nodes, our contributions addressed these issues and supported further developments of the SWoT domain. However, many challenges remain open in the convergence of SW technologies and IoT devices, and they motivate our propositions of future work to overcome limitations identified in our own contributions.

7.2 Future work

7.2.1 Short-term work: Extending the EDR ecosystem

EDR is a generic approach, supported by a vocabulary and refined by implementations of deployment strategies. The core characteristics of EDR have been studied in this thesis, but further extensions can be considered. These extensions aim at capturing different applicative requirements with new deployment strategies, but also to support the work of rule implementors.

7.2.1.1 Potential deployment strategies

$\text{EDR}_{\mathcal{T}}$ is one possible refinement of EDR, focusing on improving response time by considering observation types. In the remainder of this section, we list other parameters that may be considered to implement new deployment strategies.

Considering node capabilities: The limitations of $\text{EDR}_{\mathcal{T}}$ were discussed in Chapter §6, in particular its failure to consider the capabilities of Fog nodes in the rule deployment process. Due to the heterogeneity of Fog nodes, it is naive for a given node to consider all its neighbors equally. We intend to extend $\text{EDR}_{\mathcal{T}}$ in

order to make nodes aware of each other’s capabilities when propagating rules to neighbors. A challenge for this approach is that it should not be limited to the static characteristics of the nodes, such as memory or computing power. It should also be adaptative to a node’s dynamic state, for instance considering the battery state, or the rules that have already been forwarded to this node.

The heterogeneity of Fog nodes regarding communication capabilities is also a factor to consider. Depending on the deployed technologies, direct communication between Fog and Cloud nodes or applications might not be possible due to technical interoperability concerns. In such a topology, the Application-Direct-Processed and Cloud-Direct-Processed delivery mechanisms cannot be applied, having this direct communication as a prerequisite. In deployments where nodes communicate over ad-hoc networks, we discussed the role of border gateways as relays between the IoT and the Web. Proposing a new approach derived from our Cloud-Indirect-Processed mechanism by identifying the critical gateways that ensure technical interoperability, will allow the relaxation of our hypothesis requiring direct communication between Fog nodes and application.

Domain identification: In the second version of the S-LOR, platform introduced in [Gyrard 2017], rules are classified into domains. It is possible to represent such classification as a taxonomy, in order to capture the domains into a KB. Once represented in a KB, it is possible to use these domains to drive the propagation of rules among Fog nodes, by associating nodes and rules to domains. We draw a prospective outline of such a deployment strategy, whose implementation by an EDR refinement could be referred to as $\text{EDR}_{\mathcal{D}}$.

Let us briefly consider $\text{EDR}_{\mathcal{D}}$ through a smart campus example. The campus is composed of a library, a food court, student housing, and classroom buildings. Each of these buildings belongs to a different domain, and therefore rules considering the same type of input (*e.g.*, a temperature threshold, for simplicity) will not be executed equally in these different domains. One could expect the desirable temperature to be warmer in an apartment than in a classroom for instance. Instead of propagating all the rules measuring a temperature threshold to a node where temperature observations are available, only the rules associated to the node’s domain should be considered. Moreover, one node may belong to several domains, and for instance, all places where fire hazards should be monitored can be classified in a “Fire” domain.

The core mechanism of EDR would be unchanged by such a deployment strategy: instead of proxying production types, nodes would proxy the domain. This approach would implement a specialization logic, where depending on their context (here the domain), similar observation types are not processed with the same rules. Moreover, it would make it possible to dynamically adapt the context in the case of mobile nodes. For instance, let us consider a situation where a student with reduced mobility having a Fog node embedded in his/her wheelchair. Such a node may be associated with the “Reduced mobility” domain, triggering dedicated rules in the

different contexts it will appear. The mobility of the node means that accessibility measures (automated doors, support from staff) are only set up when the student is present in the building, and supporting proper assistance whenever needed.

Privacy awareness: Privacy is a critical concern for the IoT R&D community¹, as well as for IoT devices' end users. The recent multiplication of security breaches found in IoT systems² demonstrates the legitimacy of this concern. Shifting the paradigm from the concentration of data in remote, centralized, third-party nodes to the propagation of processing close to data producers and consumers enforces the locality of data processing. Enforcing privacy in IoT systems remains an open issue [Miorandi 2012], despite some initial work for knowledge access control in the SWoT [Alam 2010].

A refinement of EDR implementing a **privacy-aware deployment strategy** would therefore be an interesting prospective approach, that we refer to as $EDR_{\mathcal{P}}$. Such an approach would measure how the overhead of traffic, compared to the solution proposed in the present thesis, impacts performances, and find a trade-off between privacy and performances. $EDR_{\mathcal{P}}$ would rely on a partially ordered credentials definition: each data producer (typically a sensor) would be given a credential level, attached to each observation it produces, and each node would also be attributed with a credential level, inspired by [Singh 2017] or Unix user groups. The credential level of a node is typically a characteristic that would be declared by a node to its neighbors, based on the EDR announcement mechanism. A node may be forwarded a piece of data if and only if this node has a credential level that is both comparable and superior or equal to the credential of the piece of data. The credential level of a child is considered superior to that of its ancestors: the notion of context imbrication on which $EDR_{\mathcal{T}}$ is based can also be adapted to privacy. For sibling nodes, and *a fortiori* in the general case, credentials are not comparable, unless explicitly stated so.

A specificity of $EDR_{\mathcal{P}}$ compared to $EDR_{\mathcal{T}}$ is that it is possible that an observation is not only propagated upward. Indeed, the notion of lowest common ancestor as it is used in $EDR_{\mathcal{T}}$ assumes that any node may access any information, which is no longer the case in $EDR_{\mathcal{P}}$. Therefore, the rule propagation module of an $EDR_{\mathcal{P}}$ rule should not only check if a child node is a potential candidate for rule application, but also declare the child node consumer of observations collected by its parent, which is contrary to the $EDR_{\mathcal{T}}$ logic.

The notion of policy as defined in [Singh 2017] should also be considered when attributing a confidentiality level to a deduction. Since the different elements considered by the rule might be of varying confidentiality, determining the confidentiality of the rule result is a non-trivial issue, necessitating an explicit policy expressed as part of the rule. Simple examples of policy include “most restrictive” or “least restrictive”, where the confidentiality of the result is directly inherited from the

¹<https://www.slideshare.net/kartben/iot-developer-survey-2018>

²<https://internethealthreport.org/2018/spotlight-securing-the-internet-of-things/>

confidentiality of one of the rule's input, but more complex aggregation policies may be implemented as well.

7.2.1.2 Supporting rule management

Rule hosting platform: After considering improvements in the rule deployment process itself, we intend to improve the management of rules as well. The modularity of EDR rules does not facilitate their development, and requires some expertise for rule implementors. In order to improve the accessibility of rules, a sharing platform inspired by [Gyrard 2017] is also considered as a future work. Since EDR rules are compliant with Linked Open Rules principles, the platform may offer access to rules presented as Web resources, improving their reusability. Moreover, having a central repository for rules enables the approach to consider a reference version of the rule, as the dereferenced resource hosted by said repository. Since rules are identified by IRI, it is possible to incrementally modify them at runtime, so that the operation of the controlled system is not interrupted. Modifying rules allow applications to fine-tune their behavior according to a feedback loop that considers either previous responses to inputs, or external factors (*e.g.*, seasonal change, or regulation evolution).

Rule development support: Developing rules embedding deployment strategies may be a challenging task. In order to support rule implementors in their development process, the hosting platform presented in the previous paragraph could include rule-building tools. These tools will enable the construction of rules embedding existing deployment strategies such as $\text{EDR}_{\mathcal{T}}$, as well as the visualization of existing rules on the platform.

Multiple concurrent deployment strategies: So far, the deployment strategies implemented by refinements of EDR have been considered individually, with only one strategy driving rule deployment in a particular SWoT network. Since EDR is a generic approach agnostic to the deployment strategy, it is technically possible to deploy rules embedding multiple concurrent deployment strategies. However, inconsistent behaviors may be induced by the execution of contradictory deployment strategies. For instance, the decisions based on privacy considerations in $\text{EDR}_{\mathcal{P}}$ might be undermined by decisions driven by efficiency in $\text{EDR}_{\mathcal{T}}$. The identification of inconsistent strategies, and the mechanisms to prevent detrimental behaviors are challenges that should be addressed in future work.

7.2.2 Medium-term work: Bringing semIoTics and EDR together

In its current state, EDR feeds remote applications deductions, that these applications can use in a decision-making process. Such decisions may be taken by human operators, in which case the application provides the operator with a representation of the deductions received from the network, implementing a presentation

functionality as described in Section §4.2.1.2. However, it is also possible that the application reacts to the deductions by taking action without a direct human intervention. In this case, the model of autonomic computing, already introduced in Section §3.3.4.2 when discussing a home automation use case, may be applied.

Let us consider the application featured in the use case described in Section §5.3 as an autonomic agent implementing a MAPE-K loop. When applying EDR, the implementation of the MAPE-K loop is incomplete: the Monitoring and Analysis steps are indeed performed, by collecting observations and processing them with rules, but the Planning and Execution steps are not discussed.

The EDR approach aims at bursting the MAPE-K loop implemented by the application: instead of receiving all the raw observations (Monitoring) and making the deductions itself (Analysis), the application spreads its Analysis rules in the network so that both Monitoring and Analysis are distributed. In order to achieve a full distribution of the MAPE-K loop, it would be necessary to decompose the Planning modules into rules as well, in order to make decisions locally based on the deductions inferred from observations, as well as enabling the local implementation of the Execution step. In order to enable the transformation of high-level action representations, issued from the Planning step, into messages understandable by legacy devices, the mapping reversal approach introduced in Section §3.4 should be integrated into the Execution step.

A complex system may be composed of several autonomic elements, each of them implementing its own MAPE-K loop. Therefore, the decentralization of the application would be complete, with MAPE-K loop instances deployed opportunistically in the network in order to control devices at a local scale. This approach would lead to a multiscale system-of-systems IoT automation and self-configuration, by enabling the creation of interworked autonomous systems depending on each other. Multi-scale modelling is discussed in [Gassara 2017], and the concept of system-of-systems is defined in [Boardman 2006]. Generalizing autonomic computing at different levels of IoT deployments would enable a full-stack semantic interoperability, from a high-level policy expressed to drive an IoT to atomic device behavior implementing such strategy. IoT networks are then treated as holons [Koestler 1967], that is to say that they are both a whole and a part, depending on the granularity of the policy considered. As a whole, they have a purpose and they can be considered an atomic system, and, as a part, they contribute to the purpose of the system at a larger scale [Oliveira 2013]. This approach leads to the emergence of **local behaviors enforcing global policies**, which is a more scalable pattern bridging high-level, global user requirements to low-level, local actions.

7.2.3 Long-term work: Supporting natural-language interaction

User-centric design is a global approach that motivates the contributions of my work. How user requirements are collected in the first place is an actively discussed topic in this approach. The fundamental way of human communication is natural language, and usually user-centered design is driven by the interaction between

designers and users. Moreover, in this work we considered systems in which user requirements are not only considered at design time, but are also driving the behavior of the system. For SWoT systems, user preferences are usually expressed through a Graphical User Interface (GUI), as in [Kaed 2018b] or [Kasnesis 2015]. However, such expressiveness is limited to the use cases designed by the implementor of the GUI, restricting the customization for the user. The expression of requirements in natural language is a way to enable a more user-friendly interaction.

An issue with natural language interaction in the IoT is the mostly numerical nature of the exchanged data. However, the emergence of the SWoT introduces natural language resources in data annotation. Such resources enable systems to interact with the user not only via graphical interfaces, but also through conversational interfaces. We presented preliminary work for query-answering dedicated to IoT systems in [Lannes 2017]. The current popularity of vocal assistants, the ever growing capabilities of speech recognition for smart phones, and open projects fostering the production of large corpora of voice recordings³, combined to the development of chatbots, denotes the importance of conversational interfaces facing end users. Not only are direct commands and interactions enabled by such interfaces (“Switch on the bedroom reading light”), but high-level policies driving autonomic behavior can also be specified by the user. A natural language interface would therefore be a user-centered endpoint deployed on top of a joined semIoTics/EDR system.

Mark Weiser, a major figure of the pervasive computing paradigm, stated that: “The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it” [Weiser 2002]. Enabling through the SWoT conversational interactions with complex systems integrated into our environments while ensuring privacy and dynamic adaptability, is achieving a vision of the IoT where the technology does indeed disappear, to leave humans in a “smarter” environment.

³<https://voice.mozilla.org/>

Appendix

A.1 Namespaces and prefixes

All along the manuscript, the following prefixes have been used to represent namespaces:

Prefix	Namespace
dul	http://www.ontologydesignpatterns.org/ont/dul/DUL.owl
edr	https://w3id.org/laas-iot/edr
ex	http://example.org/ns
ioto	http://www.irit.fr/recherches/MELODI/ontologies/IoT-0
lc	http://vocab.org/lifecycle/schema
lmu	https://w3id.org/laas-iot/lmu
msm	http://iserve.kmi.open.ac.uk/ns/msm
ows	https://delicias.dia.fi.upm.es/ontologies/ObjectWithStates.owl
sh	http://www.w3.org/ns/shacl
sosa	http://www.w3.org/ns/sosa/
ssn	http://purl.oclc.org/NET/ssnx/ssn
time	http://w3c.org/2006/time
wsmo	http://www.wsmo.org/ns/wsmo-lite

A.2 Example EDR rule

```

prefix san: <http://www.irit.fr/recherches/MELODI/ontologies/SAN#>
prefix xsd: <http://www.w3.org/2001/XMLSchema#>
prefix fn: <http://w3id.org/sparql-generate/fn/>
prefix iter: <http://w3id.org/sparql-generate/iter/>
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
prefix adr: <https://w3id.org/laas-iot/adream#>
prefix iotl: <http://iot.ee.surrey.ac.uk/fiware/ontologies/iot-lite#>
prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn#>
prefix ioto: <http://www.irit.fr/recherches/MELODI/ontologies/IoT-0#>
prefix edr: <http://w3id.org/laas-iot/edr#>
prefix ex: <http://example.com/ns#>
prefix sh: <http://www.w3.org/ns/shacl#>
prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
prefix lmu: <http://w3id.org/laas-iot/lmu#>
prefix dul: <http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#>
prefix time: <http://www.w3.org/2006/time#>

```

```
ex:R1Spot
```

```

edr:hasTransferShape ex:R1TransferShape ;
edr:hasApplyShape ex:R1ApplicableShape ;
edr:hasDeliveryShape ex:R1ResultDeliveryShape ;
edr:hasDeductionShape ex:R1ActiveShape .

```

```
ex:R1TransferShape
```

```

a sh:NodeShape ;
a edr:TransferShape ;
a edr:NodeSensitiveComponent ;
sh:targetClass lmu:Node ;
sh:sparql [
  sh:select """
    PREFIX edr: <http://w3id.org/laas-iot/edr#>
    PREFIX lmu: <http://w3id.org/laas-iot/lmu#>
    PREFIX ssn: <http://purl.oclc.org/NET/ssnx/ssn#>
    prefix adr: <https://w3id.org/laas-iot/adream#>
    prefix ex: <http://example.com/ns#>

    SELECT $this {
      FILTER NOT EXISTS {
        $this a lmu:Node ;
        edr:producesDataOn adr:Temperature, adr:Luminosity ;
        lmu:hasUpstreamNode [
          a lmu:HostNode;
        ].
      }
      FILTER NOT EXISTS {
        {ex:R1 edr:transferredTo $this.}
        UNION
        {ex:R1 edr:transferableTo $this.}
      }
    }
  """ ;

```

].

```

ex:R1Transfer
  a sh:NodeShape ;
  sh:targetClass lmu:Node ;
  sh:rule [
    a sh:SPARQLRule ;
    sh:condition ex:R1TransferShape ;
    sh:construct """
      PREFIX ssn:<http://purl.oclc.org/NET/ssnx/ssn#>
      PREFIX ex:<http://example.com/ns#>
      PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
      PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
      PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
      PREFIX dul: <http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#>
      PREFIX edr: <http://w3id.org/laas-iot/edr#>
      PREFIX lmu:  <http://w3id.org/laas-iot/lmu#>
      CONSTRUCT {
        ex:R1 edr:transferableTo $this.
        ex:R1 edr:transferredFrom ?host.
      } WHERE {
        $this lmu:hasUpstreamNode ?host.
        ?host a lmu:HostNode.
      }
    """;
  ].

```

```

ex:R1ApplicableShape
  a sh:NodeShape ;
  a edr:ApplicableShape ;
  sh:targetClass lmu:HostNode ;
  a edr:NodeSensitiveComponent;
  sh:sparql [
    sh:select """
      PREFIX edr: <http://w3id.org/laas-iot/edr#>
      PREFIX lmu: <http://w3id.org/laas-iot/lmu#>
      PREFIX ssn:  <http://purl.oclc.org/NET/ssnx/ssn#>
      prefix adr:  <https://w3id.org/laas-iot/adream#>
      prefix ex:   <http://example.com/ns#>
      PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
      SELECT $this {
        FILTER NOT EXISTS {
          ex:R1 edr:isRuleActivable "true"^^xsd:boolean.
        }
        FILTER NOT EXISTS {
          $this a lmu:HostNode.
          $this lmu:hasDownstreamNode
            ?temperatureProvider,
            ?luminosityProvider.
          ?temperatureProvider edr:producesDataOn adr:Temperature.
          ?luminosityProvider edr:producesDataOn adr:Luminosity.
        }
        FILTER NOT EXISTS {
          ex:R1 edr:isRuleActive "true"^^xsd:boolean.
        }
      }
    """
  ].

```

```

    FILTER EXISTS {
      $this lmu:hasDownstreamNode ?lowerNode.
      FILTER(
        ?lowerNode = ?luminosityProvider ||
        ?lowerNode = ?temperatureProvider
      )
      FILTER NOT EXISTS {
        ?lowerNode edr:producesDataOn
          adr:Temperature,
          adr:Luminosity.
      }
    }
  }
}
""" ;
].

```

```

ex:R1ApplicantRule
  a sh:NodeShape ;
  sh:targetClass lmu:HostNode ;
  sh:rule [
    a sh:SPARQLRule ;
    sh:condition ex:R1ApplicableShape ;
    sh:construct """
      prefix adr: <https://w3id.org/laas-iot/adream#>
      PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
      PREFIX edr: <http://w3id.org/laas-iot/edr#>
      prefix ex: <http://example.com/ns#>
      prefix lmu: <http://w3id.org/laas-iot/lmu#>
      prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
      CONSTRUCT {
        $this edr:isInterestedIn adr:Luminosity, adr:Temperature.
        $this edr:producesDataOn ex:Symptom1.
        ?interest a rdf:Statement;
          rdf:subject $this;
          rdf:predicate edr:isInterestedIn;
          rdf:object ?partialProduction;
          edr:announceTo ?partialDataProvider.
        ex:R1 edr:isRuleActive "true"^^xsd:boolean.
        ?originator edr:consumesResult ex:R1.
      } WHERE {
        $this a lmu:HostNode.
        {
          $this lmu:hasDownstreamNode ?partialDataProvider.
          ?partialDataProvider edr:producesDataOn ?partialProduction.
          FILTER NOT EXISTS {
            ?partialDataProvider edr:producesDataOn
              adr:Luminosity,
              adr:Temperature.
          }
        }
      } UNION {
        ex:R1 edr:isRuleActivable "true"^^xsd:boolean.
      }
    ex:R1 edr:ruleOriginatedFrom ?originator.
  ]

```

```

OPTIONAL{ex:R1 edr:isActive "false"^^xsd:boolean.}
BIND(STRAFTER(str(?partialProduction), "#") AS ?productionName)
BIND(URI(CONCAT(str($this), ?productionName, "Interest"))
      AS ?interest)
}
""";
].

```

```

ex:R1ActiveShape
a sh:NodeShape ;
a edr:ActiveShape ;
sh:targetClass lmu:HostNode ;
a edr:ContentSensitiveComponent;
sh:sparql [
sh:select ""
PREFIX edr: <http://w3id.org/laas-iot/edr#>
PREFIX lmu: <http://w3id.org/laas-iot/lmu#>
PREFIX ssn: <http://purl.oclc.org/NET/ssnx/ssn#>
prefix adr: <https://w3id.org/laas-iot/adream#>
prefix ex: <http://example.com/ns#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT $this
WHERE {
FILTER NOT EXISTS {
$this a lmu:HostNode.
ex:R1 edr:isActive "true"^^xsd:boolean.
}
}
""";
].

```

```

ex:R1ResultDeliveryShape
a sh:NodeShape ;
a edr:ResultTransferShape ;
a edr:NodeSensitiveComponent;
sh:targetClass lmu:Node ;
sh:sparql [
sh:select ""
PREFIX edr: <http://w3id.org/laas-iot/edr#>
PREFIX lmu: <http://w3id.org/laas-iot/lmu#>
prefix ex: <http://example.com/ns#>

SELECT $this {
FILTER NOT EXISTS {
$this a lmu:Node ;
edr:isInterestedIn ex:Symptom1 ;
lmu:hasDownstreamNode [
a lmu:HostNode;
].
}
FILTER NOT EXISTS {
{$this edr:consumesResult ex:R1.}
}
}
]
}

```



```

    }
    """;
].

ex:R1ResultTransfer
  a sh:NodeShape ;
  sh:targetClass lmu:Node ;
  sh:rule [
    a sh:SPARQLRule ;
    sh:condition ex:R1ResultTransferShape ;
    sh:construct """
      PREFIX ex:<http://example.com/ns#>
      PREFIX edr: <http://w3id.org/laas-iot/edr#>
      PREFIX lmu:  <http://w3id.org/laas-iot/lmu#>
      CONSTRUCT {
        $this edr:consumesResult ex:R1.
      } WHERE {
        $this a lmu:Node ;
      }
    """;
  ].

ex:R1Shape
  a sh:NodeShape ;
  sh:targetClass lmu:HostNode ;
  sh:rule ex:R1 .

ex:myAbstractApp a lmu:Application;
  iotl:exposes [
    iotl:endpoint "http://localhost:8005";
  ].

ex:R1 a sh:SPARQLRule ;
  rdfs:comment "T, L -> C1";
  sh:condition ex:R1ActiveShape ;
  edr:ruleOriginatedFrom ex:myAbstractApp ;
  edr:originatingEndpoint "http://localhost:8005" ;
  sh:construct """
    PREFIX ssn:<http://purl.oclc.org/NET/ssnx/ssn#>
    PREFIX ex:<http://example.com/ns#>
    PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
    PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
    PREFIX opa:<https://w3id.org/laas-iot/adream#>
    PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
    PREFIX dul: <http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#>
    PREFIX edr: <http://w3id.org/laas-iot/edr#>
    PREFIX lmu:  <http://w3id.org/laas-iot/lmu#>
    PREFIX adr: <https://w3id.org/laas-iot/adream#>
    CONSTRUCT {
      ?deduction a rdf:Statement;
        rdf:subject ?feature;
        rdf:predicate rdf:type;
        rdf:object ex:Symptom1;
        edr:deducedAt ?now;
    }
  """;

```

```

        edr:deducedBy $this;
        edr:deducedWith ex:R1;
        edr:deducedFrom ?t_obs, ?l_obs.
    } WHERE {
        ?t rdf:type/rdfs:subClassOf* adr:Temperature;
        ssn:isPropertyOf ?feature.
        ?t_obs ssn:observationResult/ssn:hasValue/dul:hasDataValue
            ?temperatureValue;
        ssn:observedProperty ?t;
        ssn:observedBy ?temperature_sensor.
        FILTER(?temperatureValue <= '25.0'^^xsd:float)

        ?l rdf:type/rdfs:subClassOf* adr:Luminosity;
        ssn:isPropertyOf ?feature.
        ?l_obs ssn:observationResult/ssn:hasValue/dul:hasDataValue
            ?luminosityValue;
        ssn:observedProperty ?l;
        ssn:observedBy ?luminosity_sensor.
        FILTER(?luminosityValue < '800.0'^^xsd:float)

        $this a lmu:HostNode.

        FILTER NOT EXISTS {
            ?t_obs edr:usedForDeductionBy ex:R1.
            ?l_obs edr:usedForDeductionBy ex:R1.
        }
        FILTER NOT EXISTS {
            ?otherDeduction edr:deducedWith ex:R1;
            edr:deducedFrom ?t_obs, ?l_obs.
        }

        BIND(URI(CONCAT('http://example.com/ns#R1_deduction', STRUUID()))
            AS ?deduction )
        BIND(NOW() AS ?now)
    }
}
"""
ex:Symptom1 rdfs:subClassOf <http://purl.oclc.org/NET/ssnx/ssn#Property>.





```

A.3 Acronyms

AAE Action-Actuator-Effect.	IK Information/Knowledge.
ADP Application-Direct-Processed.	IoE Internet of Everything.
AI Artificial Intelligence.	IoT Internet of Things.
AR Augmented Reality.	IRI International Resource Identifier.
BAS Building Automation System.	KB Knowledge base.
BLE Bluetooth Low Energy.	LD Linked Data.
BO Base Ontology.	LDP Linked Data Platform.
CDP Cloud-Direct-Processed.	LOD Linked Open Data.
CDR Cloud-Direct-Raw.	LOV Linked Open Vocabularies.
CEP Complex Event Processing.	M2M Machine-to-Machine.
CIP Cloud-Indirect-Processed.	MAS Management, Abstraction, Semantics.
CIR Cloud-Indirect-Raw.	MEC Mobile Edge Computing.
CoAP Constrained Application Protocol.	MQTT Message Queue Telemetry Transport.
CVO Composite Virtual Object.	MSM Minimal Service Model.
DB database.	NTP Network Time Protocol.
DI Data/Information.	OBDA Ontology-Based Data Access.
DIKW Data, Information, Knowledge and Wisdom.	ODP Ontology Design Pattern.
ECA Event-Condition-Action.	OPA Open Platform for ADREAM.
EDR Emergent Distributed Reasoning.	OWL Web Ontology Language.
EWS Early Warning System.	OWS Object With States.
GUI Graphical User Interface.	PSW Physical Semantic Web.
IIoT Industrial IoT.	PW Physical Web.
	QoS Quality of Service.
	RDB Relational Database.

- RDF** Ressource Description Framework.
- RFID** Radio Frequency Identification.
- SAN** Semantic Actuator Network.
- SAREF** Smart Appliance REFERENCE.
- SDO** Standard Developing Orgaization.
- SSN** Semantic Sensor Network.
- SSO** Stimulus Sensor Observation.
- SW** Semantic Web.
- SWoT** Semantic Web Of Things.
- VO** Virtual Object.
- W3C** World Wide Web Consortium.
- WoT** Web Of Things.

A.4 Image credits

- : Fan by Douglas Santos from the Noun Project
- : Display by Ralf Schmitzer from the Noun Project
- : Server loading by Chunk Icons from the Noun Project
- : Human Sensor by Antoine Dieulesaint from the Noun Project
- Miscellaneous icons from FontAwesome

A.5 Miscellaneous remarks

- All the URL featured in the present thesis have been last visited on the 14th of August, 2018.

Bibliography

- [Abrás 2004] Chadia Abrás, Diane Maloney-Krichmar and Jenny Preece. *User-centered design*. Bainbridge, W. Encyclopedia of Human-Computer Interaction. Thousand Oaks: Sage Publications, vol. 37, no. 4, pages 445–456, 2004. (Cited in pages v, 1, 2 and 54.)
- [Aïssaoui 2016] François Aïssaoui, Guillaume Garzone and Nicolas Seydoux. *Providing interoperability for autonomic control of connected devices*. In *InterIoT*, pages 1–7, 2016. (Cited in pages 66, 73, 74, 80, 83 and 162.)
- [Al-Hazmi 2015] Y. Al-Hazmi and T. Magedanz. *MOFI: Monitoring ontology for federated infrastructures*. In 2015 IEEE International Workshop on Measurements Networking (M N), pages 1–6, Oct 2015. (Cited in page 34.)
- [Al-Osta 2017] Mahmud Al-Osta, Bali Ahmed and Gherbi Abdelouahed. *A Lightweight Semantic Web-based Approach for Data Annotation on IoT Gateways*. *Procedia Computer Science*, vol. 113, pages 186–193, 2017. (Cited in pages 74 and 75.)
- [Alam 2010] Sarfraz Alam, Mohammad M.R. Chowdhury and Josef Noll. *Senaas: An event-driven sensor virtualization approach for internet of things cloud*. In 2010 IEEE International Conference on Networked Embedded Systems for Enterprise Applications, NESEA 2010, 2010. (Cited in page 165.)
- [Aquin 2012] Mathieu Aquin. *Modularizing ontologies*. *Ontology engineering in a networked world*, vol. Springer B, pages 9–34, 2012. (Cited in pages vi and 35.)
- [Ara 2014] Safina Showkat Ara, Zia Ush Shamszaman and Ilyoung Chong. *Web-of-objects based user-centric semantic service composition methodology in the internet of things*. *International Journal of Distributed Sensor Networks*, vol. 2014, 2014. (Cited in pages 74, 80, 84 and 86.)
- [Ashraf 2010] Muhammad Ikram Ashraf, Leonardo Goratti, Jussi Haapola and Carlos Pomalaza-Raez. *Distributed semantic algorithm for power constrained publish/subscribe routing*. In 2010 Sixth International Conference on Intelligent Sensors, Sensor Networks and Information Processing, pages 91–96. IEEE, dec 2010. (Cited in pages 87, 89 and 97.)
- [Ashton 2009a] Kevin Ashton. *That ‘internet of things’ thing*. *RFID journal*, vol. 22, no. 7, pages 97–114, 2009. (Cited in page 4.)
- [Ashton 2009b] Kevin Ashton. *That ‘internet of things’ thing*. *RFID journal*, vol. 22, no. 7, pages 97–114, 2009. (Cited in pages 8, 9, 12 and 68.)

- [Atzori 2010] Luigi Atzori, Antonio Iera and Giacomo Morabito. *The Internet of Things: A survey*. Computer Networks, vol. 54, no. 15, pages 2787–2805, oct 2010. (Cited in page 72.)
- [Avancha 2004] Sasikanth Avancha, Chintan Patel and Anupam Joshi. *Ontology-driven adaptive sensor networks*. Proceedings of MOBIQUITOUS 2004 - 1st Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, no. Section 2, pages 194–202, 2004. (Cited in page 30.)
- [Barnaghi 2009] Payam Barnaghi, Stefan Meissner, Mirko Presser and Klaus Moessner. *Sense and sens' ability: Semantic data modelling for sensor networks*, 2009. (Cited in pages 20, 39, 74 and 75.)
- [Barnaghi 2012] Payam Barnaghi, Wei Wang, Cory Henson and Kerry Taylor. *Semantics for the Internet of Things: early progress and back to the future*. International Journal on Semantic Web and Information Systems, vol. 8, no. 1, pages 1–21, 2012. (Cited in pages 2, 39 and 72.)
- [Bazoobandi 2015] Hamid R. Bazoobandi, Steven De Rooij, Jacopo Urbani, Annette Ten Teije, Frank Van Harmelen and Henri Bal. *A Compact In-Memory Dictionary for RDF Data*. In ESWC, volume 9088, pages 205–220, 2015. (Cited in page 97.)
- [Ben-Alaya 2015] Mahdi Ben-Alaya, Samir Medjiah, Thierry Monteil and Khalil Drira. *Toward semantic interoperability in oneM2M architecture*. IEEE Communications Magazine, vol. 53, no. 12, pages 35–41, 2015. (Cited in pages viii, 18, 19, 20, 23, 33, 39, 45, 80, 85 and 110.)
- [Bermudez-Edo 2017] Maria Bermudez-Edo, Tarek Elsaleh, Payam Barnaghi and Kerry Taylor. *IoT-Lite: a lightweight semantic model for the internet of things and its use with dynamic semantics*. Personal and Ubiquitous Computing, vol. 21, no. 3, pages 475–487, 2017. (Cited in pages 32 and 36.)
- [Berners-Lee 2001] Tim Berners-Lee, Jim Hendler and Ora Lasilla. *The Semantic Web*. Scientific American, vol. 284, no. 5, pages 34–43, 2001. (Cited in pages 2, 4, 12, 30 and 68.)
- [Bischof 2012] Stefan Bischof, Stefan Decker, Thomas Krennwallner, Nuno Lopes and Axel Polleres. *Mapping between RDF and XML with XSPARQL*. Journal on Data Semantics, vol. 1, no. 3, pages 147–185, 2012. (Cited in pages 62 and 65.)
- [Boardman 2006] John Boardman and Brian Sauser. *The meaning of System of Systems*. In 2006 IEEE/SMC International Conference on System of Systems Engineering, pages 118–123, 2006. (Cited in page 167.)

- [Boldt 2015] Dennis Boldt, Henning Hasemann, Marcel Karnstedt, Alexander Kroller and Christian Von Der Weth. *SPARQL for networks of embedded systems*. Proceedings - 2015 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology, WI-IAT 2015, vol. 1, pages 93–100, 2015. (Cited in pages 87 and 88.)
- [Boley 2007] Harold Boley, Michael Kifer, Paula-Lavinia Pătrânjan and Axel Polleres. *Rule interchange on the web*. In Reasoning Web International Summer School, pages 269–309. Springer, 2007. (Cited in page 105.)
- [Bonomi 2012] Flavio Bonomi, Rodolfo Milito, Jiang Zhu and Sateesh Addepalli. *Fog Computing and Its Role in the Internet of Things*. In Proceedings of the first edition of the MCC workshop on Mobile cloud computing, pages 13–16, New York, New York, USA, 2012. ACM Press. (Cited in pages 21 and 68.)
- [Bousselmi 2014] Khadija Bousselmi, Zaki Brahmi and Mohamed Mohsen Gammoudi. *Cloud services orchestration: A comparative study of existing approaches*. In Advanced Information Networking and Applications Workshops (WAINA), 2014 28th International Conference on, pages 410–416. IEEE, 2014. (Cited in page 98.)
- [Bovet 2014] Gerome Bovet and Jean Hennebert. *Distributed semantic discovery for web-of-things enabled smart buildings*. In 2014 6th International Conference on New Technologies, Mobility and Security - Proceedings of NTMS 2014 Conference and Workshops, 2014. (Cited in pages 80, 81 and 87.)
- [Butler 2010] James Butler and Robert Veelenturf. *Point naming standards*. ASHRAE Journal, vol. 53, no. 2, page 14, 2010. (Cited in page 49.)
- [Cabé 2018] Benjamin Cabé. *Key Trends from the IoT Developer Survey*, 2018. (Cited in page 2.)
- [Charpenay 2015] Victor Charpenay, Sebastian Kabisch, Darko Anicic and Harald Kosch. *An ontology design pattern for IoT device tagging systems*. In Proceedings - 2015 5th International Conference on the Internet of Things, IoT 2015, 2015. (Cited in page 95.)
- [Charpenay 2018] Victor Charpenay, K Sebastian and Harald Kosch. *Towards a Binary Object Notation for RDF*. In Extended Semantic Web Conference, page 15, 2018. (Cited in pages 87, 88, 94 and 97.)
- [Charrada 2016] Faouzi Ben Charrada and Samir Tata. *An Efficient Algorithm for the Bursting of Service-Based Applications in Hybrid Clouds*. IEEE Transactions on Services Computing, vol. 9, no. 3, pages 357–367, may 2016. (Cited in pages 98 and 103.)
- [Chatzigiannakis 2012] Ioannis Chatzigiannakis, Henning Hasemann, Marcel Karnstedt, Oliver Kleine, Alexander Kröller, Myriam Leggieri, Dennis Pfisterer,

- Kay Römer and Cuong Truong. *True Self-Configuration for the IoT*. In 3rd International Conference on the Internet of Things (IOT), 2012. (Cited in pages 19 and 107.)
- [Christian Bizer 2009] Tom Heath Christian Bizer Tim Berners-Lee. *Linked Data - The Story So Far*. International Journal on Semantic Web and Information Systems (IJSWIS), vol. 5(3), pages 1–22, 2009. (Cited in page 49.)
- [Christophe 2011] Benoit Christophe, Mathieu Boussard, Monique Lu, Alain Pastor and Vincent Toubiana. *The web of things vision: Things as a service and interaction patterns*. Bell Labs Technical Journal, vol. 16, no. 1, pages 55–61, jun 2011. (Cited in pages 80, 85, 86 and 87.)
- [Colombo-Mendoza 2012] Luis Colombo-Mendoza and María Salas-Zárate. *Cloud Computing: A review of PaaS, IaaS, SaaS Service and Providers*. Lámp-sakos, no. 7, pages 47–57, 2012. (Cited in page 98.)
- [Compton 2009a] Michael Compton, Cory Henson, Laurent Lefort, Holger Neuhaus and Amit Sheth. *A survey of the semantic specification of sensors*. In International Conference on Semantic Sensor Networks, volume 522, pages 17–32, 2009. (Cited in pages 20, 71, 73, 74, 80 and 81.)
- [Compton 2009b] Michael Compton, Holger Neuhaus, Kerry Taylor and KN Tran. *Reasoning about Sensors and Compositions*. In 2nd International Semantic Sensor Networks Workshop, pages 33–48, 2009. (Cited in pages 19 and 39.)
- [Corby 2015] Olivier Corby, Catherine Faron-Zucker and Fabien Gandon. *A generic RDF transformation software and its application to an online translation service for common languages of linked data*. In ISWC, volume 9367, pages 150–165, 2015. (Cited in page 62.)
- [Corcho 2010] Oscar Corcho and Raúl García-Castro. *Five challenges for the Semantic Sensor Web*. Semantic Web, vol. 1, no. 1, pages 121–125, 2010. (Cited in pages 4 and 95.)
- [Costea 2016] Cristinel Costea, Liviu Neamt, Olivian Chiver, Electric Engineering, Cristian Cola and Vasile Sambor. *Distributed processing of data streams on the edge devices*. no. Epe, pages 20–22, 2016. (Cited in pages 74, 77 and 87.)
- [Daniele 2016] Laura Daniele, Monika Solanki, Frank Den Hartog and Jasper Roes. *Interoperability for Smart Appliances in the IoT World*. In ISWC, 2016. (Cited in page 32.)
- [Dastjerdi 2016] Amir Vahid Dastjerdi and Rajkumar Buyya. *Fog Computing: Helping the Internet of Things Realize Its Potential*. Computer, vol. 49, no. 8, pages 112–116, 2016. (Cited in page 22.)

- [Datta 2015] Soumya Kanti Datta and Christian Bonnet. *Connect and Control Things : Integrating Lightweight IoT Framework into a Mobile Application*. In 9th International Conference on Next Generation Mobile Applications, Services and Technologies, number SEPTEMBER 2015, page 6, 2015. (Cited in page 74.)
- [del Carmen Suarez de Figueroa Baonza 2010] Maria del Carmen Suarez de Figueroa Baonza. *NeOn methodology for building ontology networks : specification, shedding and reuse*. PhD thesis, 2010. (Cited in pages vi, 33 and 35.)
- [D’Elia 2017] Alfredo D’Elia, Fabio Viola, Luca Roffia, Paolo Azzoni and Tullio Salmon Cinotti. *Enabling Interoperability in the Internet of Things*. International Journal on Semantic Web and Information Systems, vol. 13, no. 1, pages 147–167, 2017. (Cited in pages 74 and 77.)
- [Desai 2015] Pratikkumar Desai, Amit Sheth and Pramod Anantharam. *Semantic Gateway as a Service architecture for IoT Interoperability*. In Kno.e.sis Publications, 2015. (Cited in pages 1, 20, 74, 75 and 107.)
- [Dey 2017] Sounak Dey, Abhijan Bhattacharyya and Arijit Mukherjee. *Semantic data exchange between collaborative robots in fog environment: Can CoAP be a choice?* GIoTS 2017 - Global Internet of Things Summit, Proceedings, 2017. (Cited in pages 74, 80, 81, 86 and 87.)
- [Euzenat 2008] Jérôme Euzenat, Axel Polleres and François Scharffe. *Processing Ontology Alignments with SPARQL*. In 2008 International Conference on Complex, Intelligent and Software Intensive Systems, pages 913–917. IEEE, 2008. (Cited in page 52.)
- [Evans 2012] Dave Evans. *The internet of everything: How more relevant and valuable connections will change the world*. Cisco IBSG, vol. 2012, pages 1–9, 2012. (Cited in page 4.)
- [Evchina 2015] Yulia Evchina, Juha Puttonen, Aleksandra Dvoryanchikova and José Luis Martínez Lastra. *Context-aware knowledge-based middleware for selective information delivery in data-intensive monitoring systems*. Engineering Applications of Artificial Intelligence, vol. 43, pages 111–126, 2015. (Cited in pages 106 and 107.)
- [Fensel 2017] Anna Fensel, Dana Kathrin Tomic and Andreas Koller. *Contributing to appliances’ energy efficiency with Internet of Things, smart data and user engagement*. Future Generation Computer Systems, vol. 76, pages 329–338, 2017. (Cited in pages 74, 78, 79, 80 and 92.)
- [Ferdinand 2004] Matthias Ferdinand, Christian Zirpins and David Trastour. *Lifting XML Schema to OWL*. Web Engineering, no. 3140, pages 354–358, 2004. (Cited in pages 61 and 75.)

- [Foteinos 2013] Vassilis Foteinos, Dimitris Kelaidonis, George Poullos, Panagiotis Vlacheas, Vera Stavroulaki and Panagiotis Demestichas. *Cognitive management for the internet of things: A framework for enabling autonomous applications*. IEEE Vehicular Technology Magazine, vol. 8, no. 4, pages 90–99, 2013. (Cited in pages 2, 80, 82 and 93.)
- [Fredj 2013] Sameh Ben Fredj, Mathieu Boussard, Daniel Kofman and Ludovic Noirie. *A Scalable IoT Service Search Based on Clustering and Aggregation*. In 2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing, pages 403–410. IEEE, aug 2013. (Cited in pages 86, 87 and 89.)
- [Gangemi 2005] Aldo Gangemi. *Ontology Design Patterns for Semantic Web Content*. History, vol. 3729, no. 4, pages 262–276, 2005. (Cited in pages vi, 35 and 36.)
- [Gassara 2017] Amal Gassara, Ismael Bouassida Rodriguez, Mohamed Jmaiel and Khalil Drira. *A bigraphical multi-scale modeling methodology for system of systems*. Computers and Electrical Engineering, vol. 58, pages 113–125, 2017. (Cited in page 167.)
- [Gorlatova 2014] M Gorlatova, J Sarik, G Grebla, M Cong and ... *Movers and shakers: Kinetic energy harvesting for the internet of things*. Acm Sigmetrics ..., pages 407–419, 2014. (Cited in page 9.)
- [Gruber 1991] Tr Gruber. *The role of common ontology in achieving sharable, reusable knowledge bases*. Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference, pages 601–602, 1991. (Cited in pages vi, 4 and 13.)
- [Guinard 2009] Dominique Guinard, Vlad Trifa, Thomas Pham and Olivier Liechti. *Towards physical mashups in the web of things*. INSS2009 - 6th International Conference on Networked Sensing Systems, pages 196–199, 2009. (Cited in page 16.)
- [Gyrard 2015] Amelie Gyrard, Martin Serrano and Ghislain A. Atemezing. *Semantic web methodologies, best practices and ontology engineering applied to Internet of Things*. In 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT), pages 412–417. IEEE, 2015. (Cited in pages 26, 27 and 31.)
- [Gyrard 2016] Amélie Amelie Gyrard, Pankesh Patel, Amit Sheth and Martin Serrano. *Building the Web of Knowledge with Smart IoT Applications*. IEEE Intelligent Systems, vol. 31, 2016. (Cited in pages 74 and 77.)
- [Gyrard 2017] Amelie Gyrard, Martin Serrano, Joao Bosco Jares, Soumya Kanti Datta and Muhammad Intizar Ali. *Sensor-based Linked Open Rules (S-LOR): An Automated Rule Discovery Approach for IoT Applications and its*

- use in Smart Cities*. In Proceedings of the 26th International Conference on World Wide Web Companion, pages 1153–1159. International World Wide Web Conferences Steering Committee, 2017. (Cited in pages 74, 77, 80, 105, 164 and 166.)
- [Hachem 2011] Sara Hachem, Thiago Teixeira and Valérie Issarny. *Ontologies for the internet of things*. In Proceedings of the 8th Middleware Doctoral Symposium on - MDS '11, number June 2009, pages 1–6, New York, New York, USA, dec 2011. ACM Press. (Cited in page 39.)
- [Hachem 2014] Sara Hachem. *Service-Oriented middleware for the large-scale mobile Internet of Things*. PhD thesis, Université de Versailles-Saint Quentin en Yvelines, 2014. (Cited in page 34.)
- [Han 2012] Son N. Han, Gyu Myoung Lee and Noel Crespi. *Towards Automated Service Composition Using Policy Ontology in Building Automation System*. In 2012 IEEE Ninth International Conference on Services Computing, pages 685–686, 2012. (Cited in page 19.)
- [Han 2014] Son N. Han, Gyu Myoung Lee and Noel Crespi. *Semantic Context-Aware Service Composition for Building Automation System*. IEEE Transactions on Industrial Informatics, vol. 10, no. 1, pages 752–761, feb 2014. (Cited in pages 80, 82, 83 and 86.)
- [Han 2017] Son N. Han and Noel Crespi. *Semantic service provisioning for smart objects: Integrating IoT applications into the web*. Future Generation Computer Systems, vol. 76, pages 180–197, 2017. (Cited in pages 74 and 80.)
- [Hasemann 2012] Henning Hasemann, Alexander Kröllner and Max Pagel. *RDF provisioning for the internet of things*. In Proceedings of 2012 International Conference on the Internet of Things, IOT 2012, pages 143–150. IEEE, oct 2012. (Cited in pages 74, 87, 88, 98 and 107.)
- [He 2012] Jing He, Yanchun Zhang, Guangyan Huang and Jinli Cao. *A smart web service based on the context of things*. ACM Transactions on Internet Technology, vol. 11, no. 3, pages 1–23, 2012. (Cited in pages 74, 80 and 83.)
- [Henson 2012] Cory Henson, Amit Sheth and Krishnaprasad Thirunarayan. *Semantic perception: Converting sensory observations to abstractions*. IEEE Internet Computing, vol. 16, no. 2, pages 26–34, 2012. (Cited in page 76.)
- [Howell 2017] Shaun Howell, Yacine Rezgui and Thomas Beach. *Integrating building and urban semantics to empower smart water solutions*. Automation in Construction, vol. 81, pages 434–448, 2017. (Cited in pages 74, 87, 92 and 95.)
- [Hussein 2016] Dina Hussein, Son N. Han, Gyu Myoung Lee, Noel Crespi and Emmanuel Bertin. *Towards a dynamic discovery of smart services in the social*

- internet of things*. Computers & Electrical Engineering, 2016. (Cited in pages 30, 80, 86, 106 and 107.)
- [itu 2012a] *Framework of the web of things*. Technical Report, International Telecommunication Union, 2012. (Cited in page 16.)
- [itu 2012b] *Overview of the web of things (Y.2060)*. Technical Report, International Telecommunication Union, 2012. (Cited in page 8.)
- [Janowicz 2010] Krzysztof Janowicz and Michael Compton. *The Stimulus-Sensor-Observation Ontology Design Pattern and its Integration into the Semantic Sensor Network Ontology*. In Proceedings of the 9th International Semantic Web Conference, 3rd International Workshop on Semantic Sensor Networks, pages 7–11, 2010. (Cited in page 42.)
- [Jara 2014] Antonio J. Jara, Alex C. Olivieri, Yann Bocchi, Markus Jung, Wolfgang Kastner and Antonio F. Skarmeta. *Semantic Web of Things: an analysis of the application semantics for the IoT moving towards the IoT convergence*. Int. J. of Web and Grid Services, vol. 10, no. 2/3, 2014. (Cited in page 72.)
- [Jurdak 2004] Raja Jurdak, Cristina Videira Lopes and Pierre Baldi. *A framework for modeling sensor networks*, 2004. (Cited in page 30.)
- [Kaed 2016] Charbel El Kaed, Imran Khan, Hicham Hossayni and Philippe Nappey. *SQenloT: Semantic query engine for industrial Internet-of-Things gateways*. 2016 IEEE 3rd World Forum on Internet of Things, WF-IoT 2016, pages 204–209, 2016. (Cited in pages 74, 80, 83, 87, 88 and 108.)
- [Kaed 2018a] Charbel El Kaed, Imran Khan, Andre Van Den Berg, Hicham Hossayni and Christophe Saint-Marcel. *SRE: Semantic Rules Engine for the Industrial Internet-Of-Things Gateways*. IEEE Trans. Industrial Informatics, vol. 14, no. 2, pages 715–724, 2018. (Cited in page 107.)
- [Kaed 2018b] Charbel El Kaed, Imran Khan, Andre Van Den Berg, Hicham Hossayni and Christophe Saint-Marcel. *SRE : Semantic Rules Engine For the Industrial Internet- Of-Things Gateways*. IEEE Transactions on Industrial Informatics, vol. 14, no. 2, pages 715–724, 2018. (Cited in pages 74, 80, 81, 83, 87, 108 and 168.)
- [Kamalinejad 2015] P Kamalinejad, C Mahapatra, Zhengguo Sheng, S Mirabbasi, V C M Leung and Yong Liang Guan. *Wireless energy harvesting for the Internet of Things*. IEEE Communications Magazine, vol. 53, no. 6, pages 102–108, 2015. (Cited in page 9.)
- [Kasnesis 2015] Panagiotis Kasnesis, Charalampos Z. Patrikakis and Iakovos S. Venieris. *Collective domotic intelligence through dynamic injection of semantic rules*. In IEEE International Conference on Communications, volume 2015-Septe, pages 592–597, 2015. (Cited in pages 106 and 168.)

- [Kelaidonis 2016] Dimitrios Kelaidonis, Angelos Rouskas, Vera Stavroulaki, Panagiotis Demestichas and Panagiotis Vlachas. *A federated Edge Cloud-IoT architecture*. EUCNC 2016 - European Conference on Networks and Communications, pages 230–234, 2016. (Cited in pages 74, 80, 82, 84 and 87.)
- [Kephart 2003] J.O. Kephart and D.M. Chess. *The vision of autonomic computing*. Computer, vol. 36, no. 1, pages 41–50, jan 2003. (Cited in pages vii and 55.)
- [Khan 2015] Imran Khan, Rifat Jafrin, Fatima Zahra Errounda, Roch Glitho, Noël Crespi, Monique Morrow and Paul Polakos. *A Data Annotation Architecture for Semantic Applications in Virtualized Wireless Sensor Networks*. 2015. (Cited in pages 74, 75, 76, 80, 81 and 87.)
- [Khandelwal 2011] Ankesh Khandelwal, Ian Jacobi and Lalana Kagal. *Linked rules: Principles for rule reuse on the web*. Lecture Notes in Computer Science, vol. 6902 LNCS, pages 108–123, 2011. (Cited in pages 105 and 120.)
- [Kharlamov 2016] Evgeny Kharlamov, Yannis Kotidis, Theofilos Mailis, Christian Neuenstadt, Charalampos Nikolaou, Özgür Özcep, Christoforos Svingos, Dmitriy Zheleznyakov, Steffen Lamparter, Ian Horrocks, Yannis Ioannidis and Ralf Möller. *Towards Analytics Aware Ontology Based Access to Static and Streaming Data (Extended Version)*. In ISWC, pages 1–22, 2016. (Cited in pages 74, 78, 80, 87 and 88.)
- [Kibria 2015] Muhammad Golam Kibria. *Knowledge based open IoT service provisioning through cooperation between physical web and WoO*. In 2015 Seventh International Conference on Ubiquitous and Future Networks, pages 395–400. IEEE, 2015. (Cited in pages 30, 80, 82, 92 and 95.)
- [Kifer 2013] Michael Kifer and Harold Boley. *RIF Overview (Second Edition)*. W3C note, W3C, February 2013. <http://www.w3.org/TR/2013/NOTE-rif-overview-20130205/>. (Cited in page 15.)
- [Kiljander 2014] Jussi Kiljander, Alfredo D’elia, Francesco Morandi, Pasi Hyttinen, Janne Takalo-Mattila, Arto Ylisaukko-Oja, Juha-Pekka Soininen and Tullio Salmon Cinotti. *Semantic Interoperability Architecture for Pervasive Computing and Internet of Things*. IEEE Access, vol. 2, pages 856–873, 2014. (Cited in pages 77, 80, 83 and 84.)
- [Kitchenham 2004] Barbara Kitchenham. *Procedures for performing systematic reviews*. Keele, UK, Keele University, vol. 33, no. TR/SE-0401, page 28, 2004. (Cited in page 69.)
- [Koestler 1967] Arthur Koestler. *The ghost in the machine*. Hutchinson, 1967. (Cited in page 167.)

- [Kopecký 2007] Jacek Kopecký, Tomas Vitvar, Carine Bournez and Joel Farrell. *SAWSDL: Semantic Annotations for WSDL and XML Schema*. IEEE Internet Computing, vol. 11, no. 6, pages 60–67, nov 2007. (Cited in pages 60, 62 and 94.)
- [Kopecký 2008] Jacek Kopecký, Karthik Gomadam and Tomas Vitvar. *hRESTS: An HTML microformat for describing RESTful Web services*. Proceedings - 2008 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2008, pages 619–625, 2008. (Cited in page 41.)
- [Köpke 2010] Julius Köpke and Johann Eder. *Semantic annotation of XML-schema for document transformations*. On the Move to Meaningful Internet Systems: OTM 2010 Workshops, vol. 6428 LNCS, pages 219–228, 2010. (Cited in pages 62 and 65.)
- [Kotis 2012a] Konstantinos Kotis and Artem Katasonov. *An ontology for the automated deployment of applications in heterogeneous IoT*. Semantic web journal, 2012. (Cited in page 34.)
- [Kotis 2012b] Konstantinos Kotis and Artem Katasonov. *Semantic interoperability on the Web of things: The semantic smart gateway framework*. Proceedings - 2012 6th International Conference on Complex, Intelligent, and Software Intensive Systems, CISIS 2012, no. May, pages 630–635, 2012. (Cited in page 95.)
- [Kotis 2012c] Konstantinos Kotis, Artem Katasonov and Jarkko Leino. *Aligning Smart and Control Entities in the IoT*. In Sergey Andreev, Sergey Balandin and Yevgeni Koucheryavy, editors, Internet of Things, Smart Spaces, and Next Generation Networking: 12th International Conference, NEW2AN 2012, and 5th Conference, ruSMART 2012, St. Petersburg, Russia, August 27-29, 2012. Proceedings, volume 7469 LNCS, pages 39–50, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. (Cited in page 95.)
- [Lannes 2017] Mathilde Lannes, Fabien Amarger, **Nicolas Seyoux** and Nathalie Hernandez. *Du langage naturel à la connaissance il n’y a qu’un pas : SWIP*. In IC, pages 229–232, Caen, France, 2017. (Cited in page 168.)
- [Le-Phuoc 2011] Danh Le-Phuoc, Hoan Quoc, Josiane Xavier Parreira and Manfred Hauswirth. *The linked sensor middleware—connecting the real world and the semantic web*. In Semantic Web Challenge 2011, number April 2005, pages 1–8, 2011. (Cited in page 58.)
- [Le-Phuoc 2012] Danh Le-Phuoc, Hoan Quoc Nguyen-Mau, Josiane Xavier Parreira and Manfred Hauswirth. *A middleware framework for scalable management of linked streams*. Web Semantics: Science, Services and Agents on the World Wide Web, vol. 16, pages 42–51, 2012. (Cited in pages 74, 79, 87 and 89.)

- [Le-Phuoc 2016] Danh Le-Phuoc, Hoan Nguyen Mau Quoc, Hung Ngo Quoc, Tuan Tran Nhat and Manfred Hauswirth. *The Graph of Things: A step towards the Live Knowledge Graph of connected things*. Web Semantics: Science, Services and Agents on the World Wide Web, 2016. (Cited in pages 20, 74, 78, 79, 87, 89 and 92.)
- [Lee 2016] Yann Hang Lee and Shankar Nair. *A Smart Gateway Framework for IOT Services*. Proceedings - 2016 IEEE International Conference on Internet of Things; IEEE Green Computing and Communications; IEEE Cyber, Physical, and Social Computing; IEEE Smart Data, iThings-GreenCom-CPSCoM-Smart Data 2016, pages 107–114, 2016. (Cited in pages 80, 82, 83, 84 and 107.)
- [Lefort 2011] Laurent Lefort and Cory Henson. *Semantic sensor network xg final report*. Technical Report June 2011, 2011. (Cited in page 31.)
- [Lefrançois 2017] Maxime Lefrançois, Antoine Zimmermann and Noorani Bakerally. *A SPARQL extension for generating RDF from heterogeneous formats*. In European Semantic Web Conference, pages 35–50. Springer, 2017. (Cited in page 50.)
- [Lemaignan 2012] Séverin Lemaignan. *Grounding the Interaction: Knowledge Management for Interactive Robots*. PhD thesis, 2012. (Cited in page 54.)
- [Li 2010] Zang Li, Chao Hsien Chu, Wen Yao and Richard a. Behr. *Ontology-driven event detection and indexing in smart spaces*. In Proceedings - 2010 IEEE 4th International Conference on Semantic Computing, ICSC 2010, pages 285–292, 2010. (Cited in page 105.)
- [Li 2015] Juan Li, Nazia Zaman and Honghui Li. *A Decentralized Locality-Preserving Context-Aware Service Discovery Framework for Internet of Things*. In 2015 IEEE International Conference on Services Computing, pages 317–323. IEEE, jun 2015. (Cited in pages 30 and 80.)
- [Lillo 2015] Paolo Lillo, Luca Mainetti, Vincenzo Mighali, Luigi Patrono and Piercosimo Rametta. *A Novel Rule-based Semantic Architecture for IoT Building Automation Systems*. In International Conference on Software, Telecommunications and Computer Networks (SoftCOM), volume 12, pages 124–131. IEEE, sep 2015. (Cited in page 106.)
- [Liu 2015] Jiaqiang Liu, Yong Li, Min Chen, Wenxia Dong and Depeng Jin. *Software-defined internet of things for smart urban sensing*. IEEE Communications Magazine, vol. 53, no. 9, pages 55–63, 2015. (Cited in pages viii, 19 and 23.)
- [Loseto 2016] Giuseppe Loseto, Saverio Ieva, Filippo Gramegna, Michele Ruta, Floriano Scioscia, Eugenio Di Sciascio and Bari I. *Linked Data (in low-resource*

-) *Platforms : a mapping for Constrained Application Protocol*. In ISWC, Kobe, 2016. (Cited in pages 87, 88 and 97.)
- [Maarala 2017] Altti Ilari Maarala, Xiang Su and Jukka Riekkii. *Semantic Reasoning for Context-aware Internet of Things Applications*. IEEE Internet of Things Journal, 2017. (Cited in pages 74, 77, 94, 102, 107, 108 and 123.)
- [Martini 2015] Barbara Martini, Federica Paganelli, Paola Cappanera, Stefano Turchi and Piero Castoldi. *Latency-aware composition of Virtual Functions in 5G*. 1st IEEE Conference on Network Softwarization: Software-Defined Infrastructures for Networks, Clouds, IoT and Services, NETSOFT 2015, 2015. (Cited in page 96.)
- [Mathew 2014] Sujith Samuel Mathew, Yacine Atif, Quan Z. Sheng and Zakaria Maamar. *Building sustainable parking lots with the Web of Things*. Personal and Ubiquitous Computing, vol. 18, no. 4, pages 895–907, 2014. (Cited in pages 74, 75, 80 and 82.)
- [Mayer 2013] Simon Mayer and Gianin Basler. *Semantic metadata to support device interaction in smart environments*. Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication - UbiComp '13 Adjunct, pages 1505–1514, 2013. (Cited in pages 80 and 85.)
- [Mell 2011] Peter Mell and Timothy Grance. *The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology*. National Institute of Standards and Technology, Information Technology Laboratory, vol. 145, page 7, 2011. (Cited in page 21.)
- [Miorandi 2012] Daniele Miorandi, Sabrina Sicari, Francesco De Pellegrini and Imrich Chlamtac. *Internet of things: Vision, applications and research challenges*. Ad hoc networks, vol. 10, no. 7, pages 1497–1516, 2012. (Cited in page 165.)
- [moz 2018] *Internet Health Report*. Technical Report, Mozilla Foundation, 2018. (Cited in page 2.)
- [Mrissa 2015] M Mrissa, L Medini, J.-P. Jamont, N Le Sommer and J Laplace. *An Avatar Architecture for the Web of Things*. Internet Computing, IEEE, vol. 19, no. 2, pages 30–38, 2015. (Cited in pages 80 and 81.)
- [Murdock 2016] Paul Murdock and al. *Semantic Interoperability for the Web of Things*, 2016. (Cited in pages 4, 30 and 65.)
- [Nachabe 2015] Lina Nachabe, Marc Girod-Genet and Bachar El Hassan. *Unified Data Model for Wireless Sensor Network*. IEEE Sensors Journal, vol. 15, no. 7, pages 3657–3667, jul 2015. (Cited in pages 30 and 39.)

- [Nachabe 2016] Lina Nachabe, Marc Girod-Genet and Bachar El Hassan. *Semantic Techniques for IOT Data and Service Management on to Smart System*. International Journal of Wireless & Mobile Networks, vol. 8, no. 4, pages 43–63, 2016. (Cited in page 80.)
- [Nagib 2016] Ahmad M. Nagib and Haitham S. Hamza. *SIGHTED: A Framework for Semantic Integration of Heterogeneous Sensor Data on the Internet of Things*. Procedia Computer Science, vol. 83, no. Ant, pages 529–536, 2016. (Cited in pages 74, 87 and 88.)
- [Nikoli 2011] Siniša Nikoli, Valentin Penca and Zora Konjovi. *Semantic Web Based Architecture for Managing Hardware Heterogeneity in Wireless Sensor Network*. In International Journal of Computer Science and Applications, volume 8, pages 38–58, 2011. (Cited in pages 20, 80, 85 and 115.)
- [Nitti 2016] Michele Nitti, Virginia Pilloni, Giuseppe Colistra and Luigi Atzori. *The Virtual Object as a Major Element of the Internet of Things: A Survey*. IEEE Communications Surveys and Tutorials, vol. 18, no. 2, pages 1228–1240, 2016. (Cited in pages 18 and 81.)
- [Oliveira 2013] F. A. Oliveira, T. Ledoux and R. Sharrock. *A Framework for the Coordination of Multiple Autonomic Managers in Cloud Environments*. In 2013 IEEE 7th International Conference on Self-Adaptive and Self-Organizing Systems(SASO), volume 00, pages 179–188, Sept. 2013. (Cited in page 167.)
- [Patel 2017] Pankesh Patel, Muhammad Intizar Ali and Amit Sheth. *On Using the Intelligent Edge for IoT Analytics*. IEEE Intelligent Systems, vol. 32, no. 5, pages 64–69, sep 2017. (Cited in pages 22, 68, 85 and 98.)
- [Pease 2017] Sarogini Grace Pease, Paul P. Conway and Andrew A. West. *Hybrid ToF and RSSI real-time semantic tracking with an adaptive industrial internet of things architecture*. Journal of Network and Computer Applications, vol. 99, no. August 2016, pages 98–109, 2017. (Cited in pages 30, 74, 77, 87 and 92.)
- [Perera 2014a] Charith Perera, Arkady Zaslavsky, Peter Christen and Dimitrios Georgakopoulos. *Context aware computing for the internet of things: A survey*. IEEE Communications Surveys and Tutorials, vol. 16, no. 1, pages 414–454, jan 2014. (Cited in pages 18 and 126.)
- [Perera 2014b] Charith Perera, Arkady Zaslavsky, Chi Harold Liu, Michael Compton, Peter Christen and Dimitrios Georgakopoulos. *Sensor search techniques for sensing as a service architecture for the internet of things*. IEEE Sensors Journal, vol. 14, no. 2, pages 406–420, 2014. (Cited in pages 80 and 86.)
- [Perera 2016] Charith Perera and Athanasios V. Vasilakos. *A knowledge-based resource discovery for Internet of Things*. Knowledge-Based Systems, vol. 109, 2016. (Cited in pages 74, 80, 84, 87 and 92.)

- [Perera 2017] Charith Perera, Yongrui Qin, Julio C. Estrella, Stephan Reiff-Marganiec and Athanasios V. Vasilakos. *Fog Computing for Sustainable Smart Cities: A Survey*. vol. 50, no. 3, 2017. (Cited in page 72.)
- [Pfisterer 2011] Dennis Pfisterer, Kay Romer, Daniel Bimschas, Oliver Kleine, Richard Mietz, Cuong Truong, Henning Hasemann, Alexander Kröller, Max Pagel, Manfred Hauswirth, Marcel Karnstedt, Myriam Leggieri, Alexandre Passant and Ray Richardson. *SPITFIRE: toward a semantic web of things*. IEEE Communications Magazine, vol. 49, no. 11, pages 40–48, nov 2011. (Cited in pages 17, 34, 74, 80, 81 and 88.)
- [Ploennigs 2017] Joern Ploennigs, Amadou Ba and Michael Barry. *Materializing the Promises of Cognitive IoT: How Cognitive Buildings are Shaping the Way*. IEEE Internet of Things Journal, vol. 4662, no. c, pages 1–8, 2017. (Cited in pages 74, 76, 79, 80, 81 and 92.)
- [Poslad 2015] Stefan Poslad, Stuart E. Middleton, Fernando Chaves, Ran Tao, Ocal Necmioglu and Ulrich Bugel. *A Semantic IoT Early Warning System for Natural Environment Crisis Management*. IEEE Transactions on Emerging Topics in Computing, vol. 3, no. 2, pages 246–257, 2015. (Cited in pages 74, 78, 80, 87, 88, 89 and 92.)
- [Puustjärvi 2015] Juha Puustjärvi and Leena Puustjärvi. *The Role of Smart Data in Smart Home: Health Monitoring Case*. Procedia Computer Science, vol. 69, pages 143–151, 2015. (Cited in page 87.)
- [Qu 2016] Chao Qu, Fagui Liu, Ming Tao and Dacheng Deng. *An OWL-S based specification model of dynamic entity services for Internet of Things*. Journal of Ambient Intelligence and Humanized Computing, vol. 7, no. 1, pages 73–82, 2016. (Cited in page 9.)
- [Reinfurt 2016] Lukas Reinfurt, Uwe Breitenbücher, Michael Falkenthal, Frank Leymann and Andreas Riegg. *Internet of things patterns*. In Proceedings of the 21st European Conference on Pattern Languages of Programs - EuroPlop '16, pages 1–21, New York, New York, USA, 2016. ACM Press. (Cited in pages 18 and 23.)
- [Rivera 2013] Janessa Rivera and Rob van der Meulen. *Gartner Says the Internet of Things Installed Base Will Grow to 26 Billion Units By 2020*. Technical Report, Gartner cabinet, 2013. (Cited in page 3.)
- [Robins 2010] David B. Robins. *Complex Event Processing*. In 2010 Second International Workshop on Education Technology and Computer Science, page 10, 2010. (Cited in page 76.)
- [Rodriguez 2010] Ismael Bouassida Rodriguez, Jérôme Lacouture and Khalil Drira. *Semantic Driven Self-Adaptation of Communications Applied to ERCMS*.

- In 2010 24th IEEE International Conference on Advanced Information Networking and Applications, pages 1292–1299. IEEE, 2010. (Cited in pages 106, 107 and 110.)
- [Rowley 2007] Jennifer Rowley. *The wisdom hierarchy: representations of the DIKW hierarchy*. Journal of Information Science, vol. 33, no. 2, pages 163–180, 2007. (Cited in page 71.)
- [Russomanno 2005] David J. Russomanno, Cartik Kothari and Omoju Thomas. *Sensor ontologies: From shallow to deep models*. In Annual Southeastern Symposium on System Theory, volume 37, pages 107–112, 2005. (Cited in page 30.)
- [Ruta 2016] Michele Ruta, Saverio Ieva, Giuseppe Loseto and Eugenio Di Sciascio. *From the Physical Web to the Physical Semantic Web: knowledge discovery in the Internet of Things*. In 14th International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies, 2016. (Cited in pages 80, 84 and 85.)
- [Sahni 2017] Yuvraj Sahni, Jiannong Cao, Shigeng Zhang and Lei Yang. *Edge Mesh: A New Paradigm to Enable Distributed Intelligence in Internet of Things*. IEEE Access, vol. 5, pages 16441–16458, 2017. (Cited in page 22.)
- [Santos 2016] Gabrielle Santos, Terry R. Payne, Valentina Tamma and Floriana Grasso. *Discovering Ontological Correspondences Through Dialogue*. In EKAW, pages 284–468, 2016. (Cited in page 95.)
- [Sarkar 2015] Chayan Sarkar, Akshay Uttama Akshay, R. Venkatesha Prasad, Abdur Rahim, Ricardo Neisse and Gianmarco Baldini. *DIAT: A scalable distributed architecture for IoT*. In IEEE Internet of Things Journal, volume 2, 2015. (Cited in pages 74, 77, 80 and 82.)
- [Scharffe 2008] François Scharffe, Jérôme Euzenat and Dieter Fensel. *Towards design patterns for ontology alignment*. In Proceedings of the 2008 ACM symposium on Applied computing - SAC '08, page 2321, New York, New York, USA, mar 2008. ACM Press. (Cited in page 36.)
- [Scioscia 2009] Floriano Scioscia and Michele Ruta. *Building a Semantic Web of Things: Issues and perspectives in information compression*. ICSC 2009 - 2009 IEEE International Conference on Semantic Computing, pages 589–594, 2009. (Cited in page 17.)
- [Seydoux 2016a] Nicolas Seydoux, Khalil Drira, Nathalie Hernandez and Thierry Monteil. *Autonomy through knowledge: how IoT-O supports the management of a connected apartment*. In Semantic Web Technologies for the Internet of Things, pages 67–78, 2016. (Cited in pages 66 and 162.)

- [Seydoux 2016b] Nicolas Seydoux, Khalil Drira, Nathalie Hernandez and Thierry Monteil. *IoT-O, a core-domain IoT ontology to represent connected devices networks*. In EKAW, 2016. (Cited in pages vi, 26, 33, 45, 65, 84, 95 and 162.)
- [Seydoux 2016c] Nicolas Seydoux, Khalil Drira, Nathalie Hernandez and Thierry Monteil. *Lowering knowledge : Making constrained devices semantically interoperable*. In International Semantic Web Conference (ISWC 2016), ISWC 2016 Posters & Demonstrations Track. CEUR-WS, 2016. Poster. (Cited in pages 26, 61, 66, 94 and 162.)
- [Seydoux 2017] Nicolas Seydoux, Khalil Drira, Nathalie Hernandez and Thierry Monteil. *Capturing the contributions of the semantic web to the IoT: a unifying vision (extended abstract)*. Semantic Web technologies for the Internet of Things, 2017. (Cited in pages 99 and 112.)
- [Seydoux 2018a] Nicolas Seydoux, Khalil Drira, Nathalie Hernandez and Thierry Monteil. *A Distributed Scalable Approach for Rule Processing: Computing in the Fog for the SWoT*. In IEEE International Conference on Web Intelligence, 2018. (Cited in pages 77, 136, 137, 143, 159 and 163.)
- [Seydoux 2018b] Nicolas Seydoux, Khalil Drira, Nathalie Hernandez and Thierry Monteil. *Reasoning on the edge or in the cloud?* Internet Technology Letters, vol. 0, no. 0, page e51, 2018. (Cited in pages 74, 77, 80, 136, 137, 143, 159 and 163.)
- [Seydoux 2018c] Nicolas Seydoux, Khalil Drira, Nathalie Hernandez and Thierry Monteil. *Towards Cooperative Semantic Computing: a Distributed Reasoning approach for Fog-enabled SWoT*. In International Conference on cooperative information systems, 2018. (Cited in pages 77, 135, 137, 159 and 163.)
- [Sezer 2018] Omer Berat Sezer, Erdogan Dogdu and Ahmet Murat Ozbayoglu. *Context Aware Computing, Learning and Big Data in Internet of Things: A Survey*. IEEE Internet of Things Journal, vol. 5, no. 1, pages 1–1, 2018. (Cited in pages 72 and 105.)
- [Sheng 2015] Zhengguo Sheng, Chinmaya Mahapatra, Chunsheng Zhu and Victor C. M. Leung. *Recent Advances in Industrial Wireless Sensor Networks Toward Efficient Management in IoT*. IEEE Access, vol. 3, pages 622–637, 2015. (Cited in page 79.)
- [Sheth 2008] Amit Sheth, Cory Henson and Satya S Sahoo. *Semantic Sensor Web*. In IEEE Internet Computing, volume 12, pages 78–83, 2008. (Cited in pages 35, 45, 61, 75 and 105.)
- [Shi 2016] Weisong Shi and Schahram Dustdar. *The promise of edge computing*. Computer, vol. 49, no. 5, pages 78–81, 2016. (Cited in page 3.)

- [Shvaiko 2013] P. Shvaiko and J. Euzenat. *Ontology Matching: State of the Art and Future Challenges*. IEEE Transactions on Knowledge and Data Engineering, vol. 25, no. 1, pages 158–176, 2013. (Cited in page 95.)
- [Singh 2017] Dhananjay Singh, Gaurav Tripathi, Antonio M. Alberti and Antonio Jara. *Semantic edge computing and IoT architecture for military health services in battlefield*. 2017 14th IEEE Annual Consumer Communications and Networking Conference, CCNC 2017, pages 185–190, 2017. (Cited in pages 74, 78 and 165.)
- [Siow 2016] Eugene Siow, Thanassis Tiropanis and Wendy Hall. *SPARQL-to-SQL on Internet of Things Databases and Streams*. In ISWC, 2016. (Cited in pages 87 and 88.)
- [Stirbu 2008] Vlad Stirbu. *Towards a RESTful plug and play experience in the Web of Things*. Proceedings - IEEE International Conference on Semantic Computing 2008, ICSC 2008, pages 512–517, 2008. (Cited in page 16.)
- [Su 2015] Xiang Su, Jukka Riekkı, Jukka K. Nurminen, Johanna Nieminen and Markus Koskimies. *Adding semantics to internet of things*. Concurrency Computation, vol. 27, no. 8, 2015. (Cited in pages 74, 87, 97 and 154.)
- [Su 2018] Xiang Su, Pingjiang Li, Jukka Riekkı, Xiaoli Liu, Jussi Kiljander, Juha-Pekka Soininen, Christian Prehofer, Huber Flores and Yuhong Li. *Distribution of Semantic Reasoning on the Edge of Internet of Things*. In IEEE UbiComp, number November, page 79, 2018. (Cited in pages viii, 20, 23, 74, 77, 94, 103, 108 and 110.)
- [Sun 2014] Yunchuan Sun and Antonio J. Jara. *An extensible and active semantic model of information organizing for the Internet of Things*. Personal and Ubiquitous Computing, vol. 18, no. 8, 2014. (Cited in pages 55 and 105.)
- [Szilagyi 2016] Ioan Szilagyi and Patrice Wira. *Ontologies and Semantic Web for the Internet of Things - a survey*. In IECON. IEEE, 2016. (Cited in pages 19, 71 and 110.)
- [Tachmazidis 2017] Ilias Tachmazidis, Sotiris Batsakis, John Davies, Alistair Duke, Mauro Vallati, Grigoris Antoniou and Sandra Stincic Clarke. *A hypercat-enabled semantic internet of things data hub*. In European Semantic Web Conference, pages 125–137. Springer, 2017. (Cited in page 34.)
- [Taneja 2017] Mohit Taneja and Alan Davy. *Resource aware placement of IoT application modules in Fog-Cloud Computing Paradigm*. In 2017 IFIP/IEEE Symposium on Integrated Network and Service Management, pages 1222–1228. IEEE, may 2017. (Cited in page 108.)
- [Thiéblin 2018] Élodie Thiéblin, Ollivier Haemmerlé, Nathalie Hernandez and Cassia Trojahn. *Task-Oriented Complex Ontology Alignment: Two Alignment*

- Evaluation Sets*. In European Semantic Web Conference, pages 655–670. Springer, 2018. (Cited in page 52.)
- [Ullah 2017] Farhan Ullah, Muhammad Asif Habib, Muhammad Farhan, Shehzad Khalid, Mehr Yahya Durrani and Sohail Jabbar. *Semantic interoperability for big-data in heterogeneous IoT infrastructure for healthcare*. Sustainable Cities and Society, vol. 34, no. March, pages 90–96, 2017. (Cited in pages 74, 78, 87 and 92.)
- [Van Woensel 2018] William Van Woensel and Syed Sibte Raza Abidi. *Optimizing Semantic Reasoning on Memory-Constrained Platforms Using the RETE Algorithm*. In ESWC, volume 10843 LNCS, pages 682–696, 2018. (Cited in pages 97 and 107.)
- [Vandenbussche 2017] Pierre Yves Vandenbussche, Ghislain A. Atemezing, María Poveda-Villalón and Bernard Vatant. *Linked Open Vocabularies (LOV): A gateway to reusable semantic vocabularies on the Web*. Semantic Web, vol. 8, no. 3, pages 437–452, 2017. (Cited in page 31.)
- [Verbelen 2012] Tim Verbelen, Pieter Simoens, Filip De Turck and Bart Dhoedt. *Cloudlets : Bringing the cloud to the mobile user*. Mcs 2012, pages 29–36, 2012. (Cited in page 84.)
- [Vlacheas 2013] Panagiotis Vlacheas, Raffaele Giaffreda, Vera Stavroulaki, Dimitris Kelaidonis, Vassilis Foteinos, George Poullos, Panagiotis Demestichas, Andrey Somov, Abdur Biswas and Klaus Moessner. *Enabling smart cities through a cognitive management framework for the internet of things*. IEEE Communications Magazine, vol. 51, no. 6, pages 102–111, 2013. (Cited in pages 20, 80, 81, 82, 86 and 93.)
- [Wang 2015a] Feng Wang, Liang Hu, Jin Zhou and Kuo Zhao. *A survey from the perspective of evolutionary process in the internet of things*. International Journal of Distributed Sensor Networks, vol. 2015, 2015. (Cited in pages 31 and 93.)
- [Wang 2015b] Wei Wang, Suparna De, Gilbert Cassar and Klaus Moessner. *An experimental study on geospatial indexing for sensor service discovery*. Expert Syst. Appl., vol. 42, pages 3528–3538, 2015. (Cited in page 34.)
- [Wang 2017] Wei Wang, Suparna De, Yuchao Zhou, Xin Huang and Klaus Moessner. *Distributed Sensor Data Computing in Smart City Applications*. In Proceedings of WoWMoM 2017, number June, 2017. (Cited in pages 30, 80, 84 and 87.)
- [Wang 2018] Shiyong Wang, Jiafu Wan, Di Li and Chengliang Liu. *Knowledge reasoning with semantic data for real-time data processing in smart factory*. Sensors (Switzerland), vol. 18, no. 2, pages 1–10, 2018. (Cited in pages 73, 74, 92 and 106.)

- [Weiser 2002] M. Weiser. *The computer for the 21st Century*. IEEE Pervasive Computing, vol. 1, no. 1, pages 19–25, jan 2002. (Cited in page 168.)
- [Wu 2017] Zhenyu Wu, Yuan Xu, Yunong Yang, Chunhong Zhang, Xinning Zhu and Yang Ji. *Towards a semantic web of things: A hybrid semantic annotation, extraction, and reasoning framework for cyber-physical system*. Sensors (Switzerland), vol. 17, no. 2, 2017. (Cited in pages 34 and 42.)
- [Xu 2016] Yi Xu and Abdelsalam Helal. *Scalable Cloud–Sensor Architecture for the Internet of Things*. IEEE Internet of Things Journal, vol. 3, no. 3, pages 285–298, 2016. (Cited in page 23.)
- [Xu 2017] Guangquan Xu, Yan Cao, Yuanyuan Ren, Xiaohong Li and Zhiyong Feng. *Network Security Situation Awareness Based on Semantic Ontology and User-Defined Rules for Internet of Things*. IEEE Access, vol. 5, pages 21046–21056, 2017. (Cited in page 105.)
- [Zanella 2014] Andrea Zanella, Nicola Bui, Angelo Castellani, Lorenzo Vangelista and Michele Zorzi. *Internet of Things for Smart Cities*. IEEE Internet of Things Journal, vol. 1, no. 1, pages 22–32, 2014. (Cited in pages viii, 2, 19, 20, 23 and 110.)
- [Zhao 2010] F. Zhao. *Sensors meet the Cloud: Planetary-scale distributed sensing and decision making*. In 9th IEEE International Conference on Cognitive Informatics (ICCI'10), pages 998–998, July 2010. (Cited in page 3.)

Abstract: This thesis is situated in the Semantic Web of things (SWoT) domain, at the interface between the Internet of Things (IoT) and the Semantic Web (SW). The integration of SW approaches into the IoT aims to tackle the important heterogeneity of resources, technologies and applications in the IoT, which creates interoperability issues hindering the deployment of IoT systems. A first scientific challenge is risen by the resource consumption of the SW technologies, inappropriate for the limited computation and communication capabilities of IoT devices. Moreover, IoT networks are deployed at a large scale, while SW technologies have scalability issues. This thesis addresses this double challenge and proposes two related contributions. The first contribution is the identification of quality criteria for IoT ontologies, leading to the elaboration of IoT-O, a modular IoT ontology. IoT-O is deployed to enrich data from a smart building, and drive semIoTics, our autonomic computing application. The second contribution is EDR (Emergent Distributed Reasoning), a generic approach to dynamically distributed rule-based reasoning. Rules are propagated from peer to peer, guided by the descriptions exchanged among nodes. EDR is evaluated in two use-cases, using both a server and a number of constrained nodes to simulate the deployment.

Keywords: Semantic Web of Things, Semantic Fog computing, Distributed rule-based reasoning, IoT interoperability

Résumé : Cette thèse porte sur le Web Sémantique des Objets (WSdO), un domaine de recherche à l'interface de l'Internet des Objets (IdO) et du Web Sémantique (WS). L'intégration des approche du WS à l'IdO permettent de traiter l'importante hétérogénéité des ressources, des technologies et des applications de l'IdO. Ceci est une source de problèmes d'interopérabilité freinant le déploiement de plateformes d'IdO. Un premier verrou scientifique est lié à la consommation en ressource des technologies du WS, là où l'IdO s'appuie sur des objets aux capacités de calcul et de communication limitées. De plus, les réseaux IdO sont déployés à grande échelle, quand la montée en charge est difficile pour les technologies du WS. Cette thèse a pour objectif de traiter ce double défi, et comporte deux contributions. La première porte sur l'identification de critères de qualité pour les ontologies de l'IdO, et l'élaboration de IoT-O, une ontologie modulaire pour l'IdO. IoT-O a été implantée pour enrichir les données d'un bâtiment instrumenté, et pour être moteur de semIoTics, notre application de gestion autonome. La seconde contribution est EDR (Emergent Distributed Reasoning), une approche générique pour distribuer dynamiquement le raisonnement à base de règles. Les règles sont propagées de proche en proche en s'appuyant sur les descriptions échangées entre noeuds. EDR est évaluée dans deux scénarii, s'appuyant sur un serveur et des noeuds contraints pour simuler le déploiement.

Mots clés : Web Sémantique des Objets, Fog computing sémantique, Raisonnement distribué à base de règle, Interopérabilité
