New Challenges on Web Architectures for the Homogenization of the Heterogeneity of Smart Objects in the Internet of Things

Víctor Caballero Codina

http://hdl.handle.net/10803/669186

UNIVERSITAT RAMON LLULL

DOCTORAL THESIS

Title	New Challenges on Web Architectures for the Homogenization of the Heterogeneity of Smart Objects in the Internet of Things
Presented by	Víctor Caballero Codina
Centre	Facultat Internacional de Comerç i Economia Digital La Salle
Department	Engineering Department
Directed by	Juan Agustín Zaballos de Diego David Vernet Bellet

RAMON LLULL UNIVERSITY

DOCTORAL THESIS

New Challenges on Web Architectures for the Homogenization of the Heterogeneity of Smart Objects in the Internet of Things

Author:
Víctor Caballero Codina

Supervisors:
Dr. Juan Agustín Zaballos de Diego
Dr. David Vernet Bellet

A thesis submitted in fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Engineering

in the

Internet Technologies & Storage Research Group
Department of Engineering

March 12, 2020

To my parents Marta and Luciano and to my grandparents Montserrat and Francisco.

This is the real secret of life — to be completely engaged with what you are doing in the here and now. And instead of calling it work, realize it is play.

Alan Watts

Declaration of Authorship

I, Víctor Caballero Codina, declare that this thesis titled, "New Challenges on Web Architectures for the Homogenization of the Heterogeneity of Smart Objects in the Internet of Things" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

Acknowledgements

I would like to express my most profound appreciation to Prof. Agustín Zaballos and PhD. David Vernet, who played a decisive role during the development of this research. They provided me with guidance, advice in the form of constructive criticism, and insightful suggestions and discussions.

I would like to extend my sincere thanks to Adrià Mallorquí, Alan Briones, PhD. Ramon Martín De Pozuelo, and Joan Camps for their practical suggestions. Special thanks to Sergi Valbuena for his dedication and fruitful discussions near the end of this thesis.

I would like to acknowledge the assistance of PhD. Guiomar Corral at the beginning of the thesis.

I'm deeply indebted to my family, who have shown unconditional support during the development of this thesis and during my life process. Without them I would not have reached this landmark.

Finally, I would like to recognise that this achievement would not have been possible without every single being that has stepped, even for a short time, into my life experience and has shaped my reality.

This work received funding from the "SUR" of the "DEC" of "Generalitat de Catalunya" (grant identification "2017 SGR 977") and ENVISERA project, funded by Spanish Ministerio de Economía y Competitividad (Plan Estatal de Investigación Científica y Técnica y de Innovación 2013-2016, grant reference CTM2015-68902-R). Part of this research has also been funded by the predoctoral grant of the "SUR" of the "DEC" of the "Generalitat de Catalunya" and by the European Social Funds (grant references "2017FI_B 00583", "2018FI_B1 00171", "2019FI_B2 00164"). In addition, we would like to thank La Salle - Universitat Ramon Llull for their support.

Abstract

This thesis deals with two novel Internet of Things (IoT) technologies and their integration to the field of the Smart Grid (SG); these technologies are the Web of Things (WoT) and the Social Internet of Things (SIoT). The WoT is an enabling technology expected to provide a scalable and interoperable environment to the IoT using the existing web infrastructure, web protocols and the semantic web. The SIoT is expected to expand further and contribute to scalability and discoverability challenges by creating a social network of agents (objects and humans). When exploring the synergy between those technologies, we aim at providing practical and empirical evidence, usually in the form of prototype implementations and empirical experimentation.

In relation to the WoT and SG, we create a prototype for the Web of Energy (WoE), that aims at addressing challenges present in the SG domain. The prototype is capable of providing interoperability and homogeneity among diverse protocols. The implementation design is based on the Actor Model, which also provides scalability in regards to the prototype. Experimentation shows that the prototype can handle the transmission of messages for time-critical SG applications.

We also take another similar research direction less focused on the SG, but for a broader range of application domains. We integrate the description of flows of execution as Finite-State Machines (FSMs) using web ontologies (Resource Description Framework (RDF)) and WoT methodologies (actions are performed on the basis of calls Hyper-Text Transfer Protocol/Secure (HTTP/S) to a Uniform Resource Locator (URL)). This execution flow, which can also be a template to allow flexible configuration at runtime, is deployed and interpreted as (and through) a Virtual Object (VO). The template aims to be reusable and shareable among multiple IoT deployments within the same application domain. Due to the technologies used, the solution is not suitable for time-critical applications. Nevertheless, it is suitable for non-time-critical applications that require the deployment of similar VOs.

Finally, we focus on another technology aimed at improving scalability and discoverability in IoT. The SIoT is emerging as a new IoT structure that links nodes through meaningful relationships. These relationships aim at improving discoverability; consequently, improving the scalability of an IoT network. We apply this new paradigm to optimize energy management at the demand side in a SG. Our objective is to harness the features of the SIoT to aid in the creation of Prosumer Community Group (PCG) (groups of energy users that consume or produce energy) with the same Demand Side Management (DSM) goal. We refer to the synergy between SIoT and SG as Social Internet of Energy (SIoE). Therefore, with the SIoE and focusing on a specific challenge, we set the conceptual basis for the integration between SIoT and SG. Initial experiments show promising results and pave the way for further research and evaluation of the proposal.

We conclude that the WoT and the SIoT are two complementary paradigms that nourish the evolution of the next generation IoT. The next generation IoT is expected to be a pervasive Multi-Agent System (MAS). Some researchers are already pointing at the Web and its technologies (e.g. Semantic Web, HTTP/S) — and more concretely at the WoT — as the environment nourishing the agents. The SIoT can enhance both the environment and the relationships between agents in this fusion. As a specific field of the IoT, the SG can also benefit from IoT advancements.

Keywords — Internet of Things, Web of Things, Social Internet of Things, Smart Grid, Prosumer, Prosumer Community Group, Web of Energy, Social Internet of Energy

Resumen

Esta tesis trata de dos de las novedosas tecnologías relacionadas con la Internet of Things (IoT) y su integración con el campo de las Smart Grids (SGs); estas tecnologías son la Web of Things (WoT) y la Social Internet of Things (SIoT). La WoT es una tecnología que se espera que provea de un entorno escalable e interoperable a la IoT usando la infraestructura web existente, los protocolos web y la web semántica. También se espera que la SIoT contribuya a solucionar los retos de escalabilidad y capacidad de descubrimiento creando una red social de agentes (objetos y humanos). Para explorar la sinergia entre estas tecnologías, el objetivo es el de proporcionar evidencia práctica y empírica, generalmente en forma de prototipos de implementación y experimentación empírica.

En relación con la WoT y las SGs, se ha creado un prototipo para la Web of Energy (WoE) que tiene como objetivo abordar los desafíos presentes en el dominio las SGs. El prototipo es capaz de proporcionar interoperabilidad y homogeneidad entre diversos protocolos. El diseño de implementación se basa en el Modelo de Actores, que también proporciona escalabilidad del prototipo. La experimentación muestra que el prototipo puede manejar la transmisión de mensajes para aplicaciones de las SGs que requieran que la comunicación se realice bajo umbrales de tiempo críticos.

También se toma otra dirección de investigación similar, menos centrada en las SGs, pero para una gama más amplia de dominios de aplicación. Se integra la descripción de los flujos de ejecución como máquinas de estados finitos utilizando ontologías web (Resource Description Framework (RDF)) y metodologías de la WoT (las acciones se realizan basándose en peticiones Hyper-Text Transfer Protocol/Secure (HTTP/S) a Uniform Resource Locators (URLs)). Este flujo de ejecución, que también puede ser una plantilla para permitir una configuración flexible en tiempo de ejecución, se implementa e interpreta como si fuera (y a través de) un Virtual Object (VO). El objetivo de la plantilla es que sea reutilizable y se pueda compartir entre múltiples despliegues de la IoT dentro del mismo dominio de aplicación. Debido a las tecnologías utilizadas, la solución no es adecuada para aplicaciones de tiempo crítico (umbral de tiempo relativamente bajo y rígido). Sin embargo, es adecuado para aplicaciones que no demandan respuesta en un tiempo crítico y que requieren el despliegue de VOs similares en cuanto al flujo de ejecución.

Finalmente, el trabajo se enfoca en otra tecnología destinada a mejorar la escalabilidad y la capacidad de descubrimiento en la IoT. La SIoT está emergiendo como una nueva estructura de la IoT que une los nodos a través de relaciones significativas. Estas relaciones tienen como objetivo mejorar la capacidad de descubrimiento; en consecuencia, mejora la escalabilidad de una red de la IoT. En este trabajo se aplica este nuevo paradigma para optimizar la gestión de la energía en el lado de la demanda en las SGs. El objetivo es aprovechar las características de la SIoT para ayudar en la creación de Prosumer Community Groups (PCGs) (grupos de usuarios que consumen o producen energía) con el mismo objetivo de optimización en el uso de la energía. La sinergia entre la SIoT y las SGs ha sido denominada Social Internet of Energy (SIoE). Por lo tanto, con la SIoE y con el foco en un desafío específico, se establece la base conceptual para la integración entre la SIoT y las SG. Los experimentos iniciales muestran resultados prometedores y allanan el camino para futuras investigaciones y evaluaciones de la propuesta.

Se concluye que la WoT y la SIoT son dos paradigmas complementarios que nutren la evolución de la próxima generación de la IoT. Se espera que la próxima generación de la IoT sea un Multi-Agent System (MAS) generalizado. Algunos investigadores ya están apuntando a la Web y sus tecnologías (por ejemplo, Web Semántica, HTTP/S) — y más concretamente a la WoT — como el entorno que nutra a estos agentes. La SIoT puede mejorar tanto el entorno como las relaciones entre los agentes en esta fusión. Como un campo específico de la IoT, las SGs también pueden beneficiarse de los avances de la IoT.

Palabras clave — Internet of Things, Web of Things, Social Internet of Things, Smart Grid, Prosumer, Prosumer Community Group, Web of Energy, Social Internet of Energy

Resum

Aquesta tesi tracta de dues de les noves tecnologies relacionades amb la Internet of Things (IoT) i la seva integració amb el camp de les Smart Grids (SGs); aquestes tecnologies son la Web of Things (WoT) i la Social Internet of Things (SIoT). La WoT és una tecnologia que s'espera que proveeixi d'un entorn escalable i interoperable a la IoT usant la infraestructura web existent, els protocols web y la web semàntica. També s'espera que la SIoT contribueixi a solucionar els reptes d'escalabilitat i capacitat de descobriment creant una xarxa social d'agents (objectes i humans). Per explorar la sinergia entre aquestes tecnologies, l'objectiu és el de proporcionar evidència pràctica i empírica, generalment en forma de prototips d'implementació i experimentació empírica.

En relació amb la WoT i les SGs, s'ha creat un prototip per al Web of Energy (WoE) que té com a objectiu abordar els desafiaments presents en el domini les SGs. El prototip és capaç de proporcionar interoperabilitat i homogeneïtat entre diversos protocols. El disseny d'implementació es basa en el Model d'Actors, que també proporciona escalabilitat del prototip. L'experimentació mostra que el prototip pot gestionar la transmissió de missatges per a aplicacions de les SGs que requereixen que la comunicació es realitzi sota llindars de temps crítics.

També es pren una altra direcció d'investigació similar, menys centrada en les SGs, però per a una gamma més àmplia de dominis d'aplicació. S'integra la descripció dels fluxos d'execució com a màquines d'estats finits utilitzant ontologies web (Resource Description Framework (RDF)) i metodologies de la WoT (les accions es realitzen basant-se en peticions Hyper-Text Transfer Protocol/Secure (HTTP/S) a Uniform Resource Locators (URLs)). Aquest flux d'execució, que també pot ser un plantilla per a permetre una configuració flexible en temps d'execució, s'implementa i interpreta com si fos (i mitjançant) un Virtual Object (VO). L'objectiu de la plantilla és ser reutilitzable i poder-se compartir entre múltiples desplegaments de la IoT dins el mateix domini d'aplicació. A causa de les tecnologies utilitzades, la solució no és adequada per a aplicacions de temps crític (llindar de temps relativament baix i rígid). No obstant això, és adequat per a aplicacions que no demanden resposta en un temps crític i que requereixen el desplegament de VOs similars en el que fa referència al flux d'execució.

Finalment, el treball s'enfoca en una altra tecnologia destinada a millorar l'escalabilitat i la capacitat de descobriment en la IoT. La SIoT està sorgint com una nova estructura de la IoT que uneix els nodes a través de relacions significatives. Aquestes relacions tenen com a objectiu millorar la capacitat de descobriment; en conseqüència, millora la escalabilitat d'una xarxa de la IoT. En aquest treball s'aplica aquest nou paradigma per optimitzar la gestió de l'energia en el costat de la demanda a les SGs. L'objectiu és aprofitar les característiques de la SIoT per ajudar a la creació de Prosumer Community Groups (PCGs) (grups d'usuaris que consumeixen o produeixen energia) amb el mateix objectiu d'optimització en l'ús de l'energia. La sinergia entre la SIoT i les SGs s'ha anomenat Social Internet of Energy (SIoE). Per tant, amb la SIoE i amb el focus en un desafiament específic, s'estableix la base conceptual per a la integració entre la SIoT i les SGs. Els experiments inicials mostren resultats prometedors i aplanen el camí per a futures investigacions i avaluacions de la proposta.

Es conclou que el WoT i la SIoT són dos paradigmes complementaris que nodreixen l'evolució de la propera generació de la IoT. S'espera que la propera generació de la IoT sigui un Multi-Agent System (MAS) generalitzat. Alguns investigadors ja estan apuntant a la Web i les seves tecnologies (per exemple, Web Semàntica, HTTP/S) — i més concretament a la WoT — com a l'entorn que nodreixi a aquests agents. La SIoT pot millorar tant l'entorn com les relacions entre els agents en aquesta fusió. Les SGs també poden beneficiar-se dels avenços de la IoT, ja que es poden considerar com una aplicació específica d'aquesta última.

Paraules clau — Internet of Things, Web of Things, Social Internet of Things, Smart Grid, Prosumer, Prosumer Community Group, Web of Energy, Social Internet of Energy

Contents

List of Figures

List of Tables

List of Algorithms

List of Listings

Acronyms

API	Application Programming Interface. 4, 16–18, 20, 29, 43–45, 61, 62, 69, 72, 99
ATHIKA	Advanced Training in Health Innovation Knowledge Alliance. 39, 56
BLE	Bluetooth Low Energy. 44
CAF	C++ Actor Framework. 25
CEC	Clean Energy Community. 2
CLOR	Co-location Object Relationship. 66, 69
CoAP	Constrained Application Protocol. 19–21, 61
CWOR	Co-work Object Relationship. 47, 66, 70, 86
DER	Distributed Energy Resource. 1, 2, 5, 14, 16, 47, 64, 71, 77, 85
DHT	Distributed Hash Table. 67
DLC	Direct Load Control. 2, 71
DSM	Demand Side Management. v, 2, 5, 10, 63, 65, 71, 72, 75–78, 82, 90, 91, 96–99
DSO	Distribtion Systems Operator. 1, 16
EA	Evolutionary Algorithm. 75
ENVISERA	Optimized HF transmission for NVIS links for remote sensors in Antarctica. xxvii, 39, 56, 63
ERP	Enterprise Resource Planning. 57, 60
ESCO	Energy Services Company. 1, 14
EV	Electric Vehicle. 2
EVSE	Electrical Vehicle Supply Equipment. 16
FSM	Finite-State Machine. v, 41–43, 45, 46, 48, 50, 52, 54–56, 59–62, 89, 90, 95
GW	Gateway. 43
HTTP	Hyper-Text Transfer Protocol. 17
HTTP/S	Hyper-Text Transfer Protocol/Secure. v, vii–x, 3, 4, 15–23, 26, 30, 34, 36, 37, 40, 41, 43–46, 50, 52–54, 56, 58, 61, 62, 68, 72, 89–93, 98, 99
HTTPS	Hyper-Text Transfer Protocol Secure. 4, 17
HVAC	Heat, Ventilation, Air Conditioning. 37, 69
ICES	Integrated Community Energy System. 2
ICT	Information and Communication Technology. 1, 5, 7, 8, 13–15, 36, 64, 85, 89, 90, 92, 97
IL	Interruptible Load. 72, 75, 87
IoE	Internet of Energy. 2, 65

IoT	Internet of Things. v–x, xxvii, 1–10, 13–26, 36, 37, 39–45, 56, 57, 61, 63, 65, 66, 69, 70, 85, 89–93, 95–99
IRI	Internationalized Resource Identifier. 54, 55
JSON	JavaScript Object Notation. 30, 50, 52
JVM	Java Virtual Machine. 24
LAN	Local Area Network. 58
LD	Linked Data. 19, 98
LMS	Learning Management System. 56, 61
MAS	Multi-Agent System. v, viii, x, 24, 75, 87, 98, 99
ME	Micro Engine. 69, 74
MG	Microgrid. 1, 64, 72, 75
ML	Machine Learning. 23
MQTT	Message Queuing Telemetry Transport. 19–21, 26–30, 36, 89
NVIS	Near Vertical Incidence Skywave. xxvii
OOR	Ownership Object Relationship. 46, 47, 66, 69–71, 86, 92, 99
OSI	Open Systems Interconnection. 18
OWL	Web Ontology Language. 19, 42, 43, 48
P2P	Peer-to-Peer. 21, 45
PAR	Peak-to-Average Ratio. 63, 72, 74, 76–84, 86, 90
PCG	Prosumer Community Group. v, vii, ix, 1, 2, 9, 10, 47, 63–65, 67, 69–79, 82, 86, 87, 90, 91, 96–98
PHP	PHP: Hypertext Preprocessor. 9, 23, 24
PLC	Programmable Logic Controller. 42
POR	Parental Object Relationship. 66
PSO	Particle Swarm Optimization. 75
PTO	Pre-Trusted Object. 67
QoS	Quality of Service. 2, 14, 21, 99
RDF	Resource Description Framework. v, vii, ix, 19, 41, 45, 49–52, 54, 55, 57, 58, 61, 90
RDFa	Resource Description Framework in Attributes. 19
RES	Renewable Energy Source. 1, 2, 40, 64
REST	Representational State Transfer. 4, 19–22, 99
RFID	Radio-Frequency IDentification. 57
RIoT	Remote Internet of Things. xxvii
RTP	Real Time Pricing. 2, 71, 72
SCADA	Supervisory Control And Data Acquisition. 16
SDU	Software Defined Utility. 64
SG	Smart Grid. v, vii–x, xxvii, 1, 2, 4, 5, 7–10, 13–17, 22, 23, 26, 31, 34, 36, 64, 65, 71, 72, 75, 76, 85, 86, 89–93, 95–99
SIoE	Social Internet of Energy. v, vii, ix, 9, 10, 63, 65, 71, 72, 85, 90, 91, 93, 96, 98

SIoT	Social Internet of Things. v, vii–x, 3–5, 8–10, 37, 46, 63, 65–70, 72–76, 78, 85, 86, 90–93, 96–99
SN	Social Network. 4, 65, 66
SNS	Social Network Service. 3, 4, 65, 66, 69
SOR	Social Object Relationship. 46, 47, 66, 69–72, 86, 92
SPARQL	SPARQL Protocol and RDF Query Language. 41, 45, 49, 52, 54, 57–59, 61, 90
SPOF	Single Point of Failure. 74, 75
SPRINT	Strategic Programme for Innovation and Technology Transfer. 39, 56
SVO	Social Virtual Object. 4, 37, 46, 47, 70, 71, 74, 76, 87, 92, 93
SWoT	Social Web of Things. 20, 65, 66
TCP	Transmission Control Protocol. 20, 21
TCP/IP	Transmission Control Protocol/Internet Protocol. 18
TD	W3C Web of Things (WoT) Thing Description. 41, 42, 61, 91, 99
TDB	Triplestore Database. 45, 52, 61
TLS	Transport Layer Security. 21
UDP	User Datagram Protocol. 20, 21
UML	Unified Modeling Language. 42, 48
URI	Uniform Resource Identifier. 22, 43, 48, 52, 55, 62
URL	Uniform Resource Locator. v, vii, ix, 19, 30, 45
USN	Ubiquitous Sensor Network. 23, 26, 37
UUID	Universally Unique Identifier. 30, 35
VM	Virtual Machine. 30
VO	Virtual Object. v, vii, ix, 4, 10, 21–23, 25–27, 29, 30, 36, 43–46, 48, 52–61, 68, 74, 89–91, 96, 99
VPP	Virtual Power Plant. 1, 14, 64, 72
W3C	World Wide Web Consortium. 41–43, 48, 52, 61, 90
WoE	Web of Energy. v, vii, ix, 8, 10, 13, 15–17, 21, 22, 24, 25, 34–37, 89, 91, 92, 95, 97, 98
WoT	Web of Things. v, vii–x, 3–5, 8–10, 13, 15–21, 23, 24, 36, 37, 39–43, 45, 61, 62, 66, 68, 69, 72, 89–93, 95–99
WS	WebSocket. 4, 15, 20, 21, 23, 26, 27, 29, 30, 34, 36, 89, 93
WSN	Wireless Sensor Network. 8, 58
WSS	WebSocket Secure. 20

Preface

I have been a user of the World Wide Web since I was young. My understanding of the Web as a tool took multiple steps. I knew that the Web could be used to search for information. When I needed it in my undergraduate studies, the Web evolved — as of my perception — to allow me to ask and obtain answers. Not so far from there, and driven by my instructors, I became to understand the technologies that build a web application. It surprised and interested me the adaptability of web applications to a myriad of devices; e.g. smartphones, computers and tablets. I believed the Web could be — and already was — an evolution towards pervasive applications. Next, I was on a scholar internship to discover that the Web was envisioned to join forces with a hyper-connected world that was somehow unmanageable. So I dived until I began to see an already-known world emerging before my eyes. Yet, this world was (*is*) emerging real. I have always been, as far as I remember, an individual willing to understand. Accordingly, I decided to plunge with these technologies that are giving birth to a new technological era. The infrastructure that underlays this new era is called Internet of Things (IoT). It is the layer capable of perceiving the physical world and act accordingly, through physical devices that sense and actuate. IoT is also used as an umbrella term, comprehending both a vision and a set of technologies. The vision is of a cyber-physical hyper-connected world. The set of technologies are the ones that will enable the realization of the vision; some are even yet to come.

This thesis is framed within the competitive research project Optimized HF transmission for NVIS links for remote sensors in Antarctica (ENVISERA). ENVISERA's primary goal is the design of a sensor network able to transmit their data from remote places using Near Vertical Incidence Skywave (NVIS). The research done in ENVISERA will also allow to extend the concept of IoT to devices located in places out of coverage of traditional systems (Remote Internet of Things (RIoT)). The IoT is still maturing, and significant challenges still need to be addressed. These challenges are also shared with RIoT. The project, which is in a growing, mature stage, demands the management of heterogeneous deployments of sensors. The definition of the word *management* implies two non-exclusive sub-definitions in this regard: management by a governing entity (e.g. humans managing objects), and self and autonomous management (e.g. objects managing themselves as a goal-oriented cluster). For this, the research group I'm part of is interested in the next-generation technologies, architectures and frameworks that are expected to thrust the present IoT closer to the IoT that connects anything and anyone anywhere anytime.

The research group has a large body of experience in Smart Cities and Smart Grids (SGs). Since the research line in RIoT is still in the final stages of investigating the physical layer, I focus on investigating the integration between next-generation technologies for IoT and the SG. Considering that the fields of IoT and SG, and RIoT have overlapping challenges, specially in the management of heterogeneous deployments, the experience gained from contributing to the integration between IoT technologies and SG is easily transferable to other and specific IoT domains such as RIoT.

1 Introduction

1.1 Background and Motivation

1.1.1 Future Grid and new era technologies

Contrary to the rapid evolution experienced in the last decade of Information and Communication Technologies (ICTs) and particularly the Internet of Things (IoT) [1], electric power distribution systems have remained exceptionally steady for a long time. Nevertheless, energy demand in the world is increasing rapidly. Worldwide energy consumption is expected to increase by nearly 50% from 2018 to 2050 [2]. The majority of the energy demand is met by non-renewable energy resources which are starting to be insufficient and produce undesirable climate changes that harm the world we live in [3]. Indeed, the report of the United Nations on Sustainable Development Goals progress [4] states: "[...] since 2012 [...] the growth of renewables (has) outpaced the growth of total energy consumption". Hence, society is moving towards the use of renewable and sustainable energy sources. Consequently, the increase in energy demand will not be manageable unless the traditional electricity grid evolves.

The Smart Grid (SG) concept, the addition of telecommunication infrastructure to the electrical domain, has enabled a plethora of new services and opportunities (e.g., accurate grid monitoring, real-time energy consumption, and customer-side generation), which are currently driving a significant revolution in the energy sector [5, 6]. SGs are conceived to improve traditional electric power networks in several dimensions [7] such as information, business models and scalability, to meet the highest standards of power quality and thus to ensure that the electric power grid is cost-effective and sustainable [8].

The SG has led to the rise of new agents and components. The new energy user does not only consume energy but performs a crucial role in the SG. The prosumer (energy producer and energy consumer [9]) as the last link in the electricity value chain is easily the most important creator of value within the SG. Prosumers have a more active role in the Future Grid. The most important connections for prosumers in the electricity value chain are the Distribtion Systems Operator (DSO) or the aggregator/retailer in addition to Energy Services Companies (ESCOs), Virtual Power Plants (VPPs) [10], Microgrids (MGs) [10] or Prosumer Community Groups (PCGs) [9, 11], which are also new components in the energy market value chain [7]. Those components are key in managing the increase of energy demand along with energy gathered from Renewable Energy Sources (RESs). They range from energy-oriented commercial businesses, ESCOs; to systems that operate Distributed Energy Resources (DERs), VPP; that can even operate autonomously as a single unit detached from the main grid, MGs; to PCGs, that describe prosumer collaboration towards a common goal.

Those new agents and components of the SG could become the most enriched elements due to their added information and technology features. However, the upgraded features of the end-user side come at the cost of requiring more technology to make them usable. Due to the complexity (i.e., stringent levels of service reliability and availability), magnitude (i.e., large-scale areas), and new agent profiles inherent to SGs, practitioners have recently addressed the digital transformation of power-electric networks by proposing flexible and future internet-based architectures [8].

The SG has an inherent heterogeneous nature which forces system architects to consider different telecommunication technologies. SGs are deployed in hostile wireless communication environments [12]; aware protocols [13] (also referred to as cognitive radio techniques) are needed to reduce communication delay [12], meet Quality of Service (QoS) needs in terms of delay, bandwidth, and data reliability, improve energy harvesting techniques [13], and provide reliable distributed sensing [14] in order to minimize interoperability issues between heterogeneous communication networks [15].

The SG is considered a special use case of the IoT, with its own requirements, coined as the Internet of Energy (IoE) [16]. A common challenge both for SG and IoT is the accessibility of deployed sensors and actuators. As long as those devices are connected to the Internet and included in the IoT, they are potentially accessible. Nonetheless, these heterogeneous devices expose diverse interfaces, which renders access to those resources difficult and non-scalable.

As stated, one of the most relevant agents in the SG and Future Grid are prosumers, individual users that consume and produce energy. A prosumer is "an energy user who generates renewable energy in his/her domestic environment and either store the surplus energy for future use or trades to interested energy customers in smart grid" [9, 17]. To meet worldwide energy demand using RESs, it has been proposed to group energy users (prosumers) to optimize generation, demand-side or storage resources and increase individual DER visibility; while allowing these groups to work autonomously to some extent. Those prosumer communities are often called PCGs, and they pursue a mutual goal and compete in the energy market as a group. Other related terms, such as [18], Integrated Community Energy Systems (ICESs) [19] and Clean Energy Communities (CECs) [20] are used to refer to the same concept [9].

Prosumers, rather than the utility, are expected to be the primary agents in the Future Grid. Management services at the demand side (prosumers' side) aim at optimizing the use of energy resources. Thus, Demand Side Management (DSM) refers to the management of energy consumption and production at the demand side. There are several DSM services that usually involve an agreement with the utility and prosumers. Those services range from using price signals to control or bias the consumption pattern of prosumers (Real Time Pricing (RTP) [21]), to allowing the utility to remotely control the operation and consumption of certain appliances (Direct Load Control (DLC) [21]). Energy usage optimization pursues the reduction of energy demand peaks to reduce blackouts and successful management of DERs such as storage systems (batteries, Electric Vehicle (EV)) and energy gathered from RES. Energy from RES cannot be gathered on demand since it depends on climate factors. Therefore, energy storage systems must store surplus energy from RES to provision the grid on demand. Effectively managing consumption patterns leads to a reduction in energy demand peaks and a decrease in overall energy consumption.

1.1.2 New era technologies

The IoT [1, 22] is emerging as the new era technology, where before were the Internet and the Web. The IoT is one of the consequences of the Internet — that which allowed to create an intercommunicated world and the World Wide Web, that builds on top of a World Wide Network. As science and technology advancements allowed, more and more devices were being connected to the World Wide Network, a hyper-connected world was and still is emerging. The hyper-connected world was coined under the umbrella term IoT, or for more concrete use cases, Internet of X. Where X receives the value of vehicles, people, radios, energy among other names.

Nevertheless, more often than not, these field-specific IoT deployments operate in silos. A lack of homogenization of data and protocols used by smart devices prevents local IoT deployments to inter-operate and pose a challenge to realize the full potential of the IoT. "Interoperability is considered as the ability of two or more systems or components to exchange information and data, and use the exchanged information and data without special effort by either system, or without any special manipulation." [23]. The Web and its technologies serve humankind well in facilitating the access and globalization of information. It is envisioned by some authors as the homogenization technologies through which rendering the IoT easily and homogeneously accessible. Yet, and although still in development these days, the Web of Things (WoT) [24], as it is called, is far from one solution fits all.

The WoT answers the challenge of protocol and data homogenization, which enables accessibility, findability, shareability and composition of WoT [24], but there is still a major problem to solve. How are these data and services found in a scalable manner? A system is space-time scalable "if it continues to function gracefully as the number of objects it encompasses increases by orders of magnitude. A system may be space-time scalable if the data structures and algorithms used to implement it are conducive to smooth and speedy operation whether the system is of moderate size or large." [25]. The WoT enables data accessibility and findability using already available web technologies. However, the discovery of billions of objects is challenging. Tackling scalability and diversity (heterogeneity) is not an easy task. WoT search engines should support local-scale and global-scale search at the same time while also considering diversity of search queries and services. For that matter, WoT search engines seem to be moving towards a distributed approach [26].

Recently, another approach to scalable discovery is gaining momentum. Some authors envision a social network of things. This social network is based on the small world phenomenon [27], which states that people is connected by short chains of acquaintances and thus, creating a worldwide network of connections where there is always a short path between the source node and the target node. After all, if there is a natural phenomenon of such characteristics that allows scalability and findability, why cannot devices create the same kind of network? The term Social Internet of Things (SIoT) [28, 29] refers to just that, a social network of things; similar to the social structure that emerged from the ones we have created for humans, namely Facebook, Google Plus or Twitter but for connected devices.

The IoT enables smart objects or *things* to connect to the Internet, enabling them to be accessed as their heterogeneity allows. This layer is heterogeneous in that smart devices are build by different vendors and use different formats to structure their data and different protocols to communicate. It is also the main reason for the existence of IoT silos, as specific IoT deployments, while interconnected locally for specific use cases, are not able to interact with the external world directly due to their heterogeneity; although they are connected to the Internet. The WoT and the SIoT build on top the IoT infrastructure layer. These two frameworks are not on top of the other and instead they provide complementary mechanisms to overcome the challenges of the IoT.

The WoT provides homogenized access to smart devices through the semantic web [30, 31] and web protocols such as Hyper-Text Transfer Protocol/Secure (HTTP/S). Homogeneous access at the semantic level allows devices to be found as they share a common framework that describes them. The accessibility layer enables findability, sharing and composition layers in the WoT. Objects can be discovered using already available search engines in addition to distributed discovery and lookup infrastructures. Similarly to a user in a Social Network Service (SNS) that shares a web resource, such as a webpage, the resources of a WoT-enabled thing can be shared. Describing resources using the same framework enables their inputs and outputs to be understood and composed easily. Hence, an important contribution of the

WoT to the IoT is that it enables things to be accessed and understood anywhere using web technologies. Once things are rendered homogeneously accessible, other valuable features can be developed. Nevertheless, the mechanism enabled by the WoT in terms of findability, at its current state, might need to integrate (or be integrated with) other approaches [26, 32].

The SIoT promotes a scalable and flexible network structure between things. The basic idea is that it enables each device to be part of a Social Network (SN) to search for required services or things (like friends in SNSs) and build reliable social relationships between them. The SIoT allows an autonomous and scalable search of services, as things search for the services they need incrementally, starting with themselves as the source node and then asking their friends for the services, the friends of their friends, and so on, biased by the trust assigned to each device. The scalability is possible thanks to the leverage of the small world phenomenon. Furthermore, the SIoT gives "the IoT a structure that can be shaped as required to guarantee network navigability, so as that object and service discovery is effectively performed and scalability is guaranteed" [28]. Socially-enabled things find eventually the services they need since there will be, eventually, a path that between every two nodes [33]. As in the case of WoT, the SIoT is not the ultimate solution to find required services (e.g. the SIoT does not address real-time specifically); nonetheless, it adds flexibility and scalability to the search space that might potentially allow faster response rates.

The Virtual Object (VO) is a major and crucial component of these technologies (IoT, WoT, SIoT), whose function is to provide enhanced functionalities to physical constrained devices. A VO "is the digital counterpart of any real (human or lifeless, static or mobile, solid or intangible) entity in the IoT" [34]. They enhance the functionalities of the physical device or devices they represent as they are deployed in more resourceful hardware than the constrained physical device. They may be enhanced with features of the WoT and the SIoT such as homogeneity and discovery capabilities. On the one hand, a WoT-enabled thing receives the name of "Web Thing", and it enhances a VO to communicate via web protocols, such as HTTP/S or WebSocket (WS) and provides a Representational State Transfer (REST) Web Application Programming Interface (API) interface and a semantic description of its characteristics and features. On the other hand, a SIoT-enabled thing receives the name of Social Virtual Object (SVO) and describes a VO with the capability of establishing new meaningful relationships based on the goals and the context of such SVO; the "social" part also implies that the SVO is capable of navigating the social network of objects.

1.1.3 Challenges

In the integration between SG, WoT and SIoT, we identify two challenges to explore and contribute to.

Interoperability among agents in the SG

Both smart devices and new agents in the SG can benefit from the interoperability-enabling technologies of the web, and more specifically, from the WoT, which provides a framework to use these web technologies to increase interoperability among IoT agents. Given the WoT framework, one has to undertake two main tasks to provide basic interoperability: (i) enable SG agents to communicate using a common protocol (e.g. Hyper-Text Transfer Protocol Secure (HTTPS)), and (ii) describe those agents and services using a common framework, namely the semantic web.

Furthermore, the semantic web provides a common framework to create models that represent domain knowledge; then, this knowledge is shared among multiple IoT deployments to reduce their isolation. However, those models are not habitually used to automatically create interoperable services but, instead, (human) developers use them to create interoperable services. In the same manner that multiple domains in the IoT benefit from WoT technologies, the SG can benefit from them.

Discovery among agents in the SG

As stated, the IoT faces challenges regarding the discovery of services. Those challenges are alleviated by the WoT and semantic web technologies by fostering interoperability among devices. Nevertheless, the huge amount of devices and services that compose the IoT renders discovery of services a challenge. Discovery of new agents in the SG is no exception; new agents, namely prosumers and DERs among others, are inherently distributed and offer new types of services compared to the web 2.0 (used by humans, mainly). Once they (their services) are rendered interoperable, distributed discovery processes are possible.

1.2 Research question and objectives

1.2.1 Research questions

The potential of the what we now call IoT is unfathomable. We are in an unprecedented era of possibilities: we have never been able to access mass data and knowledge as we are now. The opportunities we face ahead are astonishing, with only the data we are gathering. We are not even close to process all these data and extract proper knowledge. At the same time, the human growth is unsustainable and is destroying the planet that preserves humankind [35]. The energy waste has been rolling down for, relatively, so long now, and more and more effort is being put in finding a solution to a myriad of energy-related challenges [35].

The IoT is a tool to achieve a state of the world where anyone and anything is seamlessly and ubiquitously interconnected anywhere and at any-time. For that to happen, we must create a scalable and reliable network of smart devices.

Today the IoT is composed of a massive amalgamation of devices. Heterogeneity among them is pervasive. If we want to advance in pursuing a scalable network of smart devices, one of the critical steps to take is to homogenize accessibility and network findability of smart devices. There is also a need for scalable management of those devices and data, as not-so-far useful strategies start to be obsolete.

At the same time, management and optimization of energy generation and demand, which is increasing rapidly, is becoming obsolete and calls for solutions to its challenges. The SG and Future Grid technologies are the ones that enable the management and optimization of energy. Nonetheless, the pace at which Future Grid technologies progress is not as fast as in ICT technologies. For sure, a infrastructure that supports human evolution, the energy grid, should be as most reliable as possible and it demands creating ICT infrastructures tailored to its needs. This thesis aims to bring novel ICT technologies closer to SG agents. Therefore, the main research questions of this thesis are:

Q1 Can SGs and, more concretely DSM, benefit from WoT and SIoT in regards to space-time scalability and interoperability?

Q2 Can prosumers in a SG benefit from WoT and SIoT when aiming to improve energy sustainability?

1.2.2 Thesis objectives

The questions raised in Section 1.2.1 lead to plan certain objectives to answer them:

O1 Application focused

The framework in which to develop this thesis aims to be practical within the research context and methodology. This first and overall objective biases other objectives towards practicality. We conceive practicality as the empirical prototyping and validation of models, whether they are state of the art models or novel models that emerge from pursuing other objectives of this thesis.

The research methodology is a "philosophical study of methods used in addressing a research problem" [36], and encompasses research methods, which are plans for conducting research. Typical research methods include: literature review, modelling, engineering/prototyping methods, qualitative analysis, and experimentation along with quantitative or qualitative analyses [37]. Consequently, the work presented in this thesis strives for answering the research questions posed in Section 1.2.1 by using a combination of these research methods.

O2 State of the art: middleware layers on top of IoT's physical layer

The IoT is a concept whose main idea is that every device connected to the Internet is uniquely addressable. Moreover, those devices should be able to communicate with no obstacles with each other to achieve a hyper-connected and pervasive world; one that should merge into the fabric of our everyday lives. An objective of this thesis is to focus on the technologies that enable interoperability and space-time scalability. According to [38], these properties are accomplished at a greater level on the layers on top of physical and connectivity layers (see Figure 1.1). The reference model proposed by Cisco represents the IoT as a seven-layer architecture. Accordingly, IoT layers are:

1. Physical Devices and Controllers ("Things" in IoT). Those are endpoint devices that send and receive information. Besides physical devices, controllers control (aggregate) multiple physical devices.
2. Connectivity (communication and processing units). It enables communication between devices and networks through multiple protocols. It also ensures security at the network level.
3. Edge (Fog) Computing (data element analysis and transformation). This layer evaluates the data and transforms them if necessary. Data is transformed to facilitate higher-level processing and assessed on the basis of surpassing a threshold and redirected to more (or other) destinations than the upper layer. These data is sent in the form of events to the upper layer. The Edge Computing layer also performs network analytics.
4. Data Accumulation (storage). Gathers, processes, aggregates, and stores, if necessary, the events triggered by the lower layer.
5. Data Abstraction (aggregation and access). Combines data from multiple sources and creates views for applications by filtering, selecting and projecting the data. The Data Abstraction layer reconciles multiple data formats and ensures consistent semantics across data sources, that is, transforms the heterogeneity of data into homogeneity.
6. Application (reporting, analytics and control). Business-specific applications that use the data from the Data Abstraction layer. The applications at this layer interpret the date and create value.

7. Collaboration and Processes involving people and business processes. The end purpose of the IoT system is to empower people to do their tasks better. Applications use data and business logic to provide people with information so they can trigger action. People use the same application or various applications for different purposes, according to their specific needs. Therefore, this level represents a higher level than the Application layer and considers them (applications) and associated data as tools to achieve a specific goal.

Hence and according to the layers presented in Figure 1.1, this aims to focus mainly on Data Abstraction and Application layers. The domain of application of this thesis is the SG and the application of ICT/IoT-related technologies that might benefit this domain. As every research process requires, the first step to be attained is to gather enough knowledge about the state of the art of the research topic, namely, SGs and IoT.

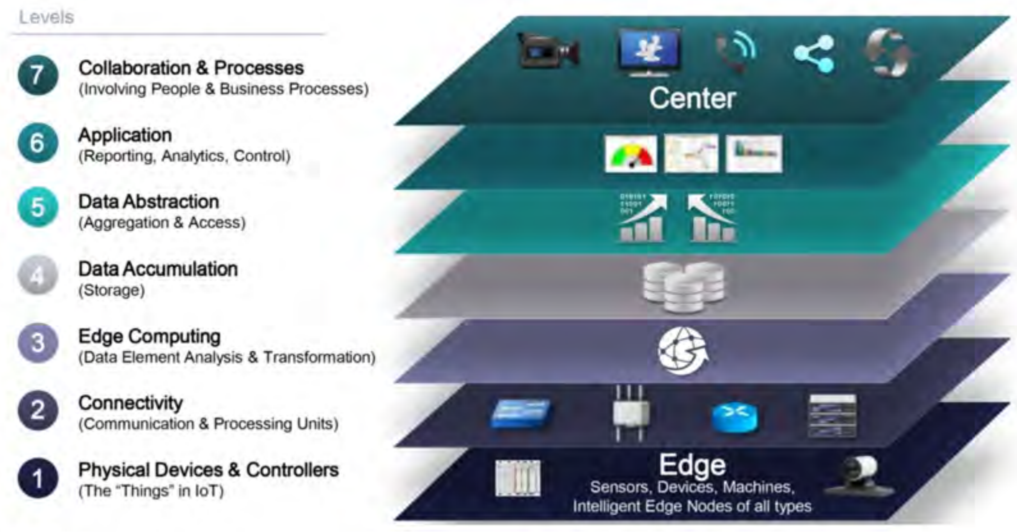

Levels
7 Collaboration & Processes
(Involving People & Business Processes)
6 Application
(Reporting, Analytics, Control)
5 Data Abstraction
(Aggregation & Access)
4 Data Accumulation
(Storage)
3 Edge Computing
(Data Element Analysis & Transformation)
2 Connectivity
(Communication & Processing Units)
1 Physical Devices & Controllers
(The "Things" in IoT)
Center
Edge
Sensors, Devices, Machines,
Intelligent Edge Nodes of all types

Figure 1.1: Cisco's IoT World Forum Reference Model ([38]).

O3 Interoperability and scalability in the IoT middleware layers

Interoperability between new agents and devices is related to their capacity to communicate homogeneously (i.e. interoperate seamlessly), avoiding the main hindrance posed by their hardware and software heterogeneity at physical and connectivity layers. Homogeneity regarding devices, applications and, generally, things in the IoT can be described as things sharing the same interface, or the same contract at multiple levels. Scalability is related to interoperability and homogeneity of things in that it enables other things to use the same interface to communicate and understand every each other. Therefore, scalability enables high reusability of abstractions over heterogeneous agents. Another perspective of scalability is related to a scalable network structure. Ideal scalability means a constant and graceful performance of the system as the number of objects it encompasses increases by orders of magnitude. Hence, interoperability between heterogeneous devices is one of the main obstacles before enabling scalable object interaction and monitoring. Otherwise, we need to keep deploying specific applications, which are usually islanded from the rest of the system and create IoT silos.

Given objectives **O1**, **O2** and **O3** we set several outcome-related challenges to accomplish. The main body of work should be published in, at least, fourth-quartile international journals. This implies acceptance of work after peer-review by members of the research community. Reviewers' constructive comments are valuable to assess the validity and originality of the research endeavour. The research endeavour implies following a research methodology; consequently, published (and validated) manuscripts need to be constructed following such methodology, which validates that the authors understand the methodology **O1**.

One of the first steps of the research methodology is gathering enough state-of-the-art information to observe the research gaps on the scientific knowledge. We find that this research gaps are usually best and more easily observed as one gets closer to the the scientific frontier. We have set our main focus on the landscape of the IoT and its integration with SGs; particularly, since the IoT is already part of the SG, our objective is to explore and integrate enhancing technologies of the IoT to SG. Accordingly, publications, each one forseen to explore and contribute to one of these enhancing technologies (namely WoT and SIoT) should include a state-of-the-art analysis — in extension as of those included in an experimental manuscript — to identify research gaps (**O1** and **O2**).

Finally, we want to contribute in the properties of scalability and interoperability of IoT integrated with SGs; our aim is to integrate novel IoT technologies with the SG (**O3**). We want to that by considering different scenarios of the SG and applying those novel technologies. To asses the validity of our approaches, we will consider experimentation (using prototypes and simulations) and qualitative and quantitative analyses (**O1**).

1.3 Contributions

This thesis contributes to two of the novel technologies expected to make the IoT scalable. The WoT, with a particular focus on the semantic web, as the homogenization technology; and the SIoT, as the technology that will enable, among other things autonomous and scalable management of smart devices. Specifically, the contents of this dissertation focus on the field of the SG.

The work presented in this dissertation is described in three published articles: [39–41]. They all bring the ICTs closer to the SG and Future Smart Grid. Articles [39, 40] base their contributions on top of the WoT and the semantic web towards energy management through the web (Web of Energy (WoE)) and device interoperability through service interoperability respectively. Work in [41] describes a novel approach towards the integration of prosumers and SIoT. The following list summarises the specific contributions of this dissertation along with the tasks performed by the author of this thesis. The specific tasks of each co-author are given in the preamble of Chapters 2, 3 and 4.

1. "Prototyping a Web-of-Energy Architecture for Smart Integration of Sensor Networks in Smart Grids Domain" [39] describes an application to integrate heterogeneous Wireless Sensor Networks (WSNs) into the SG using the WoT. To develop the integration application, the authors use a programming model used to develop concurrent and high availability applications. As such, the application developed is also a contribution of [39]. Víctor Caballero contributed to the conception of the idea and the decision of the technologies involved in the prototype. Víctor also developed the prototype, performed the experiments, and carried out the analysis.

- A preliminary version of the work was presented in the national scientific sessions "JITEL 2017" [42].

2. "Ontology-Defined Middleware for Internet of Things Architectures" [40] develops a novel application harnessing semantic web technologies and integration patterns proposed by the WoT. An application based on the findings in [39] and a new ontology derived from existing ontologies are deliverables of this work. Víctor Caballero contributed to the conception of the idea and the decision of the technologies involved in the prototype. Víctor, along with a co-author, implemented the prototype. The author of this dissertation also contributed with insightful knowledge about the technologies involved in the prototype development. The posterior analysis was carried out by Víctor and a co-author.

3. "Social Internet of Energy - A new paradigm for Demand Side Management" [41] presents a new concept, the Social Internet of Energy (SIoE), as the technology able to semi-autonomously create PCGs with a common goal: manage and optimize energy at the demand side. In the article, the authors perform a thorough analysis of the different characteristics of the SIoT and engage in defining a new model of prosumer interaction through their devices. Víctor Caballero conceived the idea with the contribution from another co-author. Víctor executed the experimentation and analysis presented in the manuscript.

    - The work was disseminated by the author of this dissertation in a conference held in Barcelona, Spain, during the "IoT Solutions World Congress 2019" [43].

Table 1.1 summarises the publications that form the compilation of this thesis:

Table 1.1: Publications of this thesis.

Name	Journal	Quartile	Impact Factor	Ref.
Prototyping a Web-of-Energy Architecture for Smart Integration of Sensor Networks in Smart Grids Domain	SENSORS	Q1	3,031	[39]
Ontology-Defined Middleware for Internet of Things Architectures	SENSORS	Q1	3,031	[40]
Social Internet of Energy – A New Paradigm for Demand Side Management	IEEE Internet of Things Journal	Q1	9,515	[41]

1.4 Roadmap

The starting point is a pre-doctoral (before enrollment in the doctorate program) publication ([44]) that explores the WoT and the most used language in web development, PHP: Hypertext Preprocessor (PHP) [45, 46], already considering the SG. Enrollment in the doctorate program happens on March 1, 2017. The first task is to analyse existing WoT architectures and associated technologies. Almost in parallel, the author starts searching for datasets related to SG. The WoT analysis, commingled with an analysis of SG's needs derives in a task to combine a WoT with the SG, continuing the work in [44], the WoE. The ongoing work is disseminated in a national scientific session [42] on September 27, 2017. Ongoing work encompasses a review of the state of the art, an architecture model for the WoE, and a prototype of the proposed architecture. Subsequently, processed, real datasets are integrated with the prototype and experiments are carried out. The work is published in an international, peer-reviewed journal on January 30, 2018 ([39]). Two future lines emerge from [39].

The first research line considers semantic web technologies. The authors find a research gap in using those technologies to directly provide an interpretable and machine-readable structure to describe the control flow of WoT-enabled smart devices. They explore diverse (semantic) ontologies to describe control flows. Thanks to the knowledge gathered in [39], the authors develop an ontology-defined middleware prototype that allows deploying VOs described by a semantic and machine-readable structure. The work is published in an international, peer-reviewed journal on March 7, 2019 ([40]).

The second research line considers the SIoT and its integration with a service of the SG, DSM. A proper exploration and state-of-the-art review on both SIoT and DSM in conjunction with PCGs is performed. To the best of the authors' knowledge, there has not been any work integrating SIoT and SG. Hence, the authors propose a synergy between SIoT and DSM, and develop proper algorithms to initiate the assessment of their proposal through experimentation with newly gathered real datasets. The work is published in an international and peer-reviewed journal on August 1, 2019 ([41]), and disseminated in the "IoT Solutions World Congress 2019" ([43]) on October 31, 2019. Further research that follows the path outlined in [41] is currently being carried out.

Figure 1.2 depicts the roadmap. The reader can better observe that some of the threads just described occurred in parallel. That is, the analysis of diverse technologies (namely WoT, SG, and SIoT) occurred while carrying out the development of prototypes and experimentation. Also, exploration, analysis and processing of datasets containing real data is done in parallel.

1.5 Thesis outline

This thesis by compilation of papers follows a "sandwitch model". After the introduction (Chapter 1), the three published articles [39–41] follow, and correspond to Chapters 2, 3 and 4 respectively. Chapter 2 ([39]) describes a novel WoE architecture based on WoT and experiments with a prototype implementation. Chapter 3 ([40]) presents an ontology-defined middleware based on WoT architecture and semantic web technologies using the experience gained in Chapter 2. Chapter 4 ([41]) coins the concept of SIoE derived from SIoT, analyses and describes its features and performs initial experimentation. Chapter 5 summarizes the results obtained in each article and discusses them. Finally, Chapter 6 concludes this document and presents future work. A copy of the articles as published or accepted is available in the Appendix.

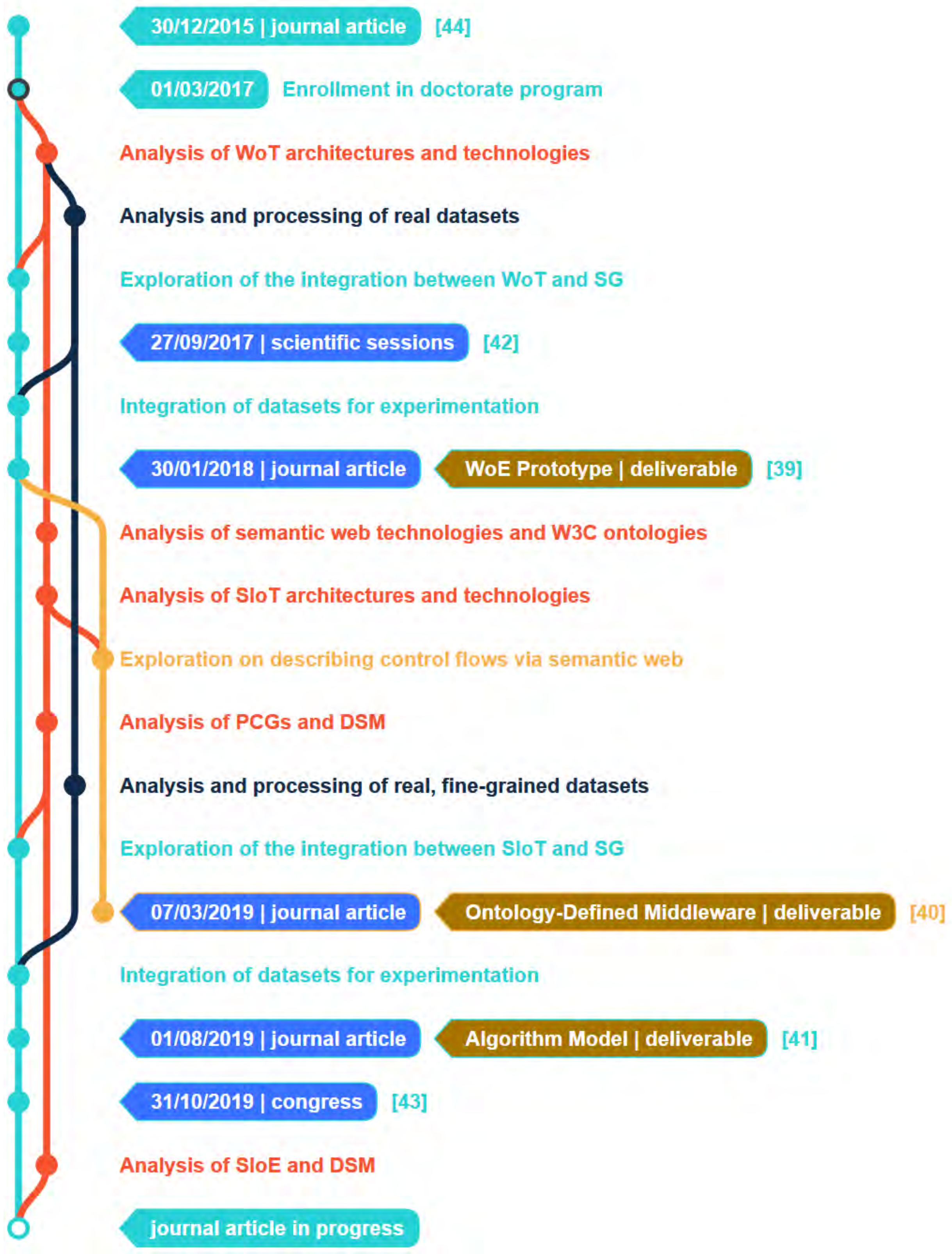

30/12/2015 | journal article
[44]
01/03/2017
Enrollment in doctorate program
Analysis of WoT architectures and technologies
Analysis and processing of real datasets
Exploration of the integration between WoT and SG
27/09/2017 | scientific sessions
[42]
Integration of datasets for experimentation
30/01/2018 | journal article
WoE Prototype | deliverable
[39]
Analysis of semantic web technologies and W3C ontologies
Analysis of SIoT architectures and technologies
Exploration on describing control flows via semantic web
Analysis of PCGs and DSM
Analysis and processing of real, fine-grained datasets
Exploration of the integration between SIoT and SG
07/03/2019 | journal article
Ontology-Defined Middleware | deliverable
[40]
Integration of datasets for experimentation
01/08/2019 | journal article
Algorithm Model | deliverable
[41]
31/10/2019 | congress
[43]
Analysis of SIoE and DSM
journal article in progress

Figure 1.2: Thesis roadmap. The roadmap reads from top to bottom. Each coloured line represents a branch. Milestones are represented by points in each branch and the text on the right describes the milestone. Relevant milestones such as publications of articles or congresses are indicated on the right by tags (rectangles filled with colour).

2 Prototyping a Web-of-Energy Architecture for Smart Integration of Sensor Networks in Smart Grids Domain

2.1 Preamble

2.1.1 Abstract

Sensor networks and the IoT have driven the evolution of traditional electric power distribution networks towards a new paradigm referred to as SG. However, the different elements that compose the ICTs layer of a SG are usually conceived as isolated systems that typically result in rigid hardware architectures which are hard to interoperate, manage, and to adapt to new situations. If the SG paradigm has to be presented as a solution to the demand for distributed and intelligent energy management system, it is necessary to deploy innovative ICT infrastructures to support these smart functions. One of the main issues of SG is the heterogeneity of communication protocols used by the smart sensor devices that integrate them. The use of the concept of the WoT is proposed in this work to tackle this problem. More specifically, the implementation of a SG's WoT, coined as the WoE is introduced. The purpose of this paper is to propose the usage of WoE by means of the Actor Model paradigm to address the latent deployment and management limitations of SGs. SG designers can use the Actor Model as a design model for an infrastructure that supports the intelligent functions demanded and is capable of grouping and converting the heterogeneity of traditional infrastructures into the homogeneity feature of the Web of Things. Conducted experimentations endorse the feasibility of this solution and encourage practitioners to point their efforts in this direction.

2.1.2 Author Contributions

Víctor Caballero and David Vernet conceived the idea and decided the technology involved in the experiments; Víctor Caballero developed the prototype; David Vernet and Agustín Zaballos supervised the work, and Agustín Zaballos contributed with relevant materials about Smart Grids domain; Guiomar Corral contributed in the related work and in the revision of the paper. All authors contributed equally to writing the paper.

2.1.3 Funding

Part of this research has been funded by the "SUR" of the "DEC" of the Generalitat de Catalunya and by the European Social Funds "2017FI_B 00583". In addition, we would like to thank La Salle (Universitat Ramon Llull) for their support.

2.1.4 Conflicts of Interest

The authors declare no conflict of interest.

2.2 Introduction

Contrary to the rapid evolution experienced in the last decade of ICTs, electric power distribution systems have remained exceptionally steady for a long time. Therefore, the SG concept, the addition of a telecommunication infrastructure to the electrical domain, has enabled a plethora of new services and opportunities (e.g., accurate grid monitoring, real-time energy consumption, and customer-side generation, etc.), which are currently driving a major revolution in the energy sector [5, 6]. In fact, SGs are conceived to improve traditional electric power networks in several dimensions [7] such as information, business models and scalability, in order to meet the highest standards of power quality and thus ensuring that the electric power grid is cost effective and sustainable [8].

Moreover, new agents and components have been raised inside this new paradigm. For example, the prosumer (load producer and load consumer) as the last link in the electricity value chain is easily the most important creator of value within the smart grid. Prosumers will be given a more active role in new business model generation. The most important connections for prosumers in the electricity value chain are the distributed system operator or the aggregator/retailer in addition to the ESCO or VPPs, which are also new components in the energy market value chain [7]. Specifically:

1. The aggregator controls low voltage power that is transferred to the usual places, where it is consumed with metering and billing functionalities.
2. ESCOs will play an important role in the future electricity market as energy-oriented commercial businesses. ESCOs can be described as specialists in providing a broad range of comprehensive energy solutions, including the design and implementation of energy saving projects, energy conservation, energy infrastructure outsourcing, power generation and energy supply and risk management.
3. A VPP can either operate large numbers of relatively small-sized generators, responsive loads and storage units on behalf of owners (in this case, prosumers) or operate its own DERs.

Those new agents and components of the SG could become the most enriched elements due to their added information and technology features. However, the upgraded features of the end user side come at the cost of requiring more technology in order to make them usable. Due to the complexity (i.e., stringent levels of service reliability and availability), magnitude (i.e., large-scale areas), and new agent profiles inherent to SGs, practitioners have recently addressed the digital transformation of power electric networks by proposing flexible and future internet based architectures [8].

As far as the communication and network protocols' field is concerned, the SG has an inherent heterogeneous nature which forces system architects to consider different telecommunication technologies. SGs are deployed in hostile wireless communication environments [12]; aware protocols [13] (also referred to as cognitive radio techniques) are needed to reduce communication delay [12], meet QoS needs in terms of delay, bandwidth, and data reliability, improve energy harvesting techniques [13], and provide reliable distributed sensing [14] in order to minimize interoperability issues between heterogeneous communication networks [15].

The IoT is introduced as the natural evolution of Internet, as it refers to the heterogeneous agglomeration of devices connected to the network. This evolution has succeeded thanks to the advances of both silicon, which have made it possible for increasingly smaller and smaller computing units to be embedded into everyday devices, as well as advances in low-power wireless protocols for such devices. The ICTs for the SGs represent a good example of this heterogeneity, where different devices, both sensors and actuators, from different vendors and using different protocols, have to work together to achieve an objective: the integration of energy and smart services. Due to this heterogeneity issue, the WoT [24] is presented as an abstraction layer that enables the homogeneous interaction between devices of different kinds using web technologies (previous work in JITEL 2017 [42]). In this sense, the grouping of the application of the methodologies provided by WoT to the management of SGs under the term WoE [44, 47] is proposed. In previous works [44], some proposals were introduced in this sense, however, it soon became apparent that another approach was needed for the development of an architecture for the WoE.

The paper is organized as follows: Section 2.3 describes the related work regarding SGs, WoT, and WoE, detailing their value proposals and associated challenges. Section 2.4 explains the problem of the direct integration of SGs, WoE, and their communications protocols in order to compose the WoE. Section 2.5 illustrates the hybrid architecture that is proposed that covers the needs of a SG infrastructure and provides the advantages of integrating the SGs' ICTs with the WoT, that is, the WoE. In Section 2.6, we conduct experiments to evaluate the performance of the prototype system and discuss the results. Section 2.7 concludes the article. Finally, Section 2.8 discusses further work.

2.3 Related Work

2.3.1 Smart Grids and Web of Energy

Over the last decade, SGs have led the revolution of the electrical grid, transforming it into a set of automated and efficiently controlled processes by the incorporation of ICTs. SGs promote electrical energy management in a distributed and flexible manner. However, the current management systems are (i) centralized regarding their management; (ii) located in independent locations between them; and (iii) managed by fragmented applications, without integration between them and only intercommunicated thanks to specific communication channels, which are generally proprietary. The main goal of SGs is to provide better services and features (also known as smart functions), for both consumers and for producers and prosumers. In addition, the increased use of distributed and renewable energy generation requires changes in the electricity management system. It is necessary to improve automation systems, distributed intelligence, real-time data mining and management to improve network control functions, simplify configuration and also reduce system recovery and self-healing times.

Recent advances in SGs have explored the feasibility of considering the electrical power distribution networks as a particular case of the IoT. Certainly, this specific domain poses appealing challenges in terms of integration, since several distinct smart devices from different vendors (wired or wireless sensors, smart meters, distributed generators, and so on), often using proprietary protocols and running at different layers, must interact to effectively deliver energy and provide a set of enhanced services and features [15]. Although the latest developments of the IoT field have definitely contributed to the physical connection of such an overwhelming amount of smart devices, several issues have arisen when attempting to provide a common management and monitoring interface for the SG [22].

To solve the integration of heterogeneous devices, the use of the concept of the WoT [24] has been proposed to access multiple devices using the same interface provided by web technologies, implying uniformity in communication protocols (HTTP/S and WS) and uniformity in the model of data representation and device discovery (Web Thing Model [48] and Semantic Web [30]). This concept is, so far, difficult to apply in real scenarios involving energy management due to the risk of creating new security vulnerabilities, the lack of devices that implement standards of the WoT and the opposition of the industry of energy to include external modules or devices in their proprietary systems.

The objective of the work presented in this paper is to create an architecture based on the IoT paradigm to manage the storage and communication needs of the SGs and at the same time link the SGs with the end user through the methodologies of the WoT. For this, a bidirectional Human-To-Machine interface is established, inspired by the WoT that allows the ubiquitous control of the energy systems, the WoE [47].

In this way, the WoE could represent an opportunity for the electric companies to have virtual representations of their more flexible devices (based on software, updatable, configurable, and with the possibility of deploying new applications on top of them), enabling a low-cost distribution and management of the electric network [49]. The WoE facilitates the sharing of data from different devices (charging points for electric vehicles, smart metering, or monitoring of substations) with third parties. In addition, the use of web technologies allows the creation of multiplatform visualization tools without installation cost, being able to offer simple and usable graphic interfaces for a greater adoption for the DSO or any user interested in the consumption or production of energy (prosumer). Thus, we can identify different fields related to energy management in which the WoE would be very useful:

1. Remote access form substations to central servers.
2. Management and monitoring of DERs.
3. Supervisory Control And Data Acquisition (SCADA) distribution to secondary substations.
4. Electrical Vehicle Supply Equipment (EVSE) management.

2.3.2 Web of Things

Although the initial predictions of one trillion devices connected to the Internet by 2015 [50] were quickly lowered to 26 billion by 2020 [51] and 20.8 billion in the same year [52], it is evident that the number of devices connected to the Internet is increasing every day. How to access all these sensors and actuators through a uniform interface is undoubtedly one of the biggest challenges of the IoT. Currently, the IoT is divided into self-contained areas, that is, there are proprietary solutions that help the integration of a specific set of devices, but this structure is far from the overall integration planned for the IoT. For this reason, the use of existing web technologies for the global integration of devices is proposed. The basic prerequisites for enabling the IoT devices on the web are two: (i) minimum capacity for data processing and (ii) connectivity to the network (it is not necessary to connect directly to the Internet).

The model proposed by the WoT [24] aims to solve the challenge of the heterogeneity of the IoT devices. This is principally achieved by generating translators or mappings between the language spoken by each device (communication protocol and data format) and the language that we can consider "universal" due to its widespread use: web technologies. Specifically, we have the HTTP/S protocol and the RESTful API [53]. These translations enable the devices to speak the same language, which makes them accessible in a homogeneous way to either human or computerized actors, like other devices. The actions to be taken on these devices can be both to act and to sense.

If we focus on the data flow, it is composed of two states that can be understood as a cycle: from heterogeneity to homogeneity and from homogeneity to heterogeneity.

1. From heterogeneity to homogeneity: a device sends data captured with a specific format and protocol through a WoT translator, which translates the data into a common format and the protocol into HTTP/S.
2. From homogeneity to heterogeneity: the actor receives this data and decides to act on the device, for example, changing its configuration. It then sends an instruction using the HTTP/S protocol and a format common to the WoT translator which, in turn, translates the protocol and format of the data into the specific protocol and format of the device.

2.4 From the Smart Grid to the Web of Energy

2.4.1 Smart Grid Architecture Modules

Our proposed SG architecture is composed of three main independent modules (see Figure 2.1). Each module is detailed in what follows.

1. Context-aware security. This module aims to individually provide the needed security level for the proper operation of every smart function. For instance, for the use case of Smart Metering, this module can update all the encryption keys of the SG once an unauthorized access to the metering infrastructure has been detected.
2. Hybrid Cloud Data Management: This module provides a data storage and processing system that intrinsically adapts to the SG's topology in a scalable and flexible way. Also, it implements an algorithm (also referred to as orchestrator) to decide whether collected data should be stored at the private cloud or could be placed at the public cloud. Such decisions are taken considering the smart functions' requirements (e.g., reliability, delay, and cybersecurity) associated with the collected data [44].
3. Web of Energy: This module provides a ubiquitous (i.e., web based) monitoring interface [44] that enables a seamless management of the whole IoT architecture of the SG. In addition to providing a mechanism to communicate humans and machines, it also enables the interactions among those IoT resource-constrained and small devices (i.e., machine to machine) through the HTTP/S protocol. Note that the aforementioned context-aware security module might decide to add an extra layer of security by switching from Hyper-Text Transfer Protocol (HTTP) to HTTPS in accordance with the device features, network status, and smart function under execution demands. This is done through an open API that both couples and decouples all the modules. Hence, this module also acts as a bridge between the distributed storage layer—that takes care of all the SG's Big Data concerns—and the context-aware security layer—that gives the necessary access control and cybersecurity mechanisms.

2.4.2 Web of Things Architecture

The concept of WoT is developed mainly in the works of Dominique Guinard, [24] and Vlad Trifa [54], where the WoT and the methods of enabling devices from IoT to WoT, respectively, are presented. In[24] the WoT is organized into four layers, each with a specific

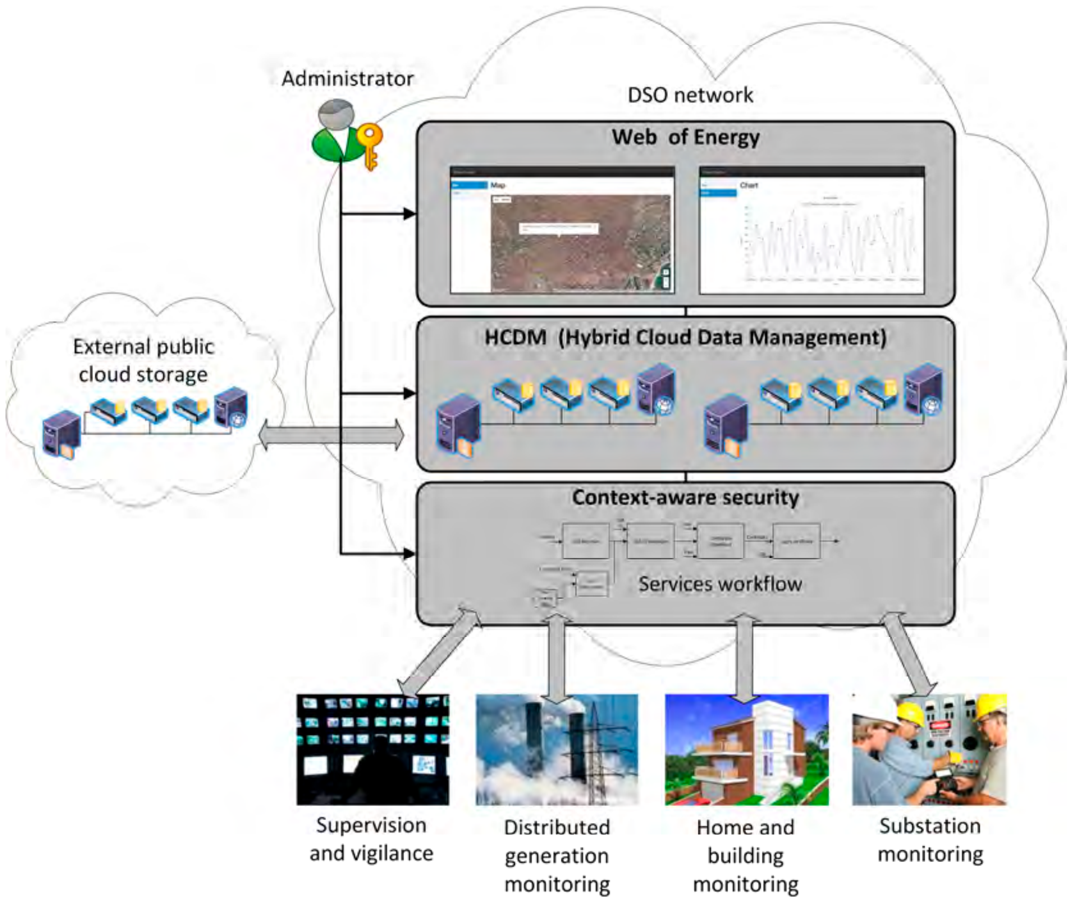

Administrator
DSO network
Web of Energy
External public cloud storage
HCDM (Hybrid Cloud Data Management)
Context-aware security
Services workflow
Supervision and vigilance
Distributed generation monitoring
Home and building monitoring
Substation monitoring

Figure 2.1: Main modules: Context-aware security, Hybrid Cloud Data Management and WoE.

function. However, these layers do not follow an isolation and encapsulation model between non-contiguous layers, as is the case with the Open Systems Interconnection (OSI) or Transmission Control Protocol/Internet Protocol (TCP/IP) model but, on the contrary, the applications can be built on top of each one (Figure 2.2), since all of them are part of the application level of the two network models mentioned above. Next, the different layers of the WoT proposed in [24] are presented.

The functions of each of these layers are the following:

1. Accessibility: It enables a consistent access to all types of IoT devices by exposing their functionalities through a RESTful HTTP/S API.

2. Findability: It provides the discovery of the representations of the different devices, uniformly modeling the access method (through the HTTP/S protocol) and establishing relations between them at the moment of their representation.

3. Sharing: It is in charge of preserving the privacy between the device representations. It also manages the authentication and authorization to access the representations by actors who do not own the device.

4. Composition: It enables the integration between the different representations of the devices that, ultimately, allows the integration between the different functionalities of the physical devices.

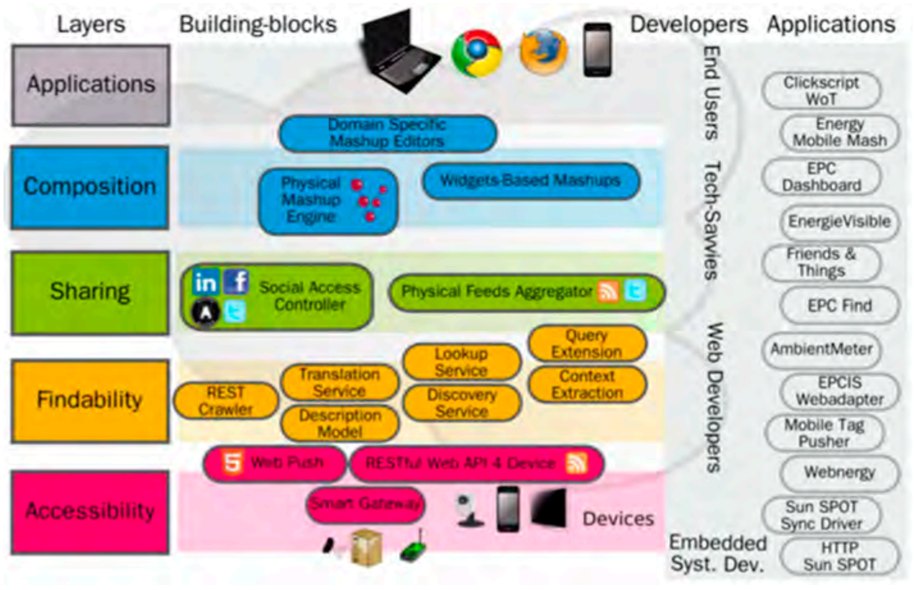


Figure 2.2: WoT layer model proposed in [24].

Accessibility Layer

The accessibility layer acts as an interface between the IoT and web technologies. Therefore, it is the closest layer to the heterogeneity of IoT protocols, such as Message Queuing Telemetry Transport (MQTT) [55] or Constrained Application Protocol (CoAP) [56], among others. In this layer, solutions are found in the form of a proxy or bridge between the IoT and HTTP/S protocols.

Two basic WoT-enabling methodologies are considered. Embedding web servers to devices, or the creation of gateways [57] that serve as aggregators or proxies between IoT protocols and web protocols. Cloud solutions are also found in this layer, generally transverse to more layers of the WoT, such as ThingWorx [58], Watson IoT Platform from IBM [59], Octoblu [60] or EVRYTHNG [61], among others. These cloud solutions offer IoT-WoT translation gateways.

Findability Layer

This layer is composed of the technologies that allow for the exploration and searchers of the different devices exposed to the WoT. Those technologies allow the publication of structured and interconnected data. The concept of REST [53] and Linked Data (LD) [31] is applied to interconnect different device representations (usually in the form of Uniform Resource Locator (URL). Semantic Web technologies [30] such as Resource Description Framework (RDF) [62], Resource Description Framework in Attributes (RDFa) [63] or Web Ontology Language (OWL) [64] are used to provide semantic meaning to both representations and connections with other representations, allowing, for example, the exposition of the information to search engines.

Sharing Layer

This layer groups all the methods that allow authentication and authorization of different actors to perform an action on the virtual representation of the device and, ultimately, on the physical device. It groups simple authentication and authorization methods from user authentication and password to the use of more advanced protocols such as API or OAuth keys.

A middleware called *Social Access Controller* is proposed in [65]. By combining the OAuth API of different social networks and the access by simple credentials to the devices (e.g., user and password), it allows to: (i) preserve the privacy of the physical devices; (ii) take advantage of the structure and function of social networks to authenticate the different actors potentially interested in participating in the actions that can be performed on the device representation; (iii) integrate the representations of the devices in social networks, creating a Social Web of Things (SWoT); and (iv) enable the publication of data aggregation using data syndication protocols like Atom [66].

Composition Layer

This layer enables the composition of the different functionalities exposed by the device's representations. Taking into account that each representation is accessible through the HTTP/S protocol, it is easy to program a script so that the different devices act in a coordinated way. The elaboration of this idea consists of providing the web user (human) with a visual interface to compose different device relationships. Solutions such as ThingWorx Composer [67], NODE-Red [68] from IBM, Octoblu [60] or IFTTT [69] exemplify the composition of physical devices (physical mashups).

2.4.3 Web of Things and Web of Energy Protocols

The main element of the WoT proposal is the use of web technologies for the transmission of information. As far as web protocols are concerned, there is the HTTP/S protocol and the WS or WebSocket Secure (WSS) protocol. HTTP/S is a request/response protocol while WS allows a bidirectional connection between the client and the server.

However, many devices do not have the necessary features to directly access the web through HTTP/S or WS. To be able to connect to the web, they need to use gateways that are responsible for translating the heterogeneity of the IoT protocols to the homogeneity of the WoT. To do this, it is necessary to establish translation bridges or mappings from IoT to WoT protocols and vice versa.

For example, both CoAP and MQTT are currently two of the most promising standard IoT protocols. CoAP, an open standard, was designed specifically for the IoT and to be directly compatible with HTTP/S [70]. It is a protocol based on request/response through User Datagram Protocol (UDP) packets and follows the same schemes as HTTP/S, allowing the creation of REST resources. The MQTT protocol, designed by IBM, is now also an open standard, but follows a publisher/subscriber paradigm over Transmission Control Protocol (TCP), so it becomes more complicated to establish a translation bridge between HTTP/S-REST and MQTT [71].

Table 2.1 shows the differences and similarities between CoAP, HTTP/S, MQTT and WS protocols. As mentioned, the mapping between CoAP and HTTP/S is direct, since the topology of both is request/response (Req/Resp). On the other hand, the topology between HTTP/S and MQTT is different, so the translation between these two protocols is more difficult to achieve. Another example of (almost) direct mapping is between WS and MQTT, since the bidirectional communication of WS can be considered a particularity of the publisher/subscriber protocol (Pub/Sub) of MQTT. Thus, HTTP/S and WS are the two web technologies with different topologies that allow translation between IoT protocols with request/response and bidirectional topologies respectively. Table 2.1 also shows other protocols that can be part of the IoT, which have more similarities with HTTP/S or WS according to their topology in the same way that occurs with CoAP and MQTT.

Table 2.1: WoT/IoT protocols.

	HTTP/S	**WS**	**CoAP**	**MQTT**	**XMPP**	**AMQP**	**DDS**
Topology	Req/Resp	Two-way, realtime	Req/Resp	Pub/Sub	Pub/Sub and Req/Resp	Pub/Sub	Pub/Sub
Architecture	Peer-to-Peer (P2P)	P2P	P2P	Broker	P2P	P2P or Broker	Global Data Space
Transport Level	TCP	TCP	UDP	TCP	TCP	TCP	TCP/UDP
Encryption	TLS	TLS	DTLS	TLS	TLS	TLS	TLS/DTLS
Authentication	TLS	TLS	DTLS	User/Pass	TLS	SASL	TLS/DTLS
QoS	-	-	Confirmable	3	-	3	23

In addition to the two protocols already presented in Table 2.1 and many other proprietary and open-source protocols for IoT, protocols that are more adapted to the specific needs of electric utilities are beginning to emerge, particularly in sectors where the infrastructures are becoming obsolete. However, those protocols become more difficult to adapt to the WoE because: (i) its specificity reduces the interest of third parties of contributing to the exposure of the devices (or sets of them) to the WoT and, therefore, the companies should invest more capital in the mapping and (ii) in case it is of interest for the company to expose some of their devices, the WoT adds a stack of new technologies, opening new security holes in their systems. These challenges are significant in the WoE, because the systems involved cover a large number of devices that use very specific protocols adapted to their needs.

2.4.4 Complementing the Web of Things Architecture

The main goal of this section is to list the key features that a WoT architecture should comply with. These features are the following:

1. Although certain IoT devices can support HTTP/S stacks, there are many of them that can only support lighter protocols due to their limited resources. Although ideally a narrow range of IoT protocols (e.g., CoAP and MQTT) would facilitate the integration of the devices to the Internet and the Web, a basic and indispensable requirement for the devices to be part of the Web is that they can be connected to the Internet.

2. The architecture must provide abstractions so that developers can interact independently with the devices of the communication protocol, either towards the heterogeneity of the IoT protocols or towards the homogeneity of the WoT protocols.

3. The architecture must be able to scale horizontally and provide self-healing for its systems. It must permit the on-demand deployment of resources.

4. In the WoE architecture, the devices have to become VO [34] or things, although VOs created from aggregations of other VOs can also be created.

5. Each VO will be accessible from an HTTP/S REST interface and, in case the features of the VO allow it, from a WS link.

6. The architecture must allow executing authentication and authorization protocols towards the different devices involved.

In our proposal, we have identified five layers that compose the architecture of the WoE (Figure 2.3). Although they have a deep relationship with the layers proposed in [24] and other emerging architectures [72], some differences can also be identified, mainly due to the fact that the layers presented below are more focused on the development of the WoE architecture.

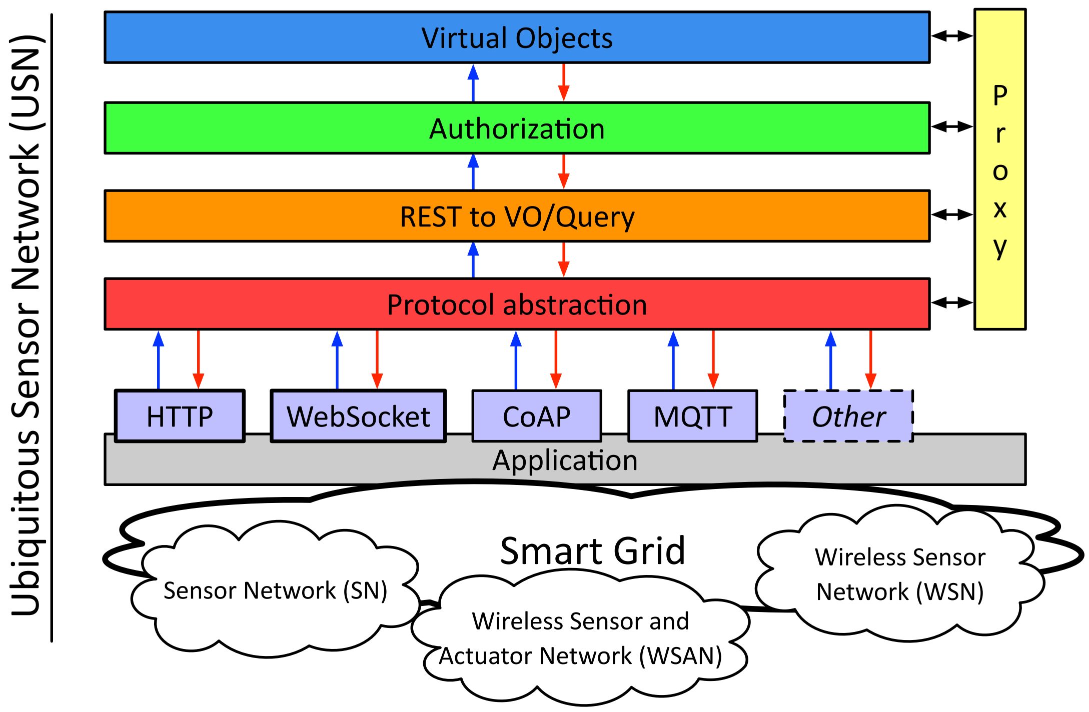


Figure 2.3: Proposed architecture for the WoE.

The description of the different layers of the proposed WoE architecture is the following:

1. Protocol Abstraction Layer: The objective of this layer is to provide an abstraction layer for developers to interact with physical devices, both for developers of architecture functionalities at the more internal layers as well as for application developers for SG that, as already specified, group different protocols. The goal is not only to expose the IoT devices as HTTP/S REST resources, but to provide developers with abstraction mechanisms of both HTTP/S and the IoT protocols. In this sense, the premise exposed in [24] of enabling access to physical devices through an HTTP/S interface is maintained.

2. REST to VO/Query: The purpose of this layer is to translate the different Uniform Resource Identifiers (URIs) generated from the information of physical devices to actions on VOs or Things. In this way, we do not act directly on physical devices but through VOs. This layer also includes an interface for more complex queries about a *Thing* or the relationships of different *Things*.

3. Authorization: This layer is responsible for requesting and granting access to VOs or *Things*.

4. VOs: When physical devices must perform actions on other physical devices they will send the instruction against a virtual representation (VO or *Thing*). The objective of this virtual representation is to increase the resources and functionalities of physical devices. We cannot provide a detailed list of added features, as they are specific to each solution. Even so, our goal is to provide a method (code injection) so that these functionalities can be added dynamically. Some common functionalities are related to reasoning (artificial intelligence) or caching.

5. Proxy Layer: The architecture of the WoT presented in [24] proposes that all representations of the devices use web technologies (e.g., HTTP/S) to communicate and expose their characteristics and achieve homogeneous access to the devices, both between devices and between devices and humans. However, using web technologies for all types of communication between specific servers of the WoT will not always be optimal. The motivation of this layer is therefore to show that efficient communication between servers can be carried out through a protocol that is not within web technologies. However, it is important to mention that even those services should have an HTTP/S interface for the integration in the WoT.

Finally, the application layer conceptualizes all those applications that are likely to interact with the architecture. Within the range of applications supported by the WoT and, therefore, the IoT, those applications of Smart Cities and SGs are grouped. Specifically, the generation of a Ubiquitous Sensor Network (USN) [15] is an application that includes the agglomeration of wired and wireless networks of sensors and actuators.

2.4.5 Web of Energy Implementation

First Approach: PHP Implementation

Our first goal in [44] was to create a high-performance WoT architecture, while experimenting with the most used web programming language up to that point, PHP, to see if it could be facilitated to the many web programmers in PHP [45, 46] the programming interfaces for the WoT, thus achieving greater adoption in less time. However, the results were not positive for the following reasons:

1. PHP web servers follow a very strict execution model that hinder the use of web technologies such as WS. This is because WS connections are permanent, whereas PHP servers delimit the connection with the client until a certain time limit or until the script execution is finished, since its main execution model is “load, execute, and die”. There are alternatives to this execution model like the one implemented in ReactPHP [73] (reactor pattern), but this library is not available for large-scale production environments such as WoT.

2. PHP, an interpreted language, is slower than compiled languages. In addition, with the execution model discussed above, main developers have never been concerned with optimizing the generated instructions or eliminating multiple memory leaks. Now, with the arrival of PHP 7, an improvement in performance has been achieved [74], although the other reasons discussed in this list are still valid.

3. There are very few libraries focused on functionalities that involve mathematical calculations intrinsic to Machine Learning (ML) or distributed computing models. This is due, once again, to the idiosyncrasies of this type of languages, which are mainly used to serve dynamic web pages.

Second Approach: Actor Model Implementation

PHP is not efficient enough nor does it have the necessary tools for a large-scale implementation of an infrastructure for the WoT, as it was concluded in [44]. Once we discarded the use of PHP to develop the infrastructure of the WoT, we looked for a programming language that had the performance, the ecosystem and the necessary tools to develop such infrastructure. The Java Virtual Machine (JVM) provides a stable execution environment (since it is supported by Oracle) and many tools and libraries (protocols, web frameworks, and Machine Learning libraries) are available for this platform.

On the other hand, we were interested in finding a programming model that facilitated the creation of distributed, flexible, and self-healing systems, mandatory features for the WoE. For this reason, the Actor Model [75, 76] seemed to be the ideal model. This model or paradigm of programming facilitates the creation of systems with those requirements and, in addition, thanks to the fact that it allows concurrency, it is capable of taking advantage of the different device computing cores [77].

The basic principles of operation of this model are that when an Actor receives a message, it can:

1. Send messages to other Actors.
2. Create new Actors.
3. Designate how to manage the next received message.

Several Actors can run concurrently although an Actor can only process one message at a time. That is why each Actor receives messages in a totally asynchronous way, storing them in a local message queue. Figure 2.4 represents this idea.

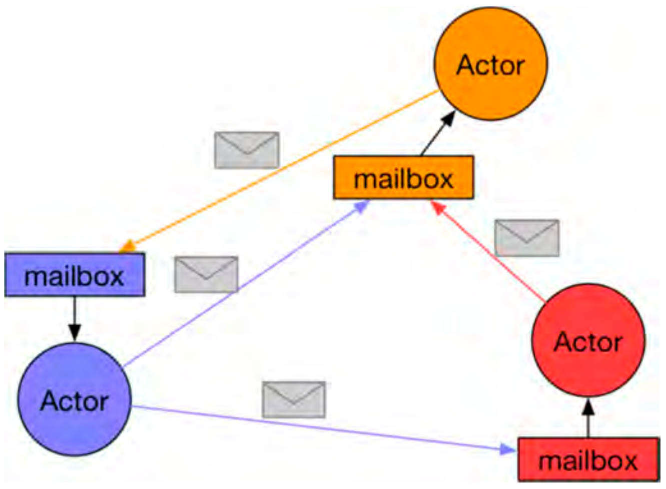


Figure 2.4: Representation of the messaging system in the Actor Model.

As can be observed, when an Actor receives a message, it is stored in a mailbox that is specific to each Actor and inaccessible to the rest of the Actors. This mechanism maintains a private status and isolates an Actor from the rest.

Although the Actor Model can be used to create agents and, therefore, Multi-Agent System (MAS) this article focuses on the exposure and use of the model per se in IoT or WoT.

In [78], authors benefit from this model to enable features such as multi-cloud and multi-tenure for IoT devices. They take advantage of the few resources that an actor needs to operate by compartmentalizing the physical devices in different software modules (actors) that are isolated and connected to different cloud infrastructures and have different owners.

In [79], the use of C++ Actor Framework (CAF) is proposed to provide developers with a high-level Operating System abstraction framework to develop IoT applications. The authors also emphasize the properties of abstraction, distribution and flexibility of this computation model.

Thanks to the principles on which the Actor Model is built, we can obtain interesting properties for the WoE infrastructure.

1. Distribution and Flexibility: One of the basic principles of the model is the creation of new actors. This facilitates the use of computing resources not only in the same node but distributed in different nodes since the communication is asynchronous and transparent between local and remote node actors. Besides being able to create new actors, they can also be eliminated and, therefore, permit the management of the use of resources.

2. Self-Healing: This property is not based on any principle of the theoretical Actor Model, but all the libraries that implement this model also implement it since the creation of Erlang [80], due to its great practical use. By definition, the state of an actor is isolated in that actor and can only communicate with the outside through messages. Self-healing takes advantage of this isolation principle to manage the failure of a given actor. For example, Akka [81] implements parental supervision. The creation of actors by another actor results in the creation of a hierarchy with a parent actor. In the event of the failure of a "child" actor, the "parent" actor can decide how to manage this failure: restart it or not, for example. Thanks to the isolation between actors and the supervision between actors, execution errors can be isolated and managed in a controlled manner. Actor state recovery mechanism involve event sourcing [82] techniques, where messages that change the state of the actor are logged and replayed when the actor restarts.

Thanks to the basic principles of an actor and the properties that derive from them, we also propose that each actor (or group of them) represents the virtualization of a physical device, increasing the resources and functionalities of such a device and isolating the execution failures of the other actors, that is, of the system. Actually, this proposal is not novel because there are different references [75, 76] that propose this same use for the IoT. Our intention is to provide a working framework and an implementation for the WoE that meets its expectations. It is also our objective to promote the use of the Actor Model because it provides important abstractions to develop distributed systems, flexible, self-healing, and designed to achieve maximum and optimal use of resources thanks to parallel computing.

2.5 Prototypes, Initial Model, and Emulation Setting

In this section, we present the successfully implemented prototypes that are governed by the techniques analyzed above.

2.5.1 Layer Prototypes

Prototypes of some of the layers described above have been implemented for the proof of concept. Specifically, the VOs Space and the Protocol Abstraction layer have been implemented. Also, a publisher/subscriber protocol for the proxy layer has been considered to communicate two remote servers. Below both are detailed:

1. VOs Space: Each physical device is represented by a VO and this in turn represented by one or more actors. This layer defines the logic and abstractions necessary for each device to be represented by one or more actors, in such a way that the data flow, taking as a starting point the reception of data from a protocol of what we consider to be heterogeneous or IoT (e.g., MQTT), is transformed into a homogeneous protocol (proxy layer) so that it can be understood by the other parts of the architecture.

2. Proxy layer: According to the needs of this proof of concept, the proxy layer has been implemented through a publisher/subscriber protocol. In this way, the VOs that represent the devices can publish the captured data and can subscribe to messages also collected by other devices or to action messages sent by other devices.

3. Protocol Abstraction Layer: Contains the implementations that interface between different protocols such as MQTT, WS, or HTTP/S.

Figure 2.5 shows the interaction between the different layers in a SG. We find, considering a top-down vision, (i) the application layer with the intelligent functions of SGs; (ii) the middleware proposed in this paper; (iii) the accessibility layer to the USN with the aggregators of multiple sensors; and finally, (iv) the sensor networks. Intentionally, this schematic has a strong resemblance to Figure 2 shown in [15]. In this case, we have focused on the middleware development for a USN.

2.5.2 Initial Model

There are three well-differentiated sections in our implementation model. These sections are the following:

1. MQTT Section: It consists of those devices that communicate through the MQTT protocol and the servers or services responsible for translating MQTT into an "understandable" protocol.

2. VOs Space and Proxy Layer: This section is made up of those modules responsible for representing each physical device through a VO and the publisher/subscriber protocol.

3. HTTP/S/WS: Analogous to the MQTT section, it includes those devices that communicate with the architecture through web protocols and the services responsible for translating these protocols into a protocol "understandable" by the internal layers of the architecture.

The architectural model is depicted in detail in Figure 2.6.

In order to transmit data to end nodes of this architecture (Device/SmartGateway and WS Client in Figure 2.6), we have developed a communication protocol at the application layer that exploits the real-time capabilities of the proposed architecture. The protocol is depicted in Figure 2.7 and explains the actions conducted by each entity involved in the communication process. Involved entities are:

1. Device: A device or smart gateway. In a SG setting, a smart meter or any other intelligent electronic device can be the Device.

2. MQTT Broker: An MQTT server or broker that handles subscriptions and publications and deliveries messages from publishers to subscribers.

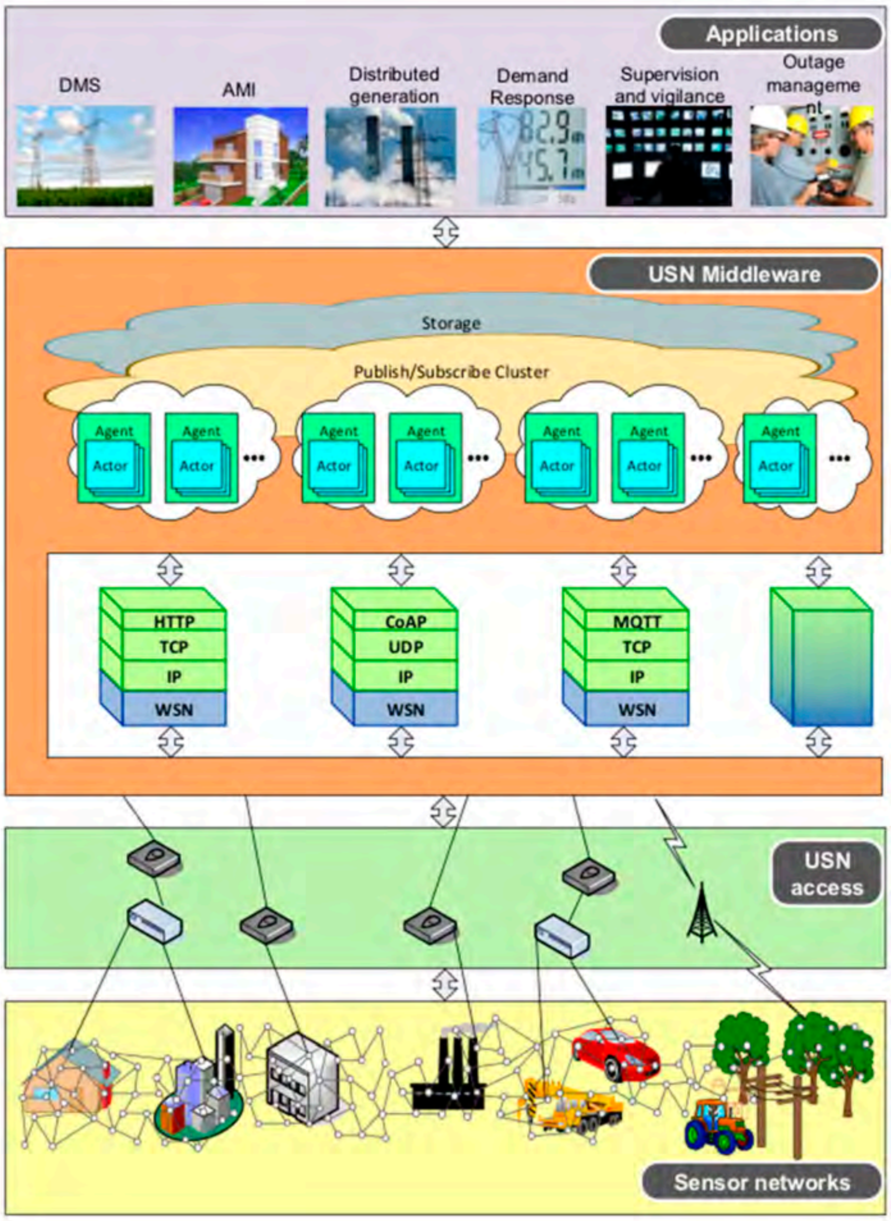


Figure 2.5: Layer schematic for a USN applied to Smart Grids.

3. MQTT ActorSystem: An Actor System responsible for translating the MQTT protocol to a homogeneous protocol. It is also responsible for allocating VOs by deploying the corresponding actors.
4. Proxy Layer: A publish/subscribe system and a database.
5. WS ActorSystem: An Actor System responsible for translating the WS protocol to a homogeneous protocol. It is also responsible for allocating VOs by deploying the corresponding actors.
6. WS Client: For example, a browser that supports the WS protocol.

Entities perform actions by sending asynchronous messages. Actions between entities are represented in Figure 2.7 by directed arrows and the associated label gives information

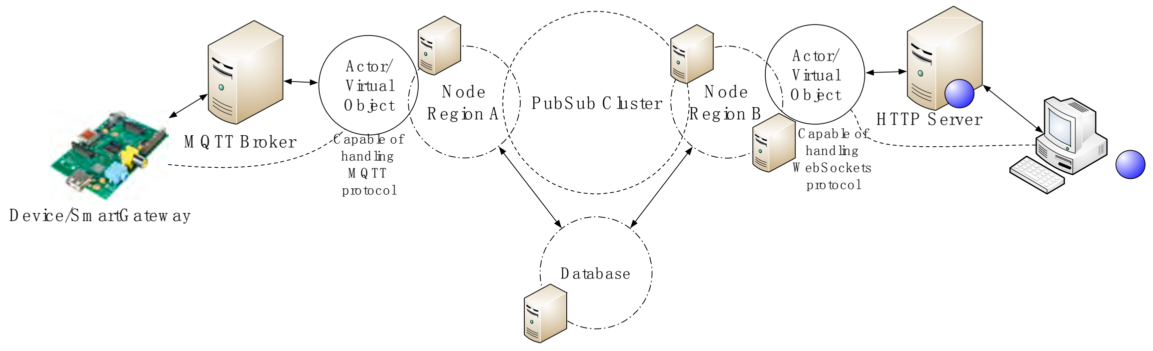


Figure 2.6: System design and architecture.

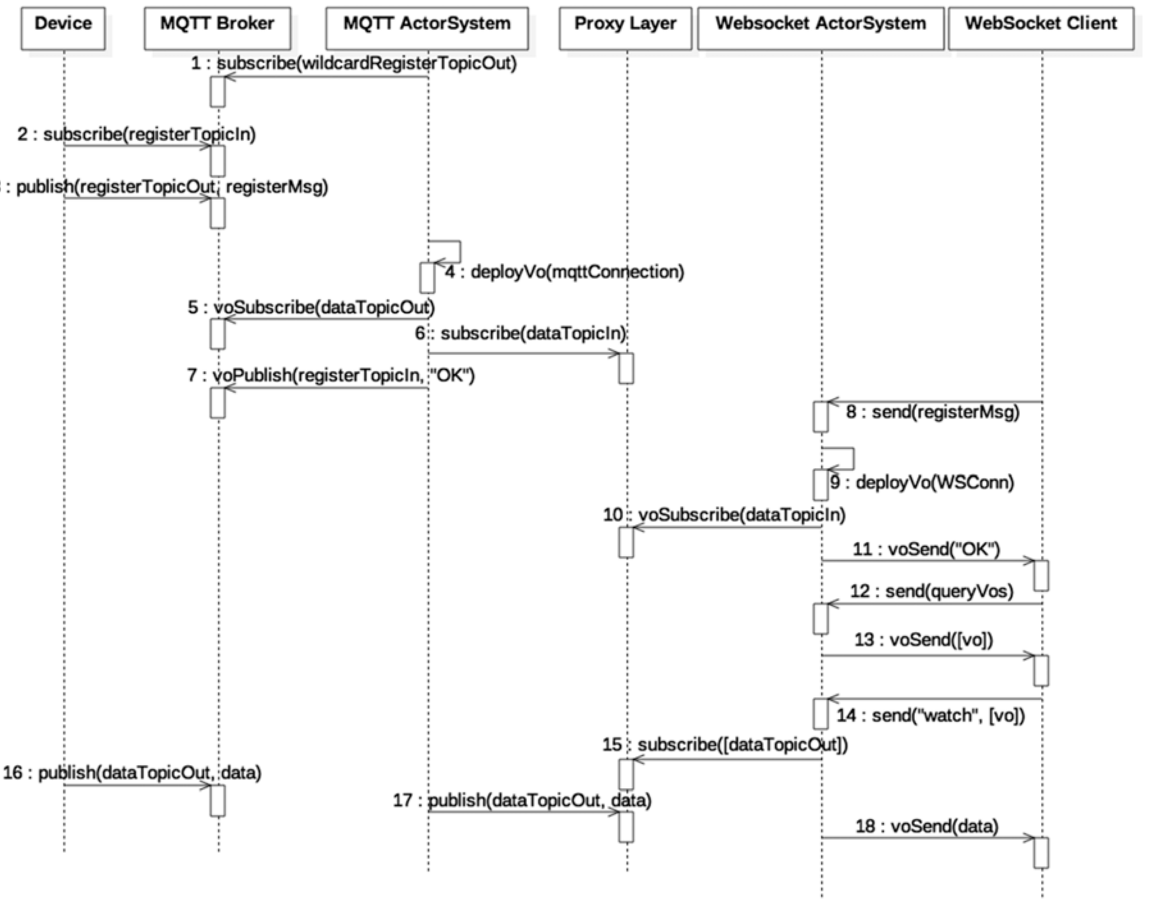

Device
MQTT Broker
MQTT ActorSystem
Proxy Layer
Websocket ActorSystem
WebSocket Client
1 : subscribe(wildcardRegisterTopicOut)
2 : subscribe(registerTopicIn)
3 : publish(registerTopicOut, registerMsg)
4 : deployVo(mqttConnection)
5 : voSubscribe(dataTopicOut)
6 : subscribe(dataTopicIn)
7 : voPublish(registerTopicIn, "OK")
8 : send(registerMsg)
9 : deployVo(WSConn)
10 : voSubscribe(dataTopicIn)
11 : voSend("OK")
12 : send(queryVos)
13 : voSend([vo])
14 : send("watch", [vo])
15 : subscribe([dataTopicOut])
16 : publish(dataTopicOut, data)
17 : publish(dataTopicOut, data)
18 : voSend(data)

Figure 2.7: Entity interaction diagram.

about the action performed and the involved and exchanged information in those actions. The information shown in parentheses can be seen as variables whose name gives semantic hints about their content. The schematic is explained as follows:

1. In step 1, an actor named MQTT Master that has been initially allocated in the MQTT ActorSystem subscribes to a wildcard topic such as *+/config/out* to receive incoming information about new devices. The plus sign in the wildcard topic means that the subscription is to all topics that match a *string* prepended by */config/out*.

2. A Device which communicates via MQTT initiates a registration process with the MQTT ActorSystem via the MQTT broker. As shown in steps 2 to 7, the Device subscribes to a topic dedicated to configuration purposes, which includes the registration process, and sends a registration request to the MQTT Master. If the registration process is successful, which is the flow shown, the MQTT Master deploys a dedicated actor responsible for handling the messages sent over the configuration topic and the data topic (*dataTopicOut*). The actor also subscribes to the Proxy Layer to receive messages directed to the device (step 6). Finally, the actor responds with an ACK ("OK" message) to the Device communicating that the registration process was successful. The VO that represents the Device and associated information is registered in the system's database (action not shown in Figure 2.7). Note that the actions prepended with the prefix "vo" (VO), (e.g., voSubscribe) correspond to actions performed by a dedicated actor, thus the voSubscribe action is performed by the actor deployed by MQTT Master in step 4. Also, the topic or message used in a "vo" action contains the ID of the VO and therefore messages and topics do not collide between Virtual Objects.

3. The WS Client sends a message to the WS ActorSystem, concretely to an actor named WS Master, to initiate a registration process that starts at step 8 and ends at step 10. If successful, the WS Master deploys a new actor which will be responsible for handling the messages sent over the previously established WS connection (wsConn). The subscription process ends at step 11 when the dedicated actor sends and ACK ("OK" message) meaning that the registration process was successful. As with the MQTT registration process, the associated information is saved into the database.

4. Steps 12 and 13, exemplify a WS Client query requesting for available VOs the dedicated actor response with appropriate data. The process of the actor querying the database is not shown in Figure 2.7. Then, in step 14, the WS Client sends a message indicating what VOs to watch or subscribe to. Once the dedicated actor receives this information, it subscribes to the corresponding VO topics to receive updates of them in step 15. Variables in square brackets "[]" indicate a list of those variables.

5. Finally, steps 16 to 18 show how messages produced by the Device are consumed by the WS Client.

Note that the process explained can be reversed. Once the devices (Device and WS Client) connect to the architecture, both can perform the same actions. In fact, at the software level, the functionalities of both VOs are encoded equally (classes in Object Oriented Programming).

2.5.3 Emulation Setting

To implement the model represented in Figure 2.6:

1. Mosquitto has been used as the MQTT server [83].

2. Akka [81] has been used through the Scala API as an implementation of the Actor Model in Regions A and B. Note that the Erlang Virtual Machine allows actors to be hot deployed and that the characteristics of the Java Virtual Machine make this functionality harder to implement. Nevertheless, the OSGi framework provides the necessary means to accomplish this task, which could be useful for implementing Software Defined Utilities in the same way as used in Software Defined Networks [5, 49]. Both

in Erlang and Akka, an actor does not correspond to a system thread, but instead, a few threads, usually as many as the number of cores, are scheduled to process each actor mailbox when new messages arrive in order to optimize resources.

3. For the Proxy Layer with the publish/subscribe function, a publish/subscribe cluster implementation has been used with Akka Cluster.

4. The Play Framework [84] has been used as WS and HTTP/S server.

5. MongoDB [85] has been used as the database to store available VOs.

In order to emulate the maximum number of network hops and distributed resources to approximate a real scenario, we have used the maximum number of lab resources that are at our disposal. We have used a MacBook Pro to simulate a large number of Devices and two Virtual Machines (VMs). Each VM is allocated in a different physical computer. Table 2.2 shows the specifications for each lab resource and the entities/services (Figure 2.7) allocated in them.

Table 2.2: Resources used in the emulation setting and service allocation.

Resource	**Specifications**	**Services**
VM 1	2 × 2.4 GHz Intel Xeon (4 cores per processor), 4 GB RAM	Mosquitto, MQTT ActorSystem
VM 2	3 × 2.4 GHz Intel Xeon (4 cores per processor), 6 GB RAM	Proxy Layer, HTTP/S/WS Server and ActorSystem
MacBook Pro	2.7 GHz Intel Core i7 (2 cores per processor), 8 GB RAM	MQTT Devices, WS Client

2.6 Experimentation and Results

In this section, we present an analysis of the successfully implemented prototypes that are governed by the techniques analyzed above. Our goal is to analyze the real-time performance of our system and to study how it behaves under different types of load. Concretely, the experiments measure the time interval from the moment the sensor data is sent by the Device or Smart Gateway until it arrives at the WS Client.

The minimum message size includes the message id, the VO id, the payload, the unit of measure and the time the value was sensed. With payload or value equal to 0 bytes, the minimum message size is 132 bytes. The message id and VO id are raw Universally Unique Identifiers (UUIDs) (36 bytes per UUID with hyphens) but an extra compression could be applied by encoding them to Base64 (22 Bytes) or Base85 (20 bytes), although the latter encoding is not URL safe. The timestamp is represented as a 13-byte string. Finally, the unit of measure is represented as 1 byte and the remaining bytes account for JavaScript Object Notation (JSON)-encoding characters. Extra compression can be achieved using binary formats [86] but it was not within the scope of the project.

The legend in each experiment shows the mean, minimum, and maximum difference from the sending of a message from the Device to the reception of the same message by the WS Client. From now on, we will refer to this difference as t (*timedelta*). Therefore, we define $\partial t = rt - st$ for a message, where rt is the reception time and st is the sending time. We also define:

$$mean = mean\,(\partial t_i) \tag{2.1}$$

$$max = max\,(\partial t_i) \tag{2.2}$$

$$min = min\,(\partial t_i) \tag{2.3}$$

2.6.1 Experiment 1

The Dataport-PecanStreet project [87] provides a huge dataset of smart meter data. We have extracted the data from January 2015. The dataset contains data from 633 households and records of energy consumption and production at 15 min interval. Each record contains information about a maximum of 66 smart appliances per household but each household has no more than 15 smart appliances generating smart meter readings. A total of 124,331,328 individual smart meter records are available and 28,257,120 contain energy usage readings. For this reason, the payload size has been fixed to 4 bytes to fit a smart meter reading in each message and load tests are performed until total transmitted messages per burst resemble a scenario in which each household owns 15 smart appliances. As the experiments will be performed in an emulation setting and the goal is to perform stress tests, the interval of 15 min per smart meter reading has not been taken into account. We also perform 3 bursts where each device sends a message. The interval between bursts is of 1 s. Bursts are performed in order to analyze how actor mailboxes behave. Table 2.3 summarizes the description of the experiment. Figure 2.8 shows the results.

Table 2.3: Experiment 1 description setting.

Variable	**Value**
Devices	{1k, 2k, 3k, 4k, 5k, 6k, 7k, 9k}
Payload Size (bytes)	{4}
Burst Interval (ms)	{1000, 5000}
Bursts	{3}

As shown in Figure 2.8, the mailboxes of the actors tend to saturate at each sample, but the timedelta of the first message is less than 500 ms. A high percentage of messages are affected by the queue size, as the mean shape is closer to the maximum.

Increasing the burst interval to 5 s, smooths the timedelta mean as shown in Figure 2.9. This means that a higher percentage of timedeltas compared with Figure 2.8 is closer to the minimum. It also shows that the maximum is 1.3 s approximately, compared to 5 s in Figure 2.8. The results obtained conform to time needs in non-critical SG applications [8, 49, 88].

2.6.2 Experiment 2

The experiment description setting 2 is similar to the experiment description setting 1 but instead of increasing the number of simulated devices, it emulates the aggregation of messages in the payload field. The experiment shows 1000 devices sending messages with payloads that aggregate messages, emulating that each device is a Smart Gateway that aggregates messages. Table 2.4 summarizes the experiment description.

The results of the experiment are shown in Figure 2.10. Note that the x-axis shows aggregated messages instead of sent messages. For example, the message size of a message that aggregates messages is calculated as:

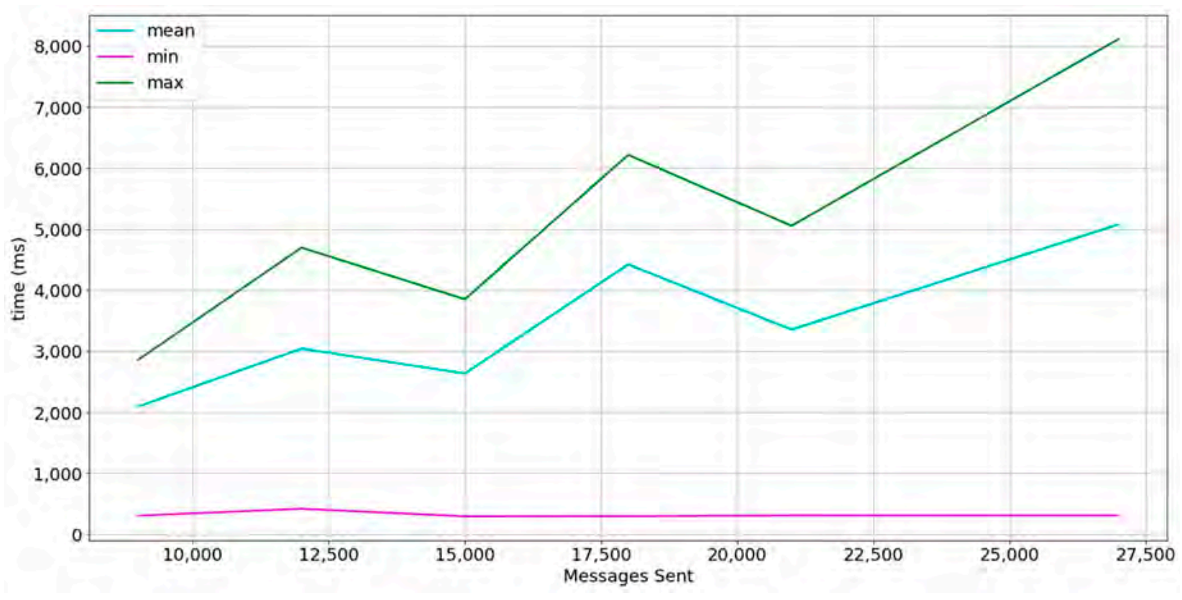


Figure 2.8: Results in experiment description setting 1 at a 1-s interval between bursts. The line in cyan represents the mean timedelta, the line in magenta represents the minimum timedelta, and the line in green represents the maximum timedelta.

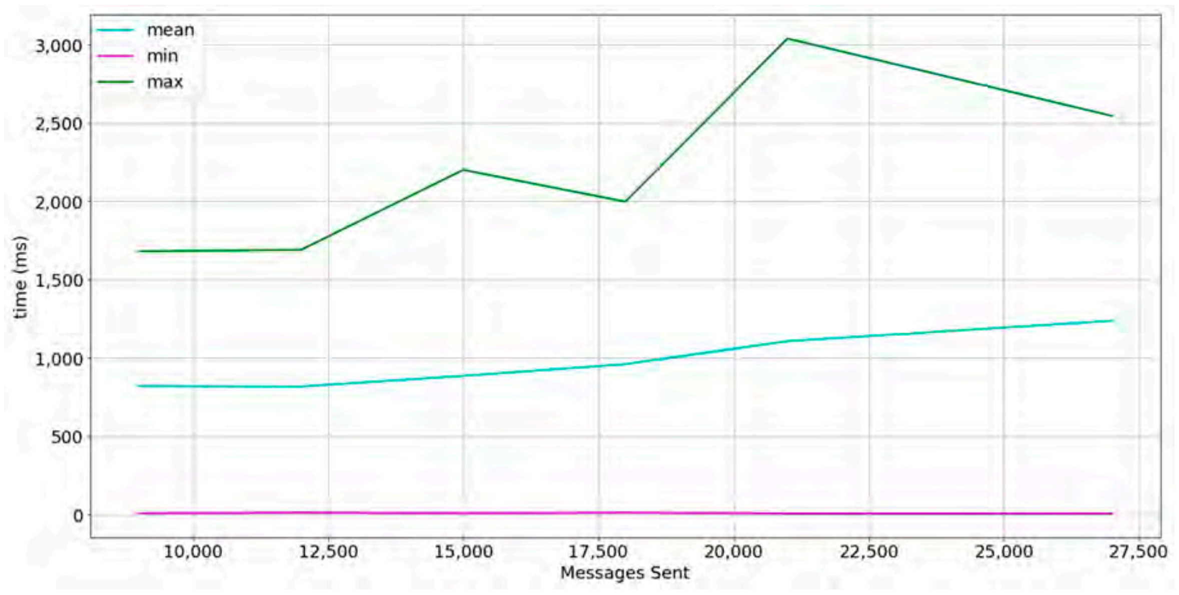


Figure 2.9: Results in experiment description setting 1 at a 5-s interval between bursts. The line in cyan represents the mean timedelta, the line in magenta represents the minimum timedelta, and the green line represents the maximum timedelta.

$$\begin{split} &\mathit{messageSize}\,(\mathit{aggregatedMessages}) \\ &= \mathit{baseSize} + \mathit{aggregatedMessages} * (\mathit{baseSize} + \mathit{payload}) \end{split} \tag{2.4}$$

where $\mathit{payload} = 4$ bytes, $\mathit{baseSize} = 132$ bytes. If a Smart Gateway sends two aggregated messages, the total payload is calculated as:

Table 2.4: Experiment 2 description.

Variable	Value
Devices	{1000}
Payload Size (bytes)	{272, 408, 544, 680, 816, 952, 1224}
Burst Interval (ms)	{1000, 5000}
Bursts	{3}

$$messageSize\,(2) = 132 + 2 * (132 + 4) = 132 + 272 = 404 \text{ bytes.}$$

The difference of the mean, minimum, and maximum parameters between samples at 1 s interval and 5 s interval are not abrupt, which means that reducing the interval between bursts from 5 s to 1 s does not have a great impact on performance, even with messages that have a payload equal to 1224 bytes. We conclude that our system is capable of delivering 1000 messages aggregating nine messages in their payload within a $\partial t < 1.42$ s at an interval of 1 s between 1000 messages, which is equal to 9000 aggregated messages delivered in less than 1.5 s and at a mean rate of 600 ms. The experiments start when 2 messages are aggregated, since the aggregation of 1 message is nonsense.

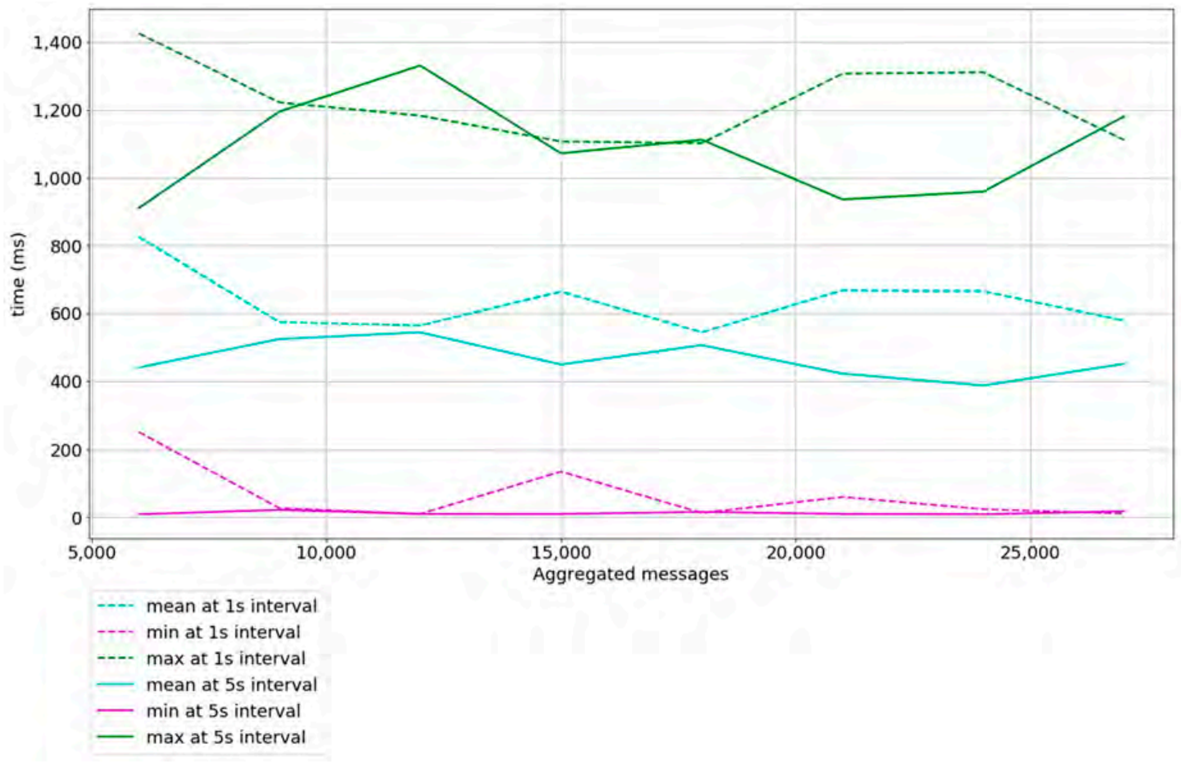


Figure 2.10: Results in experiment description setting 2 at 1 and 5-s interval between bursts. Dashed lines represent experiments at a 1-s interval between bursts and continuous lines represent experiments at a 5-s interval between bursts. Cyan lines represent the mean timedelta, magenta lines represent the minimum timedelta, and lines in green represent the maximum timedelta.

2.6.3 Discussion between Experiments 1 and 2

In Figure 2.11 a comparison between experiments 1 and 2 with the description setting where the time between bursts is 5 s is shown. The mean timedelta obtained in both experiments shows that a better performance can be achieved by aggregating messages (experiment 2). The slope of the mean function in experiment 1 is approximately 0 and the slope of the mean function in experiment 2 is positive. No messages were lost in either experiment 1 nor 2. While results in experiment 1 conform to time needs of non-critical SG applications, results in experiment 2 conform to some of time needs in critical SG applications [8, 88, 89]. We also conclude that message aggregation is preferred when possible, as it reduces the overheads of the entire system.

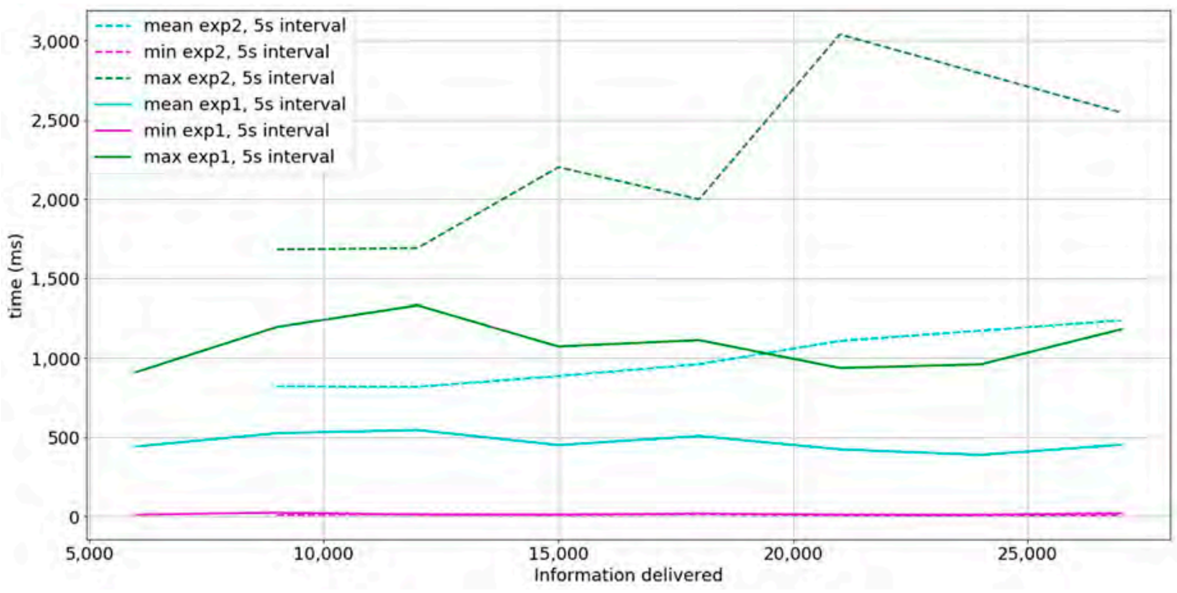


Figure 2.11: Comparison between experiment 1 and experiment 2 at a 5-s interval between bursts. x-axis equals to the total information delivered and refers to both individual messages at experiment 1 and aggregated messages at experiment 2. Dashed lines represent measurements in experiment 2 at a 5-s interval between bursts. Continuous lines represent measurements in experiment 1 at a 5-s interval. Cyan lines indicate the mean timedelta, magenta lines indicate the minimum timedelta, and green lines represent the maximum timedelta.

2.6.4 Web of Energy, Proof of Concept

To conclude the experiments, we directed our efforts to developing a web portal to visualize the data available through the HTTP/S and the WS protocol, which prototypes the WoE. 1000 devices have been simulated, emitting synthetic smart meter data every 5 s. These sensors are distributed throughout the city of Barcelona. Figure 2.12 shows one of the functionalities of the system, a heat map updated every 5 s in near real-time. However, Figure 2.12 only shows a few of those sensors because the web browser performance is rapidly decreased as more and more objects need to be painted. A possible solution could be to aggregate smart meter readings and show the aggregation value in the heat map.

Figure 2.13 shows the result of consulting the value of one of these sensors by clicking

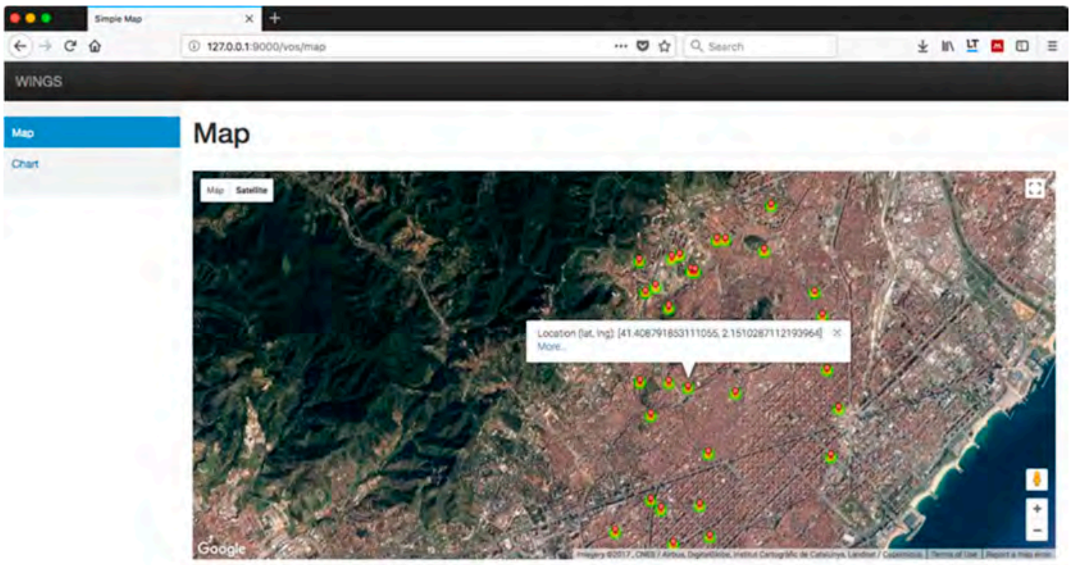


Figure 2.12: Graphic representation of sensor location.

in the "More . . . " link shown in Figure 2.12. The chart label corresponds to the UUID of the sensor. The values in the vertical axis correspond to the measurements obtained by the sensor. This simulated sensor reads voltage, so a "V" inside the parentheses is displayed. The values in the horizontal axis show the time at which the voltage reading is obtained, every 5 s. The data shown updates in near real-time.

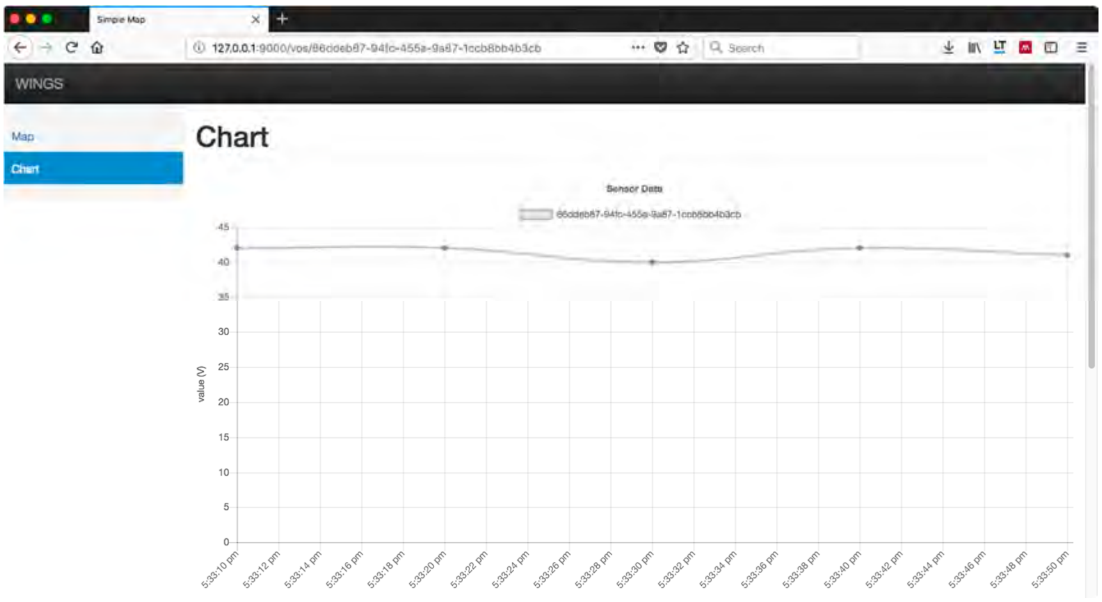


Figure 2.13: Chart showing the synthetic values emitted by a sensor every 5 s.

The application is designed with the ability to interact with each of these sensors, although in this initial version only the query of values is allowed. All queries are executed in near real-time and without delays, despite the large volume of data. This confirms that the design of the prototype is consistent and allows the extension of the rest of the WoE.

2.6.5 Discussion and Conclusions

A WoE prototype based on the Actor Model has been implemented. Our goals were: (i) to address some of the key issues related to ICT in SG systems while demonstrating the benefits of the Actor Model in such ICT-SG use cases and (ii) to implement a prototype for further research. The SG can be seen as a specific IoT application, with its own constraints and requirements. Specifically, we have addressed the heterogeneity of protocols by means of the WoT and some of the near real-time requirements of SG ICT systems. We have created VOs, a common abstraction in IoT scenarios, by means of the Actor Model. Each VO is represented by at least one actor, a lightweight reactive agent. Heterogeneity is addressed in the prototype system by the capacity of each implemented actor to translate heterogeneous protocols (MQTT, for example) to web protocols (HTTP/S or WS). Performance tests show that the initial model meets some of the time needs of time-critical SG-ICT applications when VOs aggregate several smart meters (messages that aggregate messages). In such cases, the timedelta mean slope remains neutral even at bursts of 9000 aggregated messages at a 1-s interval between bursts. Tests involving more devices have to be carried out in order to explore the scalability possibilities of the system. Note that system is flexible in that the actors are allocated/deployed when needed. Experiments also show that the system is suitable for non-time-critical SG traffic when messages cannot be aggregated before reaching the prototype system. Finally, a simple web-based interface to visualize sensor location and sensor readings in near real-time has been implemented as the WoT interface to the SG, WoE.

The results encourage us to keep investigating in this direction. We acknowledge that our system and experiments do not cover all of the scalability and flexibility requirements in SG ICT systems but we have presented the benefits of the Actor Model for such systems and developed an initial prototype for further research and development.

2.7 Conclusions

While the overestimations on the initial prediction of the volume of internet connections were acknowledged and duly rectified, the fact that our society is becoming increasingly connected to the internet is undeniable. This acceleration has been promoted thanks to the advances of both silicon, which makes it possible to embed increasingly smaller computing units in everyday devices, and advances in low-power wireless protocols. At the same time, internet and web applications targeted at controlling and managing internet-connected devices are being continually improved. We are being encouraged to integrate internet-connected devices into our lives, where social network services serve as the easy-to-use interface between people and devices.

It is obvious that the WoT is continually extending to different application domains. Nevertheless, in order to continue this trend, it is essential that the software applications involved are built following guidelines that guarantee their performance, accessibility, and high availability. In this sense, an overall development model agreed upon and approved by different authors is becoming increasingly specific.

In this paper, the term WoE is used to refer to the critical features that an architecture of the WoT must fulfill to be applied to the domain of SGs. We highlight the requirements related to the implementation of smart functions, such as the distribution, resilience and self-healing of the system as well as the difficulties of renewing the traditional energy generation and management systems, both in terms of device and the software, which are generally interrelated. Given those challenges, the use of the Actor Model is proposed to overcome them. This paradigm is designed specifically to perform the modeling of concurrent and distributed systems, thus directly improving the inherent characteristics of distributed systems. In this sense, a proposal of middleware architecture using this paradigm for the creation of a USN has been presented in this paper.

2.8 Future Work

The future work to be done is focused on two different aspects. On the one hand, we want to expand the WoE application, providing it with more functionalities and allowing us to interact with the devices. We also want to integrate different types of sensors, with unequal capacities and performances.

On the other hand, characteristics of scalability, discoverability, and interoperability are only partially tackled by the WoT as it serves as the abstraction layer that comprises technologies that enable homogeneous communication among devices by means of HTTP/S and semantic web ontologies. However, a global IoT or WoT, the latter being an IoT empowered by web technologies, still has scalability and discoverability issues. As the number of Internet-connected devices grows, the lack of global scalability and discoverability functionalities will become more evident. In this regard, a novel IoT paradigm, the SIoT is gaining momentum.

In this novel paradigm, communication between SVOs becomes mandatory and the amount of messages exchanged between devices is expected to increase. Several works aim to provide example scenarios where SIoT can be leveraged [90–92], to name a few. In this sense, we highlight [92], where the authors use thing-to-thing social relationships to encourage communication to optimize energy usage by the Heat, Ventilation, Air Conditioning (HVAC) system in their laboratory. Results show that the comfort-aware HVAC system ensures users' thermal comfort, while at the same time reducing energy costs with respect to static and traditional methods. An ongoing challenge for us is the study of the implications of reactive, asynchronous communication together with computation distribution properties of the Actor Model in a SIoT system, in terms of performance and how these abstractions facilitate the development of this kind of applications. Further work on this subject encompasses the creation of proper simulation tools or modules to facilitate the simulation of SIoT enabled nodes and networks at multiple network layers (e.g., applications installed on top of constrained devices and fog or cloud applications).

3 Ontology-Defined Middleware for Internet of Things Architectures

3.1 Preamble

3.1.1 Abstract

The IoT scenario is composed of an amalgamation of physical devices. Those physical devices are heterogeneous in their nature both in terms of communication protocols and in data exchange formats. The WoT emerged as a homogenization layer that uses well-established web technologies and semantic web technologies to exchange data. Therefore, the WoT enables such physical devices to the web, they become Web Things. Given such a massive number of services and processes that the IoT/WoT enables, it has become almost mandatory to describe their properties and characteristics. Several web ontologies and description frameworks are devoted to that purpose. Ontologies such as SOSA/SSN or OWL-S describe the Web Things and their procedures to sense or actuate. For example, OWL-S complements SOSA/SSN in describing the procedures used for sensing/actuating. It is, however, not its scope to be specific enough to enable a computer program to interpret and execute the defined flow of control. In this work, it is our goal to investigate how we can model those procedures using web ontologies in a manner that allows us to directly deploy the procedure implementation. A prototype implementation of the results of our research is implemented along with an analysis of several use cases to show the generality of our proposal.

3.1.2 Author Contributions

Víctor Caballero and David Vernet conceived the idea and decided the technologies involved in the prototype; Víctor Caballero and Sergi Valbuena implemented the prototype; Víctor Caballero contributed with insightful knowledge about the technologies involved; Agustín Zaballos and David Vernet supervised the execution of the work and contributed with insightful discussions and ideas to improve the overall quality of the work; Agustín Zaballos is also the link with the European projects Advanced Training in Health Innovation Knowledge Alliance (ATHIKA) and Strategic Programme for Innovation and Technology Transfer (SPRINT). All authors contributed equally to writing the paper.

3.1.3 Funding

This work received funding from the "Agència de Gestió d'Ajuts Universitaris i de Recerca (AGAUR)" of "Generalitat de Catalunya" (grant identification "2017 SGR 977") and Optimized HF transmission for NVIS links for remote sensors in Antarctica (ENVISERA) project, funded by Spanish Ministerio de Economía y Competitividad (Plan Estatal de Investigación Científica y Técnica y de Innovación 2013–2016, grant reference CTM2015-68902-R). Part of this research has also been funded by the "SUR" of the "DEC" of the "Generalitat de

Catalunya" and by the European Social Funds (grant reference "2018FI_B1 00171"). In addition, we would like to thank La Salle-Universitat Ramon Llull for their support.

3.1.4 Acknowledgements

Icons used in Figure 3.3 made by Freepick and Gregor Cresnar at `www.flaticon.com`.

3.1.5 Conflicts of Interest

The authors declare no conflict of interest.

3.2 Introduction

The recent growth of the Internet has fostered the interaction between many heterogeneous technologies. The term IoT [1] has been coined as the umbrella term that refers to this new reality. The heterogeneity of the IoT encompasses both hardware and software. At the hardware level, an amalgamation of devices from multiple vendors take part in the IoT. At the software level, those devices use multiple Internet protocols and data exchange formats. Device manufacturers choose what seems to fit their needs in describing and operating their devices. Hence, the current IoT scenario is filled with devices/Things that are not fully interoperable. Even stronger lack of interoperability can be found in vertical domain specific silos of IoT. In order to facilitate the communication between devices and the integration of them into systems and systems of systems, researchers have conducted work that aims at providing a common communication interface. The WoT [24] provides a common interface by translating the heterogeneity into the homogeneity of the Web 3.0. To achieve a common language, the WoT refers to HTTP/S as the standard communication protocol and to Semantic Web [30] as the common information exchange structure.

An example that takes advantage of both IoT and WoT technologies is as follows. In the context of a Smart Home, there are several electrical appliances which consume or produce energy. On the one hand, home appliances that consume energy can be an air conditioner, vacuum cleaner, washing machine, clothes dryer, dishwasher, oven, stove, mobile charger, lights and so on. On the other hand, devices that produce energy usually take advantage of RESs like photo-voltaic panels. In order to take advantage of both production and consumption, those devices need to exchange information between themselves. A course of action would be, for example, to autonomously turn on the clothes dryer during the day when there is enough sunlight to produce enough energy for it to function. It is very unlikely, though, that they all use the same communication protocol and data exchange format. Some will use wireless protocols like Bluetooth, ZigBee, Z-Wave or 6LoWPAN, or wired ones like Ethernet, RS-232, or USB, to name a few. The WoT integration pattern proposes to use gateways as translators between those protocols (some are proprietary) and HTTP/S, along with data exchange format translation, which occurs from a variety of formats such as EXI, JSONB or unstructured data to a semantically annotated Web 3.0 standards, such as JSON or XML. Scenarios like the one presented are very likely to occur. Each scenario, however, will comprehend different types of devices that use different protocols and different data exchange formats. The simple scenario of pairing a light bulb from brand A with a switch from brand B can be different in each household. In such a simple use case, the task of describing each specific scenario can be tedious. Therefore, there is a need to describe the execution flow or behaviour of those devices in an abstract manner, thus the description can be used for multiple and diverse devices with the same capabilities.

3.2.1 Motivation

A large number of heterogeneous devices share the same capabilities and concrete services, with concrete, shared behaviours can be requested from them. Normally, the description of the services and the execution process that they follow are specified and implemented in multiple manners, which pose an obstacle for the shareability and interpretation of their execution flow. A great effort in providing a common abstraction and interoperability layer is being done in the framework of the IoT/WoT. The World Wide Web Consortium (W3C), for example, is leading an effort in order to provide a common communication and abstraction layer. There is also a growing tendency to describe services using Semantic Web standards such as OWL-S [93], W3C Web of Things (WoT) Thing Description (TD) [94] or the oneM2M Base Ontology [95]. Our vision is that we can share and reuse a concrete flow of execution among multiple and heterogeneous IoT deployments using existing standards. We can leverage Semantic Web Ontologies to model, and execute by means of a specialized engine, the behaviour of the devices that lay in the lower layer of the IoT, by providing a common template (abstraction) that defines such behaviour. This template can be reused by as many Things and services as needed in a variety of heterogeneous deployments.

3.2.2 Contributions

This work aims to address the challenge of describing and executing the same behaviour in multiple and heterogeneous IoT scenarios. In heterogeneous IoT scenarios, where devices from different manufacturers are aimed at providing the same service, it is often found that their heterogeneity prevents developers from reusing the same solution for each device. We address that challenge by using Finite-State Machines (FSMs) to model their behaviour, translating the FSM into in abstract description framework, namely RDF [62], and developing an engine capable of interpreting and executing the RDF model. The main contributions of this work are:

1. Abstraction of flow of control or behaviour over the heterogeneity of the IoT using existing standards such as RDF, SPARQL Protocol and RDF Query Language (SPARQL), FSM and HTTP/S. This abstraction brings reusability.
2. A prototype implementation of an engine and web interface that allows direct deployment, interpretation and execution of a concrete instance obtained by materializing the abstraction.
3. Analysis of several use cases that demonstrates the generality of the abstraction.

This article is organized as follows. Section 3.3 gives a brief overview of IoT/WoT standards relevant to our goal and analyses related work on control of flow/behaviour modelling using Semantic Web technologies. Section 3.4 gives an overview of the architecture that supports our approach. Section 3.5 introduces and models the use case that is used to explain our solution and Section 3.6 enhances the model using ontologies. Section 3.7 explains how we use the template (abstraction) and execute the behaviour described in it. Section 3.8 analyses several use cases in relation to our approach and finally Section 3.9 concludes the paper and presents future work.

3.3 State of the Art

Given the massive number of services and processes that IoT enables, it has become almost mandatory to describe such services and processes to enhance interoperability, allowing them to be automatically discovered and called by multiple, heterogeneous systems [96]. Ontologies such as OWL-S [93] and low-level specifications such as the TD [94] or the oneM2M Base Ontology [95] can be used together to describe IoT/WoT systems, fostering interoperability. OWL-S helps software agents to discover the web service that will fulfil a specific need. Once discovered, OWL-S provides the necessary language constructs to describe how to invoke the service. It allows describing inputs and outputs. Thanks to the high-level description of the service, it is possible to compose multiple services to perform more complex tasks. In OWL-S, service description is organized into four areas: the *process model*, the *profile*, the *grounding* and the *service*. Specifically, the process model describes how a service performs its tasks. It includes information about inputs, outputs, preconditions and results. Similarly, the oneM2M Global Initiative [97] defines a standard for machine-to-machine communication interoperability at the semantic level, the one M2M Base Ontology, which is a high-level ontology designed to foster interoperability among other ontologies using equivalences and alignments. The TD "is a central building block in the W3C Web of Things (WoT) and can be considered as the entry point of a Thing [...]. The TD consists of semantic metadata for the Thing itself, an interaction model based on WoT's Properties, Actions, and Events paradigm, a semantic schema to make data models machine-understandable, and features for Web Linking to express relations among Things". Both oneM2M Base Ontology and the ontology defined in TD strive for interoperability among multiple IoT applications and platforms, each one covering a large set of use cases, so there is also a work in progress to align the oneM2M ontology with the TD ontology [98]. Orthogonal to these ontologies, the SOSA/SSN ontology is an ontology for describing sensors and their observations. Among other concepts, it defines the class sosa:Procedure, which "is a reusable workflow, protocol, plan, algorithm, or computational method that can be used, among others, to specify how an observation activity has been made [...], (the sosa:Procedure) can be either a record of how the actuation has been performed or a description of how to interact with an actuator (i.e., the recipe for performing actuations)" [99]. How much detail is provided to define such procedures is beyond the scope of SOSA/SSN. It is our vision that detailed and deployable procedure descriptions could fit into and be orthogonal to the models just presented. This work aims at modelling the flow of control for a given service. Therefore, our focus is on modelling the procedure/behaviour of Things using ontologies in a manner that allows us to directly deploy the behaviour implementation.

Work aimed at modelling the behaviour of Things using FSMs and OWL exists in the literature. On [100], they aim to represent Unified Modeling Language (UML) FSMs using OWL, performing an almost one-to-one translation between UML concepts and OWL classes. Although their mapping from UML to OWL allows for a more machine-readable information structure, its complexity makes it unpractical to use for our objective. The work done in [101] presents a simpler FSM model. UML is used to specify platform independent navigation guides in web applications. They use OWL to describe a model for FSMs which serves as a meta-model for semantic web descriptions of the navigation guides on the Semantic Web. There also exists some scientific literature devoted not only to create a model to express the behaviour of a service but also to interpret and execute the behaviour. In [102], the aim is to develop an FSM that a special server can read and translate to executable entities. These executable entities are executed later by robots. They successfully satisfy their objective, their OWL FSM is domain specific and it includes properties that solve the complexity of their use case but make the FSM too complex for our goal. The goal in [103] is to model an FSM that can be easily translated into Programmable Logic Controller (PLC) code. This model is domain-agnostic as it is not directly merged with the PLC domain. Works

[102, 103] show how a semantic FSM can be translated and executed by machines, although their application field is industrial and not on the WoT. In our approach, we do not want to express or translate the model in languages other than OWL so it maintains the benefits that OWL gives. Our aim is to build a model that is directly interpreted by different machines.

Our solution aims to be as middleware-agnostic as possible, being a building block for middleware that implement other technical challenges. Therefore, we rely on standard and well-known architectures to contextualize our proposal. We have considered two big standardization and recommendation bodies devoted to promoting interoperability among IoT verticals. An international partnership project for IoT standards, the oneM2M Global Initiative [97] aims to minimize M2M service layer standards by consolidating standardization activities among several member organizations. The W3C Web of Things Working Group [94] aims to develop standards for the Web of Things; their goal is also to reduce fragmentation between IoT silos. For this work, we are interested in the architecture standards and recommendations that oneM2M and W3C WoT Working Group propose. As our objective is to contextualize and provide a prototype implementation for our solution, the W3C WoT Architecture recommendation seems more suitable for that purpose for its simplicity. The architecture is explained in Section 3.4.

3.4 Architecture

The objective of this paper to provide an abstraction mechanism to describe and execute the behaviour of Things in any WoT architecture. This section explains the architecture used in our proposal. There is a common trend on IoT architectures to represent physical devices with virtual representations or agents that enhance their capabilities. Such virtual representations are called Virtual Object (VO) [34]. VOs are usually placed in fog or cloud middleware [34, 39, 104]. The behaviour templates developed in this work are interpreted and executed where those VO reside, thus more resources are available to execute the behaviour that the templates describe. The first layer in the seminal work of the WoT [24] defines an Accessibility Layer that enables each physical device to the web by translating each IoT protocol to HTTP/S and providing a common RESTful (REpresentational State Transfer) API to interact with them. As seen in Figure 3.1, we assume that physical devices are exposed via HTTP/S in order to interact with the VOs. A basic division between layers can be made. The first layer is composed of physical devices, which is the Perception layer. The second layer is responsible for enabling physical devices to the web, the Accessibility Layer. The third layer, the Virtual Object layer, is composed of VOs that interact with the Perception layer via the Accessibility layer. As explained in [24] and in the W3C WoT Architecture [94], Things can become Web Things (web-enabled Things) by three mechanisms. If they have enough computing resources to implement the HTTP/S protocol, they can be directly connected to the web. Otherwise, they can use Gateways (GWs) or cloud/fog infrastructures that translate their proprietary protocols or non-standard data exchange formats into web standards. The template developed in this paper is embodied by a VO. Each VO is given the capability to interact with physical devices through the Accessibility layer. Therefore, actions encoded by an HTTP/S call the Accessibility layer, which in turn informs the physical device to actuate in the real world. Thanks to this action abstraction, relationships can be made between one VO and multiple physical devices. It is only necessary to change the URI actions refer to; this is explained in more detail in Section 3.6.3. The architecture is shown in Figure 3.1 where different settings can be appreciated. The following lists the settings, from left to right:

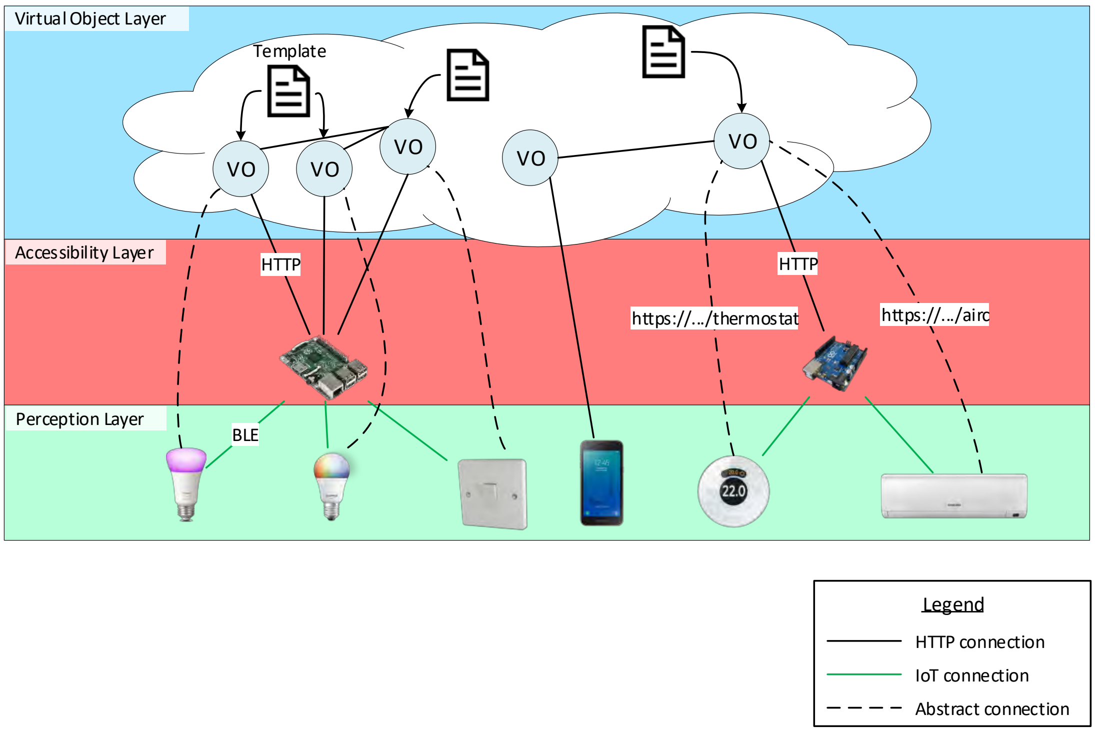


Figure 3.1: Network architecture used in our proposal. Three layers are depicted: (1) Perception layer, where physical devices reside; (2) the translation between IoT protocols and Hyper-Text Transfer Protocol/Secure (HTTP/S) is made at the Accessibility layer and; (3) the Virtual Object layer is comprised of virtual representations of physical devices, each Virtual Object (VO) embodies a template that describes the behaviour for physical devices. Straight lines represent direct connections between nodes, which means that no intermediate protocol or data exchange format is needed (nodes at the Accessibility layer are responsible for performing such actions). Dashed lines represent that the VO at one end of the line handles the physical device at the other end of the line. Finally, black straight lines represent that nodes are connected through HTTP/S and that there exists an HTTP/S Application Programming Interface (API) compatible with nodes at both ends of the line. Green, straight lines represent that nodes are connected through IoT protocols, such as Bluetooth Low Energy (BLE).

1. The first setting is composed of two smart light bulbs from different brands (they use different communication protocols and/or data exchange formats) and a light switch at the Perception layer. There is a Raspberry Pi at the Accessibility layer that translates IoT protocols from the physical devices to an HTTP/S API. Finally, three VOs dwell in the VO layer and each one embodies a template. Note that the same template is used for each one of the VOs that communicate with the smart light bulbs, which means that the same template is used to control both of them. The template for the light switch describes the behaviour for controlling (turn on/off) both smart light bulbs. The control is made at the VO layer. More precisely, the switch VO sends commands and receives messages from the two VOs that represent the smart light bulbs.

2. The second setting is composed of a smartphone at the Perception layer that directly connects to a VO at the VO layer through HTTP/S. To lighten the image, the template the VO embodies is not drawn. The interaction is made possible via an app installed at the smartphone.

3. The third setting comprehends a thermostat and an air conditioner machine at the Perception layer. They both connect to an Arduino board at the Accessibility layer, which exposes their functionality as an HTTP/S API. In this scenario, a single VO is responsible for managing the thermostat and the air conditioner. This can be achieved by simply using both URLs, *https://.../thermostat* and *https://.../airc*, as the target for the template actions. Finally, this VO communicates with the VO that represents the smartphone, enabling for controlling the temperature of the room.

The model architecture depicted in Figure 3.1 shows the interaction patterns between physical devices and our proposal (templates) in a WoT architecture. One of the features of interest in our proposal is that it is middleware-agnostic. Therefore, critical challenges such as full-fledged security and privacy are implemented by concrete middleware. For example, the VICINITY project [105] defines a Virtual Neighbourhood and is a platform that links various ecosystems providing interoperability as a service for infrastructures in the IoT. It tackles privacy by allowing the owner of Things to define sharing access rules of devices and services. The communication between VICINITY Nodes occurs between a P2P network based on those rules. This network supports VICINITY Nodes with security services such as end-to-end encryption and data integrity [106]. The IoT ecosystems are connected to VICINITY through the VICINITY Node. Hence, security and privacy outside VICINITY is handled by the manager of the IoT ecosystem. Security in that ecosystem could be an exchange of identifiers and encryption keys between VOs and physical devices or gateways, representing the weakest phase when security could be compromised.

Integration with other middleware involves three main efforts: (1) to implement an engine to interpret the flow of execution defined by the FSM template (which can be reused by middleware developed with the same programming language); (2) to adapt the input and output channels of the FSM engine with an HTTP/S interface provided by the middleware; and to provide an RDF and SPARQL API to the FSM engine.

As shown in Figure 3.2, our approach complements third-party middleware by providing several desired properties. By combining RDF and the model of execution flow that FSMs provide, we enable a programming abstraction. The programming abstraction can be shared among multiple engine implementations in different middleware. The ease of deployment is facilitated by the fact that behaviour and executor are decoupled. Behaviours for web-enabled physical devices can be created ad hoc when needed and deployed, as the necessary software to execute the behaviour already exists. A Triplestore Database (TDB) or RDF API and SPARQL endpoint is also needed for the FSM engine in order to store the template and save the data received during runtime. Guards, conditions and actions use these data (see Section 3.6). As seen in Figure 3.2, our approach relies on HTTP/S in order to enable communication with the Perception Layer. Therefore, it is mandatory for third-party middleware to have an HTTP/S interface. Two pathways are shown in Figure 3.2. The first pathway assumes a sensor (thermostat/thermometer) with an HTTP/S client that pushes data to the middleware. The HTTP/S server in the middleware receives this request and sends it to the input interface of the FSM engine and triggers the FSM logic, which is read from the template instance. The FSM does not return any response with data as it may not be available at that time. However, the middleware is still able to respond to the HTTP/S request. The second pathway is the actuator pathway. There, the actuator is directly addressable as it is an HTTP/S server. Following the process triggered by the sensor, the FSM may generate an action to turn on the actuator. For example, if the temperature is higher than 27 °C, the course of action would be to turn on the air conditioner. This logic and action are written in the FSM template instance. The FSM generates an action which is sent to the HTTP/S client provided by the middleware. An HTTP/S request is sent to the actuator and the actuator may respond with valuable data, which is then used as the input of the FSM.

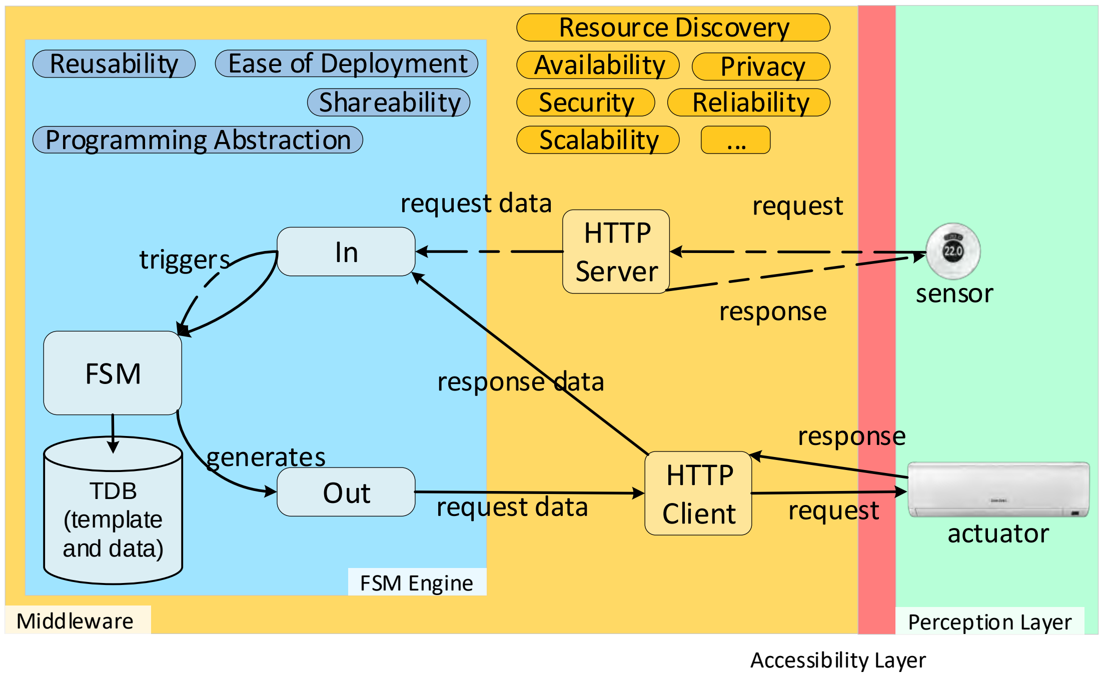


Figure 3.2: Functional blocks for third-party middleware integration. The layers are: FSM Engine and Third-party middleware (Middleware), Accessibility Layer and Perception Layer. The FSM Engine complements the middleware. Only two components are mandatory in the middleware layer to enable our approach. Arrows indicate the flow of information between components in each layer. Dashed lines represent the sensor pathway and straight lines represent the actuator pathway.

3.5 An Example to Model

This section introduces the use case that is modelled throughout the paper. Our objective is to model a template for a VO that allows users to exchange their energy profile information and check if they are compatible. For the sake of simplicity, we define “compatibility” of energy profiles as “similarity” of energy profiles. The use case is framed in the context of the SIoT, which endows devices with the ability to belong to social networks of devices. The Lysis [91] platform is an implementation that enables the SIoT. There, VOs are extended to become social and create relationships; they become SVOs. Several types of relationships can be created. We are interested in Ownership Object Relationship (OOR) and Social Object Relationship (SOR) relationships. OORs are established among heterogeneous objects that belong to the same user. SORs are defined when objects come into contact (close range) because their owners come in touch with each other during their lives. The overall context of the use case we want to model is as follows:

1. A user interested in being part of a prosumer community to benefit from the community and achieve certain goals installs a discovery app on his/her smartphone. The smartphone becomes an SVO [91].
2. The potential PCG user installs apps or allocates apps in the cloud that enable retrieving data from his/her energy-related devices (devices that produce or consume energy), enabling social capabilities for each one (SVO).
3. Devices and DERs that consume and/or produce energy owned by an owner establish an OOR relationship (Figure 3.3, 1). This relationship is then leveraged to obtain energy-related data from DERs that belong to the same owner. These data can be both aggregated and stored or stored individually for each device.
4. The data obtained is used to build an energy user profile during a time period. The profile is then stored as part of the discovery app, and thus enabling the smartphone and other social devices to retrieve such information. A social device is a device that the user (prosumer) usually carries with himself/herself.
5. When the user establishes a social relationship with another potential user (Figure 3.3, 2), both social devices (i.e., smartphones) also establish social relationships (Figure 3.3, 3) and exchange energy profiles and goals. Then, the SVOs determine if they are compatible or not.

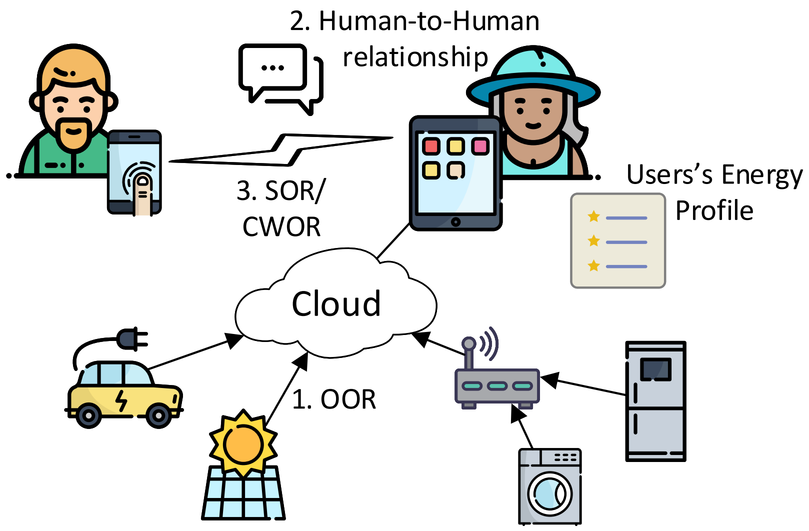


Figure 3.3: PCG formation leveraging SIoT. Firstly, prosumers and appliances establish an Ownership Object Relationship (OOR) via the augmented Social Virtual Object (SVO) (1). When prosumers come close together and establish and interact (2), their devices establish Social Object Relationship (SOR) or Co-work Object Relationship (CWOR) relationships and exchange the energy profiles of the prosumers (3).

Our objective is to model the behaviour expressed in Item 5. The model starts describing the steps that take place when one SVO meets another SVO and ends when the compatibility results between them have been exchanged. The initial peering process is assumed to be done by a service implemented in Lysis [91]. In addition, the energy profile is supposed to exist in the device. The process is described as:

1. Wait for a peer to be available.
2. Send the energy profile to the peer.
3. Wait until the peer's energy profile is received.
4. Check the compatibility between the device's and peer's profiles.
5. Inform of the compatibility result to the peer. The results are exchanged because the compatibility check of each device could be different.
6. Wait until the peer's compatibility result has been received.

To model the behaviour, we have chosen an FSM because it enables us to describe a sequence of actions to be executed and the use of conditions to modify the execution's flow. In order to visualize the FSM, we first model the behaviour using StarUML [107] following UML standards. Figure 3.4 shows the execution flow using states and conditions. The aim of UML is to provide a standard manner to model the design and behaviour of a system; it is made to be comprehended by humans and not by machines. However, our interest is to enable machines to interpret and execute this behaviour. OWL provides explicit meaning, making it easier for machines to automatically process and integrate information. From the options discussed in Section 3.3, we have chosen the FSM ontology developed in [101]. Some properties and classes have been added to integrate it with the web, others have been renamed to follow W3C naming conventions on ontologies. The modified ontology is used to describe the system's behaviour. Note that the FSM is designed in a manner that both VOs (each one interacting with one of the physical devices shown in Figure 3.3) share the same behaviour.

3.6 Behaviour Modelling Using Ontologies

Throughout the rest of the paper, the prefix fsm: is used to shorten its URI. The siot: prefix represents some classes and properties specific to the domain of the use case. Finally, the: prefix is used to reference the file or ontology where the own model is stored.

3.6.1 Skeleton of the FSM

The first step is to define the skeleton of the FSM, it contains the instance of the machine, its states and its transitions.

As shown in Listing 3.1, an FSM is defined by the class fsm:StateMachine and the states as fsm:State. The initial state is also defined as fsm:InitialState, the final state as fsm:FinalState and intermediate states as fsm:SimpleState, these three classes extend from fsm:State. An fsm:StateMachine instance should have their states associated with the property fsm:hasStateMachineElement. Each state can have some exit or entry actions; they are associated with the property fsm:hasEntryAction or fsm:hasExitAction.

The transitions between states are described by the class fsm:Transition. As seen in Listing 3.2, a transition has a source state with the property fsm:hasSourceState and a target state with fsm:hasTargetState. A transition can also contain none or more guards associated with the property fsm:hasGuard.

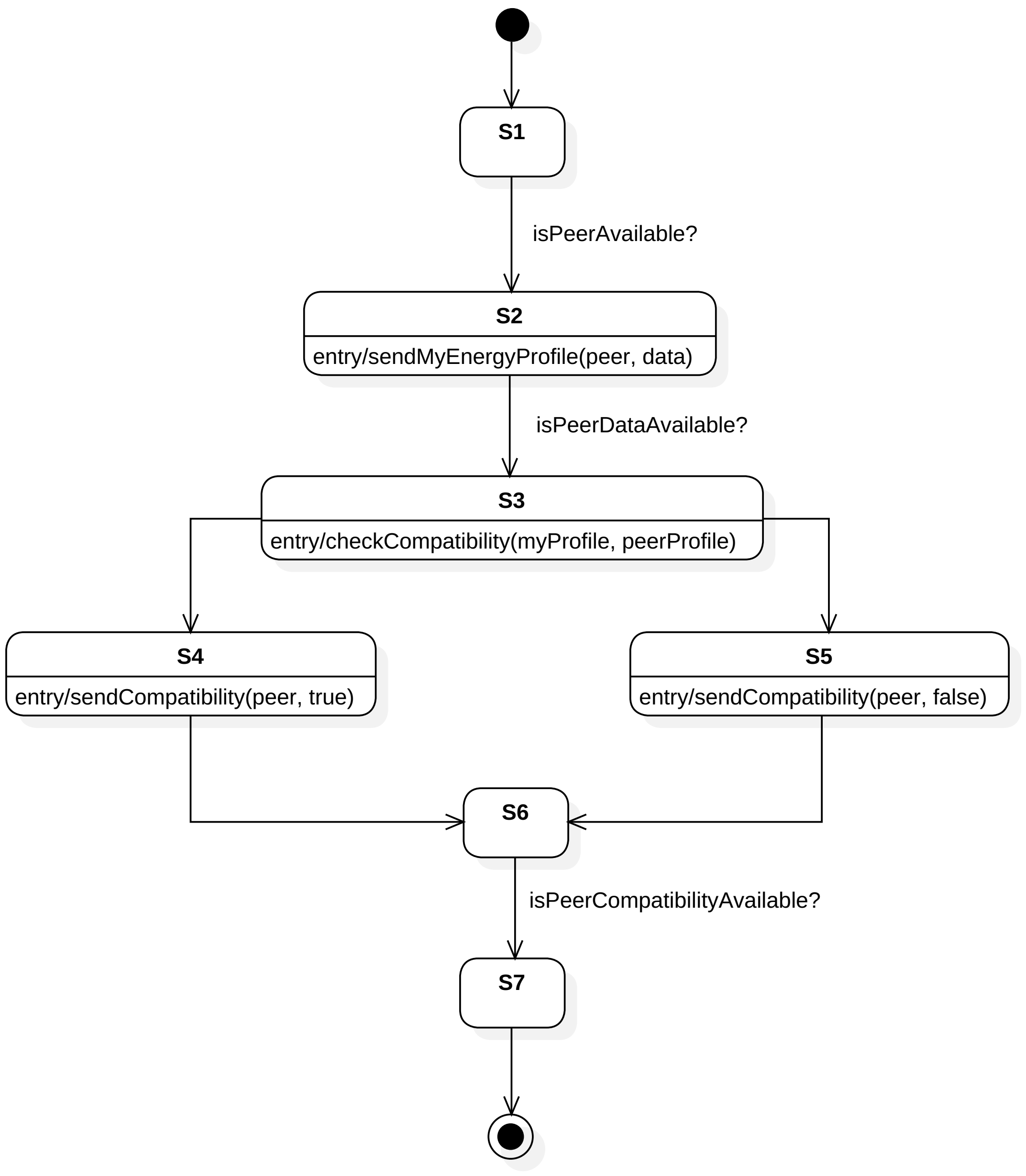

S1
isPeerAvailable?
S2
entry/sendMyEnergyProfile(peer, data)
isPeerDataAvailable?
S3
entry/checkCompatibility(myProfile, peerProfile)
S4
entry/sendCompatibility(peer, true)
S5
entry/sendCompatibility(peer, false)
S6
isPeerCompatibilityAvailable?
S7

Figure 3.4: FSM representation using UML.

3.6.2 Guards and Conditions

The purpose of a guard, defined by the class fsm:Guard, is to provide a method to evaluate if the transition has to be travelled. The guard can contain none or more conditions to express what has to be evaluated. It can also have none or more actions to be executed if the guard evaluates true.

A condition is defined with the class fsm:Condition; it contains a body object marked as fsm:Body that represents the condition's content. This body has a string associated with fsm:hasContent that contains the actual condition represented as a SPARQL query. The body should also include its type associated with fsm:hasBodyType to indicate the type of the body's content. Even though it is possible to use different types, an ASK SPARQL query is needed when used on conditions, as the main focus is to work directly with RDF data and ontologies. The reason to use the type property is that the body object is also used as the

```
### The FSM
:siot_fsm rdf:type owl:NamedIndividual ,
                   fsm:StateMachine ;
          fsm:hasStateMachineElement :S1_WatingForPeerState ,
                                     :S2_InformationExchangeState ,
                                     :S3_CheckCompatibilityState ,
                                     :S4_IsCompatibleState ,
                                     :S5_NotCompatibleState ,
                                     :S6_WaitingForPeerCompatibilityState ,
                                     :S7_CompareCompatibilitiesState .


### States
:S1_WatingForPeerState rdf:type owl:NamedIndividual ,
                                fsm:InitialState ,
                                fsm:State .

:S2_InformationExchangeState rdf:type owl:NamedIndividual ,
                                      fsm:SimpleState ,
                                      fsm:State ;
                             fsm:hasEntryAction :sendMyData .

:S3_CheckCompatibilityState rdf:type owl:NamedIndividual ,
                                     fsm:SimpleState ,
                                     fsm:State ;
                            fsm:hasEntryAction :checkCompatibility .

:S4_IsCompatibleState rdf:type owl:NamedIndividual ,
                               fsm:SimpleState ,
                               fsm:State ;
                      fsm:hasEntryAction :sendImCompatible .

### ...

:S7_CompareCompatibilitiesState rdf:type owl:NamedIndividual ,
                                         fsm:FinalState ,
                                         fsm:State .
```

Listing 3.1: FSM and states.

body and the value for the header "Content-Type" to be sent on HTTP/S requests, where it can take other formats such as RDF or JSON. Some conditions are modelled on Listing 3.4. The procedure to evaluate the transitions is:

- Retrieve all the state's exit transitions. Note that only one of these transitions should be true at the same time, otherwise different executions of the same FSM may lead to different results if the order of the evaluation of transitions changes.
- For each transition, retrieve its guards (Listing 3.3). If it has no guards, then that transition is considered feasible. If any guard is true, then the transition is also feasible, which means that an OR condition is applied between all the guards. The utility of having multiple guards is to have different actions that are executed only under certain cases and not each time the transition is travelled.
- For each guard, retrieve its conditions. Another OR operation is applied between these conditions, and if any of them is true, the guard also evaluates to true. It is important

to evaluate all the guards and to not stop when one of them is true, as all the guards' actions must be executed if their guard is true.

- The body of each condition should be an ASK query. An ASK query is a type of query that contains a pattern of triples, it searches for a set of triples that match the pattern. If any matching set is found, the query returns true, and it is false otherwise. The query is evaluated against the RDF database. The condition evaluates the same as the ASK query.

```
:S1_to_S2_Transition rdf:type owl:NamedIndividual ,
                              fsm:Transition ;
                     fsm:hasSourceState :S1_WatingForPeerState ;
                     fsm:hasTargetState :S2_InformationExchangeState ;
                     fsm:hasTransitionGuard :S1_to_S2_T_Guard .

:S3_to_S4_Transition rdf:type owl:NamedIndividual ,
                              fsm:Transition ;
                     fsm:hasSourceState :S3_CheckCompatibilityState ;
                     fsm:hasTargetState :S4_IsCompatibleState ;
                     fsm:hasTransitionGuard :S3_to_S4_T_Guard .
```

Listing 3.2: Transitions.

```
:S1_to_S2_T_Guard rdf:type owl:NamedIndividual ,
                           fsm:Guard ;
                  fsm:hasGuardCondition :IsPeerAvailable .

:S3_to_S4_T_Guard rdf:type owl:NamedIndividual ,
                           fsm:Guard ;
                  fsm:hasGuardCondition :IsPeerCompatible .
```

Listing 3.3: Guards.

```
:IsPeerCompatible rdf:type owl:NamedIndividual ,
                           fsm:Condition ;
                  fsm:hasContent
        """
            PREFIX siot: <file:///D:/projects/ontologies/siot/siot.owl#>
            PREFIX owl: <http://www.w3.org/2002/07/owl#>
            PREFIX fsm: <file:///D:/projects/ontologies/fsm/fsm.owl#>

            ASK {
                self: siot:hasPeer ?peer .
                self: siot:hasCompatibility ?compatibility .
                ?compatibility siot:withPeer ?peer .
                ?compatibility fsm:hasContent \"true\" .
            }
        """ .
```

Listing 3.4: Conditions.

3.6.3 Actions

An action requests or sends data via HTTP/S to another node in the architecture. The HTTP Vocabulary in RDF 1.0 [108] ontology by W3C is used to define actions. An action is defined with the class fsm:Action and http:Request at the same time. An action has a body, associated with fsm:hasBody, which is sent with the request. The body follows the same structure used in the conditions; it can have different types like RDF (Listing 3.5), JSON and others. Moreover, a timeout can be specified to limit the amount of time that an action waits for the response, it is specified in milliseconds with the property fsm:hasTimeoutInMs. A request method like GET or POST is indicated with the property http:mthd. This property references an object and not a literal, W3C has already defined the standard method objects in [108], these objects are the preferred ones to use. Therefore, as templates are not bound to perform actions using the same URI, a template can perform actions on multiple nodes. This, in conjunction with the FSM model, enables the templates or, more precisely, their instantiation, the VOs, to interact with multiple physical devices and services, such as other VOs. This is the case of the third scenario described in Figure 3.1, where a single VO is able to orchestrate two different physical devices. The VO also communicates with the smartphone described in the second scenario via its VO.

The action has a URI to express where the request is done. This URI can be expressed in different ways; our two approaches are:

- Use an absolute URI with the property *http:absoluteURI* to express the final address as in Listing 3.6. This is used when we know in advance where the request is done and that the address will not change.

- Use a prototype URI with the property fsm:hasPrototypeURI, shown in Listing 3.7. A prototype is an object of the class fsm:Prototype that defines the structure of a URI with one or more placeholders that are replaced during runtime. A schema of the classes and properties involved is presented in Figure 3.5. This is used when we do not know in advance the final address or when it changes frequently. An absolute URI is computed from the prototype each time the action needs to be executed. The replacements of parameters are not done on the ontology; this newly computed URI is built separately in program memory. A prototype defines the URI's structure with a string linked with fsm:hasStructure, for example *http://localhost:9000/[fsm_id]/send_data*, where *[fsm_id]* is the placeholder. The prototype has an fsm:Parameter for each placeholder, linked via the property fsm:hasParameter. The purpose of the parameter is to indicate what placeholder needs to be replaced and to provide a function that generates the value to fill the placeholder. Each parameter has the placeholder itself as a string (like *[fsm_id]*) linked with fsm:hasPlaceholder and a SPARQL query as a string linked with fsm:hasQuery. The query must be a SELECT query that returns only one value. The value returned by the query is used to replace on the URI the placeholder with the same parameter key.

3.7 Execution

This section explains how the proposed ontology-defined behaviours can be executed. We have implemented the template interpreter and embedded it in a web server. We have used Java as the programming language to implement the template interpreter and executor. Apache Jena [109] has been used as the TDB and SPARQL engine. Play Framework 2

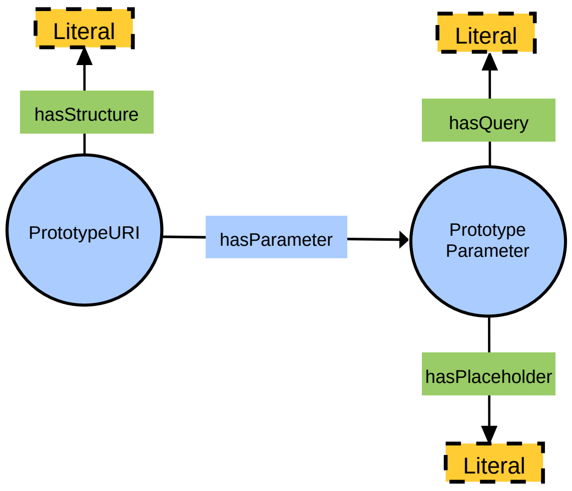


Figure 3.5: Schema of PrototypeURI.

```
:sendImNotCompatibleBody rdf:type owl:NamedIndividual ,
                                  fsm:Body ;
                         fsm:hasBodyType fsm:rdf ;
                         fsm:hasContent
        """
            @prefix siot: <file:///D:/projects/ontologies/siot/siot.owl#> .
            @prefix owl: <http://www.w3.org/2002/07/owl#> .
            @prefix fsm: <file:///D:/projects/ontologies/fsm/fsm.owl#> .

            self: siot:hasCompatibility ?compatibility .
            ?compatibility siot:withPeer self:myPeer .
            ?compatibility fsm:hasContent \"false\" .
        """ .
```

Listing 3.5: Bodies.

```
:checkCompatibility rdf:type owl:NamedIndividual ,
                             fsm:Action ,
                             http:Request ;
                    fsm:hasBody :checkCompatibilityBody ;
                    http:absoluteURI <http://localhost:9000/check_compatibility> ;
                    http:mthd <http://www.w3.org/2011/http-methods#POST> .
```

Listing 3.6: Action with absolute URI.

[84] has been used to implement the web service. Akka [81], an Actor Model toolset, has facilitated the task of deploying the two VOs used in the example and enabled us to endow the engine with a reactive programming model. This means that the engine only "wakes up" when there is a new request for that VO, that is when new data is available. Both Play Framework 2 and Akka ease the task of implementing asynchronous actions. As explained in Section 3.6.3, the amount of time an action waits for a response (HTTP/S response) can be specified. Nevertheless, actions are performed sequentially and asynchronously. If actions A_1 and A_2 with fsm:hasTimeoutInMs 1 s and 5 s respectively are to be performed when entering a state, both actions will execute at the same time, without A_2 waiting for A_1 response.

```
:sendImNotCompatible rdf:type owl:NamedIndividual ,
                              fsm:Action ,
                              http:Request ;
                     fsm:hasBody :sendImNotCompatibleBody ;
                     fsm:hasPrototypeURI :sendDataPrototype ;
                     http:mthd <http://www.w3.org/2011/http-methods#POST> .

:sendDataPrototype rdf:type owl:NamedIndividual ,
                            fsm:PrototypeURI ;
            fsm:hasParameter :peerUriParameter ;
            fsm:hasStructure "[peer_uri]/send_data" .

:peerUriParameter rdf:type owl:NamedIndividual ,
                           fsm:PrototypeParameter ;
            fsm:hasPlaceholder "[peer_uri]" ;
            fsm:hasQuery
        """
                PREFIX siot: <file:///D:/projects/ontologies/siot/siot.owl#>

                SELECT (str(?peer) as ?peer_uri)
                WHERE
                {
                    self: siot:hasPeer ?peer
                }
        """ .
```

Listing 3.7: Action with prototype URI.

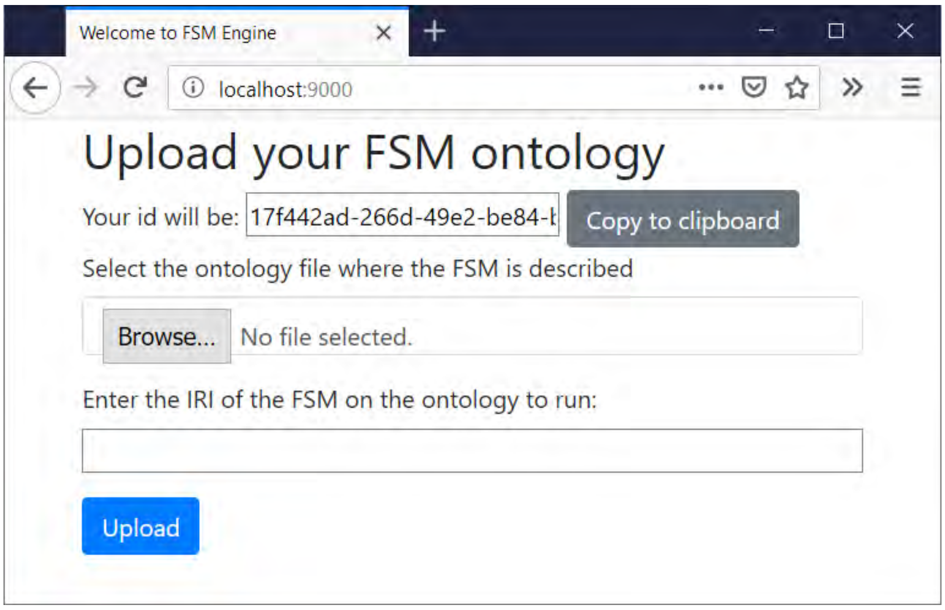
No other FSM operation like checking guards will be performed until the actions finish. An action finishes if it receives a response within its response timeout or it has not received any response during that time. We have made this implementation decision to prevent the VO from being blocked while waiting for the response.

As shown in Figure 3.6, users are provided with a graphic interface where they can upload their models. Users are asked to upload the file with the FSM ontology and the Internationalized Resource Identifier (IRI) that identifies the instance of fsm:StateMachine to be executed. When accessing the web the user is given an ID that identifies the FSM instance. This ID is also used to identify the VO's unique endpoint that it will be listening HTTP/S requests from. The VO is also able to send HTTP/S requests to both web interfaces of physical Things and other VOs. Each VO is provided with a RDF database that stores all the data received and generated during the execution of the FSM. The RDF database uses a base prefix ":" that uniquely identifies the data. The IRI of this prefix is built from the server URL (e.g., *http://localhost:9000/*) and the previous auto-generated ID. The IRI has a # appended at the end to refer to objects inside this domain. The final IRI looks like *http://localhost:9000/[fsm_id]#*.

In order to ease how conditions and actions are built, the VO automatically adds the RDF database's ":" prefix to all RDF data and SPARQL queries that are written on the ontology or sent to the server. The VO also adds a prefix called "self:" that refers to the same IRI that ":" but without the final #, so the FSM instance is able to reference itself. The IRI of self: is then *http://localhost:9000/[fsm_id]*.

The endpoint of the VO provides an API with three generic actions that enable communication with the FSM instance:

- Get data: it returns the result of a SPARQL SELECT query. The query is sent to the



Upload your FSM ontology

Your id will be: 17f442ad-266d-49e2-be84-t

Select the ontology file where the FSM is described

No file selected.

Enter the IRI of the FSM on the ontology to run:

Figure 3.6: Graphic interface where users can upload their models of FSMs. Users are requested to load their FSM ontology and the IRI that identifies the concrete instance of the FSM. Users are also provided with an ID that uniquely identifies the VO that will use the FSM.

server on the request's body. It is a GET request and the URI is *http://localhost:9000/[fsm_id]/get_data*.

- Send data: it saves RDF data on the RDF database. The RDF data to be stored is sent on the request's body. It is a POST request and the URI is *http://localhost:9000/[fsm_id]/send_data*.

- Execute operation: it executes a SPARQL query like INSERT on the RDF database. The query to execute is sent on the request's body. It is a POST request and the URI is *http://localhost:9000/[fsm_id]/execute_operation*.

After the user sends the data via the initial form, a VO is deployed. During the deployment process, the FSM ontology is read and pre-loaded. The process is as follows:

1. Load the FSM ontology into Apache Jena.

2. Search on the model and read the FSM instance of type fsm:StateMachine that has the IRI specified by the user.

3. Read all the states and states' actions. All entry and exit actions are also read.

4. For each state, read the transitions that have the state as the source are searched. For each transition, retrieve their guards, conditions and guards' actions.

In the initialization process, the VO applies default values to some properties if they are not specified on the ontology:

- Action timeout: if no timeout is specified for an action, the VO sets a default timeout of 1 second for that action.

- Action method: if no method is specified for an action, the VO sets the method to GET.
- Action body: if the body is not specified, it is set to an empty string.
- Body type: if no type is specified for a body, it is treated as plain text.

After the initialization process, the VO starts to execute the FSM. The model explained in Section 3.5 and modelled through the paper has been executed using the prototype implementation. Two VOs are deployed and instantiated using the same template. Conditions *isPeerAvailable?*, *isPeerDataAvailable?* and *isPeerCompatibilityAvailable?* are fulfilled by external services. Those services are mocked as the main purpose of the prototype implementation is to validate the feasibility and consistency of our approach. They return true or the necessary data so the FSM can transition to the next state. In order to visualize the execution flow of the FSM in real time via a web browser, we have endowed the VO with a WebSocket interface.

3.8 Use Case Analysis

This section analyses several use cases to assess the suitability and to identify the strengths and weaknesses of our solution.

3.8.1 Use Cases

We have considered seven use cases with different qualitative characteristics. Those use cases are gathered or related to European projects where the authors participate. Three of them relate to the Healthcare IoT for equipment tracking, personal monitoring and medicine dispense, they are related to the ATHIKA project. One of them considers logistics in Industry 4.0, gathered from the project SPRINT. The aim of both projects is to transfer knowledge, and thus a learning enhancement system is analyzed. The Smart City use case involving intelligent trash bins is related to the ENVISERA project. Finally, an analysis is performed over the use case explained in Section 3.5 which is related to the Smart Grid [39, 49], it is referred to as the "Prosumers" use case.

We also provide a qualitative analysis of the effort needed to integrate our approach in different types of middleware. In order to do that, we classify middleware in use cases according to the types described in [110, 111]. We assume that they support the HTTP/S protocol both as server and client. Healthcare middleware is considered service-based. Service-based middleware is more secure than cloud-based middleware in that cloud security and privacy are managed by the cloud provider. However, end-to-end encryption is still challenging in IoT middleware. Actor-based middleware provide the capability of embedding lightweight actors in resource-constrained, thus enabling security capabilities to end devices. Actor-based middleware is not considered for healthcare use cases as many challenges are still to be solved [110]. An event-based middleware is considered for the use cases of logistics in Industry 4.0, intelligent trash bins and prosumers. Concretely, an event-based and actor-based (no embedded actors in the physical device) middleware is assumed [39]. Event-based architectures are recurrent in Smart City, Industry 4.0 and Smart Grid systems as they allow massive data ingestion. Finally, the prototype implementation presented in Section 3.7 is considered to be used between web-enabled devices and a Learning Management System (LMS) system.

Hospital Equipment Tracking

Hospital equipment, like stretchers and wheelchairs, can be tracked to obtain its position and whether they are being used or not. The tracking is used to optimize the utilization of resources, to keep track of the inventory and to reserve the equipment. Our VO can be used to represent each piece of equipment. The VO will be asking the sensors for the current position and for other values like the weight they are carrying. For instance, when the VO detects a significant change on the weight it can send a message to a centralized service.

Hospital Personal Monitoring

On a hospital, patients identity, status or position can be monitored through IoT to increase their safety. The medical staff can also be tracked to optimize their workflow. The VO used for the patient and the staff are different and there is a one-to-one relationship between any user (patient or staff) and a VO. The VO can be used to retrieve the current position and patients' vital signs. If the patient exits their room or some value of a vital sign gets out of a healthy range, the VO can send a notification to the medical team. The staff position is also useful, for example, to alert the professional closer to the patient.

Intelligent Medication Dispenser

An intelligent medication dispenser is a machine that gives patients the correct medication at the right time and also checks for incompatibilities between medicines. This is a perfect tool to avoid problems like the ones explained by the authors in [112] such as a lethal combination of medicines, recalled medicines, not taking medicines on the right time or taking the same dose twice. The VO can implement the logic of deciding which medicines to dispense. First, it will wait for the identification of the patient (for instance, with an Radio-Frequency IDentification (RFID) card). Second, the VO will retrieve the information about the medication assigned to the patient by a doctor (this information has been previously added). Then, it will select the medication that needs to be taken at that time and verify that it had not been taken already. The VO also checks the compatibility with other medicines. It uses an external database that describes medicine compatibility in RDF format. Then, the VO executes a SPARQL query with the necessary reasoning logic. If any incompatibility is found, the patient is notified. The VO can be also programmed to alert patients at the time they have to take the medication through a smart-alarm or a smartphone.

Logistics on Industry 4.0

Logistics on warehouses can be made more efficient through automatic processing of the incoming items. The author in [113] introduces a use case where all pallets of items have an RFID tag. The RFID tags are read on their arrival by RFID readers. Multiple tags can be read at the same time as their position on the pallet is not important as long as they are in the range of the reader. The tag can provide information like the product identifier, the number of items in the pallet or additional information like fragile content or notes written by the sender or the driver. The stock of the warehouse is updated through an Enterprise Resource Planning (ERP) application. A VO can implement this process. For each RFID tag processed, the VO sends the information of the tag to the ERP system. The VO can ask the ERP for the best place to store the incoming item or it can even implement the logic of deciding the best place to store it via queries and reasoning. Finally, the VO will notify a worker or robot to carry the item to its storage place.

Learning Enhancement

The learning performance of students can be improved by monitoring the environment of the classroom and the status of the student. The work done in [114] explains a way to achieve this; they use the technology of WSNs to integrate temperature, humidity, illumination, the concentration of CO_2 sensors and human emotions detection cameras on wireless networks. Some parameters like temperature are used to automatically adjust the room temperature, but others like the students' mood may need the attention of the professor. A VO can be used to actively monitor all the parameters and adjust the conditions of the room to optimal values; it can also alert the professor about the students that are losing focus. We assume that the VO has a one-to-one relationship with the classroom (temperature, humidity, CO_2, ...) and one-to-many with the students in a classroom, but it is a relationship with a reduced and limited set of entities (students).

Intelligent Trash Bin

The intelligent trash bin is an already used technology on Smart Cities that monitors the state of trash bins and enables optimization on the waste collection. Authors in [115] propose an architecture that implements this technology; the bin contains an Arduino and attached sensors that collect information about waste level, temperature, humidity, motion and weight. The information of the sensors is sent to the cloud where patterns to optimize the collection of waste are extracted. A previous step can be achieved using edge computing to analyze the physical state of the bin to prevent heavy damage, as the analysis on the cloud may not be real time. The one-to-one or one-to-many relation between a VO and an intelligent trash bin or set of trash bins (if they are placed close to one another) can be established. The VO is placed on the edge cloud. Each time, the VO receives the report from the Arduino, before sending it to the cloud, it will analyze the values of humidity, toxicity (with additional sensors), motion or weight to detect potential problems like fire, vandalism/bad use or a possible toxic hazard. If any of these problems are detected, the VO sends an alert to the trash management system.

3.8.2 Analysis and Discussion

This section provides a qualitative analysis of the described use cases. A summary of the analysis is provided in Table 3.1. The following metrics are considered to provide the analysis:

- Time reliability. This metric considers the limitations of the technologies that have been used to develop our solution, namely RDF, SPARQL and HTTP/S. They allow interoperability and shareability, the main focus of our proposal. However, they may be computationally heavy, especially SPARQL. This solution may not be suitable for time-critical applications. Specialized systems are better suitable to run time-critical applications. The speed of the network due to congestion can be mitigated with a dedicated Local Area Network (LAN). A Low score means that the VO cannot keep up with the timing requirements of the use case. A High score means that the use case does not need accurate timing and that some delay is acceptable.
- Scalability. Scalability represents how the VO performance scales with the number of Things that it manages. The total performance of the system degrades when the number of Things controlled increases. Usually, the performance degradation is linear and the biggest impact will be at time reliability because the available resources will

be shared between more Things. Having more Things to control usually means that there will be more states and actions to perform, and that will translate in more delays. A Low score means that the manner in which the system is designed provokes a performance degradation as more Things are integrated into the system.

- Reusability. We consider reusability as the degree of how easy it is to reuse the generated template or templates for other similar use cases. For example, a template used by a VO that senses the room temperature and actuates accordingly to increase the comfort of the inhabitants can be easily used on other rooms or buildings. It may be only necessary to change the temperature thresholds. The temperature is represented as the value to be queried and not hardcoded in the execution flow or template. This example has a High reusability. On the other hand, a template that describes the behaviour of a robot that assembles car pieces has Low reusability. There will be substantial differences in the assembly logic of different models of cars and between companies.
- Suitability. Suitability is the degree of fitness between the behaviour design that the current FSM ontology allows and the desired functionalities/behaviour of the use case. If none or few modifications are needed, the suitability is considered High. If major modifications are needed, then suitability is Low.
- Incompatibilities (with domain-specific ontologies). The degree of incompatibilities between the base ontologies used by the proposed solution and the ontologies of the use case domain. A score of High means that it is hard to integrate our solution with domain-specific ontologies. It is possible to have a score of None if there are no incompatibilities.
- Outsourcing. Outsourcing refers to the need to use external services that provide operations that the FSM cannot do. If the VO is independent or uses few external services, then the outsourcing is considered Low. VOs that act as service aggregators have high outsourcing. The value of this metric does not have an explicit positive or negative meaning, its purpose is to describe a characteristic of the VO behaviour.
- Query complexity. Query complexity is an estimation of how heavy the queries of the use case are. Heavy means that the number of returned triples is very high or/and the query execution time is expected to be long (more than half a second). A High complexity means that heavyweight queries will be performed during the execution flow.
- Knowledge complexity. Knowledge complexity refers to how much knowledge about ontologies, SPARQL and behaviour representation with FSMs the user requires to implement the system for a concrete use case. A High score means that the user necessitates a High degree of knowledge to represent the behaviour of the system. A Low score denotes that few experiences are needed as it is easy to represent the behaviour.
- Integration Effort. Integration effort considers the required effort to integrate our system in the middleware/architectures assumed for each use case. If the middleware supports actors or VOs, the effort is considered Low or Medium.

Time reliability is High in the cases where it is not important if the system has delays greater than one second, for instance, it is not critical if the VO of the medicine dispenser has a delay of two seconds. It is Medium where a second of difference is important like

Table 3.1: Use cases benchmark.

	Hospital Equipment Tracking	Hospital Personal Monitoring	Intelligent Medication Dispenser	Logistics on Industry	Learning Enhancement	Intelligent Trash Bin	Prosumers
Time reliability	High	Medium	High	Medium	High	High	High
Scalability	High	Medium	High	High	High	High	High
Reusability	High	Medium	High	Medium	High	High	High
Suitability	High	High	High	High	High	High	High
Incompatibilities	None	None	None	None	None	None	None
Outsourcing	Low	Low	Medium	Medium	Low	Low	Medium
Query complexity	Low	Low	Medium	Low	Low	Low	Low
Knowledge complexity	Low	Low	Medium	Low/Med.	Low	Low	Low
Integration Effort	Medium	Medium	Medium	Medium	Low	Medium	Medium

monitoring the health of a patient (a fast alert could save the patient's life). In the case of Logistics, it also critical if, for each item to be unloaded, there is a delay of some seconds because a lot of items are expected to be processed fast and that can sum up to a big delay.

Scalability gets a Medium score on hospital personal monitoring because adding more Things to the VO may cause bigger delays that are not desired. This is because if more constants are to be monitored on the patient, the tasks and logic that the VO performs increase.

Reusability is Medium on the logistics use case as the desired behaviour about how the items are stored and processed may change for each company or warehouse. The idea behind templates is to define a behaviour abstracted from the physical implementation to provide reusability. It is also Medium in the use case of hospital personal monitoring as each hospital may have different workflows to optimize the performance of the staff. Reusability is High in other use cases as we consider that the behaviour/execution flow can be defined generically.

Suitability is High in all use cases as the VO is able to execute the desired behaviour. There are no incompatibilities between our proposed solution and domain-specific ontologies. Our solution relies on ontologies in order to be as compatible as possible with other ontologies. Domain-specific ontologies are expected to extend higher level ontologies such as SOSA/SSN in order to promote template reusability.

Outsourcing is Medium where the system has some operations that rely on external processes. The medication dispenser relies on an external catalogue in order to check medicine compatibility. The logistics use case scores Medium at outsourcing because it relies on the ERP to perform its tasks; some operations are too complex to be implemented by a FSM or need specialized software.

Finally, query complexity is Low in almost all cases as the queries are usually with local data and are simple. It is Medium on the medication dispenser because there may be some heavy queries that check the compatibility between all the medications of the patient. Knowledge complexity is Medium on the dispenser as the queries for medicine compatibility check can be complex to write. On the logistics, the queries can be also complex if the ERP does not provide the location to store the items; queries of searching the optimal place should be written in this case.

Our approach is easily applicable in learning enhancement, intelligent trash bin and prosumers use cases. We consider that they have a generic structure (High reusability) and that the modelling of their behaviour is straightforward with Low query and knowledge complexity. They are not time-critical use cases so they score High in time reliability. Their score is High on scalability, as each VO has a relation one-to-one or one to a reduced and limited set of entities to control. Prosumers use case has a Medium score on Outsourcing as it uses external services to get informed if a peer is available or for the complex operation of checking the energy profile compatibility.

The integration effort with any concrete middleware comprehends (1) implementing the FSM engine in the programming language used by the middleware; (2) implementing a software adapter to connect the FSM inputs and outputs to the HTTP/S server and client interfaces of the middleware; and (3) providing an RDF and SPARQL API to the FSM engine. If the middleware provides a TDB, it is only necessary to connect the FSM engine to the database. Otherwise, it needs to be implemented. For that reason, the integration effort is considered Medium in almost all use cases and respective type of middleware. Once this initial effort is done, the effort remains to create the template of the desired behaviour using the ontologies and considerations described in this manuscript. Regarding the learning enhancement use case, a simple web service like the one presented in Section 3.7 can be used as the orchestrator between web-enabled devices and the LMS. Note that the purpose of the prototype is only to provide a web execution environment to the FSM engine. It lacks common web security and privacy mechanisms such as authentication and authorization. If the integration between our approach and the middleware is not possible or requires too much effort (e.g., in middleware that does not have the concept of VO/(Web-)Thing), our approach can be used at the application layer as a microservice that composes with the middleware's HTTP/S API.

3.9 Conclusions and Future Work

Given the heterogeneity that characterizes the IoT, a novel middleware-agnostic approach that allows for describing and executing the behaviour of devices has been proposed and implemented. The objective is to allow the reusability and shareability of the execution flow among multiple and heterogeneous IoT deployments. The approach relies heavily on existing standards to promote interoperability and reusability. FSMs are used as the model to create the execution flow using ontologies. Our work is contextualized in a reference architecture recommended by W3C, the W3C WoT Architecture. We have used the concept of VO, which are virtual counterparts of physical devices as the computational entities that run the concrete instances of the templates. Several use cases have been analyzed to asses the viability and suitability of our solution. Relying on standards such as RDF, SPARQL and HTTP/S has some drawbacks. They tend to be more heavyweight than an ad hoc solution. For example, HTTP/S can be replaced by CoAP (a lightweight, IoT version of HTTP/S) and the data model can be replaced by a SQL or NoSQL database. For that reason, our approach is not well suited for time-critical applications such as monitoring and reasoning over patients' vital signs in a hospital. However, our approach is well-suited for IoT scenarios that are non-time-critical and with a low level of variability between each deployment. It allows for reusing behaviours for heterogeneous physical devices with the same set or subset of capabilities.

In future works, we expect to fully apply a TD interface to the VOs. Our aim is to extract and generate part of the TD definition from the FSM instance. This will also enable the alignment with other ontologies such as the oneM2M Base Ontology and enable service composition. Given that each state has its own inputs and outputs, research is needed to identify which Properties, Actions and Events (according to the TD model) should be exposed. SPARQL performance is computationally heavy and our solution only allows to send SPARQL queries to external services. We plan to add the capability of sending non-SPARQL requests to external services described using ontologies such as TD ontology or oneM2M Base Ontology. Our goal was to facilitate deployments in similar IoT scenarios. Nevertheless, the task of creating the template ontology can be tedious, especially if the template has multiple states and actions with prototype URIs. Our approach has some restrictions imposed by the FSM model and the WoT architecture (we assume that physical devices expose an HTTP/S API). Therefore, we plan to create a visual tool to hasten the creation of FSM templates using visual building blocks like in StarUML. The IDE will be used to create the template in a visual manner and to automatically translate the visual representation to the ontological one.

4 Social Internet of Energy - A new paradigm for Demand Side Management

4.1 Preamble

4.1.1 Abstract

The creation of prosumer communities leveraging the Smart Grid domain to improve energy production and consumption patterns has been proposed in the literature. We propose to use a novel Internet of Things paradigm, namely the Social Internet of Things, to enable appliance-level metering (sub-metering) and to create prosumer communities dynamically. We use the term Social Internet of Energy to refer to the integration between prosumers and devices via social relationships. This paper explores the diverse technologies that will enable the Social Internet of Energy. On the one hand, we explore the different aspects and value propositions of the Social Internet of Things, applying this novel paradigm to the creation of prosumer communities. On the other hand, we explore Demand Side Management, its synergy with Social Internet of Things and the properties and challenges that may arise. Concretely, in this paper we aim at grouping prosumers by their energy load shape compatibility and then reschedule intra-group load shapes. We propose a clustering model and we develop two clustering heuristics to distribute prosumers among Prosumer Community Groups and one Peak-to-Average Ratio prediction heuristic in order to reduce the Peak-to-Average Ratio and the amplitude of energy peaks per cluster. Then, on the basis of a real dataset, we perform simulations and analyze and discuss the results, paving the way for further research.

4.1.2 Author Contributions

Víctor Caballero conceived the idea with contributions from Agustín Zaballos. Víctor Caballero executed the analysis and experimentation presented in this work. David Vernet and Agustín Zaballos supervised the execution of the work and contributed with insightful discussions and ideas to improve the overall quality of the work.

4.1.3 Funding

This work received funding from the “Agència de Gestió d’Ajuts Universitaris i de Recerca (AGAUR)” of “Generalitat de Catalunya” (grant identification “2017 SGR 977”) and ENVISERA project, funded by Spanish Ministerio de Economía y Competitividad (Plan Estatal de Investigación Científica y Técnica y de Innovación 2013-2016, grant reference CTM2015-68902-R). Part of this research has also been funded by the "SUR" of the "DEC" of the "Generalitat de Catalunya" and by the European Social Funds (grant reference "2019FI_B2 00164"). In addition, we would like to thank La Salle-Universitat Ramon Llull for their support.

4.1.4 Acknowledgements

Icons used in Figure 3.3 made by Freepick and Gregor Cresnar at www.flaticon.com.

4.1.5 Conflicts of Interest

The authors declare no conflict of interest.

4.2 Introduction

Energy demand in the world is increasing rapidly. Worldwide energy consumption is expected to increase by 29.7% from 2017 to 2050 [116]. The majority of the energy demand is met by non-renewable energy resources which are starting to be insufficient and produce undesirable climate changes that harm the world we live in [3]. Hence, society is moving towards the use of renewable and sustainable energy sources. In order to meet worldwide energy demand using RESs, it has been proposed to group energy users (prosumers) to optimize generation, demand-side or storage resources and increase individual DER visibility, while allowing these groups to work autonomously to some extent. Those features are and will be possible thanks to the underlying SG infrastructure. The SG can be seen as a hybrid system composed of the electric power system and a heterogeneous ICT infrastructure [49]. Several solutions have emerged, aiming to increase the visibility of individual DERs and optimize system operation. Proposed solutions for the Communityware Smart Grid [117] aim to create prosumer clusters to make them gain market visibility and reduce system operation costs. Three main clustering solutions have emerged: Virtual Power Plants, Microgrids and Goal-Oriented Prosumer Community Groups. A brief description of each one is given in the following list.

1. Microgrid: A MG is an integrated energy system consisting of distributed energy resources and multiple electrical loads operating as a single, autonomous grid either in parallel or "islanded" from the existing utility power grid [10].

2. Virtual Power Plant: VPPs rely upon software systems to remotely and automatically dispatch and optimize generation, demand-side or storage resources in a single, secure Web-connected system. VPPs represent an "Internet of Energy" tapping existing grid networks to tailor electricity supply and demand for a customer, maximizing value for both end-user and distribution utility through software innovations. [10].

3. Goal-Oriented Prosumer Community Group: In contrast, PCGs virtually interconnect prosumers that may not be technically connected. According to [17], a Goal-Oriented PCG is defined as "a network of prosumers having relatively similar energy sharing behaviors and interests, which make an effort to pursue a mutual goal and jointly compete in the energy market". As a result, PCGs can create a dynamic ecosystem of cooperating prosumers [11, 17].

The work presented in this article emerges from the concept of Smart Grid flexibility and Software Defined Utility (SDU), presented in [49], from knowledge gathered in the FP7 European projects INTEGRIS (INTelligent Electrical GRId Sensor Communications) [118] and FINESCE (Future INternEt Smart Utility ServiCEs) [119]. Thus, the proposed approach aims to be a technological solution to achieve the so-called Smart Grid flexibility by some European utilities such as ENEL, ESB or Iberdrola. In order to provide increased Smart Grid flexibility, there are some works related to the Future Smart Grid [120] carried out by the authors [118, 119].

PCGs excel for their dinamicity at managing resources, leveraging a SG infrastructure. The SG can be considered a particular case of the IoT [39], referred as the IoE [16]. IoE aims at optimizing the efficiency of the energy infrastructure by creating a distributed network of sensors and energy generators. In this situation, a novel paradigm, namely the SIoT can contribute in many manners, enabling a Social Internet of Energy.

The SIoE aims at enhancing the connectivity and relationships established among the users and devices in the IoE (and Future Smart Grid) by leveraging the technology offered by the SIoT. In this work, we will focus on explaining how PCGs can be dynamically allocated and then how those groups can be allocated in a hierarchical/fractal style. The main contributions of this article are:

- Explore how the SG can harness the SIoT technology to improve energy management at the demand side (DSM).
- Analyze which technologies are involved when combining DSM and SIoT in a SG.
- Advance in the research of the combination of DSM and SIoT through experimentation and evaluation using cluster-based segmentation.

This article is organized as follows. Section 4.3 gives a brief overview of how the SIoT emerged and its characteristics. Section 4.4 proposes to apply the SIoT paradigm to the SGs and explains the benefits of such approach to create communities of prosumers. Section 4.5 presents a use case: to apply SIoT to DSM and analyzes the manners in which SIoT-enabled devices can cooperate. Section 4.6 presents the DSM model that will be used to create communities of prosumers. Section 4.7 explains the tests that have been performed to evaluate the presented approach and analyzes and discusses the results. Section 4.8 proposes a new hierarchical clustering algorithm that nourishes from the conclusions drawn in Section 4.7. Finally, the article is concluded in Section 4.9, where future work is also presented.

4.3 Background: Social Internet of Things

The Social Internet of Things (SIoT) is one of the last and novel configurations proposed in the IoT framework and, according to a generational analysis of the IoT [1], it belongs to the third generation of IoT. The SIoT envisions interactions among humans and things, besides human-to-human and simple object-to-object interactions, the former enabled by SNSs and the latter already tackled by the IoT vision. By leveraging SNs new logical configurations involving both humans and things or networks of things can be created. In the future SIoT, humans and things are not logically separated and instead, can be represented as nodes of the same network, both providing services [121]. For example, in [24] a SWoT is proposed, where users can access physical devices through SNSs authorization mechanisms and physical devices (things) can publish to the SNS by the same mechanisms: SNSs delegated access mechanisms such as OAuth. The work done in [24] is also relevant as SNS friend relationships are used as the trust model in order to authorize users to perform actions that involve physical devices (web-enabled things) owned by another entity. The realization of the SIoT paradigm regarding object communities will bring desired properties to the IoT:

1. Scalable navigability via relationships.
2. Flexibility, as each node identifies its friends and uses those connections to perform scalable service discovery and matchmaking.

3. Trustworthiness that, given a specific goal, will allow nodes to use their experience (individual or collective) to decide which service is most reliable to serve that goal, based on the level of trust.

The novel SIoT paradigm is evolving rapidly, and thus appropriate reviews have emerged. As explained, the SIoT first steps where in integrating SNSs with the IoT. In [24], the WoT, a web-enabling paradigm for the IoT, and SNSs where merged together, enabling the interaction between things (mainly human-to-thing interactions). The work done in [122] provides a good review on the evolution from the IoT to the SWoT, and reviews main architectures and work related to SWoT. For example, several works use SNS to authenticate and authorize owners' friends to access their devices [65, 123]. The review article [122] presents early work on blending IoT, Web technologies and SNSs; thus, work related to SIoT understood as networks and communities of objects is excluded. The integration between SNs and IoT continues with the idea of objects creating networks and communities among themselves [28]. Researches in this direction aim to describe how social relationships between objects or things are established and what mechanisms are used to establish those connections. In fact, a review on the cluster between IoT and SIoT [124] highlighted the benefits of integrating objects with social relationships, the emerging architectures of SIoT and the challenges and open research issues for this new paradigm.

As in human social relationships, object relationships are also based on "feelings" about each other. In the SIoT paradigm, those feelings are referred as trust properties and serve as the computation parameters to decide in which degree an object can be trusted, and thus to measure the healthiness of the relationship [125, 126]. In this sense, trust management mechanisms and social network structures in SIoT are reviewed in [127] and [128], along with several works contributing to the SIoT. Also, in the same manner as human relationships, trust relationships among SIoT objects can be vulnerable to some attacks such as bad-mouthing or whitewashing attacks [129]. The following sections briefly describe the SIoT properties that will be leveraged in the remainder of this paper and give appropriate references for further reading:

4.3.1 Relationships

According to [29], an object can establish five types of relationships or friendships:

- Parental Object Relationship (POR): Defined among similar objects, built at the same period by the same manufacturer and can be used, for example, to share optimal configuration.
- Co-location Object Relationship (CLOR): Established among heterogeneous objects used always in the same place. Some of those relationships will be established regardless of the need for object cooperation, but they are useful to create short links in the network for discoverability purposes.
- Co-work Object Relationship (CWOR): Created among heterogeneous objects that need to collaborate to provide a service. For example: sensors in a city to obtain temperature measurements.
- Ownership Object Relationship (OOR): This relationship is established among heterogeneous objects that belong to the same user.
- Social Object Relationship (SOR): Defined when objects come into contact (close range) because their owners come in touch with each other during their lives. Those relationships, which usefulness varies from friend to friend, enable the creation of networks

and communities of objects, aiming to improve IoT scalability and navigability and to provide the user with the desired service or service composition.

4.3.2 Trust management

Trustworthiness management in the SIoT involves the formulation of trust mechanisms to understand how to build a reliable system based on the behavior of other objects. Trust involves, at the basic level, two objects, the trustor and the trustee in a relationship context, such as its goal. Trust properties are reviewed in [125], where the authors present several ways in which trust can be computed. The computation of trust can involve multiple parameters. It is straightforward to think that the computation only involves the trustor and the trustee. While this is true, it can also involve opinions and recommendations from other nodes or context in which the trust relationship is created, including past experiences. Two types of trust levels are analyzed in [125], objective trustworthiness and subjective trustworthiness.

- Objective trustworthiness: The information about each node is distributed and stored using a Distributed Hash Table (DHT) structure. This information is visible to every node but is only managed and stored by Pre-Trusted Objects (PTOs).
- Subjective trustworthiness: Each node computes the trustworthiness of its friends on the basis of its own experience and the experience of its friends. If the node providing a service is not a direct friend of the one requesting it, the trustworthiness is calculated by word of mouth through a chain of friendships.

4.3.3 Service Discovery

Regarding the scalability of the discovery process and navigability, each thing is cognitively able to query a service through its established relationships and evaluate the trustworthiness of the returned service. This form of tackling scalability and navigability from a social community perspective is based on the conclusions that people are tied by short chains of acquaintances [27] and that there are structural clues in a social network that help people find a short path even without a global knowledge of the network [130]. For example, a novel Resource Discovery Mechanism based on Preference and Movement Pattern Similarity with the aim to build sub-communities of similar preference and movement nodes is proposed in [131]. Then, virtual global communities are formed based on the sub-community similarity as depicted in Figure 4.1, improving the search performance. In addition, the authors also design a resource discovery algorithm that dynamically adjusts the search radius to balance the performance and communication overhead.

4.4 Social Internet of Things and Goal-Oriented Prosumer Community Groups

We envision that SIoT technologies can bring goal-oriented PCGs closer to reality. The following subsections explain the synergy that can be achieved by applying SIoT paradigm to PCGs:

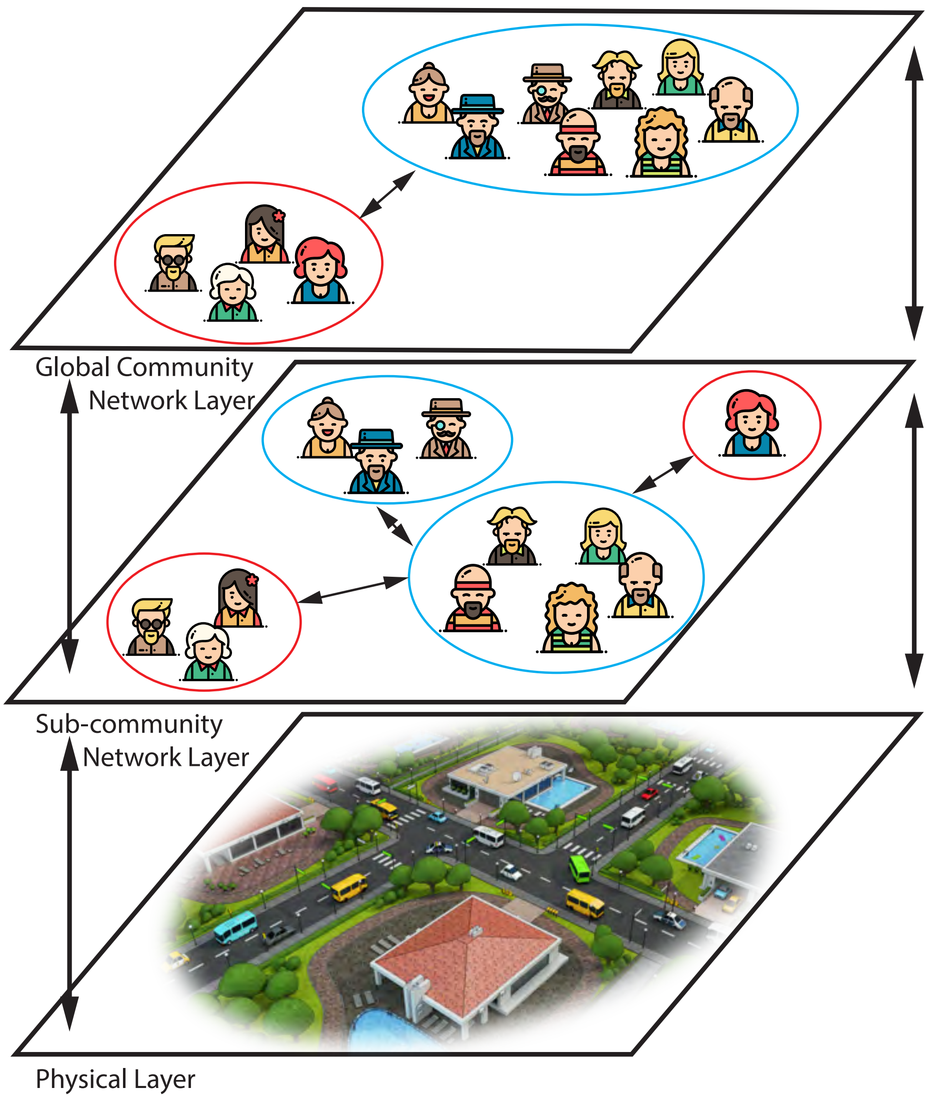


Figure 4.1: Three-layer network structure. Sub-communities are created on the basis of preference and movement similarity. Then, virtual global communities are created on the basis of sub-community similarity.

4.4.1 Homogeneity

The SIoT envisions the creation of social networks of things, where they can intercommunicate with each other. To achieve this goal, every thing or VO [34] must be capable of "speaking" the same language. The WoT paradigm uses the HTTP/S to achieve communication protocol inter-operability among things and Semantic Web technologies [30] to achieve message format homogeneity and ease discoverability. For example, authors in [132] implement a Semantic Search Engine to discover available world wide web services. The Lysis platform [91] implements the WoT paradigm and only requires physical devices (for example, smartphones) to be registered in the cloud platform. Once registered, they become a social-enabled thing. Thanks to the relations a thing can establish with other things, such thing can be notified by a "friend" (another thing) to install a specific application that provides the desired functionality. The example in [91] involves a user's goals (to arrive at the hotel where she has made a reservation), a smartphone and an airport multimedia infopoint. They establish a SOR. Thanks to the SOR, the airport multimedia infopoint suggests the user to install a SocialMobility App, which, in turn, leverages the Co-location Object Relationship (CLOR) relationships established by the airport multimedia infopoint and its surrounding services, like taxi service and bus station service. Finally, the user is able to receive information about the transportation services she needs to arrive at the hotel.

Goal-Oriented Prosumer Community Groups as explained in [17] make use of energy consumption and production data to perform statistical analysis and assign each prosumer to a group. Users (prosumers) usually do not own devices from the same brand. This means that devices, which could be renewable generators or smart meters, cannot easily interconnect with each other; neither from a local perspective (i.e by means of a local area network or short-range protocol) nor from a virtualization perspective (i.e cloud APIs), as they are vendor specific. IoT technologies and platforms allow devices to connect to each other and expose a standardized API to ease and homogenize access. Therefore, in the sense of homogenization, PCG can greatly benefit from IoT and WoT technologies such as the one presented in [91]. An energy-related use case that leverages SIoT is given in [92], where the authors use thing-to-thing relationships to optimize energy usage by the HVAC system in their laboratory. Results show that the comfort-aware Smart HVAC system ensures users' thermal comfort, while at the same time reduces energy costs with respect to static and traditional approaches.

4.4.2 Leveraging object relationships for Prosumer Community Groups

Regarding social relationships between objects, two straightforward use cases emerge related to Prosumer Community Groups.

Prosumer Ownership

The use case deals with the aforementioned heterogeneity. This use case is similar to the one presented in [92] and is as follows. Prosumers are users that own devices that produce/consume energy. To create a PCG, the basic social network that is needed to create is the one composed of all devices that produce or consume energy, owned by the same prosumer. After enabling those devices to be part of the homogeneous IoT they will, according to the SIoT relationships, establish an OOR. The just established OOR will enable them to easily find and communicate with each other and will allow the creation of Micro Engines (MEs) [91] that aggregate multiple individual functionalities. This will allow data aggregation whenever it is necessary through MEs and ad-hoc feedback, as smart appliances will be able to be the target of a supervision strategy. For example, according to the user's goals and the PCG goals, a smart appliance can show a warning to the user if he/she wants to turn the appliance on (i.e, a washing machine) in a condition unfavorable to attain such goals.

Goal-Oriented Prosumer Community Groups formation

This second use case relates PCGs dynamic formation and SIoT. The authors in [133] propose to create a Social Network Service to assist in the creation of Goal-Oriented Prosumer Community Groups. It is our belief that the first step, PCG formation, can be greatly assisted by SIoT technologies. Several sub-steps are involved in PCG formation. The following list enumerates them and highlights how they can be completely o partially achieved thanks to SIoT technologies.

1. Prosumer's energy behaviour: Current energy behaviour can be implicitly analyzed by observing quantitative energy consumption and production. An IoT infrastructure for data aggregation can be enough to query these measurements. However, the trustworthiness of the obtained data can be compromised. For example, the prosumer could change consumption/production data to obtain a benefit by joining a PCG with pre-qualification criteria that otherwise he/she couldn't have applied for. The SIoT offers two trust mechanisms: objective trustworthiness and subjective trustworthiness [125]. In the current setting, where it is of great importance to avoid SIoT-related attacks to prevent community deterioration, subjective trustworthiness mechanisms are preferred, as they are able to isolate malicious nodes, even with 70% of malicious nodes [125].

2. Prosumer Community Group dynamic formation: Ad-hoc networks are the basis of SIoT and, as such, some of the SIoT basic relationships and discovery mechanisms are based on social interactions such as SOR and Co-work Object Relationship (CWOR). Therefore, prosumers that belong to a social group are more likely to share the same energy prosumption behaviours and energy-related goals. Hence, ad-hoc social relationships between objects can have a beneficial impact on PCG formation, but it will not replace human intervention. For example, a goal-oriented PCG formation mechanism has to be aware of energy-related goals that prosumers want to achieve in the future. Prosumers can input their desired energy-related goals via a web-service or a mobile app. This information is then shared among and via "friends" via SIoT relationships, as it is done in [91], such process is highlighted in Section 4.4.1. Human-to-human interaction between prosumers and authorized agents is tackled in [133], which we envision that could be nourished by a lower layer, the SIoT, leveraging social relationships between objects to exchange prosumer's energy consumption and production data and their goals, optimizing potential PCG target prosumers and network traffic.

As demonstrated in [33], a network with a giant component can be generated by only using three types of relationships: OOR, SOR and CWOR, the same ones we propose to create PCGs. This means that, eventually, there will exist a path between all SIoT-enabled devices. Even if CWOR relationships are excluded, the creation of the giant component is still possible. The authors also conclude that new SIoT search mechanisms can take advantage of external metrics (distance metrics between SIoT objects characteristics that are external of the network) to improve their performance. Sections 4.6 to 4.8 are devoted to find object distance in the context of PCGs.

4.4.3 Goal-Oriented Prosumer Community Group dynamic formation

This section illustrates the envisioned process of dynamic group formation that leverages the SIoT paradigm. Assuming that a user is able to perform actions in a SIoT cloud platform such as Lysis [91]:

1. A user interested in being part of a PCG to benefit from the community and achieve certain goals installs PCG discovery app on his/her smartphone. The smartphone becomes a SVO [91].

2. The potential PCG user installs apps or allocates apps in the cloud that enable retrieving data from his/her energy-related devices (devices that produce or consume energy), enabling social capabilities for each one (SVO).

3. Devices and DERs that consume and/or produce energy owned by an owner establish an OOR relationship (Figure 4.2, 1). This relationship is then leveraged to obtain energy-related data from DERs that belong to the same owner. These data can be both aggregated and stored or stored individually for each device.

4. The data obtained is used to build an energy user profile along a time period. The profile is then stored as part of the PCG app, and thus enabling the smartphone and other social devices to retrieve such information. A social device is a device that the user (prosumer) usually carries with himself/herself. We specifically refer to social devices as the key enablers to create SORs between users, as demonstrated in [33].

5. When the user establishes a social relationship with another potential user (Figure 4.2, 2), both social devices (i.e, smartphones) also establish social relationships (Figure 4.2, 3) and exchange energy profiles and goals. Then, the SVOs determine if a PCG needs to be created or one of the users is eligible to be part of an existing PCG (i.e, the PCG the other user is part of).

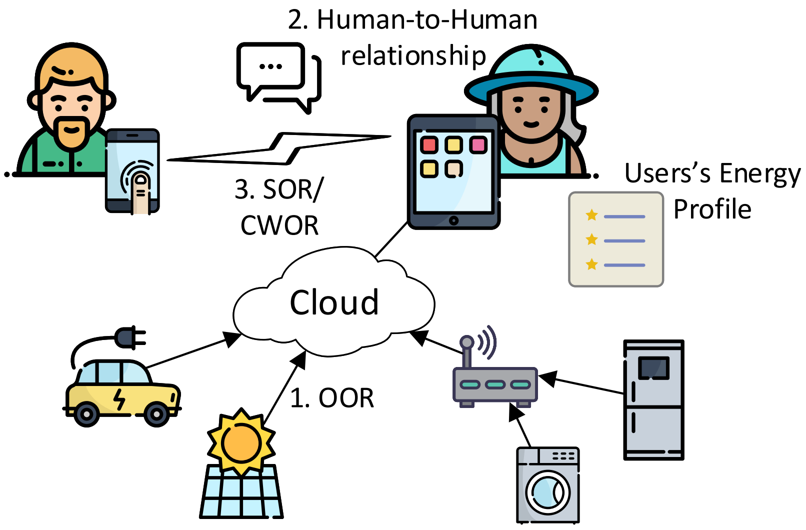


Figure 4.2: PCG formation leveraging SIoT. Firstly, prosumers and appliances establish an OOR via the augmented SVOs (1). When prosumers come close together and establish and interact (2), their devices establish SOR or CWOR relationships and exchange the energy profiles of the prosumers (3).

4.5 Use case: Demand Side Management and Social Internet of Things

This section presents a use case where a SG-enabled feature, namely DSM, is complemented and improved by leveraging SIoE.

The SG improves reliability performance and allows customers' responsiveness by encouraging better efficiency decisions. DSM represents an integral part of SG. DSM is the modification of consumer's demand for energy through various methods to better match the demand for power with the supply. Some examples of DSM programs are DLC and RTP. "DLC programs for residential load management are based on an agreement between the utility company and the customers. The utility, or an aggregator, can remotely control the operations and energy consumption of certain appliances such as lighting, thermal comfort equipment, refrigerators, and pumps" [21]. "According to RTP programs, the price of electricity varies at different hours of the day and each user is expected to individually respond to the time-differentiated prices by shifting its own load from the high-price hours to the low-price hours" [21]. In this context, Interruptible Loads (ILs) are consumers who agree to be interrupted, as required and within constraints, to maintain system security or reduce market prices, and are usually compensated by paying reduced tariffs.

We present an example scenario using SIoE for DSM as follows: a user utilizes a specific mobile application to enable SIoT interactions. The application uses the energy profile gathered by querying the user's home appliances. The application also asks the user to set time periods where the smart appliances can function; for example, a user is usually at home at 17:00 and he can have the clothes in the washing machine at 17:30. He only needs to run the washing machine once, and it can be from 17:30 to 19:00. In that way, the user sets the time period from 17:30 to 19:00. Finally, the application shares the energy profile with other users using SIoT relationships, for example, using SORs. When two applications detect that both user's energy profiles are compatible, they create a small PCG. Given two or more prosumers $a, b, c, ...$, we define their "compatibility" as how much they contribute in reducing their join Peak-to-Average Ratio (PAR). Their compatibility increases as $par(a + b + c + ...)$ gets colser to 1. The process can be repeated for each new user that meets a user in the PCG via SIoT relationships. This process aims to create multiple PCGs that try to autonomously reduce overall energy peaks and improve energy generation efficiency. In case a user needs to start an appliance out of his autonomously scheduled time, the application leverages SIoT relationships within its PCG to find a "friend" able to re-schedule the time when his appliances will start functioning to stabilize load curve as much as possible between both users. Which will, in turn, stabilize the load curve of the PCG they are in. This process is repeated until the overall load curve is optimized for the situation. The process just described can harness the potentialities that integration with the WoT provides. Related work in merging the WoT and SG for web-enabled, energy-aware smart homes describes a similar DSM use case. The authors in [134] enable smart appliances to the web by providing an HTTP/S API that allows the interaction with them by sending queries and commands. It is proposed that the SG Utility informs about variable electricity tariffs to a Home Controller, which in turn tries to reschedule the working period of the smart appliances. Hence, the WoT is used as a set of standards to create an energy-aware smart home.

4.5.1 Properties

In the presented scenario, we can identify properties and issues to be addressed. The following is a list comprehending those properties and next sections analyze each one in more detail.

- The SIoT paradigm enables local rather than global search and navigability, although this does not mean that distant nodes cannot establish a relationship between themselves. What is to be tackled in this regard is the eligibility of a node to be a member of a cluster (PCG) and how to search for new members.
- There are several clusters of users (PCGs). This means that users are distributed into groups according to some variables. By splitting users into groups, energy management can be greatly improved as in MG, VPP and PCG.

- A scheduling algorithm is needed in order to schedule the times when electrical appliances should start in order to reduce energy consumption peaks. The scheduling algorithm can be based on a global knowledge assumption or agent-based knowledge, meaning that each agent/user contributes to the scheduling process without sending all information to a "global" entity that executes the algorithm.
- In order to evaluate scenarios involving combinations of the items presented before, we can assess two properties: communication efficiency and overall energy efficiency.

Cluster organization

As it has been already mentioned, we expect to create multiple PCG using the SIoT paradigm (Figure 4.3). Yet, the inter- and intra-cluster organization and interaction has to be explored. Once the appropriate members to form the PCG have been found, two main intra-cluster member organization can be analyzed.

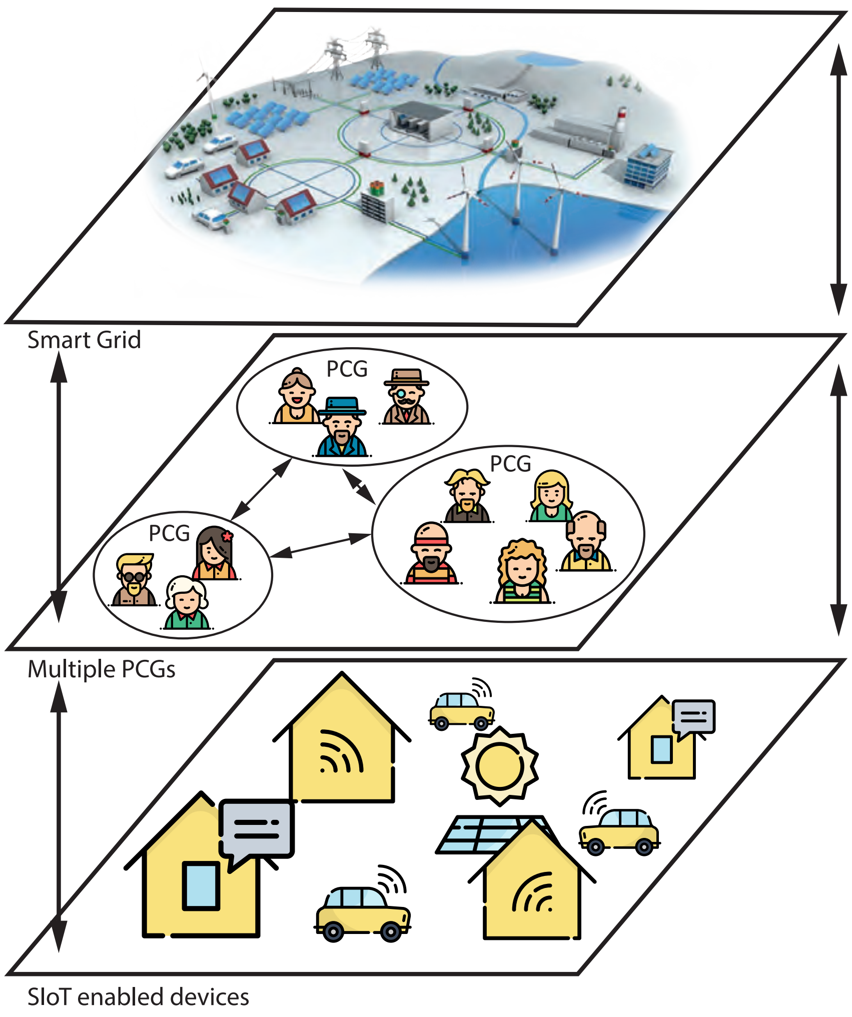


Figure 4.3: Multiple PCG structure.

- Master-slave organization: The cluster has a cluster leader or master node. The master node is responsible for running the scheduling algorithm with the information of its slave nodes. The master node has to be capable of running a scheduling algorithm within the time constraints set to achieve an optimal energy management, it should have enough resources for achieving such a task. A master-slave organization also supposes a Single Point of Failure (SPOF). The SPOF problem can be partially alleviated as the SIoT paradigm assumes the use of VOs to represent the virtual counterparts of physical devices. Those SVOs are usually allocated in the cloud as in [91], which means that devices are augmented with virtual resilience and higher computing power. Otherwise, if a complete cloud model is followed, a model where the functionalities of the master node are replaced by a component at the aggregation layer can be implemented. Following the approach implemented in Lysis [91], each SVO is able to perform its own optimization process. This information is then aggregated per PCG at the aggregation layer via a ME. This "aggregation layer" model is not strictly a master-slave organization as understood in the literature. However, both models follow the same structure. For the sake of simplicity, and to avoid adding more classifications, we have considered this “aggregation layer” approach as a master-slave organization. Disambiguation and clarifying explanations are added when necessary.

- Fully distributed: The cluster is organized without a cluster leader and all the nodes participate in the scheduling process. They do not share their information with all the nodes in the cluster but only with the ones they need to set their own scheduling. Therefore, algorithms that come into play in this fully distributed organization are mainly agent-based, asynchronous and distributed algorithms.

- Distributed computation: Another cluster organization which is not to be confused with the master-slave organization explained before (although it can be part of it) is that the nodes involved in the cluster are also part of the computation resources used by the master node. Therefore, this cluster organization refers to the parallelization of the computation and the nodes’ organization to attain such goals. For example, in [135] nodes are used to distribute the computation of a parallel genetic algorithm to optimize energy consumption in a building.

Scheduling algorithms

In order to distribute energy consumption processes over time in a way that reduces the PAR a scheduling/planning algorithm is needed. A first division of the type of algorithms considers how the knowledge is distributed among all nodes. Thus, global-knowledge algorithms and agent-based algorithms can be considered.

- Global-knowledge algorithms need all the information gathered by the PCG to extract an optimal energy plan for the members of the PCG. Therefore, there should be a component that aggregates the information, either a cluster leader (master node) or an aggregator component at the aggregation layer. Global-knowledge algorithms align with the master-slave cluster organization as all information is eventually conveyed to the master node or at the aggregator. Note that although one can think of the utility as the master node, a further subdivision into PCGs can be applied. Then, slave nodes can convey the information needed to the master node to run the algorithm and master nodes can send the information to the utility when needed.

- Agent-based and distributed algorithms account for keeping the core information in each node where each node only shares the information needed to schedule a proper plan. Those algorithms align with the fully-distributed cluster organization.

A second division considers the type of algorithm to be used. They usually consider energy efficiency and/or user comfort as a stochastic optimization problem. The explanation that follows pretends to give an overview of the types of algorithms used in DSM but is far from an exhaustive analysis of the state of the art. The reader can refer to [136] for an in-depth analysis. Appropriate references to revisions of such algorithms are provided when available. Some work analyzed in this section does not focus on load shifting *per se*, but it is relevant in that the reduction in electricity price has a relation with energy saving.

Global-knowledge DSM techniques for load shifting need the information of all ILs before running. Evolutionary Algorithms (EAs) (which include genetic algorithms) redistribute shiftable loads in order to achieve the best solution based on the objective function of load shifting and user comfort [137–139]. Particle Swarm Optimization (PSO) algorithms also prove to be successful in load shifting and scheduling of ILs [140–142].

Other works present in the literature focus on game theory to solve these optimization problems. Game-theoretic algorithms can be distributed easily. Indeed decision theory fields such as game-theory or evolutionary game-theory combined with MASs is recurrent in modern literature [143]. Moreover, MASs applied to SGs and, more concretely, to MG management and DSM is gaining momentum as a solution to give autonomicity in the management of such complex systems [144, 145]. References [143–145] support the idea that a mid-large-scale SG is likely to be managed in a cooperatively and more autonomously manner with multiple agents involved. Therefore, the SIoT paradigm presents itself as a dynamic, flexible and scalable organization solution to such approach.

A structural perspective on Demand Side Management and Social Internet of Things

So far, EA, PSO algorithms and game-theoretic algorithms in DSM contexts have been presented in order to introduce the reader to global- and local-knowledge algorithms. It is obvious that other types of algorithms have been used for DSM apart from those. The focus of this section is to analyze the implications of global-knowledge algorithms and local-knowledge algorithms or, rather, the implications of different cluster organizations in merging DSM and SIoT.

If the PCG is organized in a master-slave structure:

- The master node is responsible for running the scheduling algorithm and thus should have enough resources to compute the result.
- As the information of each member of the cluster will be conveyed to the master node, each slave node should have a direct relationship to the master node.
- Considering a possible master node failure and in order to avoid a SPOF, slave nodes must form a strongly connected component (graph theory). If the current master node goes down, a new master node can be chosen by means of a distributed quorum.
- If a similar structure is followed but with an aggregator component instead of a master node, the aggregator component may go down due software or underlying hardware malfunction. Since it is a component in the cloud, it can be automatically re-instantiated.

If the PCG is organized as a fully distributed cluster with no master node:

- Each node cooperates with other nodes in the PCG. Nodes themselves are responsible for computing their rescheduling on the basis of the information of other nodes.

- Each node sends their predicted scheduling information to a subset of nodes in their PCG. Given that load consumption can be predicted [146, 147], some nodes are more likely to be compatible with each other, meaning that they are more likely to achieve a lower PAR.
- Although the intra-cluster rescheduling computation is not done in a master-slave structure, PCGs need to communicate in order to optimize overall PAR. A master node should be elected for this purpose, but note that the failure of the node and consequently, the election of a new one, is not as resource-consuming as in the master-slave organization when it requires a master node.

Solution evaluation

Resulting solutions when evaluating the application of SIoT to the SG can be assessed in two manners. The first one involves communication-related aspects and deals with:

- The searchability of new members to include in a PCG.
- The navigability among the members that pertain to a PCG and between different PCGs.
- The performance of the communication. Special attention must be paid to the relationships between nodes of PCGs and the number of messages exchanged.
- There is also the possibility of moving the SVOs from the cloud to the edge cloud in order to reduce communication overhead. The work presented in [148] analyzes this possibility and associated challenges and shows that moving 75% of SVOs to the edge cloud reduces network overhead considerably.

The second manner of evaluation concerns the energy efficiency of the proposed solutions. As already pointed out, there are several methodologies or algorithms that help in predicting energy consumption loads and even adapt to changes. The fitness of an algorithm will be ultimately measured by its capacity to optimize energy efficiency while being able to leverage the scalability, flexibility and navigability provided by the SIoT paradigm.

4.6 Clustering Model

In this section, we will explain the model we propose in this work and future ones. According to [136], there are six (6) basic DSM services (Figure 4.4):

1. Peak Clipping: Reduction of grid load.
2. Conservation: Reduction of the load during the day by reducing overall consumption. For example, by using more energy-efficient appliances.
3. Load Building: Increase the load by increasing the overall consumption.
4. Valley Filling: Improvement of system load factors such as PAR by increasing load during off-peak periods.
5. Flexible Load Shape: Specific contracts and tariffs that allow the control of prosumer's equipment.

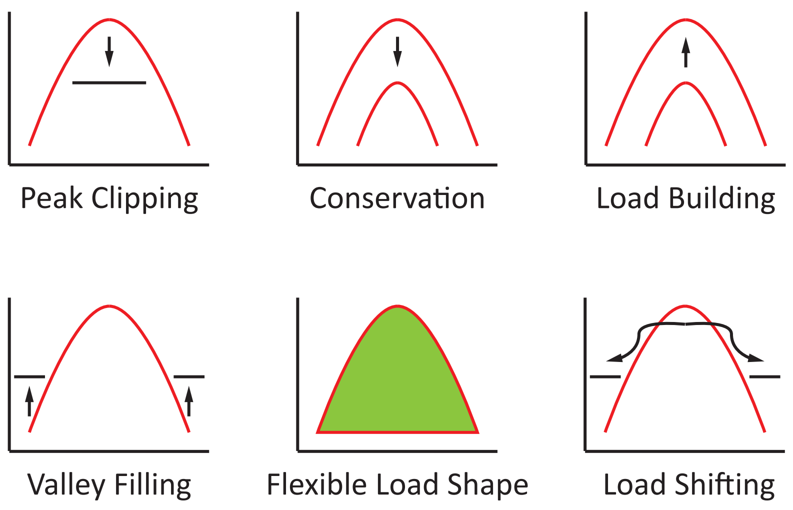


Figure 4.4: Load shifting effects. Vertical axis is power and horizontal axis is time.

6. Load Shifting: Reduction of grid load during peak demand and load building across off-peak periods.

As the purpose of DSM is to achieve those effects when desired, the granularity of the model allows the representation of each consumer's appliance as a load. Also, as the number of residential DERs increases, energy generation and storage must also be represented. Each load type should abide by certain rules, imposed by their characteristics and prosumer preferences. Some loads cannot be shifted and allow to reduce the energy they demand by changing their working mode; other loads should be treated as non-shiftable loads because the user does not allow the system to shift the load, or simply they cannot be shifted; energy generated by DERs may not be storable and a load building effect should take place to leverage the extra energy produced; and so on. By rescheduling positive loads (energy consumption) and negative loads (energy generation) better system load factors can be achieved. The DSM model that follows is the simplest one, it considers only electricity consumption and the PAR as the metric to measure the achieved optimization after clustering and rescheduling the loads. We acknowledge, however, that the DSM challenge is a multi-objective challenge. For example, only aiming to reduce the PAR may lead to an increase in the amplitude of all energy peaks without the proper policy; we address this problem in Algorithm 4.2.

In order to create PCGs or clusters, our model aims to create clusters of compatible prosumers to reduce the inter-cluster PAR by reducing the intra-cluster PAR. Let $\mathcal{U}$ be the set of users and $\mathcal{T}$ the set of time slots, where $U \triangleq |\mathcal{U}|$ and $T \triangleq |\mathcal{T}|$. The symbol $\triangleq$ indicates that letter in the left hand side is the cardinality of the set on the right hand side. For each $u \in \mathcal{U}$, we define the power consumption vector as

$$\vec{l_{u,T}} = [l_u^1, ..., l_u^t, ..., l_u^T] \tag{4.1}$$

where l_u^t is the energy consumption of user u at time slot t. As the model allows for appliance granularity, the set of appliances that belong to a user u are represented by $\mathcal{A}_u$ and $A_u \triangleq |\mathcal{A}_u|$. Then, l_u expands to the matrix

$$lm_{u_{A_u,T}} = \begin{bmatrix} lm_{1,1} & \dots & lm_{1,t} & \dots & lm_{1,T} \\ lm_{2,1} & \dots & lm_{2,t} & \dots & lm_{2,T} \\ \vdots & & \vdots & & \vdots \\ lm_{A_u,1} & \dots & lm_{A_u,t} & \dots & lm_{A_u,T} \end{bmatrix} \tag{4.2}$$

Total load per user $u \in \mathcal{U}$ during T time slots is denoted by L_u

$$L_u = \sum_{t=1}^{T} l_u^t \tag{4.3}$$

the PAR is then denoted by

$$par(u)_{u\in\mathcal{U}} = \frac{\max_{t\in\mathcal{T}} l_u^t}{\frac{1}{T} L_u} \tag{4.4}$$

and is defined in a similar manner by each appliance at a more granular level and by each cluster in a less granular level. In fact, intra-cluster consumption can be represented by a vector similar to the one presented in Equation (4.1) and then dis-aggregated as per consumer in a similar manner as in Equation (4.2).

The objective is to create clusters of prosumers as much compatible as possible and with the capability of self-managing their energy needs as a community. For the sake of simplicity, the time subindex T in $\vec{l_{u,T}}$ is omitted in the descriptions that follow. The set of load shapes is defined as $\mathcal{L}$ where $\mathcal{L} = \{\vec{l_{u1}}, ..., \vec{l_{u_i}}, ..., \vec{l_{uU}}\}$. We define the set of clusters as $\mathcal{C}$ and its cardinality as $K \triangleq |\mathcal{C}|$, where $\mathcal{C} = \{C_1, ..., C_k, ..., C_K\}$ and $C_k \subset \mathcal{L}$. All members in $\mathcal{C}$ are in the same microgrid. Moreover, members of a cluster k cannot pertain to other clusters: $\forall (C_k, C_m) \in \mathcal{C}; k \neq m : C_k \cap C_m = \varnothing$. Given two or more prosumers $l_{ux}, l_{uy}, l_{uz}, ...$ where $x \neq y \neq z$, we define their "compatibility" as how much they contribute in reducing their join PAR ($par(l_{ux} + l_{uy} + l_{uz} + ...)$). For each cluster C, it is desirable to maximize their compatibility, which means to minimize their PAR to be as closer to 1 (lowest value) as possible. Given that each cluster is a subset of $\mathcal{L}$, the compound PAR of all clusters is equal to $par(\mathcal{L})$, Equation (4.5).

$$\forall C_k \in \mathcal{C} : par(\bigcup C_k) = par(\mathcal{C}) = par(\mathcal{L}) \tag{4.5}$$

In the same manner as $\vec{l_u}$ expands to $lm_{u_{A_u,T}}$ (Equation (4.2)), a cluster $C \in \mathcal{C}$ can be collapsed by calculating the summation of all profile shapes $\vec{l_u} \in C$, that is the cluster profile shape. We will refer to the latter as $\vec{C} = \sum_{i=1}^{|C|} \vec{l_{u_i}}$ for each $\vec{l_{u_i}} \in C$. Also $\vec{\mathcal{C}} = \sum_{k=1}^{K} \vec{C_k} = \sum_{i=1}^{U} \vec{l_{u_i}}$.

4.6.1 Algorithm

This section presents the algorithm used to create consumer communities and reschedule their consumption in order to minimize the PAR. Our objective is to assess the viability of reducing the PAR per community and experiment how PAR is reduced after consumption rescheduling per community. It is, however, not in the scope of this article to simulate inter-community communication to reduce global PAR. Also, no contribution from SIoT relationships is expressed in this algorithm. The rationale is that the algorithm and the analysis of the results will serve as the reference measurements given a concrete dataset. We rely on the giant component (Section 4.4.2) as the enabler to eventually create PCGs that will include all the members of the dataset. Once reference measures and external metrics are known, we will be able to study the formation of new meaningful relationships to improve navigability for this specific use case (Demand Side Management). Future improvements are explained in Section 4.9. Note that the word *cluster* is used to refer to PCGs as this word is commonly used in algorithms, but we do not use a conventional clustering algorithm.

Given a dataset *Dataset* containing user load profiles, create K clusters ($C_k \in \mathcal{C}$) that minimize $dist(\mathcal{C})$ to make each cluster similar to one another in terms of PAR while minimizing the PAR of each one (Algorithm 4.1). We will refer to $dist(\mathcal{C})$ as the clustering heuristic. Then, for each cluster, reschedule its elements so that $par(C_k) > par(C_k^{'})$ (Algorithm 4.2). In order to get an optimal solution, Algorithm 4.1 is run $len(D) * K$ times and the solution with the lowest $dist(\mathcal{C})$ is taken. Each time Algorithm 4.1 is run, K points/seeds are randomly chosen from the initial dataset, and each seed is the initial element of the cluster C_k. Note that operations such as $\mathcal{C}$ += C_k represent the replacement of element C_k in set $\mathcal{C}$. On the contrary, operations such as c = $C_k + d$ represent adding the point d to the cluster C_k and assigning the result to c, without modifying C_k nor d (notice that there is not an equal sign in the operation).

Algorithm 4.1: First step of the algorithm.

```
Data: Dataset
Result: $\mathcal{C}$
$\mathcal{C}$ = $\mathcal{S}$ = seeds($K$, $Dataset$)  ▷ Randomly chooses $K$ points from dataset $Dataset$
foreach $d$ in $Dataset - \mathcal{S}$ do
    c = $C_1 + d$
    cSet = $\mathcal{C}$ + c
    bestDistance = $dist(cSet)$
    bestCSet = cSet
    foreach $C_k \in \mathcal{C}$ do
        c = $C_k + d$
        cSet = $\mathcal{C}$ + c
        if $bestDistance > dist(cSet)$ then
            bestDistance = $dist(cSet)$
            bestCSet = cSet
    $\mathcal{C}$ = bestCSet
```

In Algorithm 4.2, instead of applying the rescheduling process to the cluster at once using a resource-consuming algorithm, we choose to reschedule one dis-aggregated appliance ($lm_{a_u,t}$) at a time. Doing so, we pursue to avoid consuming too many resources using stochastic methods for planning and rescheduling. We envision each cluster to be able to apply load rescheduling of all of its members, and the node applying the algorithm could be a commodity hardware. The "rescheduleOnce" algorithm works as follows:

1. Given a cluster C, find which member has the highest contribution in increasing $par(C)$.

2. For all the appliances of that member, reschedule when each of them should be functioning using Algorithm 4.3 in order to reduce $par(C)$. We assume that each appliance can be freely rescheduled. Then, given an appliance, permutations for each of its elements are tried. A permutation is implemented if Algorithm 4.3 evaluates "true". The arguments passed to Algorithm 4.3 are: $\vec{C}$ as the original cluster load shape and $\vec{C}'$ as the original load shape but with a permutation in the load shape of one of its contained appliances.

3. Finally, repeat from Item 1 until there is no significant change in the value of the intra-cluster PAR.

Algorithm 4.2: Second step of the algorithm. This algorithm is applied to each $C_k \in \mathcal{C}$ and produces a new set $\mathcal{C}'$ where each cluster has a lower PAR.

```
Data: $C_k$
Result: $C_k'$
$C_k' = C_k$
clusterSize = $|C_k'|$
memory = Memory(clusterSize)
                                        ▷ An empty list of clusterSize elements
push(memory, $par(C_k')$)
i = 0
do
    if $i == clusterSize$ then
        i = 0
        reset(memory)
                                        ▷ Delete all elements on the list
    else
        i = i + 1
    end
    $C_k'$ = rescheduleOnce($C_k'$)
    push(memory, $par(C_k')$)
while $i$ != $clusterSize$ or $significantChange(memory)$
```

Algorithm 4.3: Predicts if the permutation will reduce the cluster PAR. $max(\vec{c})$ is an element-wise operation that returns the maximum value in $\vec{c}$. $count(\vec{c}, n)$ is an element-wise operation that counts how many elements equal to n are in $\vec{c}$.

```
Data: $\vec{C}, \vec{C'}$
maxIsLowerThanBefore = $max(\vec{C'}) < max(\vec{C})$
maxIsEqualAsBefore = $max(\vec{C'}) == max(\vec{C})$
lessMaxsThanBefore = $count(\vec{C'}, max(\vec{C'})) < count(\vec{C}, max(\vec{C}))$
if $lessMaxsThanBefore$ and $maxIsEqualAsBefore$ or $maxIsLowerThanBefore$ then
    true
else
    false
```

4.7 Implementation and results

4.7.1 Implementation

As our objective is to develop a strong and consistent model for future extension, the programming language chosen has been Scala [149]. Scala offers a strong and static type system, which reduces the chances of errors during runtime, as opposed to dynamically typed languages such as Python [150]. Scala also offers algebraic and statistical libraries, although the ones found in the Python community have more functionality. Therefore, Scala has been used to implement the model and algorithms explained in this article.

4.7.2 Tests and results

To test the clustering and rescheduling method proposed in Section 4.6, part of the large dataset provided by Dataport [87] is used. We have selected households which have a smart meter installed with individual circuits for each appliance. 189 enrolled households in 2015-01-01 had a smart meter (eGauge device) with individual circuits for each appliance. eGauge readings are per hour, meaning that each record contains power readings for up to 12 circuits during 24 time spans.

The objectives of the following tests are to discover how well perform Algorithms 4.1, 4.2 and 4.3. Different heuristics are tested in Algorithm 4.1. A good solution is one that distributes prosumers among several clusters and keeps the cluster PAR as low as possible. Algorithm 4.2 is then run using the output of Algorithm 4.1 as the input. The metrics used to measure how well is a solution are explained along with the meaning of each column of the table that presents the results. Tests have been run for several K, using two different $dist(\mathcal{C})$ metrics (from now on, $dist(\mathcal{C})$ metrics are referred as *aggregate metrics*). Results are presented in tables. In order to refer to each row of the table, we will take the value of K as the row index. The values in each column are the mean of several runs. The meaning of the different columns are (for consistency, the same names and associated meaning are used throughout the text):

- points per cluster: is an array enclosed by square brackets ("[]") and each value separated by a comma (",") that represents the number of points or $\vec{l_u}$ assigned to each cluster. The values of this column contain the *best* cluster configuration. Our criteria to choose the *best* cluster configuration is that it has the lowest "s1. max m" among all runs. The rationale for choosing "s1. max m" over "s1. agg m " is that the variations in the latter are almost negligible.

- s1. agg m (Step 1 Aggregate Metric): the values correspond to the result after applying the chosen $dist(\mathcal{C})$ in Algorithm 4.1.

- s1. max m (Step 1 Maximum Metric): each value corresponds to the max $par(C_k)$ after Algorithm 4.1 is applied.

- s1 peak (kW) (Step 1 peak (kW)): the values in this column correspond to the aggregate maximum energy peak max l_C^t after Algorithm 4.1 is applied.

- s2. agg m, s2. max m, s2 peak (kW) have the same meaning as their "s1" counterparts except that they are calculated after Algorithm 4.2 is applied.

- total m (Total Metric) is calculated as the $par(Dataset')$ where $d' \in Dataset' | d' \in \mathcal{C}'$ after Algorithm 4.1 and Algorithm 4.2 have been applied. That is, the PAR of all points (load shapes) as if there was only one big cluster. Note that Algorithm 4.2 only operates per cluster, so it is the case that this value increases or decreases without following any pattern.

The first aggregate metric tested is:

$$dist(\mathcal{C}) = \frac{\sum_{k=1}^{K} par(C_k)}{K} \tag{4.6}$$

That is, the average of PARs of each cluster. Results are presented in Table 4.1. The values for $K = 1$ are the same as those without applying any clustering algorithm, as there is only one big cluster, which we can take as a reference case. It can be appreciated that the PAR is 1.37 at the beginning and it decreases down to 1.28 after applying Algorithm 4.2. Also, after applying Algorithm 4.2 and due to Algorithm 4.3, energy peak lowers from 676 kW to 633 kW. However, for $K \neq 1$, the aggregate metric used is not fair when distributing points across clusters. It tends to create one big cluster and then several small ones or clusters with zero (0) elements. The following inequations hold true: "s1. max m" > "s2. max m", "s1. peak (kW)" > "s2. peak (kW)" for any number of K, which means that Algorithm 4.2 improves the PAR and reduces the energy peak of each cluster. "s1. max m" and "s2. max m" also tend to increase as K increases. As the column "total m" indicates, the real PAR remains almost the same and it improves the value for $K = 1$ sometimes. However, our objective is to create multiple PCGs and the aggregate metric used does not serve for that purpose, as it tends to group users in one single cluster.

In order to address the unfairness of the first experiment, we propose to use the following aggregate metric:

$$dist(\mathcal{C}) = par(\vec{\mathcal{C}_{par}}) | \vec{\mathcal{C}_{par}} = \forall C_k \in \mathcal{C} \rightarrow par(C_k) \tag{4.7}$$

$\vec{\mathcal{C}_{par}}$ is a vector whose components are the PAR of each one of the clusters (for each $C_k \in \mathcal{C}$, calculate $par(C_k)$). Then, $par(\vec{\mathcal{C}_{par}})$ is the PAR of that vector. PAR is calculated using Equation (4.4). The intention of the aggregate metric just presented is to equally distribute the PAR of each cluster by reducing the difference between $par(C_k)$, which also better distributes the number of users assigned to each cluster. Results are shown in Table 4.2. An inspection of the column "points per cluster" shows that the users are more equally distributed across clusters. The results in "s1. agg m" show how equal are the PARs of each cluster. In fact, for $K \neq 1$, "s1. agg m" is one (1), which means that the value present in the column "s1. max m", which is the highest PAR value among the clusters, is equal for all clusters. Again, thanks to Algorithm 4.2, PAR and the maximum energy peak is reduced for each K after its application, as shown in Figures 4.5 and 4.6. Figure 4.5 shows how the maximum aggregate PAR tends to increase as K increases as there are fewer elements per cluster and therefore, it is less likely that the elements in each cluster are able to compensate the per-cluster metric. It is, however, possible to reverse this tendency after the rescheduling process, as less rescheduling movements are able to achieve a better metric or worsen it. Figure 4.6 shows how peak power decreases after applying Algorithm 4.3, but it does not follow a specific pattern, as the main goal is to reduce the PAR. "total m" column shows that the real PAR remains almost the same. Note also that for $K = 2$ there is an improvement regarding $K = 1$, which we can take as the reference solution as it has the lowest PAR ("s1. max m" tends to increase as K increases). Clusters contain almost the same number of points, 95 and 94 respectively, PAR for both clusters is the same as for $K = 1$ and, after applying Algorithm 4.2, "total m" is lower than for $K = 1$. This solution is not only better in terms of the values presented, it is also better because Algorithm 4.2 can be run in parallel for each one of the clusters and thus alleviating the centralization of computing resources needed to apply DSM, which is one of the main objectives of this work.

4.7.3 Discussion

Two heuristics (metrics) have been applied to distribute consumers (points) across several clusters and then their energy load profiles have been rescheduled to improve the PAR per cluster. The first proposed heuristic (Equation (4.6)) tends to group the majority of consumers into one single cluster, whereas the second heuristic (Equation (4.7)) distributes consumers more fairly, meaning the number of consumers per cluster is more egalitarian. We choose the latter heuristic (Equation (4.7)) as better because our objective is to create PCGs

Table 4.1: Simulation results: Average Aggregate.

K	points per cluster	s1. agg m	s1. max m	s1. peak (kw)	s2. agg m	s2. max m	s2. peak (kw)	total m
1	[189]	1.371114	1.371114	676.160150	1.284301	1.284301	633.348667	1.284301
2	[0, 189]	1.185557	1.371114	676.160150	1.142150	1.284301	633.348667	1.284301
3	[18, 171, 0]	1.141561	1.386598	657.846300	1.108997	1.289023	611.553717	1.279029
4	[20, 0, 4, 165]	1.126272	1.394957	641.170733	1.096543	1.278069	587.445167	1.262169
5	[0, 8, 12, 153, 16]	1.125646	1.419435	609.907067	1.099749	1.302855	559.814567	1.271343
6	[0, 25, 11, 138, 11, 4]	1.122694	1.430808	586.851050	1.088956	1.255731	515.042300	1.224344

Table 4.2: Simulation results: PAR Aggregate.

K	points per cluster	s1. agg m	s1. max m	s1. peak (kw)	s2. agg m	s2. max m	s2. peak (kw)	total m
1	[189]	1.371114	1.371114	676.160150	1.284301	1.284301	633.348667	1.284301
2	[95, 94]	1.000000	1.371114	341.647200	1.009352	1.287300	320.762883	1.275193
3	[42, 34, 113]	1.000007	1.461032	492.058050	1.021967	1.316365	443.335850	1.303961
4	[57, 63, 41, 28]	1.000127	1.485290	265.469917	1.049070	1.390578	240.844383	1.320364
5	[50, 21, 40, 46, 32]	1.000355	1.637695	243.410700	1.077833	1.426521	190.383117	1.308784
6	[33, 13, 71, 29, 24, 19]	1.010122	1.726447	383.596300	1.063379	1.415940	294.851850	1.298971

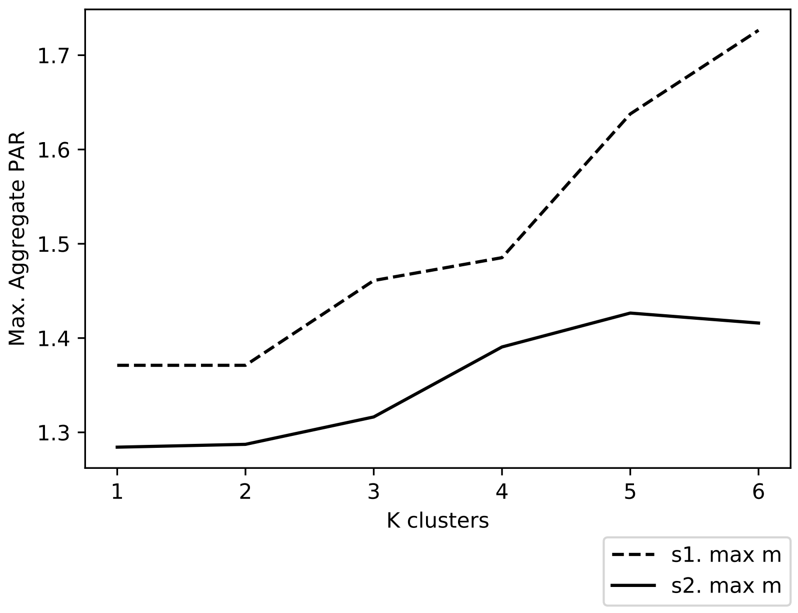


Figure 4.5: Comparative between Maximum Aggregate PAR before (dashed line) and after (solid line) applying the rescheduling algorithm. Values (Y-axis) per each K (X-axis) are computed using Equation (4.7).

that are able to perform an auto-rescheduling process in order to reduce both the intra-cluster PAR and maximum energy peak (kW). Using the dataset provided by Dataport to evaluate the algorithms, we found that the best solution is obtained when $K = 2$ and Equation (4.7) is used as the aggregate metric, which not only improves the reference case for $K = 1$ but also allows to distribute computation resources. The prediction heuristic (Algorithm 4.3) used to decrease the PAR and maximum energy peak in Algorithm 4.2 has proven to work successfully. However, the rescheduled loads obtained are not optimal due to the implementation, as it only uses a linear heuristic to choose the elements to reschedule,

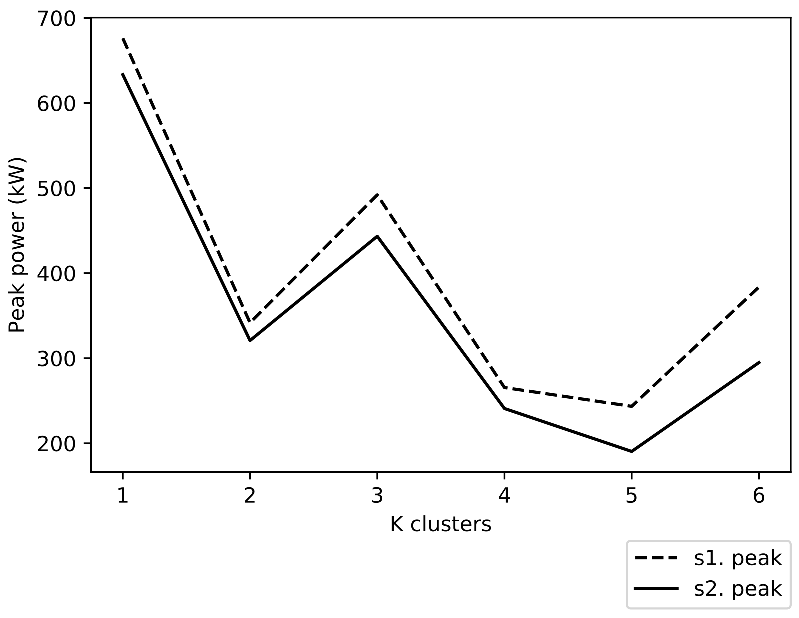


Figure 4.6: Comparative between Peak power (kW) before (dashed line) and after (solid line) applying the rescheduling algorithm. Values (Y-axis) per each K (X-axis) are computed using Equation (4.7).

avoiding brute force and thus reducing computation costs.

4.8 Hierarchical clustering

From the conclusions drawn in Section 4.7.3 and in order to improve Algorithm 4.1, we have developed a hierarchical clustering algorithm that supersedes Algorithm 4.1. Next sections explain the rationale of the algorithm and perform some experiments.

We are interested in finding and grouping $(\vec{l_{u_x}}, \vec{l_{u_y}}) \in \mathcal{L}; x \neq y : \vec{l_{u_x}} + \vec{l_{u_y}} = 0$. The justification is that $par(\vec{l_{u_x}} + \vec{l_{u_y}}) = 1$, which is the lowest possible value of PAR. In order to do so, we first need to define the *centroid* operation.

Given a generic set $\mathcal{S}$ and $\vec{s} \in \mathcal{S}$, the centroid operation is defined as:

$$centroid(\mathcal{S}) = \frac{1}{|\mathcal{S}|} \sum_{i=1}^{|\mathcal{S}|} \vec{s_i} \tag{4.8}$$

Our objective is now to find a virtual mirror image $\vec{l'_{u_y}}$ for $\vec{l_{u_x}} \in \mathcal{L}$. Without losing generality, the mirror image for $s \in \mathcal{S}$ is calculated as shown in Equation (4.9). Equation (4.9) reads as: given the direction and length from $centroid(\mathcal{S})$ to s, find $mirror\_image(s)$ which is at the same length from $centroid(\mathcal{S})$ but in opposite direction.

$$mirror\_image(s) = \vec{l'_{u_y}} = 2 * centroid(\mathcal{S}) - s \tag{4.9}$$

To find a real mirror image $\vec{l_{u_y}} \in \mathcal{L}$ for $\vec{l_{u_x}} \in \mathcal{L}$, we apply the k-nearest-neighbour algorithm to $\vec{l'_{u_y}}$ in the set $\mathcal{L}$ where $k = 1$ (1-nearest-neighbour). Also, in order to limit the size of the cluster in terms of total energy consumption, we have added the following constraint:

only add $\vec{l_{u_y}}$ the to cluster if the total energy consumption of the cluster when $\vec{l_{u_y}}$ is added is below a threshold. We expect this threshold to be set by the utility. As we do not have any specific constraint, we have calculated the threshold value as the mean energy consumption per member in C. Given that C is a cluster where $\vec{l_{u_y}} \notin C$ and $C^{'}$ is the same cluster with $\vec{l_{u_y}}$, add $\vec{l_{u_y}}$ to C only if the average energy consumption per member in $C^{'}$ is less or equal to the average energy consumption per member in C. Stop adding members to the cluster otherwise. After obtaining the clusters, they are grouped using the same procedure without the total energy consumption restriction. The grouping process stops when the desired number of clusters K is met.

4.8.1 Tests and results

This section presents the results obtained when applying the hierarchical clustering heuristic just presented. The objective is to compare the results with the ones obtained in Table 4.2. However, we do not apply a rescheduling algorithm. The rescheduling algorithm implemented in this work still does not use inter-cluster communication. This means that when rescheduling is applied, it achieves a local solution, rather than a global one and thus it is possible that it worsens the $par(Dataset^{'})$. In this new hierarchical clustering, the number of leaf clusters is higher than in Algorithm 4.1. The lack of inter-cluster communication produces a larger negative impact than in the experiments performed in Section 4.7. We have performed experiments taking samples from 50% to 100% of the total dataset at 10% steps. The sampling procedure from 50% to 100% shuffles the entire dataset and grabs the desired number of samples. Then the algorithm is applied. The procedure is repeated multiple times and the results presented are the mean of all runs. The results presented in Table 4.3 are using the 100% of the dataset. They clearly outperform the results obtained in Table 4.2. If we look at the column "s1. agg m", it increases slightly more rapidly using hierarchical clustering, but the increases are negligible. "s1 max m" and "s1. peak" both outperform the results in Table 4.2. The increment of "s1. max m" from $K = 1$ to $K = 6$, for Table 4.2 is 26% and for Table 4.3 is 8%. This means that the degree on which "s1. max m" increases related to the increase in the number of clusters is lower using the proposed heuristic. Also, if we inspect "s1. peak (kw)", values in Table 4.2 don't show any specific pattern, while values in Table 4.3 decrease as the number of clusters grows. This occurs as expected: if the number of clusters grows, members per cluster decrease and thus the peak power of the cluster decreases.

In Figure 4.7 it can be appreciated that as the number of available prosumers grows, "s1. max m" decreases. This occurs as expected: if there are less available prosumers to cluster, there are fewer chances of finding a prosumer closer to $\vec{l^{'}_{u_y}}$. This reasoning is also reflected in "s1. agg m", as it slightly decreases as the sample size grows.

4.9 Conclusion and Future work

Electrical energy demand is increasing rapidly and is usually met by non-renewable energy resources which are starting to be scarce, and thus novel solutions empowered by the integration of ICT and the electric power system, the SG, are emerging. The SG challenge can be studied as a particular case of the wider and broader IoT challenge, with its own specific needs. It is envisioned that DERs will be an integral part of the new electrical grid. In this work, we have presented our vision of the integration of the SG and a novel IoT paradigm, the SIoT. The term Social Internet of Energy (SIoE) is used to refer to this synergy. It allows

Table 4.3: Simulation results: Hierarchical clustering

K	points per cluster	s1. agg m	s1. max m	s1. peak (kw)
1	[189]	1.371114	1.371114	676.160150
2	[111, 78]	1.000665	1.374143	408.915451
3	[90, 60, 39]	1.006628	1.397967	277.239788
4	[35, 51, 51, 52]	1.015163	1.429992	226.177060
5	[81, 53, 33, 17, 5]	1.026647	1.462162	201.312466
6	[27, 23, 21, 24, 35, 59]	1.034796	1.492908	173.663762

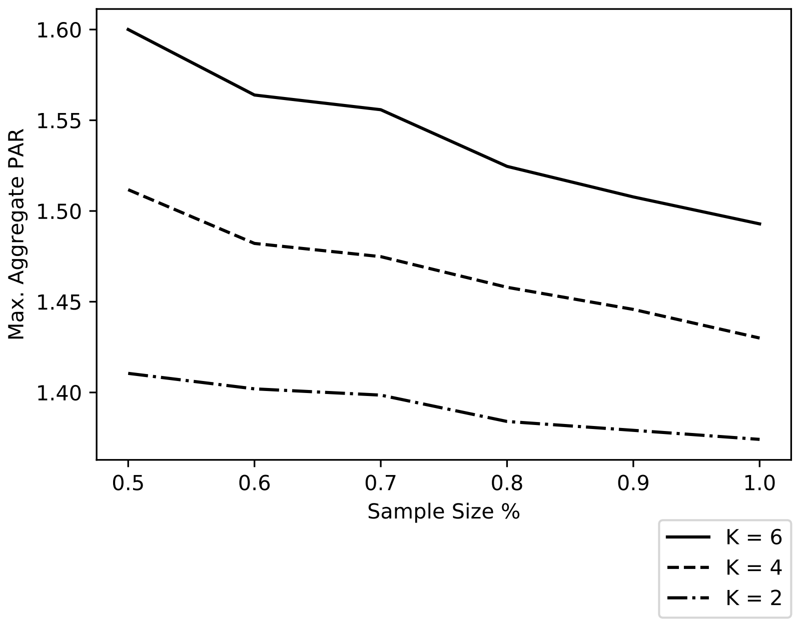


Figure 4.7: Comparative between "s1 max m" for different sample sizes when K is 2, 4 and 6.

physical devices that consume or produce energy to create social relationships to improve overall scalability. At the same time, it allows prosumers in a SG to create social relationships in order to optimize their energy usage. It has been explained how SIoT relationships can help in integrating multiple energy consuming devices both at the owner level, by leveraging Ownership Object Relationship, and at the community level, by leveraging Social Object Relationship and Co-work Object Relationship. We have started to develop a model in order to test algorithms and heuristics that aim at evaluating the viability of the proposal. Evaluation tests conducted in this regard have the objective of finding a heuristic to create PCGs and then reschedule those groups individually. A publicly available dataset has been used to perform the tests. From two proposed heuristics to assign prosumers to each PCG, one has been found to distribute them in a fair manner. A PAR prediction heuristic has been used in order to perform the rescheduling process, which has been proven to reduce both PAR and the amplitude of energy peaks. From the conclusions drawn in those experiments, we have developed a new hierarchical clustering algorithm that outperforms the previous one. Having more than one PCG implies that the rescheduling process, which is an individual process for each PCG, can be performed in parallel. Although there is a long pathway in order to evaluate the proposal explained in this article, the results and the associated discussion presented in Sections 4.7 and 4.8 will serve as the basis for our future work. Those experiments are valuable in that they have served to obtain mechanisms to evaluate object similarity (compatibility) and egalitarian cluster distribution. They also show, given a specific scenario, the values expected for different metrics regarding energy efficiency. Once this is known, and using network modelling tools, we can simulate and explore when are those groups and networks created and how their structure is. Those analyses will serve to analyze and improve navigability and search performance by creating new connections if necessary. They will also be used to include dynamic allocation and modification of clusters/PCGs. Future work aims at:

- Improving the model, not only theoretically, but the implementation (code) that supports the algorithms, by adding: ILs, generation, and storage models and the capacity to add rules per prosumer and appliance.
- Improving network structure by including energy profile compatibility metrics during its creation and modification.
- Developing new dynamic clustering algorithms to update clusters on the basis of changes on energy profiles and prosumer-related information.
- Once the proper algorithms and heuristics are found, the next step will be to apply MAS theory and game theoretic algorithms to distribute the computations among prosumer devices (SVOs).

5 Results

This chapter presents and summarizes the results obtained in each publication [39–41]. For detailed results of each article, refer to the respective source. Then, the results are discussed and related to the objectives posed in this dissertation (Section 1.2.2):

5.1 Prototyping a Web-of-Energy Architecture for Smart Integration of Sensor Networks in Smart Grids Domain

This work [39] has a two objectives: (i) to present and analyze the state of the art between SG and WoT and, (ii) to build a prototype of the WoT architecture applied to the SG using the previous analysis. Consequently, it analyzes the main challenges before the realization of a SG and the characteristics and enabling features and architecture of the WoT to solve SG challenge. The analysis results in an extensive understanding of the WoT and the SG challenges the WoT can contribute to overcome. The authors refer to the synergy between SG and WoT with the term WoE. Then, the analysis is materialized by building a WoE architecture prototype. The prototype is created using the Actor Model, which is envisioned as a programming paradigm that not only is well fitted for the SG due to its characteristics of self-healing and flexibility, but for the challenges exposed by the IoT. The Actor Model is used to create VOs. Each VO is represented by at least one actor: a lightweight, reactive agent. Heterogeneity is addressed in the system by the capacity of each implemented actor to translate heterogeneous protocols (MQTT, for example) to web protocols (HTTP/S or WS). Using the Actor Model also renders the system flexible. One of the three proprieties of actors is that when they receive a new message can decide to deploy a new actor or terminate an existing one, which means that flexibility and scalability within the system is inherited from the Actor Model.

Quantitative performance tests based on real datasets show that the prototype meets some of the time needs of time-critical SG-ICT applications when VOs aggregate several smart meters. In such cases, the timedelta (between dispatching and receiving messages) mean slope remains neutral even at bursts of 9000 aggregated messages at a 1-s interval between bursts. Experiments also show that the system is suitable for non-time-critical SG traffic when messages cannot be aggregated before reaching the prototype system. Also, a simple web-based interface to visualize sensor location and sensor readings of 1000 devices in near real-time is implemented as the WoE interface.

5.2 Ontology-Defined Middleware for Internet of Things Architectures

Given the heterogeneity that characterizes the IoT, a novel middleware-agnostic approach that allows for describing and executing the behaviour of devices is proposed and implemented in [40]. The objective is to allow the reusability and shareability of the execution flow among multiple and heterogeneous IoT deployments. The approach relies heavily on existing standards to promote interoperability and reusability. FSMs are used as the model to create the execution flow using web ontologies. The work is contextualized in a reference architecture recommended by W3C, the W3C WoT Architecture. It uses the concept of VO as the computational entities that run the concrete instances of FSM templates. FSM templates define the execution flow of a IoT deployment using semantic web technologies. The templates allow reusing behaviours for heterogeneous physical devices with the same set or subset of capabilities.

Use cases are qualitatively analyzed to asses the viability and suitability of the proposed solution. Relying on standards such as RDF, SPARQL and HTTP/S has some drawbacks. They tend to be more heavyweight than an ad-hoc solution. For that reason, our approach is not well suited for time-critical applications such as time-critical SG-ICT applications or for monitoring and reasoning over patients' vital signs in a hospital. However, the approach is well-suited for IoT scenarios that are non-time-critical and with a low level of variability between each deployment. Hospital equipment tracking or learning enhancement environments are examples of non-time-critical deployments. The approach is useful as it allows to abstract over the (potentially) heterogeneous devices between diverse deployments for the same scenario.

5.3 Social Internet of Energy - A new paradigm for Demand Side Management

The work on [41] focuses on integrating a novel IoT paradigm, the SIoT with PCGs. The work has two objectives: (i) to present and analyze the state of the art of SIoT and, (ii) to integrate the SIoT with new and valuable SG agents, prosumers, using the previous analysis. Novel studies consider groups of prosumers as entities that can be more valuable in the SG. Therefore the approach described in the article focuses on PCGs and a specific goal related to DSM. The work describes how SIoT relationships can help in integrating multiple energy-consuming devices both at the owner level and at the community level. A new term, the SIoE is coined to refer to the synergy between SIoT and SG. The manuscript explores the proprieties of the SIoE regarding the specific application use case for PCGs and DSM. The proprieties explored include cluster/group organization, scheduling algorithms for DSM available in the literature and indicators for evaluating the viability of the proposal.

Then, the authors develop a first part of the model to test algorithms and heuristics that aim at evaluating the viability of the proposal. Evaluation tests conducted in this regard have the objective of finding a heuristic to create PCGs and then reschedule those groups individually. A publicly available dataset is used to perform quantitative tests. From two proposed heuristics to assign prosumers to each PCG, only one of them is found to distribute them fairly. A PAR prediction heuristic is used to perform the rescheduling process, which is proven to reduce both PAR and the amplitude of energy peaks. From the conclusions drawn in those experiments, the authors develop a new hierarchical clustering algorithm that outperforms the previous one. The final results achieve distributed and egalitarian PCGs without using a brute-force algorithm. Indeed, the improved hierarchical algorithm, besides achieving better quantitative results in terms of DSM-related measurements, is computationally faster; the goal is embedded in the manner in which to search members for a PCG. Having more than one PCG implies that the rescheduling process, which is an individual process for each PCG, can be performed in parallel. Those experiments are valuable in that they serve to obtain mechanisms to evaluate object similarity (compatibility) and egalitarian cluster distribution. They also show, given a specific scenario, the values expected for different metrics regarding energy efficiency.

5.4 Discussion

The authors perform a comparative and exhaustive analysis on WoT architecture in correlation with challenges of SGs in [39]. To contribute to solving some challenges posed by the SG, a WoE architecture is proposed, and a prototype system based on the architecture is built. Functionality tests show that the system is reliable for some time-critical SG applications. Also, the authors consider using the Actor Model, as the programming model to overcome part of the challenges posed by the SG such as the proprieties of flexibility and self-healing. Few works in the literature consider using the Actor Model as the programming paradigm for the IoT. Yet, its inherent properties seem to be perfectly suited for most of the applications derived from the IoT. The work in [39] contributes to the proofs and rationales that the Actor Model is well suited for developing IoT applications. The prototype system partially tackles the scalability challenge via the use of the Actor Model. Scalability in terms of interoperability and homogenization of device description is tackled by semantic web technologies. Nevertheless, the scalability in terms of performance of the semantic web is still facing major challenges [151]. Therefore, the authors shift their focus on another complementary technology, which is envisioned to provide better scalability.

The work in [40] derives from research in the state of the art done in [39]. The authors deepen into semantic web technologies and find a research gap in using those technologies to directly provide an interpretable and machine-readable structure to describe the control flow of WoT-enabled smart devices. Moreover, the objective in creating that structure is its ability to be reused and shared among IoT-WoT deployments. It is analogous and a research step towards flow composition applications such as ThingWorx Composer [67], NODE-Red [68] from IBM, Octoblu [60] or IFTTT [69]. The most differential trait is that it brings semantic web technologies closer to the end-user. The authors also use the Actor Model to deploy lightweight VOs that provide an HTTP/S interface. The authors perform a qualitative analysis of the solution presented in the manuscript. Relying on semantic web technologies has its drawbacks. They are more heavyweight than ad-hoc solutions. The approach is not well suited for time-critical applications. Nevertheless, it is well-suited for IoT scenarios that are non-time-critical and with a low level of variability between each deployment. The goal pursued in this proposal is to facilitate deployments in similar IoT scenarios, which is achieved using web ontologies to describe control flows to be universally interpreted. Nevertheless, the task of creating the template ontology can be tedious, especially if the template has multiple states and actions. Further work to reduce this hindrance must be done. Also, VO are not fully described by a TD interface. Describing VOs using a TD interface increases interoperability and automation, as VOs expose their capabilities in a machine-readable manner.

To the best of our knowledge, there is no work applying the SIoT paradigm to the SGs. In [41], the authors present a specific use case related to prosumers in the SG. The use case is based on the novel concept of PCGs. PCGs are expected to add even more value to prosumers in a SG. The hypothesis is that in such specific use case, the SG can benefit from the features of a novel IoT paradigm called SIoT. The work presents a non-comprehensive but exhaustive analysis on WoT to SIoT evolution and the proprieties of SIoT. Then, with reference to the PCG use case described, the authors analyze the contributions of integrating the SIoT with SGs and DSM. The term SIoE is coined to refer to this integration. Consequently, after proposing the integration between the technology and the field of application, the authors start to study its feasibility empirically. As a starting point, they aim to group prosumers based on a common goal. They develop an initial model and algorithm to group prosumers in PCGs based on their compatibility. Experimentation and derived results are successful in demonstrating the viability of the proposal. Nevertheless, a lot more research has to be conducted to detail the proposal further and to demonstrate its viability as a whole.

5.4.1 Accomplishment of research objectives

The objective **O1** aims at applying the research methodology in every scientific contribution contained in this dissertation. The research methodology encompasses several research methods. Exploratory methods in the form of a review of the state of the art are included in [39–41]. Modelling methods are used in [39–41]. Whereas [39, 40] model architectures and software components, [41] presents a formal model (part of which is already found in reviewed literature) and develops several algorithms. Prototypes are built in [39, 40]; a prototype of the WoE architecture and a prototype of an ontology-defined middleware are created respectively. The WoE architecture is tested quantitatively, and the prototype of an ontology-defined middleware is tested qualitatively. [41] uses the simulation method to extract quantitative results on the performance of the algorithms.

The objective of **O2** is the exploration of the state of the art regarding the domains approached in this dissertation. Overall, the research is aimed at integrating novel ICT technologies (IoT, WoT, and SIoT) with the SG and Future Grid. One of the main objectives of works [39, 41] is to present the state of the art regarding WoT [24] and SIoT [28, 29] in regards with SGs. Whereas the synergy between web technologies and SGs is studied in the literature, there is no study considering SIoT and SG to the best of our knowledge. The authors also perform a state-of-the-art analysis on semantic web ontologies [30, 31] to describe services and controls of flow in [40]. Then, the authors leverage the ontologies analysed to promote re-usability and shareability of control flows.

In relation to Cisco's IoT World Forum Reference Model depicted in Figure 1.1, both technologies, WoT and SIoT are concerned with the Data Abstraction Layer and are complementary to each other. Nevertheless, both consider, at least, the layers of Physical Devices, Connectivity and Edge Computing.

The WoT describes gateways as adaptors from heterogeneous communication protocols to protocols of the web (HTTP/S) and heterogeneous data structures to a homogeneous one: semantic web technologies. Consequently, it describes elements at the Edge Computing layer (gateways) and their connection with physical devices. Transformation of heterogeneous data to representations as instances of web ontologies can also be considered as a task for the Data Abstraction Layer. Upper layers of the WoT, findability, sharing and composition layers, allow the creation of applications which add value (business-case specific) to the accessible data.

The SIoT is more concerned with the scalability of IoT. It considers both heterogeneous protocols and homogeneous ones (HTTP/S). The use of each type is specific to the level in which the SIoT framework is used. For example, a short-range protocol such as Bluetooth might be used to advertise (to a higher-level system) that two individuals have come close together, and create a SOR. On the other hand, devices owned by the same individual might create a OOR with the owner by the owner having to register those devices to a web service; therefore, relationships are not created using short-range protocols but HTTP/S and the internal protocols the service uses. The SIoT considers SVOs as the virtual representations (Data Abstraction layer) of physical devices, enabling social functionalities (e.g. create and modify relationships and store them or executing algorithms to find a required service by navigating the social graph). The WoT is complementary to the SIoT by providing homogeneous accessibility features such as the use of HTTP/S or the description of SVOs (and the physical devices they represent) using semantic web technologies.

The aim of objective **O3** is to provide the SG with scalable and interoperable technologies. The characteristic of interoperability is considered in [39–41]. [39] harnesses the WoT (including semantic web technologies) to facilitate access to sensors and actuators. On the one hand, web protocols such as HTTP/S and WS are the homogenising technologies in regards to communication protocols. On the other hand, semantic web technologies allow describing devices using the same description framework. Web protocols and semantic web technologies are also crucial in [40] as the technologies to create shareable and reusable templates. The SIoE explored in [41] also considers the use of web technologies to enable interoperability between SVOs. The property of scalability is often attached to the web, although, as already explained, it does not address the challenge of scalability fully. In this regard, works [39–41] consider web technologies (WoT) to promote scalability. The SIoT, however, is expected to be an advancement in solving the challenge of scalability. Consequently, [41] harnesses the SIoT in favour of a specific function of the SG.

Overall, the authors have accomplished the concretisations of the objectives as expressed in Section 1.2.2. The authors have published 3 articles in first-tier international journals. For each one of them, they have used the research methodology effectively; first, focusing on an enhancing technology for the IoT considering improvements in scalability or interoperability; second, by exploring the state of the art in regards to the technology; and third, by applying the technology to the field of SG through experimentation.

6 Conclusion and Further Work

6.1 The relationship between scalability and interoperability

The consideration of a scalable system — considering the case of a true, global and pervasive IoT that connects anything — implies some interconnection between each object encompassed in the system as it grows. Otherwise, there would not be a system which is scalable, but separate systems completely isolated from each other. Even if IoT systems are unintentionally defined in silos and are not easily and homogeneously interoperable, they conform a higher-level system. It is then the task of tech-professionals to adapt and provide ad-hoc solutions for better integration of those silos with each other to, ultimately, build a more interoperable system of systems.

Non-interoperable solutions are usually not reusable across other systems and for other use cases in comparison, for instance, with web microservices [152, 153]. Everything that stems from a non-reusable solution, is likely to be non-reusable unless a new course of action is taken (solutions that aim at achieving requirements that prevent them from being conceived and originated from reusable solutions are out of the scope of this dissertation). Therefore, there are huge efforts towards describing sets of technologies and standards aiming at interoperability and reusability of systems. Web technologies seem one of the most viable solutions to the interoperability challenge, specifically to semantic interoperability [154, 155]. Consequently, the WoT and semantic web technologies are the main drivers for building interoperable systems. As a consequence and considering the SG field of application, the work done in [39] describes an WoT architecture for the SG, named WoE. It considers interoperability among SG devices and performance requirements. The authors perform an analysis on the WoT architceture and relate its enabling features with the functionalities needed by the SG. The authors propose a WoE architecture that fosters interoperability among devices or things using common protocols used in the WoT and IoT, which are also analyzed. They rely upon the Actor Model, a model for distributed concurrent computing to provide of scalable performance to the proposed architecture. Based on the proposed architecture, the authors build a prototype of a WoE. They perform quantitative analyses on this prototype's performance. They perform two types of experiments, the first one considers devices that send, individually, messages through the prototype; the second experiment varies in that those messages are aggregated as if there was a gateway between the prototype and the physical devices aggregating clusters of devices. While the former experiment shows that it is not suitable for time-critical SG applications, the latter does. Aggregating data in a single message reduces the number of messages sent over the wire; thus, latency is reduced.

From the knowledge and experience gathered on [39], the authors deepen on semantic interoperability in [40]. They observe that semantic web ontologies are widely used to enable machine-readable descriptions between a comprehensive range of domains. Indeed semantic web ontologies were conceived to describe any knowledge domain. Nonetheless, few works aimed at using ontologies to describe the precise steps of a process exist in the literature. Based on an ontology to describe FSMs, they work towards a method to define a machine-readable flow of execution for a given process, with the purpose to be interpreted and executed by generic software. The objective is to allow users to describe reusable and shareable flows of execution. They use the Actor Model used in [39] to dynamically deploy new VOs, whose flow of execution is described by web ontologies. They also rely on the connectivity technologies of the WoT to enable connectivity interoperability. Finally, the authors perform a qualitative analysis of integrating their solution in different scenarios. They conclude that due to the slowness of the technologies used, it is not well suited for time-critical applications. However, it can ease the deployment of multiple VOs with the same flow of execution in non-time-critical scenarios.

Scalability implies that systems keep gracefully performing even if the number of objects they encompass grows; and the number of objects in a global IoT, whether if we consider objects as systems or objects as devices, is certainly growing at an unbelievable rate. Scalable discovery of dynamic and fast-changing data is difficult [32]. Indeed, companies and organisations with huge amounts of resources achieve managing and discovering large amounts of information with ad-hoc systems (e.g. Facebook and the huge number of user profiles it manages across the globe). Those companies and their fascinating engineering solutions represent a great landscape of technological advancements in managing such amount of data. Yet, scalability and performance challenges still arise in those companies, and they are continually striving in solving them. A true global IoT not only encompasses the amount of information included in those companies but much more. One only has to realise that the number of Internet-connected devices a single owner owns is grater than 3 and that predictions say that this number will outgrow to 9 by 2025 [156]. Whether predictions are more or less accurate, the number of connected devices is growing fast.

Enter the SIoT, a novel IoT paradigm that changes the perspective from requiring an external discovery service, to an ego-centric one. Each node in the SIoT is the starting point for discovering new and useful services, and its friends in the first social ring the first ones the node is going to ask for those services, and so on. The search process is no longer started from an external, centralised node, but from the requesting node itself. Advanced solutions such as distributed sharding for better performance, considers physical geolocation and other features to increase search performance for vast amounts of data; but they stem from traditional database structures adapted to higher performance and scalability needs. The basis on which the SIoT is conceived shifts this philosophy completely. It is not to say that such databases cannot support the SIoT, but that the structure of the majority of them is not the optimal fit and may pose a research challenge. Consequently, the SG as an use case of the IoT, can harness the scalability properties of the SIoT.

On [41] the authors focus on new agents, prosumers, and a specific goal towards DSM. The growth in the number of devices owned by the same user also includes electric appliances. Recent literature on managing and empowering prosumers has coined the term of PCGs, which are groups of prosumers pursuing the same goal. The act of grouping them and managing them implies new functionalities for computational agents in the SG. This task demands solutions that should be scalable, as the number of prosumers and owned appliances will grow. Also, prosumers are expected to take valuable and proactive action in the SG and thus call for user-centric solutions. The authors, under the term SIoE, describe how each property of the SIoT can be harnessed to enhance functionalities in the SG. The authors explore how to enhance connectivity between prosumers and devices using SIoT relationships and develop an algorithm to group prosumers that pursue a mutual DSM goal.

6.2 Research objectives and research questions

This thesis focuses on the integration between ICT technologies (IoT, WoT, SIoT) and SGs. The SG itself is a solution to a challenge of sustainable energy management, as the energy demand is starting to grow faster than ever. The main contributions of this thesis by compilation of manuscripts are comprised in [39–41]. They all contribute to the integration of middleware technologies with the SG, with the specific objective of bringing scalable interoperability among its agents (**O3**). Some do that directly [39, 41], while others expand their scope and result in a technological output with a much broader field of application [40].

As in every research process, all manuscripts contained in this thesis explore the state of the art in regards to their respective technologies (**O2**). The authors explore the web technologies that conform the WoT in [39]. They also explore the characteristics of a SG. From these explorations and the ongoing research experience, the authors become aware that enabling a true SG is not possible by only tackling web technologies, although they are a big step towards the Future Grid. Rather, they have to search for complementary solutions. In [41], the authors follow a similar approach as in [39]. They focus on a specific agent in the SG, the prosumer, provide an analysis of the SG, and relate the needs for scalability and connectivity of PCG with features of the SIoT. Finally, the authors on [40] provide a state of the art analysis on a specific technology of the semantic web: web ontologies that aim at providing semantic interoperability between device communication. All works focus on interoperability and scalability for a global IoT, or SG as a special case of IoT.

Furthermore, a combination of research methods, including exploration of the associated literature, modelling of proposed architectures and modelling of algorithms, and prototype building for qualitative or quantitative analysis have been used (**O1**).

The accomplishment of these objectives provides enough information and leads to answer the research questions posed at the beginning of this dissertation. The first research question (**Q1**) formulates:

> *Can SGs and, more concretely DSM, benefit from WoT and SIoT in regards to space-time scalability and interoperability?*

After the work performed in this dissertation, the answer is affirmative. The SG is considered a specific deployment of the IoT. Two main properties and challenges of the IoT are scalability and interoperability. Consequently, these two properties (and challenges) are inherited in the SG. Through harnessing novel IoT paradigms, all three works [39–41] deal with those properties. More specifically:

- [39] brings scalability and interoperability to SG via the WoE, which includes harnessing the WoT (and semantic web) and the Actor Model. Semantic web technologies and web protocols provide interoperability, whereas scalability is considered using both web technologies and the Actor Model. Results show that the prototype build using the architecture proposed is capable of transmitting up to 9000 messages in less than a mean of 600 ms, which is suitable for some of the time-critical applications in the SG. Also, a web visual interface is built to demonstrate real-time visualisation of sensor data.

- [40] focuses on interoperability, reusability and shareability. The work mainly focuses on interoperability using semantic web ontologies. The work transcends the scope of the SG and does not relate to it specifically; however, a qualitative use case analysis considering SG is performed considering the use case presented in [41]. The solution developed in this work is not well suited for time-critical applications as the technologies used in the approach are still maturing are currently heavyweight. Nevertheless, the SG-related use case analysed does not require immediate responses (non-time-critical application), and thus the approach is well suited for the use case.

- [41] is a proposal to bring the scalability of the SIoT to SG and coins its synergy with the term SIoE. The work considers the use case of a DSM service concerning groups of prosumers (PCGs). The proposal describes how prosumers can harness properties of the SIoT, such as thing-to-thing relationships and trust mechanisms between things and their interactions. The WoT/WoE is also present in those interactions to provide interoperability in communications between things. Subsequently, the work performed in [39, 40] is complementary to the proposal.

And the second research question (Q2) formulates:

Can prosumers in a SG benefit from WoT and SIoT when aiming to improve energy sustainability?

The answer is partially affirmative. On the one hand, prosumers, as new agents in the SG, are expected to adopt IoT-related technologies since those technologies will be part of the SG and aim at easing the interaction between prosumers and the SG. Therefore, the rationale exposed in relation to Q1 apply in this response. On the other hand, and considering the SIoE, this question remains partially answered. One of the objectives of DSM services is to improve energy sustainability by reducing the use of energy resources and therefore reduce the emission of greenhouse gas. Prosumers, situated at the demand side, are expected to have a crucial role in the SG. The work performed in [41] elaborates a proposal to integrate the SIoT with prosumers (PCGs) and the use case of DSM. The [41] considers energy consumption models and develops algorithms to advance in the validation of the proposal. The results of the simulations are a step towards the validation of the proposal. However, the proposal is still in its infancy, and there is still not enough evidence to provide a conclusive answer.

6.3 A note on the future of MAS, the Web and SIoT

Full integration between the Web and MAS is gaining traction. At the early years of the Web, Web and MAS were only juxtaposed: they integrated with MAS using Web protocols (HTTP/S) to communicate. Nevertheless, the vision of full integration comprehends the merge between MAS and the whole architecture of the Web. The Web is expected to provide an environment for agents and non-autonomous entities and information. The evolution of the Web is pushing towards an interlinked fabric of information (semantic web technologies, LD) and entities (agents), interacting with and acting on this information [157]. The work presented in this thesis is related to this vision, although it does not directly build on this vision. However, we find that it glimpses some of the building blocks for a MAS integration with the Web. In the following rationale, we explain in which aspects our work relates to the integration between MAS and the Web. We also highlight that some challenges that arise from integrating MAS and the Web can be potentially addressed from the perspective of the SIoT.

Firstly, we relate our work with the Actor Model, reactive agents, and its integration (not juxtaposition) with the Web. The Actor Model was conceived to model distributed and concurrent systems of artificial intelligence. An actor in this model is a computation agent; a first-class citizen that operates concurrently and asynchronously. In works [39] and [40], we use actors and integrate them with web technologies. In [39], we not only use the HTTP/S protocol as a transport mechanism, but also provide a REST HTTP/S API. Furthermore, in [40] we integrate the semantic web with actors. We describe a flow of execution defined in a semantic manner as the agent's states and transitioning conditions by which to operate. So far, the reactive agent has been a core concept of our architectures. They have been so because our understanding of the WoT as a distributed system of multiple agents, or things, or VOs with their autonomous capabilities.

Secondly, we relate the SIoT with MAS. The SIoT provides the framework to create a web of agents related by goal-oriented links. Accordingly, the types of relationships are labelled with the pursuing goal in navigating the relationship. Social relationships imbue a meaning — a goal, a set of potential actions — in their type. For example, an OOR means that object B has an ownership relationship with a higher-order object A (A owns B), and a possible action (or set of actions) of A to gather and monitor the data sensed by B. In that sense, proper relationships between objects create meaningful paths of communication.

The authors in [157] describe the following set of challenges (among others) for integrating MAS and the Web: "they are populated with non-textual resources that cannot be simply indexed and ranked based on term frequency; autonomous agents would conduct searches for a broader range of purposes than mere information needs and would require structured query and result capabilities to fulfill their needs" [157]. The semantic web is capable of modelling any domain. When applied to the WoT, it can provide the description of what is and what the object can do (TD) and the description of its goal-oriented relation with other objects: what path has this object to travel to achieve the desired goal? Henceforth, a single object — a single agent — cannot rank resources "based on term frequency". Still, it can index resources by type of relationship and trust in that relationship, and by the set of goals that might be achieved following the relationship. Indexing does not happen objectively but by terms subjective to the agent's experience (e.g. trust, QoS) and relative to the agent's context (e.g. location, friends).

6.4 Further work

The integration between MAS, WoT and SIoT definitely poses a new perspective in which to look at the future IoT. Therefore, further work, either derived directly from the works in this thesis or from the gained experience, will be performed from this new point of view.

This thesis encompasses three manuscripts. The authors in [39] highlight two research paths, which are followed by [40] and [41].

On the one hand, future directions in [40] consider technical aspects such as enhancing a fully REST API compliant with TD and providing a visual tool to create execution flows. The research aims at evaluating the user experience and easiness by using the visual tool. We want to provide users — and especially non-tech users — with a visual tool to create and modify these flows of execution. If this is finally achieved, the WoT (semantic web technologies, TD, and functioning VOs and agents) will be closer to the non-tech user.

On the other hand, the SIoT is emerging as one of the solutions to enable a more scalable IoT. We relate SIoT to SG in [41]. They propose an integration of these two fields for a specific use case, DSM. We perform initial experiments in creating groups of prosumers that have the same goal. Nevertheless, there is a huge body of work to validate the proposal made in [41]. The near-future work comprehends (i) improving the load profile model on the basis existing and more complete models in the literature, (ii) researching new algorithm models to enable distributed and parallel rescheduling of appliances to optimize energy usage, (iii) improving network structure by including energy profile compatibility metrics, and (iv) developing new dynamic clustering algorithms to update clusters based on changes in energy profiles and prosumer-related information.

Bibliography

[1] L. Atzori, A. Iera, and G. Morabito, “Understanding the Internet of Things: definition, potentials, and societal role of a fast evolving paradigm,” *Ad Hoc Networks*, vol. 56, pp. 122–140, Mar. 2017, ISSN: 15708705. DOI: 10.1016/j.adhoc.2016.12.004.

[2] “International Energy Outlook 2019,” pp. 1–21, Sep. 2019. [Online]. Available: https://www.eia.gov/outlooks/ieo/.

[3] X. Yu, C. Cecati, and T. Dillon, “The New Frontier of Smart Grids,” *IEEE Industrial Electronics Magazine*, vol. 5, no. 3, pp. 49–63, 2011, ISSN: 1932-4529. DOI: 10.1109/MIE.2011.942176.

[4] Secretary-General, “Report of the Secretary-General on SDG Progress 2019,” pp. 1–64, Jul. 2019, ISSN: 2518-3915. [Online]. Available: https://sustainabledevelopment.un.org/content/documents/24978Report_of_the_SG_on_SDG_Progress_2019.pdf.

[5] R. Martín de Pozuelo, M. Ponce de León, J. Howard, A. Briones, J. Horgan, and J. Sánchez, “Software defined utility: A step towards a flexible, reliable and low-cost smart grid,” in *International Conference on Smart Grid Systems*, 2016. DOI: 10.12720/sgce.5.4.280-289.

[6] V. Gungor, D. Sahin, T. Kocak, S. Ergut, C. Buccella, C. Cecati, and G. Hancke, “Smart Grid and Smart Homes: Key Players and Pilot Projects,” *IEEE Industrial Electronics Magazine*, vol. 6, no. 4, pp. 18–34, Dec. 2012, ISSN: 1932-4529. DOI: 10.1109/MIE.2012.2207489.

[7] J. Rodríguez-Molina, M. Martínez-Núez, J.-F. Martínez, and W. Pérez-Aguiar, “Business Models in the Smart Grid: Challenges, Opportunities and Proposals for Prosumer Profitability,” *Energies*, vol. 7, no. 9, pp. 6142–6171, Sep. 2014, ISSN: 1996-1073. DOI: 10.3390/en7096142.

[8] J. Navarro, A. Zaballos, A. Sancho-Asensio, G. Ravera, and J. E. Armendariz-Inigo, “The Information System of INTEGRIS: INTelligent Electrical GRId Sensor Communications,” *IEEE Transactions on Industrial Informatics*, vol. 9, no. 3, pp. 1548–1560, Aug. 2013, ISSN: 1551-3203. DOI: 10.1109/TII.2012.2228869.

[9] E. Espe, V. Potdar, and E. Chang, “Prosumer communities and relationships in smart grids: A literature review, evolution and future directions,” *Energies*, vol. 11, no. 10, 2018, ISSN: 19961073. DOI: 10.3390/en11102528.

[10] P. Asmus, “Microgrids, Virtual Power Plants and Our Distributed Energy Future,” *Electricity Journal*, vol. 23, no. 10, pp. 72–82, 2010, ISSN: 10406190. DOI: 10.1016/j.tej.2010.11.001.

[11] A. J. D. Rathnayaka, V. M. Potdar, T. Dillon, O. Hussain, and S. Kuruppu, “Goal-Oriented Prosumer Community Groups for the Smart Grid,” *IEEE Technology and Society Magazine*, vol. 33, no. 1, pp. 41–48, 2014, ISSN: 02780097. DOI: 10.1109/MTS.2014.2301859.

[12] G. A. Shah, V. C. Gungor, and Akan, "A Cross-Layer QoS-Aware Communication Framework in Cognitive Radio Sensor Networks for Smart Grid Applications," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 3, pp. 1477–1485, Aug. 2013, ISSN: 1551-3203. DOI: 10.1109/TII.2013.2242083.

[13] A. A. Khan, M. H. Rehmani, and M. Reisslein, "Cognitive Radio for Smart Grids: Survey of Architectures, Spectrum Sensing Mechanisms, and Networking Protocols," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 860–898, 2016, ISSN: 1553-877X. DOI: 10.1109/COMST.2015.2481722.

[14] H. Maziku and S. Shetty, "Software Defined Networking enabled resilience for IEC 61850-based substation communication systems," in *2017 International Conference on Computing, Networking and Communications, ICNC 2017*, IEEE, IEEE, Jan. 2017, pp. 690–694, ISBN: 9781509045884. DOI: 10.1109/ICCNC.2017.7876213.

[15] A. Zaballos, A. Vallejo, and J. Selga, "Heterogeneous Communication Architecture for the Smart Grid," *IEEE Network*, vol. 25, no. 5, pp. 30–37, Sep. 2011, ISSN: 0890-8044. DOI: 10.1109/MNET.2011.6033033.

[16] N. Bui, A. Castellani, P. Casari, and M. Zorzi, "The Internet of Energy: A Web-Enabled Smart Grid System," *IEEE Network*, vol. 26, no. 4, pp. 39–45, 2012, ISSN: 0890-8044. DOI: 10.1109/MNET.2012.6246751.

[17] A. D. Rathnayaka, V. M. Potdar, T. Dillon, and S. Kuruppu, "Framework to manage multiple goals in community-based energy sharing network in smart grid," *International Journal of Electrical Power & Energy Systems*, vol. 73, pp. 615–624, Dec. 2015, ISSN: 01420615. DOI: 10.1016/j.ijepes.2015.05.008.

[18] F. Olivier, D. Marulli, D. Ernst, and R. Fonteneau, "Foreseeing New Control Challenges in Electricity Prosumer Communities," *IREP Symposium-Bulk Power System Dynamics and Control X*, 2017.

[19] B. P. Koirala, E. Koliou, J. Friege, R. A. Hakvoort, and P. M. Herder, "Energetic communities for community energy: A review of key issues and trends shaping integrated community energy systems," *Renewable and Sustainable Energy Reviews*, vol. 56, pp. 722–744, 2016, ISSN: 18790690. DOI: 10.1016/j.rser.2015.11.080.

[20] E. M. Gui and I. MacGill, "Typology of future clean energy communities: An exploratory structure, opportunities, and challenges," *Energy Research and Social Science*, vol. 35, no. October, pp. 94–107, 2018, ISSN: 22146296. DOI: 10.1016/j.erss.2017.10.019.

[21] P. Siano, "Demand response and smart grids—A survey," *Renewable and Sustainable Energy Reviews*, vol. 30, pp. 461–478, Feb. 2014, ISSN: 13640321. DOI: 10.1016/j.rser.2013.10.022.

[22] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of things: Vision, applications and research challenges," en, *Ad Hoc Networks*, vol. 10, no. 7, pp. 1497–1516, Sep. 2012, ISSN: 15708705. DOI: 10.1016/j.adhoc.2012.02.016.

[23] M. L. Zeng, "Interoperability," *Knowledge Organization*, vol. 46, no. 2, pp. 122–146, 2019. [Online]. Available: https://www.isko.org/cyclo/interoperability.

[24] D. Guinard, "A Web of Things Application Architecture - Integrating the Real-World into the Web," en, PhD thesis, University of Fribourg, 2011, pp. 1–220. DOI: 10.3929/ethz-a-006713673.

[25] A. B. Bondi, “Characteristics of scalability and their impact on performance,” *Proceedings Second International Workshop on Software and Performance WOSP 2000*, pp. 195–203, 2000. DOI: 10.1145/350391.350432.

[26] N. K. Tran, Q. Z. Sheng, M. A. Babar, and L. Yao, “Searching the Web of Things: State of the art, challenges, and solutions,” *ACM Computing Surveys*, vol. 50, no. 4, 2017, ISSN: 15577341. DOI: 10.1145/3092695.

[27] J. Travers and S. Milgram, “An Experimental Study of the Small World Problem,” *Phychology Today*, vol. 32, no. 4, pp. 179–197, 1977, ISSN: 0038-0431. DOI: 10.1016/B978-0-12-442450-0.50018-3.

[28] L. Atzori, A. Iera, and G. Morabito, “SIoT: Giving a social structure to the internet of things,” *IEEE Communications Letters*, vol. 15, no. 11, pp. 1193–1195, 2011, ISSN: 10897798. DOI: 10.1109/LCOMM.2011.090911.111340.

[29] L. Atzori, A. Iera, G. Morabito, and M. Nitti, “The social internet of things (SIoT) - When social networks meet the internet of things: Concept, architecture and network characterization,” *Computer Networks*, vol. 56, no. 16, pp. 3594–3608, 2012, ISSN: 13891286. DOI: 10.1016/j.comnet.2012.07.010.

[30] N. Shadbolt, T. Berners-Lee, and W. Hall, “The Semantic Web Revisited,” *IEEE Intelligent Systems*, vol. 21, no. 3, pp. 96–101, May 2006, ISSN: 1541-1672. DOI: 10.1109/MIS.2006.62.

[31] C. Bizer, T. Heath, and T. Berners-Lee, “Linked Data - The Story So Far,” *International Journal on Semantic Web and Information Systems*, vol. 5, no. 3, pp. 1–22, 2009, ISSN: 1552-6283. DOI: 10.4018/jswis.2009081901.

[32] S. Pattar, R. Buyya, K. R. Venugopal, S. S. Iyengar, and L. M. Patnaik, “Searching for the IoT resources: Fundamentals, requirements, comprehensive review, and future directions,” *IEEE Communications Surveys and Tutorials*, vol. 20, no. 3, pp. 2101–2132, 2018, ISSN: 1553877X. DOI: 10.1109/COMST.2018.2825231.

[33] C. Marche, L. Atzori, A. Iera, L. Militano, and M. Nitti, “Navigability in Social Networks of Objects: The Importance of Friendship Type and Nodes’ Distance,” in *2017 IEEE Globecom Workshops (GC Wkshps)*, Singapore: IEEE, Dec. 2017, pp. 1–6, ISBN: 978-1-5386-3920-7. DOI: 10.1109/GLOCOMW.2017.8269111.

[34] M. Nitti, V. Pilloni, G. Colistra, and L. Atzori, “The Virtual Object as a Major Element of the Internet of Things: A Survey,” *IEEE Communications Surveys and Tutorials*, vol. 18, no. 2, pp. 1228–1240, 2016, ISSN: 1553877X. DOI: 10.1109/COMST.2015.2498304.

[35] Independent Group of Scientitsts appointed by the Secretary-General, “Global Sustainable Development Report 2019: The Future is Now - Science For Achieving Sustainable Development,” pp. 1–252, 2019. [Online]. Available: https://sustainabledevelopment.un.org/content/documents/24797GSDR_report_2019.pdf.

[36] I. Inuwa, *Concept of Research Methodology in an Academic Research Report Writing*, 2016. DOI: 10.13140/RG.2.2.33631.05285.

[37] J. W. Creswell and J. D. Creswell, *Research design: Qualitative, quantitative, and mixed methods approaches*. Sage publications, 2017.

[38] Cisco, *The Internet of Things Reference Model*, 2014. [Online]. Available: http://cdn.iotwf.com/resources/71/IoT_Reference_Model_White_Paper_June_4_2014.pdf.

[39] V. Caballero, D. Vernet, A. Zaballos, and G. Corral, "Prototyping a Web-of-Energy Architecture for Smart Integration of Sensor Networks in Smart Grids Domain," *Sensors*, vol. 18, no. 2, 2018. DOI: 10.3390/s18020400.

[40] V. Caballero, S. Valbuena, D. Vernet, and A. Zaballos, "Ontology-Defined Middleware for Internet of Things Architectures," *Sensors*, vol. 19, no. 5, pp. 1–1163, Mar. 2019, ISSN: 1424-8220. DOI: 10.3390/s19051163.

[41] V. Caballero, D. Vernet, and A. Zaballos, "Social Internet of Energy - A new paradigm for Demand Side Management," *IEEE Internet of Things Journal*, pp. 1–16, 2019. DOI: 10.1109/JIOT.2019.2932508.

[42] V. Caballero, D. Vernet, A. Zaballos, and G. Corral, "Web of Energy: hacia la integración inteligente para las redes de sensores en Smart Grids," in *Proceedings XIII Jornadas de Ingenieria Telematica - JITEL2017*, Valencia: Universitat Politècnica València, Sep. 2017, pp. 30–39, ISBN: 9788490485958. DOI: 10.4995/JITEL2017.2017.6499.

[43] CEEC and XRE4S, *Industry 4.0: Novel strategies to develop energy solutions using IoT*, 2019. [Online]. Available: https://ris3catenergia.wordpress.com/2019/11/04/feedback-from-the-side-event-at-the-iot-solutions-world-congress-2019/ (visited on 11/21/2019).

[44] D. Vernet, A. Zaballos, R. Martín de Pozuelo, and V. Caballero, "High Performance Web of Things Architecture for the Smart Grid Domain," *International Journal of Distributed Sensor Networks*, vol. 11, no. 12, pp. 1–347 413, Dec. 2015, ISSN: 1550-1477. DOI: 10.1155/2015/347413.

[45] S. Cass, "The 2016 top programming languages," *IEEE Spectrum*, 2016.

[46] W3techs.com, *Usage Statistics and Market Share of Server-side Programming Languages for Websites*. [Online]. Available: https://w3techs.com/technologies/overview/programming_language/all (visited on 12/07/2017).

[47] J. Navarro, A. Sancho-Asensio, A. Zaballos, V. Jiménez-Ruano, D. Vernet, and J. E. Armendáriz-Iñigo, "The Management System of INTEGRIS: Extending the Smart Grid to the Web of Energy," in *Proceedings of the 4th International Conference on Cloud Computing and Services Science, Barcelona, Spain*, 2014, pp. 3–5.

[48] D. C. Vlad Trifa Dominique Guinard, *Web Thing Model*, 2015. [Online]. Available: https://www.w3.org/Submission/wot-model/.

[49] R. Martín de Pozuelo, A. Zaballos, J. Navarro, and G. Corral, "Prototyping a Software Defined Utility," *Energies*, vol. 10, no. 6, pp. 1–818, Jun. 2017, ISSN: 1996-1073. DOI: 10.3390/en10060818.

[50] J. Iwata, *Making Markets: Smarter Planet*. [Online]. Available: https://www.ibm.com/investor/events/investor0512/presentation/05_Smarter_Planet.pdf (visited on 11/12/2017).

[51] Gartner, *Gartner Says the Internet of Things Installed Base Will Grow to 26 Billion Units By 2020*. [Online]. Available: https://www.gartner.com/newsroom/id/2636073 (visited on 12/07/2017).

[52] ——, *Gartner Says 6.4 Billion Connected "Things" Will Be in Use in 2016, Up 30 Percent From 2015*. [Online]. Available: http://www.gartner.com/newsroom/id/3165317 (visited on 12/07/2017).

[53] R. T. Fielding, *Architectural Styles and the Design of Network-based Software Architectures*. University of California, Irvine Doctoral dissertation, 2000, vol. 54, pp. 1–162, ISBN: 0599871180. DOI: 10.1.1.91.2433.

[54] M. V. Trifa, "Building Blocks for a Participatory Web of Things: Devices, Infrastructures, and Programming Frameworks," PhD thesis, ETH, 2011, pp. 1–190. DOI: 10.3929/ethz-a-006716400.

[55] OASIS, *MQTT Version 3.1.1*, 2014. [Online]. Available: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html.

[56] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)," 2014, ISSN: 2070-1721. DOI: 10.17487/rfc7252.

[57] V. Trifa, S. Wieland, D. Guinard, and T. Bohnert, "Design and implementation of a gateway for web-based interaction and management of embedded devices," *the 2nd International Workshop on Sensor Network Engineering (IWSNE 09)*, no. March, pp. 1–14, 2009. DOI: 10.1002/(SICI)1099-1190(199909/10)9:5<309::AID-NEM339>3.3.CO;2-.

[58] ThingWorx, *Enterprise IoT Solutions and Platform Technology*. [Online]. Available: https://www.ptc.com/en/products/iot (visited on 12/07/2017).

[59] IBM, *IBM Watson IoT - IoT Platform*. [Online]. Available: https://www.ibm.com/internet-of-things/platform/watson-iot-platform (visited on 12/07/2017).

[60] Octoblu, *Octoblu | Integration of Everything*. [Online]. Available: https://octoblu.github.io/ (visited on 12/07/2017).

[61] EVRYTHNG, *EVRYTHNG IoT Smart Products Platform*. [Online]. Available: https://evrythng.com/ (visited on 12/07/2017).

[62] World Wide Web Consortium, *RDF 1.1 Concepts and Abstract Syntax*, 2014. [Online]. Available: https://www.w3.org/TR/rdf11-concepts/ (visited on 01/08/2019).

[63] B. Adida, M. Birbeck, S. McCarron, and S. Pemberton, "RDFa in XHTML: Syntax and processing," *Recommendation, W3C*, vol. 7, 2008.

[64] S. Bechhofer, "OWL: Web ontology language," in *Encyclopedia of Database Systems*, Springer, 2009, pp. 2008–2009. DOI: 10.1007/978-0-387-39940-9_1073.

[65] D. Guinard, M. Fischer, and V. Trifa, "Sharing using social networks in a composable Web of Things," in *2010 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, IEEE, IEEE, Mar. 2010, pp. 702–707, ISBN: 978-1-4244-6605-4. DOI: 10.1109/PERCOMW.2010.5470524.

[66] J. Gregorio and B. De hOra, "The atom publishing protocol," Tech. Rep., 2007. DOI: 10.17487/RFC5023.

[67] 1Worx, *ThingWorxComposer™ - 1Worx*. [Online]. Available: http://www.1worx.co/the-thingworx-platform/thingworx-composer/ (visited on 12/07/2017).

[68] IBM, *Node-RED*. [Online]. Available: https://nodered.org (visited on 12/07/2017).

[69] B. Ur, M. Pak Yong Ho, S. Brawner, J. Lee, S. Mennicken, N. Picard, D. Schulze, and M. L. Littman, "Trigger-Action Programming in the Wild," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems - CHI '16*, New York, New York, USA: ACM Press, 2016, pp. 3227–3231, ISBN: 9781450333627. DOI: 10.1145/2858036.2858556.

[70] E. Dijk, A. Rahman, T. Fossati, S. Loreto, and A. Castellani, "Guidelines for HTTP-CoAP Mapping Implementations," 2016.

[71] M. Collina, G. E. Corazza, and A. Vanelli-Coralli, "Introducing the QEST broker: Scaling the IoT by bridging MQTT and REST," in *2012 IEEE 23rd International Symposium on Personal, Indoor and Mobile Radio Communications - (PIMRC)*, IEEE, IEEE, Sep. 2012, pp. 36–41, ISBN: 978-1-4673-2569-1. DOI: 10.1109/PIMRC.2012.6362813.

[72] M. Khan, B. Silva, and K. Han, "A Web of Things-Based Emerging Sensor Network Architecture for Smart Control Systems," *Sensors*, vol. 17, no. 2, pp. 1–332, Feb. 2017, ISSN: 1424-8220. DOI: 10.3390/s17020332.

[73] ReactPHP, *Reactphp/react*. [Online]. Available: https://github.com/reactphp/react (visited on 12/07/2017).

[74] Zend.com, *Get performance insight into the upcoming release of PHP 7*. [Online]. Available: http://www.zend.com/en/resources/php7_infographic (visited on 12/07/2017).

[75] C. Hewitt, P. Bishop, and R. Steiger, "Session 8 Formalisms for Artificial Intelligence A Universal Modular ACTOR Formalism for Artificial Intelligence," in *Advance Papers of the Conference*, Stanford Research Institute, vol. 3, 1973, pp. 1–235.

[76] G. Agha, "Actors: A Model of Concurrent Computation in Distributed Systems," PhD thesis, Cambridge, MA, USA, 1986, ISBN: 0-262-01092-5.

[77] H. Sutter, "The free lunch is over: A fundamental turn toward concurrency in software," *Dr. Dobb's journal*, vol. 30, no. 3, pp. 202–210, 2005.

[78] D. Diaz Sanchez, R. Simon Sherratt, P. Arias, F. Almenarez, and A. Marin, "Enabling actor model for crowd sensing and IoT," in *2015 International Symposium on Consumer Electronics (ISCE)*, IEEE, IEEE, Jun. 2015, pp. 1–2, ISBN: 978-1-4673-7365-4. DOI: 10.1109/ISCE.2015.7177779.

[79] R. Hiesgen, D. Charousset, T. C. Schmidt, and M. Wählisch, "Programming Actors for the Internet of Things," *ERCIM NEWS*, pp. 1–25, 2015.

[80] J. Armstrong, *Programming Erlang: software for a concurrent world*. Pragmatic Bookshelf, 2007.

[81] Lightbend, *Akka*. [Online]. Available: https://akka.io/ (visited on 02/16/2019).

[82] M. Fowler, *Event Sourcing*. [Online]. Available: https://martinfowler.com/eaaDev/EventSourcing.html (visited on 01/13/2018).

[83] Mosquitto.org, *An Open Source MQTT v3.1 Broker*. [Online]. Available: http://mosquitto.org/ (visited on 12/07/2017).

[84] Lightbend, *Play Framework 2*. [Online]. Available: https://www.playframework.com/ (visited on 02/16/2019).

[85] *MongoDb*. [Online]. Available: https://www.mongodb.com/ (visited on 01/13/2018).

[86] C. Bormann and P. Hoffman, "Concise Binary Object Representation (CBOR)," 2013. [Online]. Available: https://tools.ietf.org/html/rfc7049.

[87] *DataPort - PecanStreet*, 2018. [Online]. Available: https://dataport.pecanstreet.org (visited on 01/13/2018).

[88] J. M. Selga, A. Zaballos, and J. Navarro, "Solutions to the Computer Networking Challenges of the Distribution Smart Grid," *IEEE Communications Letters*, vol. 17, no. 3, pp. 588–591, Mar. 2013, ISSN: 1089-7798. DOI: 10.1109/LCOMM.2013.020413.122896.

[89] Á. Silos, A. Señís, R. de Pozuelo, and A. Zaballos, "Using IEC 61850 GOOSE Service for Adaptive ANSI 67/67N Protection in Ring Main Systems with Distributed Energy Resources," *Energies*, vol. 10, no. 11, pp. 1–1685, Oct. 2017, ISSN: 1996-1073. DOI: 10.3390/en10111685.

[90] L. Atzori, R. Girau, S. Martis, V. Pilloni, and M. Uras, "A SIoT-aware approach to the resource management issue in mobile crowdsensing," in *2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN)*, IEEE, IEEE, Mar. 2017, pp. 232–237, ISBN: 978-1-5090-3672-1. DOI: 10.1109/ICIN.2017.7899418.

[91] R. Girau, S. Martis, and L. Atzori, "Lysis: A platform for iot distributed applications over socially connected objects," *IEEE Internet of Things Journal*, vol. 4, no. 1, pp. 40–51, 2017, ISSN: 23274662. DOI: 10.1109/JIOT.2016.2616022.

[92] C. Marche, M. Nitti, and V. Pilloni, "Energy efficiency in smart building: A comfort aware approach based on Social Internet of Things," in *2017 Global Internet of Things Summit (GIoTS)*, IEEE, Jun. 2017, pp. 1–6, ISBN: 978-1-5090-5873-0. DOI: 10.1109/GIOTS.2017.8016267.

[93] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, *et al.*, "OWL-S: Semantic markup for web services," *W3C member submission*, vol. 22, no. 4, 2004. [Online]. Available: https://www.w3.org/Submission/OWL-S/.

[94] World Wide Web Consortium, *W3C Web of Things Working Group*. [Online]. Available: https://www.w3.org/WoT/WG/ (visited on 02/17/2019).

[95] OneM2M, *oneM2M Base Ontology*. [Online]. Available: www.onem2m.org/technical/published-drafts (visited on 02/13/2019).

[96] V. Charpenay, S. Käbisch, and H. Kosch, "Semantic data integration on the web of things," in *Proceedings of the 8th International Conference on the Internet of Things - IOT '18*, New York, New York, USA: ACM Press, 2018, pp. 1–8, ISBN: 9781450365642. DOI: 10.1145/3277593.3277609.

[97] OneM2M, *oneM2M Global Initiative*. [Online]. Available: http://www.onem2m.org/ (visited on 02/18/2019).

[98] ——, "WoT Interworking," oneM2M, Tech. Rep., 2018, pp. 1–28. [Online]. Available: http://ftp.onem2m.org/work.

[99] A. Haller, K. Janowicz, S. J. Cox, M. Lefrançois, K. Taylor, D. Le Phuoc, J. Lieberman, R. García-Castro, R. Atkinson, and C. Stadler, "The modular SSN ontology: A joint W3C and OGC standard specifying the semantics of sensors, observations, sampling, and actuation," *Semantic Web*, pp. 1–24, 2018, ISSN: 22104968. DOI: 10.3233/SW-180320.

[100] Y. Belgueliel, M. Bourahla, and M. Brik, "Towards an Ontology for UML State Machines," *Lecture Notes on Software Engineering*, vol. 2, no. 1, pp. 116–120, 2014, ISSN: 23013559. DOI: 10.7763/LNSE.2014.V2.106.

[101] P. Dolog, "Model-driven navigation design for semantic web applications with the UML guide," in *Engineering Advanced Web Applications*, 2004, pp. 1–12, ISBN: 978-1589490468.

[102] M. Haage, J. Malec, A. Nilsson, M. Stenmark, and E. A. Topp, "Semantic Modelling of Hybrid Controllers for Robotic Cells," *Procedia Manufacturing*, vol. 11, pp. 292–299, 2017, ISSN: 23519789. DOI: 10.1016/j.promfg.2017.07.108.

[103] W. Pessemier, G. Deconinck, G. Raskin, P. Saey, and H. van Winckel, "Developing a PLC-friendly state machine model: lessons learned," in *Proc. of SPIE*, G. Chiozzi and N. M. Radziwill, Eds., vol. 9152, Jul. 2014, pp. 1–915 208, ISBN: 9780819496201. DOI: 10.1117/12.2054881.

[104] B. Negash, T. Westerlund, and H. Tenhunen, "Towards an interoperable Internet of Things through a web of virtual things at the Fog layer," *Future Generation Computer Systems*, vol. 91, pp. 96–107, Feb. 2019, ISSN: 0167739X. DOI: 10.1016/j.future.2018.07.053.

[105] Y. Guan, J. C. Vasquez, J. M. Guerrero, N. Samovich, S. Vanya, V. Oravec, R. Garcia-Castro, F. Serena, M. Poveda-Villalon, C. Radojicic, C. Heinz, C. Grimm, A. Tryferidis, D. Tzovaras, K. Dickerson, M. Paralic, M. Skokan, and T. Sabol, "An open virtual neighbourhood network to connect IoT infrastructures and smart objects — Vicinity: IoT enables interoperability as a service," in *2017 Global Internet of Things Summit (GIoTS)*, IEEE, Jun. 2017, pp. 1–6, ISBN: 978-1-5090-5873-0. DOI: 10.1109/GIOTS.2017.8016233.

[106] The Vicinity Consortium, "2nd Open Call. Technical Details," Vicinity Consortium, Tech. Rep., 2019, pp. 1–20. [Online]. Available: https://vicinity2020.eu/vicinity/sites/default/files/documents/vicinity_oc2_technical_guidelines_v04.pdf.

[107] MKLab, *StarUML*. [Online]. Available: http://staruml.io (visited on 01/08/2019).

[108] World Wide Web Consortium, *HTTP Vocabulary in RDF 1.0*. [Online]. Available: https://www.w3.org/TR/HTTP-in-RDF10/ (visited on 02/10/2019).

[109] Apache, *Apache Jena*. [Online]. Available: https://jena.apache.org/about_jena/about.html (visited on 01/08/2019).

[110] A. H. H. Ngu, M. Gutierrez, V. Metsis, S. Nepal, and M. Z. Sheng, "IoT Middleware: A Survey on Issues and Enabling technologies," *IEEE Internet of Things Journal*, vol. 4, no. 1, pp. 1–20, 2016, ISSN: 2327-4662. DOI: 10.1109/JIOT.2016.2615180.

[111] M. A. Razzaque, M. Milojevic-Jevric, A. Palade, and S. Cla, "Middleware for internet of things: A survey," *IEEE Internet of Things Journal*, vol. 3, no. 1, pp. 70–95, 2015, ISSN: 23274662. DOI: 10.1109/JIOT.2015.2498900.

[112] C. E. M. Gomes, V. F. Lucena, F. Yazdi, and P. Gohner, "An intelligent medicine cabinet proposed to increase medication adherence," in *2013 IEEE 15th International Conference on e-Health Networking, Applications and Services (Healthcom 2013)*, IEEE, Oct. 2013, pp. 737–739, ISBN: 978-1-4673-5801-9. DOI: 10.1109/HealthCom.2013.6720776.

[113] A. Gilchrist, *Industry 4.0 - The Industrial Internet of Things*. Berkeley, CA: Apress, 2016, ISBN: 978-1-4842-2046-7. DOI: 10.1007/978-1-4842-2047-4.

[114] C.-K. Chiou and J. C. R. Tseng, "An intelligent classroom management system based on wireless sensor networks," in *2015 8th International Conference on Ubi-Media Computing (UMEDIA)*, IEEE, Aug. 2015, pp. 44–48, ISBN: 978-1-4673-8270-0. DOI: 10.1109/UMEDIA.2015.7297426.

[115] J. M. Gutierrez, M. Jensen, M. Henius, and T. Riaz, "Smart Waste Collection System Based on Location Intelligence," in *Procedia Computer Science*, vol. 61, Elsevier, Jan. 2015, pp. 120–127. DOI: 10.1016/j.procs.2015.09.170.

[116] "International Energy Outlook 2018," pp. 1–21, Sep. 2018. [Online]. Available: https://www.eia.gov/outlooks/archive/ieo18/.

[117] K. Stamatis, “Communityware Smartgrid,” in *21st International Conference on Electricity Distribution*, 2011, pp. 6–9.

[118] *FP7 INTEGRIS Project Website*. [Online]. Available: https://cordis.europa.eu/project/rcn/93726_en.html (visited on 12/13/2018).

[119] *FP7 FI-PPP FINESCE Project Website*. [Online]. Available: http://www.finesce.eu/ (visited on 12/13/2018).

[120] G. Kariniotakis, L. Martini, C. Caerts, H. Brunner, and N. Retiere, “Challenges, innovative architectures and control strategies for future networks: the Web-of-Cells, fractal grids and other concepts,” *CIRED - Open Access Proceedings Journal*, vol. 2017, no. 1, pp. 2149–2152, Oct. 2017, ISSN: 2515-0855. DOI: 10.1049/oap-cired.2017.1287.

[121] A. M. Ortiz, D. Hussein, S. Park, S. N. Han, and N. Crespi, “The cluster between internet of things and social networks: Review and research challenges,” *IEEE Internet of Things Journal*, vol. 1, no. 3, pp. 206–215, 2014, ISSN: 23274662. DOI: 10.1109/JIOT.2014.2318835.

[122] T.-Y. Chung, I. Mashal, O. Alsaryrah, V. Huy, W.-H. Kuo, and D. P. Agrawal, “Social Web of Things: A Survey,” *2013 International Conference on Parallel and Distributed Systems*, pp. 570–575, 2013, ISSN: 1521-9097. DOI: 10.1109/ICPADS.2013.102.

[123] A. Kamilaris and A. Pitsillides, “Social networking of the Smart Home,” *21st Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, pp. 2632–2637, 2010. DOI: 10.1109/PIMRC.2010.5671783.

[124] L. Atzori, A. Iera, and G. Morabito, “The Internet of Things: A survey,” *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010, ISSN: 13891286. DOI: 10.1016/j.comnet.2010.05.010.

[125] M. Nitti, R. Girau, and L. Atzori, “Trustworthiness Management in the Social Internet of Things,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 5, pp. 1253–1266, May 2014, ISSN: 1041-4347. DOI: 10.1109/TKDE.2013.105.

[126] M. Nitti, R. Girau, L. Atzori, A. Iera, and G. Morabito, “A subjective model for trustworthiness evaluation in the social Internet of Things,” in *2012 IEEE 23rd International Symposium on Personal, Indoor and Mobile Radio Communications - (PIMRC)*, IEEE, Sep. 2012, pp. 18–23, ISBN: 978-1-4673-2569-1. DOI: 10.1109/PIMRC.2012.6362662.

[127] W. Abdelghani, C. A. Zayani, I. Amous, and F. Sèdes, “Trust management in social internet of things: a survey,” in *Conference on e-Business, e-Services and e-Society*, Springer, 2016, pp. 430–441, ISBN: 978-3-319-45233-3. DOI: 10.1007/978-3-319-45234-0_39.

[128] Z. Yan, P. Zhang, and A. V. Vasilakos, “A survey on trust management for Internet of Things,” *Journal of Network and Computer Applications*, vol. 42, pp. 120–134, Jun. 2014, ISSN: 10848045. DOI: 10.1016/j.jnca.2014.01.014.

[129] F. Bao, I. R. Chen, and J. Guo, “Scalable, adaptive and survivable trust management for community of interest based Internet of Things systems,” in *2013 IEEE Eleventh International Symposium on Autonomous Decentralized Systems (ISADS)*, IEEE, Mar. 2013, pp. 1–7, ISBN: 978-1-4673-5070-9. DOI: 10.1109/ISADS.2013.6513398.

[130] J. Kleinberg, “The small-world phenomenon: An algorithmic perspective,” in *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, ACM, 2000, pp. 163–170.

[131] Z. Li, R. Chen, L. Liu, and G. Min, "Dynamic Resource Discovery Based on Preference and Movement Pattern Similarity for Large-Scale Social Internet of Things," *IEEE Internet of Things Journal*, vol. 3, no. 4, pp. 581–589, 2016, ISSN: 23274662. DOI: 10.1109/JIOT.2015.2451138.

[132] A. Kamilaris, S. Yumusak, and M. I. Ali, "WOTS2E: A search engine for a Semantic Web of Things," in *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, IEEE, Dec. 2016, pp. 436–441, ISBN: 978-1-5090-4130-5. DOI: 10.1109/WF-IoT.2016.7845448.

[133] A. Rathnayaka, V. Potdar, and S. Kuruppu, "Design of Smart Grid Prosumer Communities via Online Social Networking Communitie," *International Journal for Infonomics*, vol. 5, no. 1, pp. 544–556, 2012. DOI: 10.20533/iji.1742.4712.2012.0062.

[134] A. Kamilaris, Y. Tofis, C. Bekara, A. Pitsillides, and E. Kyriakides, "Integrating Web-Enabled Energy-Aware Smart Homes to the Smart Grid," *International Journal On Advances in Intelligent Systems*, vol. 5, no. 1, pp. 15–31, 2012, ISSN: 1942-2679. DOI: 10.1.1.476.3440.

[135] C. Yang, H. Li, Y. Rezgui, I. Petri, B. Yuce, B. Chen, and B. Jayan, "High throughput computing based distributed genetic algorithm for building energy consumption optimization," *Energy and Buildings*, vol. 76, pp. 92–101, 2014, ISSN: 03787788. DOI: 10.1016/j.enbuild.2014.02.053.

[136] B. P. Esther and K. S. Kumar, "A survey on residential Demand Side Management architecture, approaches, optimization models and methods," *Renewable and Sustainable Energy Reviews*, vol. 59, pp. 342–351, Jun. 2016, ISSN: 13640321. DOI: 10.1016/j.rser.2015.12.282.

[137] M. AboGaleela, M. El-Sobki, and M. El-Marsafawy, "A two level optimal DSM load shifting formulation using genetics algorithm case study: Residential loads," in *IEEE Power and Energy Society Conference and Exposition in Africa: Intelligent Grid Integration of Renewable Energy Resources (PowerAfrica)*, IEEE, Jul. 2012, pp. 1–7, ISBN: 978-1-4673-2550-9. DOI: 10.1109/PowerAfrica.2012.6498651.

[138] V. Jayadev and K. Swarup, "Optimization of microgrid with demand side management using Genetic Algorithm," pp. 1–6, 2013. DOI: 10.1049/ic.2013.0124.

[139] C. Bharathi, D. Rekha, and V. Vijayakumar, "Genetic Algorithm Based Demand Side Management for Smart Grid," *Wireless Personal Communications*, vol. 93, no. 2, pp. 481–502, Mar. 2017, ISSN: 0929-6212. DOI: 10.1007/s11277-017-3959-z.

[140] M. Pedrasa, T. Spooner, and I. MacGill, "Scheduling of Demand Side Resources Using Binary Particle Swarm Optimization," *IEEE Transactions on Power Systems*, vol. 24, no. 3, pp. 1173–1181, Aug. 2009, ISSN: 0885-8950. DOI: 10.1109/TPWRS.2009.2021219.

[141] A. Sepulveda, L. Paull, W. G. Morsi, H. Li, C. P. Diduch, and L. Chang, "A novel demand side management program using water heaters and particle swarm optimization," in *2010 IEEE Electrical Power & Energy Conference*, IEEE, Aug. 2010, pp. 1–5, ISBN: 978-1-4244-8186-6. DOI: 10.1109/EPEC.2010.5697187.

[142] P. Faria, Z. Vale, J. Soares, and J. Ferreira, "Demand Response Management in Power Systems Using Particle Swarm Optimization," *IEEE Intelligent Systems*, vol. 28, no. 4, pp. 43–51, Jul. 2013, ISSN: 1541-1672. DOI: 10.1109/MIS.2011.35.

[143] Panait, Luke, L. Panait, and S. Luke, “Cooperative Multi-Agent Learning: The State of the Art,” *Autonomous Agents and Multi-Agent Systems*, vol. 3, no. 11, pp. 387–434, 2005, ISSN: 13872532. DOI: 10.1007/s10458-005-2631-2.

[144] R. B. Smith, L. R. Phillips, H. E. Link, and L. Weiland, “Agent-based control of distributed infrastructure resources.,” Sandia National Laboratories (SNL), Albuquerque, NM, and Livermore, CA, Tech. Rep. January 2006, Jan. 2006, pp. 1–118. DOI: 10.2172/883135.

[145] C.-S. Karavas, K. Arvanitis, and G. Papadakis, “A Game Theory Approach to Multi-Agent Decentralized Energy Management of Autonomous Polygeneration Microgrids,” *Energies*, vol. 10, no. 11, pp. 1–1756, Nov. 2017, ISSN: 1996-1073. DOI: 10.3390/en10111756.

[146] N. Fumo, “A review on the basics of building energy estimation,” *Renewable and Sustainable Energy Reviews*, vol. 31, pp. 53–60, Mar. 2014, ISSN: 13640321. DOI: 10.1016/j.rser.2013.11.040.

[147] H.-x. Zhao and F. Magoulès, “A review on the prediction of building energy consumption,” *Renewable and Sustainable Energy Reviews*, vol. 16, no. 6, pp. 3586–3592, Aug. 2012, ISSN: 13640321. DOI: 10.1016/j.rser.2012.02.049.

[148] I. Farris, R. Girau, L. Militano, M. Nitti, L. Atzori, A. Iera, and G. Morabito, “Social Virtual Objects in the Edge Cloud,” *IEEE Cloud Computing*, vol. 2, no. 6, pp. 20–28, Nov. 2015, ISSN: 2325-6095. DOI: 10.1109/MCC.2015.116.

[149] M. Odersky, “The Scala Experiment-Can We Provide Better Language Support for Component Systems?” *Conference Record of the Acm Symposium on Principles of Programming Languages*, vol. 33, pp. 1–166, 2006.

[150] E. Meijer and P. Drayton, “Static typing where possible, dynamic typing when needed: The end of the cold war between programming languages,” *Revival of Dynamic Languages*, 2004.

[151] P. Yuan, L. Lin, Z. Kou, L. Liu, and H. Jin, “Big RDF Data Storage, Computation, and Analysis: A Strawman’s Arguments,” in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, vol. 2019-July, IEEE, Jul. 2019, pp. 1693–1703, ISBN: 978-1-7281-2519-0. DOI: 10.1109/ICDCS.2019.00168.

[152] E. Yuan, “Architecture interoperability and repeatability with microservices: An industry perspective,” *Proceedings - 2019 IEEE/ACM 2nd International Workshop on Establishing a Community-Wide Infrastructure for Architecture-Based Software Engineering, ECASE 2019*, pp. 26–33, 2019. DOI: 10.1109/ECASE.2019.00013.

[153] M. A. Jarwar, M. G. Kibria, S. Ali, and I. Chong, “Microservices in web objects enabled IoT environment for enhancing reusability,” *Sensors (Switzerland)*, vol. 18, no. 2, 2018, ISSN: 14248220. DOI: 10.3390/s18020352.

[154] M. Noura, M. Atiquzzaman, and M. Gaedke, “Interoperability in Internet of Things: Taxonomies and Open Challenges,” *Mobile Networks and Applications*, vol. 24, no. 3, pp. 796–809, 2019, ISSN: 15728153. DOI: 10.1007/s11036-018-1089-9.

[155] D. Andročec, M. Novak, and D. Oreški, “Using semantic web for internet of things interoperability: A systematic review,” *International Journal on Semantic Web and Information Systems*, vol. 14, no. 4, pp. 147–171, 2018, ISSN: 15526291. DOI: 10.4018/IJSWIS.2018100108.

[156] B. Safaei, A. M. H. Monazzah, M. B. Bafroei, and A. Ejlali, “Reliability side-effects in Internet of Things application layer protocols,” in *2017 2nd International Conference on System Reliability and Safety (ICSRS)*, vol. 2018-Janua, IEEE, Dec. 2017, pp. 207–212, ISBN: 978-1-5386-3322-9. DOI: 10.1109/ICSRS.2017.8272822.

[157] A. Ciortea, S. Mayer, F. Gandon, O. Boissier, A. Ricci, and A. Zimmermann, “A Decade in Hindsight: The Missing Bridge Between Multi-Agent Systems and the World Wide Web,” *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 1659–1663, 2019. DOI: 10.5555/3306127.3331893.

Published articles

The following pages present the articles encompassed in this dissertation as published or accepted. Article references are: [39], [40] and [41].

sensors

MDPI

Article

Prototyping a Web-of-Energy Architecture for Smart Integration of Sensor Networks in Smart Grids Domain

Víctor Caballero *, David Vernet, Agustín Zaballos and Guiomar Corral

Engineering Department, Universitat Ramon Llull (URL), La Salle, 08022 Barcelona, Spain; dave@salleurl.edu (D.V.); zaballos@salleurl.edu (A.Z.); guiomar@salleurl.edu (G.C.)

* Correspondence: vcaballero@salleurl.edu; Tel.: +34-93-290-2436

Received: 15 December 2017; Accepted: 25 January 2018; Published: 30 January 2018

Abstract: Sensor networks and the Internet of Things have driven the evolution of traditional electric power distribution networks towards a new paradigm referred to as Smart Grid. However, the different elements that compose the Information and Communication Technologies (ICTs) layer of a Smart Grid are usually conceived as isolated systems that typically result in rigid hardware architectures which are hard to interoperate, manage, and to adapt to new situations. If the Smart Grid paradigm has to be presented as a solution to the demand for distributed and intelligent energy management system, it is necessary to deploy innovative IT infrastructures to support these smart functions. One of the main issues of Smart Grids is the heterogeneity of communication protocols used by the smart sensor devices that integrate them. The use of the concept of the Web of Things is proposed in this work to tackle this problem. More specifically, the implementation of a Smart Grid's Web of Things, coined as the Web of Energy is introduced. The purpose of this paper is to propose the usage of Web of Energy by means of the Actor Model paradigm to address the latent deployment and management limitations of Smart Grids. Smart Grid designers can use the Actor Model as a design model for an infrastructure that supports the intelligent functions demanded and is capable of grouping and converting the heterogeneity of traditional infrastructures into the homogeneity feature of the Web of Things. Conducted experimentations endorse the feasibility of this solution and encourage practitioners to point their efforts in this direction.

Keywords: smart grids; sensor networks; web of things

1. Introduction

Contrary to the rapid evolution experienced in the last decade of Information and Communication Technologies (ICTs), electric power distribution systems have remained exceptionally steady for a long time. Therefore, the Smart Grid concept, the addition of a telecommunication infrastructure to the electrical domain, has enabled a plethora of new services and opportunities (e.g., accurate grid monitoring, real-time energy consumption, and customer-side generation, etc.), which are currently driving a major revolution in the energy sector [1,2]. In fact, Smart Grids are conceived to improve traditional electric power networks in several dimensions [3] such as information, business models and scalability, in order to meet the highest standards of power quality and thus ensuring that the electric power grid is cost effective and sustainable [4].

Moreover, new agents and components have been raised inside this new paradigm. For example, the prosumer (load producer and load consumer) as the last link in the electricity value chain is easily the most important creator of value within the smart grid. Prosumers will be given a more active role in new business model generation. The most important connections for prosumers in the electricity value chain are the distributed system operator or the aggregator/retailer in addition to the Energy

Services Company/provider (ESCO) or Virtual Power Plants (VPPs), which are also new components in the energy market value chain [3]. Specifically:

- The aggregator controls low voltage power that is transferred to the usual places, where it is consumed with metering and billing functionalities.
- The Energy Service Companies (ESCOs) will play an important role in the future electricity market as energy-oriented commercial businesses. ESCOs can be described as specialists in providing a broad range of comprehensive energy solutions, including the design and implementation of energy saving projects, energy conservation, energy infrastructure outsourcing, power generation and energy supply and risk management.
- A VPP can either operate large numbers of relatively small-sized generators, responsive loads and storage units on behalf of owners (in this case, prosumers) or operate its own Distributed Energy Resources (DERs).

Those new agents and components of the Smart Grid could become the most enriched elements due to their added information and technology features. However, the upgraded features of the end user side come at the cost of requiring more technology in order to make them usable. Due to the complexity (i.e., stringent levels of service reliability and availability), magnitude (i.e., large-scale areas), and new agent profiles inherent to Smart Grids, practitioners have recently addressed the digital transformation of power electric networks by proposing flexible and future internet based architectures [4].

As far as the communication and network protocols' field is concerned, the Smart Grid has an inherent heterogeneous nature which forces system architects to consider different telecommunication technologies. Smart Grids are deployed in hostile wireless communication environments [5]; hence, channel status aware protocols [6] (also referred to as cognitive radio techniques) are needed to reduce communication delay [5], meet Quality of Service (QoS) needs in terms of delay, bandwidth, and data reliability, improve energy harvesting techniques [6], and provide reliable distributed sensing [7] in order to minimize interoperability issues between heterogeneous communication networks [8].

The Internet of Things (IoT) is introduced as the natural evolution of Internet, as it refers to the heterogeneous agglomeration of devices connected to the network. This evolution has succeeded thanks to the advances of both silicon, which have made it possible for increasingly smaller and smaller computing units to be embedded into everyday devices, as well as advances in low-power wireless protocols for such devices. The ICTs for the Smart Grids represent a good example of this heterogeneity, where different devices, both sensors and actuators, from different vendors and using different protocols, have to work together to achieve an objective: the integration of energy and smart services. Due to this heterogeneity issue, the Web of Things (WoT) [9] is presented as an abstraction layer that enables the homogeneous interaction between devices of different kinds using web technologies (previous work in JITEL 2017 [10]). In this sense, the grouping of the application of the methodologies provided by WoT to the management of Smart Grids under the term Web of Energy (WoE) [11,12] is proposed. In previous works [12], some proposals were introduced in this sense, however, it soon became apparent that another approach was needed for the development of an architecture for the WoE.

The paper is organized as follows: Section 2 describes the related work regarding Smart Grids, Web of Things, and Web of Energy, detailing their value proposals and associated challenges. Section 3 explains the problem of the direct integration of Smart Grids, Web of Things, and their communications protocols in order to compose the Web of Energy. Section 4 illustrates the hybrid architecture that is proposed that covers the needs of a Smart Grid infrastructure and provides the advantages of integrating the Smart Grids' ICT with the Web of Things, that is, the Web of Energy. In Section 5, we conduct experiments to evaluate the performance of the prototype system and discuss the results. Section 6 concludes the article. Finally, Section 7 discusses further work.

2. Related Work

2.1. Smart Grids and Web of Energy

Over the last decade, Smart Grids have led the revolution of the electrical grid, transforming it into a set of automated and efficiently controlled processes by the incorporation of ICTs. Smart Grids promote electrical energy management in a distributed and flexible manner. However, the current management systems are (i) centralized regarding their management; (ii) located in independent locations between them; and (iii) managed by fragmented applications, without integration between them and only intercommunicated thanks to specific communication channels, which are generally proprietary.

The main goal of Smart Grids is to provide better services and features (also known as smart functions), for both consumers and for producers and prosumers. In addition, the increased use of distributed and renewable energy generation requires changes in the electricity management system. It is necessary to improve automation systems, distributed intelligence, real-time data mining and management to improve network control functions, simplify configuration and also reduce system recovery and self-healing times.

Recent advances in Smart Grids have explored the feasibility of considering the electrical power distribution networks as a particular case of the IoT. Certainly, this specific domain poses appealing challenges in terms of integration, since several distinct smart devices from different vendors (wired or wireless sensors, smart meters, distributed generators, and so on), often using proprietary protocols and running at different layers, must interact to effectively deliver energy and provide a set of enhanced services and features [8]. Although the latest developments of the IoT field have definitely contributed to the physical connection of such an overwhelming amount of smart devices, several issues have arisen when attempting to provide a common management and monitoring interface for the Smart Grid [13].

To solve the integration of heterogeneous devices, the use of the concept of the WoT [9] has been proposed to access multiple devices using the same interface provided by web technologies, implying uniformity in communication protocols (HTTP and WebSockets) and uniformity in the model of data representation and device discovery (Web Thing Model [14] and Semantic Web [15]). This concept is, so far, difficult to apply in real scenarios involving energy management due to the risk of creating new security vulnerabilities, the lack of devices that implement standards of the WoT and the opposition of the industry of energy to include external modules or devices in their proprietary systems.

The objective of the work presented in this paper is to create an architecture based on the IoT paradigm to manage the storage and communication needs of the Smart Grids and at the same time link the Smart Grids with the end user through the methodologies of the WoT. For this, a bidirectional Human-To-Machine interface is established, inspired by the WoT that allows the ubiquitous control of the energy systems, the WoE [11].

In this way, the WoE could represent an opportunity for the electric companies to have virtual representations of their more flexible devices (based on software, updatable, configurable, and with the possibility of deploying new applications on top of them), enabling a low-cost distribution and management of the electric network [16]. The WoE facilitates the sharing of data from different devices (charging points for electric vehicles, smart metering, or monitoring of substations) with third parties. In addition, the use of web technologies allows the creation of multiplatform visualization tools without installation cost, being able to offer simple and usable graphic interfaces for a greater adoption for the Distribution Systems Operator (DSO) or any user interested in the consumption or production of energy (prosumer). Thus, we can identify different fields related to energy management in which the WoE would be very useful:

- Remote access form substations to central servers.
- Management and monitoring of DERs.

- Supervisory Control and Data Acquisition (SCADA) distribution to secondary substations.
- Electrical Vehicle Supply Equipment (EVSE) management.

2.2. Web of Things

Although the initial predictions of one trillion devices connected to the Internet by 2015 [17] were quickly lowered to 26 billion by 2020 [18] and 20.8 billion in the same year [19], it is evident that the number of devices connected to the Internet is increasing every day. How to access all these sensors and actuators through a uniform interface is undoubtedly one of the biggest challenges of the IoT. Currently, the IoT is divided into self-contained areas, that is, there are proprietary solutions that help the integration of a specific set of devices, but this structure is far from the overall integration planned for the IoT. For this reason, the use of existing web technologies for the global integration of devices is proposed. The basic prerequisites for enabling the IoT devices on the web are two: (1) minimum capacity for data processing and (2) connectivity to the network (it is not necessary to connect directly to the Internet).

The model proposed by the WoT [9] aims to solve the challenge of the heterogeneity of the IoT devices. This is principally achieved by generating translators or mappings between the language spoken by each device (communication protocol and data format) and the language that we can consider "universal" due to its widespread use: web technologies. Specifically, we have the HTTP protocol and the RESTful APIs [20]. These translations enable the devices to speak the same language, which makes them accessible in a homogeneous way to either human or computerized actors, like other devices. The actions to be taken on these devices can be both to act and to sense.

If we focus on the data flow, it is composed of two states that can be understood as a cycle: from heterogeneity to homogeneity and from homogeneity to heterogeneity.

- From heterogeneity to homogeneity: a device sends data captured with a specific format and protocol through a WoT translator, which translates the data into a common format and the protocol into HTTP.
- From homogeneity to heterogeneity: the actor receives this data and decides to act on the device, for example, changing its configuration. It then sends an instruction using the HTTP protocol and a format common to the WoT translator which, in turn, translates the protocol and format of the data into the specific protocol and format of the device.

3. From the Smart Grid to the Web of Energy

3.1. Smart Grid Architecture Modules

Our proposed Smart Grid architecture is composed of three main independent modules (see Figure 1). Each module is detailed in what follows.

- Context-aware security. This module aims to individually provide the needed security level for the proper operation of every smart function. For instance, for the use case of Smart Metering, this module can update all the encryption keys of the Smart Grid once an unauthorized access to the metering infrastructure has been detected.
- Hybrid Cloud Data Management: This module provides a data storage and processing system that intrinsically adapts to the Smart Grid's topology in a scalable and flexible way. Also, it implements an algorithm (also referred to as orchestrator) to decide whether collected data should be stored at the private cloud or could be placed at the public cloud. Such decisions are taken considering the smart functions' requirements (e.g., reliability, delay, and cybersecurity) associated with the collected data [12].
- Web of Energy: This module provides a ubiquitous (i.e., web based) monitoring interface [12] that enables a seamless management of the whole IoT architecture of the Smart Grid. In addition to providing a mechanism to communicate humans and machines, it also enables the interactions

among those IoT resource-constrained and small devices (i.e., machine to machine) through the HTTP protocol. Note that the aforementioned context-aware security module might decide to add an extra layer of security by switching from HTTP to HTTPS in accordance with the device features, network status, and smart function under execution demands. This is done through an open API that both couples and decouples all the modules. Hence, this module also acts as a bridge between the distributed storage layer—that takes care of all the Smart Grid's Big Data concerns—and the context-aware security layer—that gives the necessary access control and cybersecurity mechanisms.

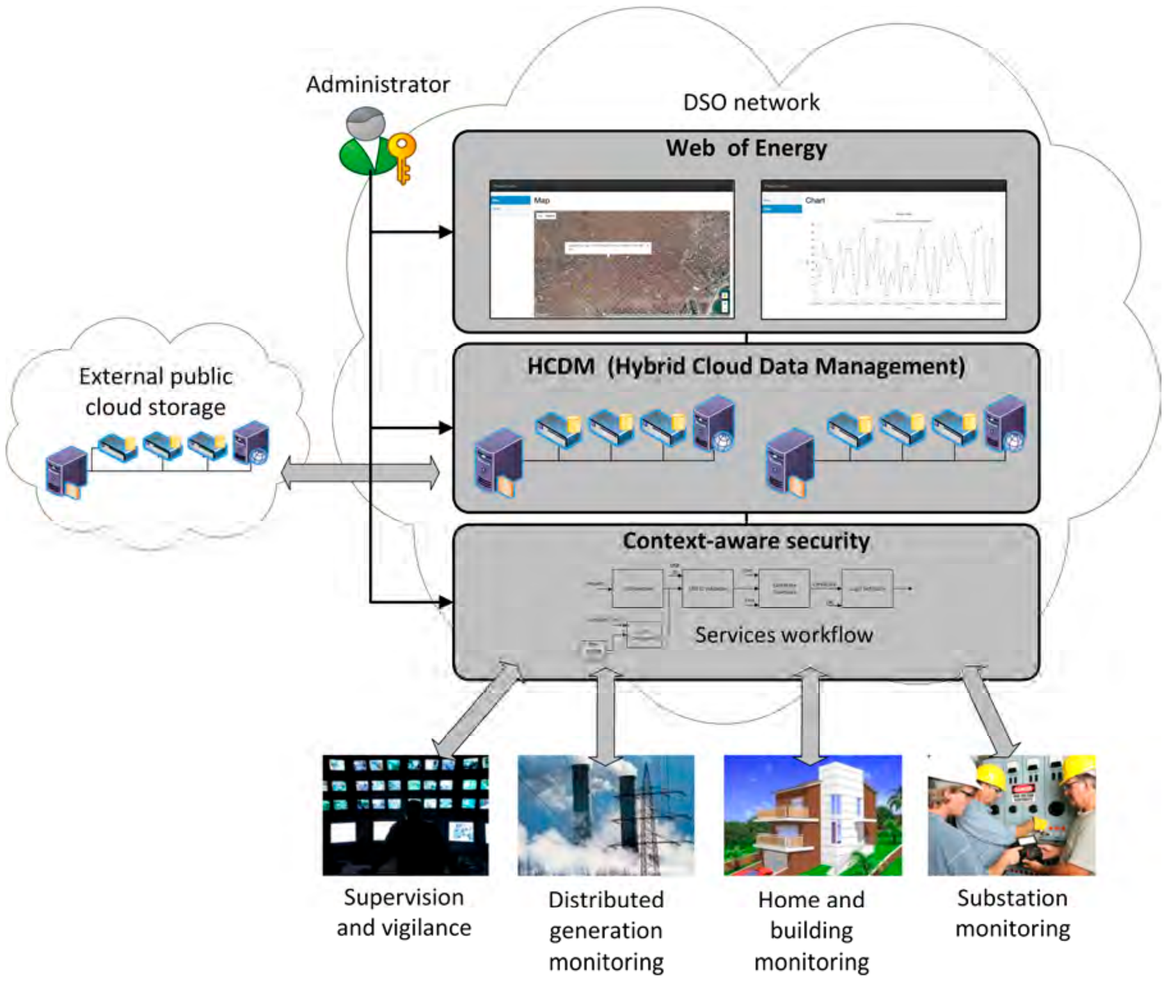

Administrator
DSO network
Web of Energy
Map
Chart
External public cloud storage
HCDM (Hybrid Cloud Data Management)
Context-aware security
Services workflow
Supervision and vigilance
Distributed generation monitoring
Home and building monitoring
Substation monitoring

Figure 1. Main modules: Context-aware security, Hybrid Cloud Data Management and Web of Energy.

3.2. Web of Things Architecture

The concept of WoT is developed mainly in the works of Dominique Guinard [9] and Vlad Trifa [21], where the WoT and the methods of enabling devices from IoT to WoT, respectively, are presented. In [9] the WoT is organized into four layers, each with a specific function. However, these layers do not follow an isolation and encapsulation model between non-contiguous layers, as is the case with the OSI (Open Systems interconnection) or TCP/IP (Transmission Control Protocol/Internet Protocol) model but, on the contrary, the applications can be built on top of each one (Figure 2), since all of them are part of the application level of the two network models mentioned above. Next, the different layers of the WoT proposed in [9] are presented.

The functions of each of these layers are the following:

1. Accessibility: It enables a consistent access to all types of IoT devices by exposing their functionalities through a RESTful HTTP API.
2. Findability: It provides the discovery of the representations of the different devices, uniformly modeling the access method (through the HTTP protocol) and establishing relations between them at the moment of their representation.
3. Sharing: It is in charge of preserving the privacy between the device representations. It also manages the authentication and authorization to access the representations by actors who do not own the device.
4. Composition: It enables the integration between the different representations of the devices that, ultimately, allows the integration between the different functionalities of the physical devices.

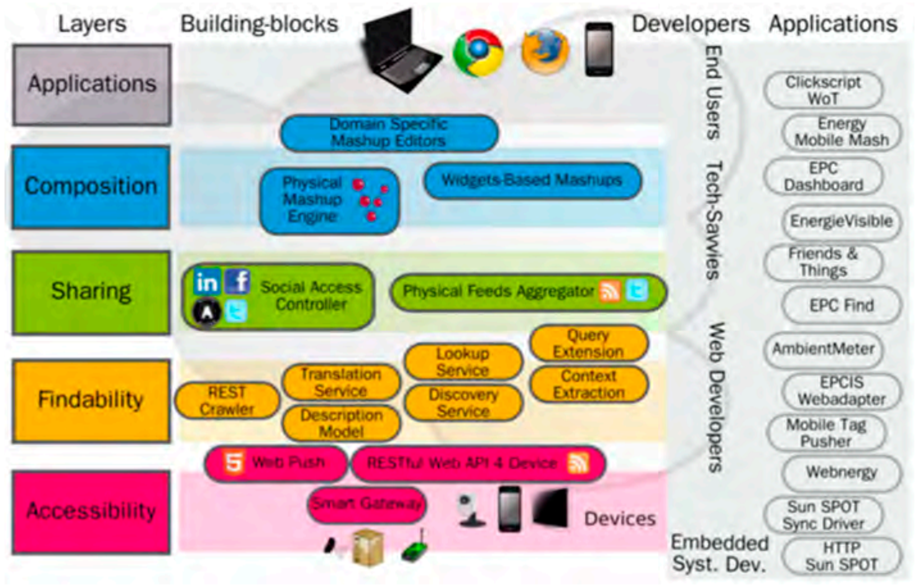


Figure 2. Web of Things layer model proposed in [9].

3.2.1. Accessibility Layer

The accessibility layer acts as an interface between the IoT and web technologies. Therefore, it is the closest layer to the heterogeneity of IoT protocols, such as MQTT (Message Queuing Telemetry Transport) [22] or CoAP (Constrained Application Protocol) [23], among others. In this layer, solutions are found in the form of a proxy or bridge between the IoT and HTTP protocols.

Two basic WoT-enabling methodologies are considered. Embedding web servers to devices, or the creation of gateways [24] that serve as aggregators or proxies between IoT protocols and web protocols. Cloud solutions are also found in this layer, generally transverse to more layers of the WoT, such as ThingWorx [25], Watson IoT Platform from IBM [26], Octoblu [27] or EVRYTHNG [28], among others. These cloud solutions offer IoT-WoT translation gateways.

3.2.2. Findability Layer

This layer is composed of the technologies that allow for the exploration and searchers of the different devices exposed to the WoT. Those technologies allow the publication of structured and interconnected data. The concept of REST (Representational State Transfer) [20] and Linked-Data [29] is applied to interconnect different device representations (usually in the form of URLs—Uniform Resource Locators). Semantic Web technologies [15] such as RDF (Resource Description Framework) [30], RDFa (Resource Description Framework in Attributes) [31] or OWL (Web Ontology Language) [32] are used to provide semantic meaning to both representations and connections with other representations, allowing, for example, the exposition of the information to search engines.

3.2.3. Sharing Layer

This layer groups all the methods that allow authentication and authorization of different actors to perform an action on the virtual representation of the device and, ultimately, on the physical device. It groups simple authentication and authorization methods from user authentication and password to the use of more advanced protocols such as API or OAuth keys.

A middleware called *Social Access Controller* is proposed in [33]. By combining the OAuth APIs of different social networks and the access by simple credentials to the devices (e.g., user and password), it allows to: (i) preserve the privacy of the physical devices; (ii) take advantage of the structure and function of social networks to authenticate the different actors potentially interested in participating in the actions that can be performed on the device representation (iii) integrate the representations of the devices in social networks, creating a Social Web of Things; and (iv) enable the publication of data aggregation using data syndication protocols like Atom [34].

3.2.4. Composition Layer

This layer enables the composition of the different functionalities exposed by the device's representations. Taking into account that each representation is accessible through the HTTP protocol, it is easy to program a script so that the different devices act in a coordinated way. The elaboration of this idea consists of providing the web user (human) with a visual interface to compose different device relationships. Solutions such as ThingWorx Composer [35], NODE-Red [36] from IBM or Octoblu [27] or IFTTT [37] exemplify the composition of physical devices (physical mashups).

3.3. Web of Things and Web of Energy Protocols

The main element of the WoT proposal is the use of web technologies for the transmission of information. As far as web protocols are concerned, there is the HTTP(S) protocol and the WebSocket (WS) or WebSocket Secure (WSS) protocol. HTTP is a request/response protocol while WebSocket allows a bidirectional connection between the client and the server.

However, many devices do not have the necessary features to directly access the web through HTTP or WebSocket. To be able to connect to the web, they need to use gateways that are responsible for translating the heterogeneity of the IoT protocols to the homogeneity of the WoT. To do this, it is necessary to establish translation bridges or mappings from IoT to WoT protocols and vice versa.

For example, both CoAP and MQTT are currently two of the most promising standard IoT protocols. CoAP, an open standard, was designed specifically for the IoT and to be directly compatible with HTTP [38]. It is a protocol based on request/response through UDP (User Datagram Protocol) packets and follows the same schemes as HTTP, allowing the creation of REST resources. The MQTT protocol, designed by IBM, is now also an open standard, but follows a publisher/subscriber paradigm over TCP, so it becomes more complicated to establish a translation bridge between HTTP-REST and MQTT [39].

Table 1 shows the differences and similarities between CoAP, HTTP, MQTT and WebSocket protocols. As mentioned, the mapping between CoAP and HTTP is direct, since the topology of both is request/response (Req/Resp). On the other hand, the topology between HTTP and MQTT is different, so the translation between these two protocols is more difficult to achieve. Another example of (almost) direct mapping is between WebSocket and MQTT, since the bidirectional communication of WebSocket can be considered a particularity of the publisher/subscriber protocol (Pub/Sub) of MQTT. Thus, HTTP and WebSocket are the two web technologies with different topologies that allow translation between IoT protocols with request/response and bidirectional topologies respectively. Table 1 also shows other protocols that can be part of the IoT, which have more similarities with HTTP or WebSocket according to their topology in the same way that occurs with CoAP and MQTT.

Table 1. WoT/IoT protocols.

	HTTP	WebSocket	CoAP	MQTT	XMPP	AMQP	DDS
Topology	Req/Resp	Two-way, realtime	Req/Resp	Pub/Sub	Pub/Sub and Req/Resp	Pub/Sub	Pub/Sub
Architecture	P2P	P2P	P2P	Broker	P2P	P2P or Broker	Global Data Space
Transport Level	TCP	TCP	UDP	TCP	TCP	TCP	TCP/UDP
Encryption	TLS	TLS	DTLS	TLS	TLS	TLS	TLS/DTLS
Authentication	TLS	TLS	DTLS	User/Pass	TLS	SASL	TLS/DTLS
QoS	-	-	Confirmable	3	-	3	23

In addition to the two protocols already presented in Table 1 and many other proprietary and open-source protocols for IoT, protocols that are more adapted to the specific needs of electric utilities are beginning to emerge, particularly in sectors where the infrastructures are becoming obsolete. However, those protocols become more difficult to adapt to the WoE because: (1) its specificity reduces the interest of third parties of contributing to the exposure of the devices (or sets of them) to the WoT and, therefore, the companies should invest more capital in the mapping and (2) in case it is of interest for the company to expose some of their devices, the WoT adds a stack of new technologies, opening new security holes in their systems. These challenges are significant in the WoE, because the systems involved cover a large number of devices that use very specific protocols adapted to their needs.

3.4. Complementing the Web of Things Architecture

The main goal of this section is to list the key features that a WoT architecture should comply with. These features are the following:

- Although certain IoT devices can support HTTP stacks, there are many of them that can only support lighter protocols due to their limited resources. Although ideally a narrow range of IoT protocols (e.g., CoAP and MQTT) would facilitate the integration of the devices to the Internet and the Web, a basic and indispensable requirement for the devices to be part of the Web is that they can be connected to the Internet.
- The architecture must provide abstractions so that developers can interact independently with the devices of the communication protocol, either towards the heterogeneity of the IoT protocols or towards the homogeneity of the WoT protocols.
- The architecture must be able to scale horizontally and provide self-healing for its systems. It must permit the on-demand deployment of resources.
- In the WoE architecture, the devices have to become virtual objects [40] or things, although virtual objects created from aggregations of other virtual objects can also be created.
- Each virtual object will be accessible from an HTTP REST interface and, in case the features of the virtual object allow it, from a WebSocket link.
- The architecture must allow executing authentication and authorization protocols towards the different devices involved.

In our proposal, we have identified five layers that compose the architecture of the WoE (Figure 3). Although they have a deep relationship with the layers proposed in [9] and other emerging architectures [41], some differences can also be identified, mainly due to the fact that the layers presented below are more focused on the development of the WoE architecture.

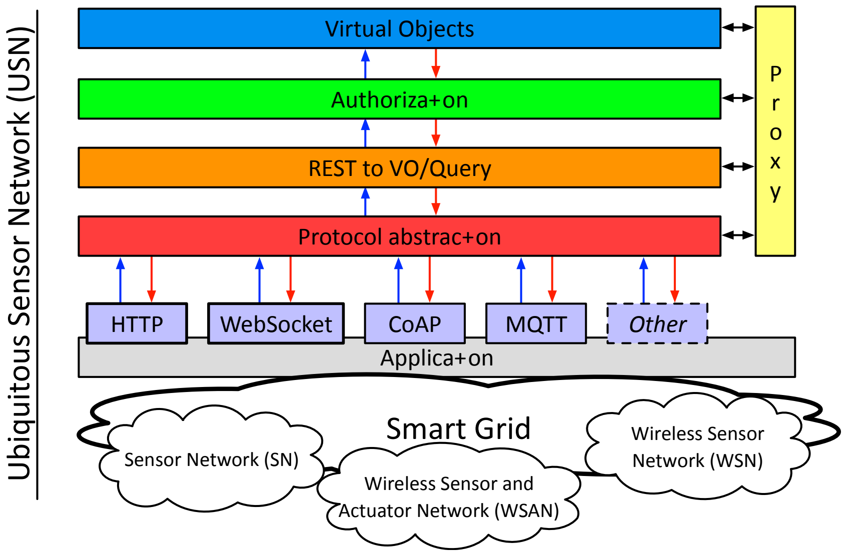


Figure 3. Proposed architecture for the Web of Energy.

The description of the different layers of the proposed WoE architecture is the following:

1. Protocol Abstraction Layer: The objective of this layer is to provide an abstraction layer for developers to interact with physical devices, both for developers of architecture functionalities at the more internal layers as well as for application developers for Smart Grid that, as already specified, group different protocols. The goal is not only to expose the IoT devices as HTTP REST resources, but to provide developers with abstraction mechanisms of both HTTP and the IoT protocols. In this sense, the premise exposed in [9] of enabling access to physical devices through an HTTP interface is maintained.
2. REST to VO/Query: The purpose of this layer is to translate the different URIs generated from the information of physical devices to actions on virtual objects or Things. In this way, we do not act directly on physical devices but through virtual objects. This layer also includes an interface for more complex queries about a *Thing* or the relationships of different *Things*.
3. Authorization: This layer is responsible for requesting and granting access to virtual objects or *Things*.
4. Virtual Objects: When physical devices must perform actions on other physical devices they will send the instruction against a virtual representation (virtual object or *Thing*). The objective of this virtual representation is to increase the resources and functionalities of physical devices. We cannot provide a detailed list of added features, as they are specific to each solution. Even so, our goal is to provide a method (code injection) so that these functionalities can be added dynamically. Some common functionalities are related to reasoning (artificial intelligence) or caching.
5. Proxy Layer: The architecture of the WoT presented in [9] proposes that all representations of the devices use web technologies (e.g., HTTP) to communicate and expose their characteristics and achieve homogeneous access to the devices, both between devices and between devices and humans. However, using web technologies for all types of communication between specific servers of the WoT will not always be optimal. The motivation of this layer is therefore to show that efficient communication between servers can be carried out through a protocol that is not within web technologies. However, it is important to mention that even those services should have an HTTP interface for the integration in the WoT.

Finally, the application layer conceptualizes all those applications that are likely to interact with the architecture. Within the range of applications supported by the WoT and, therefore, the IoT, those applications of Smart Cities and Smart Grids are grouped. Specifically, the generation of a Ubiquitous Sensor Network [8] is an application that includes the agglomeration of wired and wireless networks of sensors and actuators.

3.5. Web of Energy Implementation

3.5.1. First Approach: PHP Implementation

Our first goal in [12] was to create a high-performance WoT architecture, while experimenting with the most used web programming language up to that point, PHP (PHP: Hypertext Preprocessor), to see if it could be facilitated to the many web programmers in PHP [42,43] the programming interfaces for the WoT, thus achieving greater adoption in less time. However, the results were not positive for the following reasons:

- PHP web servers follow a very strict execution model that hinder the use of web technologies such as WebSocket. This is because WebSocket connections are permanent, whereas PHP servers delimit the connection with the client until a certain time limit or until the script execution is finished, since its main execution model is "load, execute, and die". There are alternatives to this execution model like the one implemented in React PHP [44] (reactor pattern), but this library is not available for large-scale production environments such as WoT.
- PHP, an interpreted language, is slower than compiled languages. In addition, with the execution model discussed above, main developers have never been concerned with optimizing the generated instructions or eliminating multiple memory leaks. Now, with the arrival of PHP 7, an improvement in performance has been achieved [45], although the other reasons discussed in this list are still valid.
- There are very few libraries focused on functionalities that involve mathematical calculations intrinsic to Machine Learning or distributed computing models. This is due, once again, to the idiosyncrasies of this type of languages, which are mainly used to serve dynamic web pages.

3.5.2. Second Approach: Actor Model Implementation

PHP is not efficient enough nor does it have the necessary tools for a large-scale implementation of an infrastructure for the WoT, as it was concluded in [12]. Once we discarded the use of PHP to develop the infrastructure of the WoT, we looked for a programming language that had the performance, the ecosystem and the necessary tools to develop such infrastructure. The JVM (Java Virtual Machine) provides a stable execution environment (since it is supported by Oracle) and many tools and libraries (protocols, web frameworks, and Machine Learning libraries) are available for this platform.

On the other hand, we were interested in finding a programming model that facilitated the creation of distributed, flexible, and self-healing systems, mandatory features for the WoE. For this reason, the Actor Model [46,47] seemed to be the ideal model. This model or paradigm of programming facilitates the creation of systems with those requirements and, in addition, thanks to the fact that it allows concurrency, it is capable of taking advantage of the different device computing cores [48].

The basic principles of operation of this model are that when an Actor receives a message, it can:

- Send messages to other Actors
- Create new Actors
- Designate how to manage the next received message

Several Actors can run concurrently although an Actor can only process one message at a time. That is why each Actor receives messages in a totally asynchronous way, storing them in a local message queue. Figure 4 represents this idea.

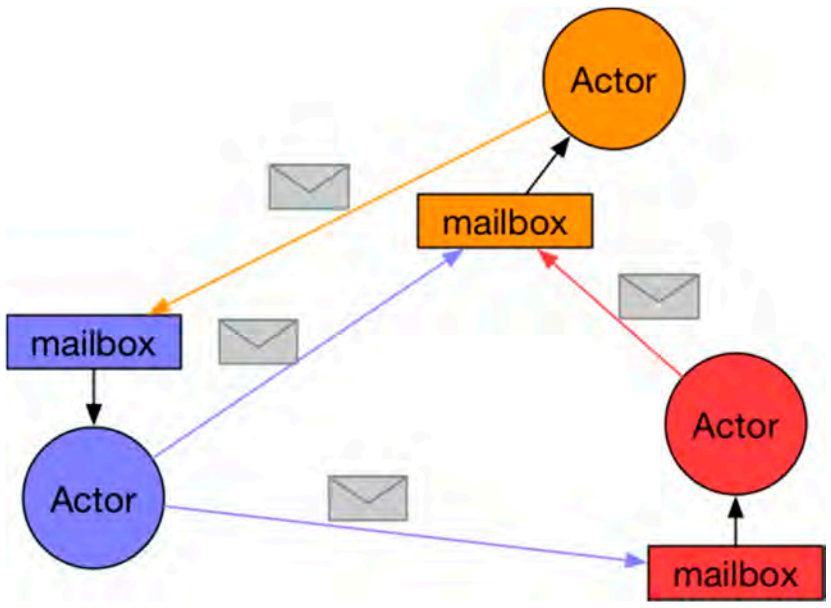


Figure 4. Representation of the messaging system in the Actor Model.

As can be observed, when an Actor receives a message, it is stored in a mailbox that is specific to each Actor and inaccessible to the rest of the Actors. This mechanism maintains a private status and isolates an Actor from the rest.

Although the Actor Model can be used to create agents and, therefore, Multi-Agent Systems (MASs), this article focuses on the exposure and use of the model per se in IoT or WoT.

In [49], authors benefit from this model to enable features such as multi-cloud and multi-tenure for IoT devices. They take advantage of the few resources that an actor needs to operate by compartmentalizing the physical devices in different software modules (actors) that are isolated and connected to different cloud infrastructures and have different owners.

In [50], the use of CAF (C++ Actor Framework) is proposed to provide developers with a high-level Operating System abstraction framework to develop IoT applications. The authors also emphasize the properties of abstraction, distribution and flexibility of this computation model.

Thanks to the principles on which the Actor Model is built, we can obtain interesting properties for the WoE infrastructure.

- Distribution and Flexibility: One of the basic principles of the model is the creation of new actors. This facilitates the use of computing resources not only in the same node but distributed in different nodes since the communication is asynchronous and transparent between local and remote node actors. Besides being able to create new actors, they can also be eliminated and, therefore, permit the management of the use of resources.
- Self-Healing: This property is not based on any principle of the theoretical Actor Model, but all the libraries that implement this model also implement it since the creation of Erlang [51], due to its great practical use. By definition, the state of an actor is isolated in that actor and can only communicate with the outside through messages. Self-healing takes advantage of this isolation principle to manage the failure of a given actor. For example, Akka [52] implements parental supervision. The creation of actors by another actor results in the creation of a hierarchy with a parent actor. In the event of the failure of a "child" actor, the "parent" actor can decide how to manage this failure: restart it or not, for example. Thanks to the isolation between actors and the supervision between actors, execution errors can be isolated and managed in a controlled manner. Actor state recovery mechanism involve event sourcing [53] techniques, where messages that change the state of the actor are logged and replayed when the actor restarts.

Thanks to the basic principles of an actor and the properties that derive from them, we also propose that each actor (or group of them) represents the virtualization of a physical device, increasing the resources and functionalities of such a device and isolating the execution failures of the other actors, that is, of the system. Actually, this proposal is not novel because there are different references [46,47] that propose this same use for the IoT. Our intention is to provide a working framework and an implementation for the WoE that meets its expectations. It is also our objective to promote the use of the Actor Model because it provides important abstractions to develop distributed systems, flexible, self-healing, and designed to achieve maximum and optimal use of resources thanks to parallel computing.

4. Prototypes, Initial Model, and Emulation Setting

In this section, we present the successfully implemented prototypes that are governed by the techniques analyzed above.

4.1. Layer Prototypes

Prototypes of some of the layers described above have been implemented for the proof of concept. Specifically, the Virtual Objects Space and the Protocol Abstraction layer have been implemented. Also, a publisher/subscriber protocol for the proxy layer has been considered to communicate two remote servers. Below both are detailed:

- Virtual Objects Space: Each physical device is represented by a virtual object and this in turn represented by one or more actors. This layer defines the logic and abstractions necessary for each device to be represented by one or more actors, in such a way that the data flow, taking as a starting point the reception of data from a protocol of what we consider to be heterogeneous or IoT (e.g., MQTT), is transformed into a homogeneous protocol (proxy layer) so that it can be understood by the other parts of the architecture.
- Proxy layer: According to the needs of this proof of concept, the proxy layer has been implemented through a publisher/subscriber protocol. In this way, the virtual objects that represent the devices can publish the captured data and can subscribe to messages also collected by other devices or to action messages sent by other devices.
- Protocol Abstraction Layer: Contains the implementations that interface between different protocols such as MQTT, WebSocket, or HTTP.

Figure 5 shows the interaction between the different layers in a Smart Grid. We find, considering a top-down vision, (i) the application layer with the intelligent functions of Smart Grids; (ii) the middleware proposed in this paper; (iii) the accessibility layer to the Ubiquitous Sensor Network (USN) with the aggregators of multiple sensors; and finally, (iv) the sensor networks. Intentionally, this schematic has a strong resemblance to Figure 2 shown in [8]. In this case, we have focused on the middleware development for a USN.

4.2. Initial Model

There are three well-differentiated sections in our implementation model. These sections are the following:

- MQTT Section: It consists of those devices that communicate through the MQTT protocol and the servers or services responsible for translating MQTT into an "understandable" protocol.
- Virtual Objects Space and Proxy Layer: This section is made up of those modules responsible for representing each physical device through a virtual object and the publisher/subscriber protocol.
- HTTP/WebSocket: Analogous to the MQTT section, it includes those devices that communicate with the architecture through web protocols and the services responsible for translating these protocols into a protocol "understandable" by the internal layers of the architecture.

The architectural model is depicted in detail in Figure 6.

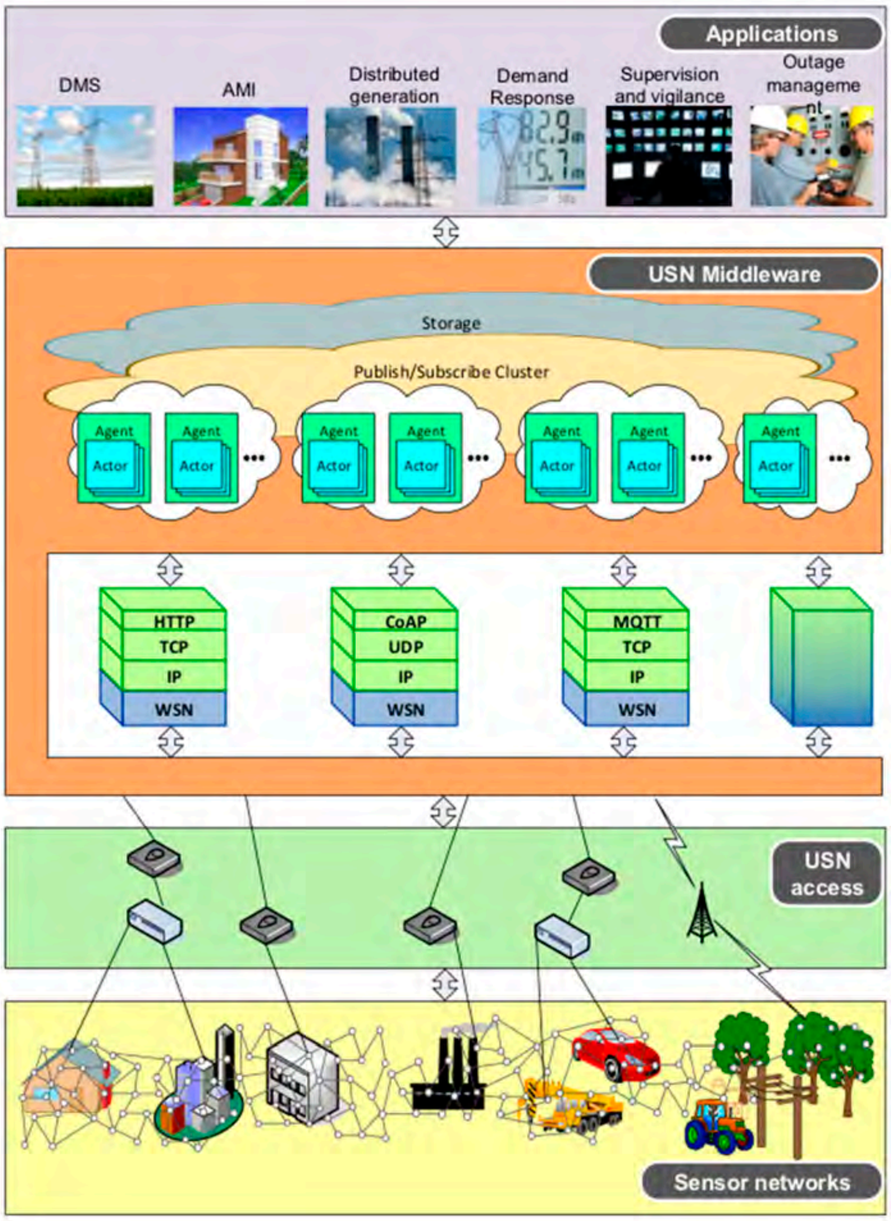


Figure 5. Layer schematic for a Ubiquitous Sensor Network (USN) applied to Smart Grids.

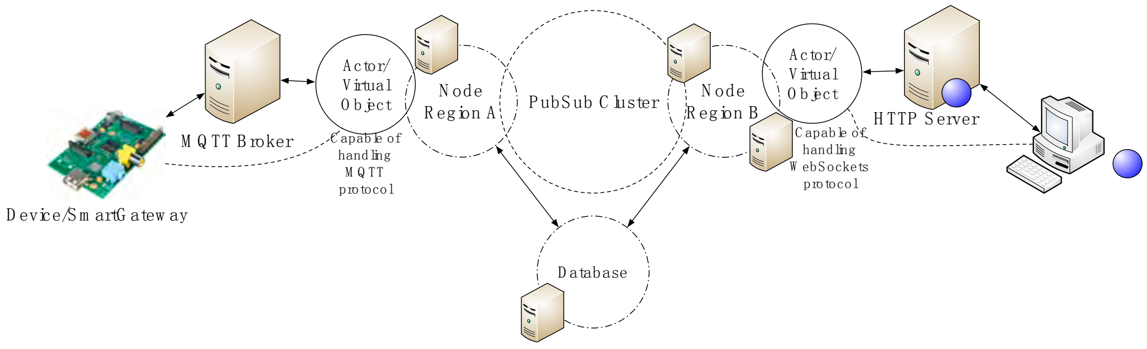


Figure 6. System design and architecture.

In order to transmit data to end nodes of this architecture (Device/SmartGateway and WebSocket Client in Figure 6), we have developed a communication protocol at the application layer that exploits the real-time capabilities of the proposed architecture. The protocol is depicted in Figure 7 and explains the actions conducted by each entity involved in the communication process. Involved entities are:

- Device: A device or smart gateway. In a Smart Grid setting, a smart meter or any other intelligent electronic device can be the Device.
- MQTT Broker: An MQTT server or broker that handles subscriptions and publications and deliveries messages from publishers to subscribers.
- MQTT ActorSystem: An Actor System responsible for translating the MQTT protocol to a homogeneous protocol. It is also responsible for allocating virtual objects by deploying the corresponding actors.
- Proxy Layer: A publish/subscribe system and a database.
- WebSocket ActorSystem: An Actor System responsible for translating the WebSocket protocol to a homogeneous protocol. It is also responsible for allocating virtual objects by deploying the corresponding actors.
- WebSocket Client: For example, a browser that supports the WebSocket protocol.

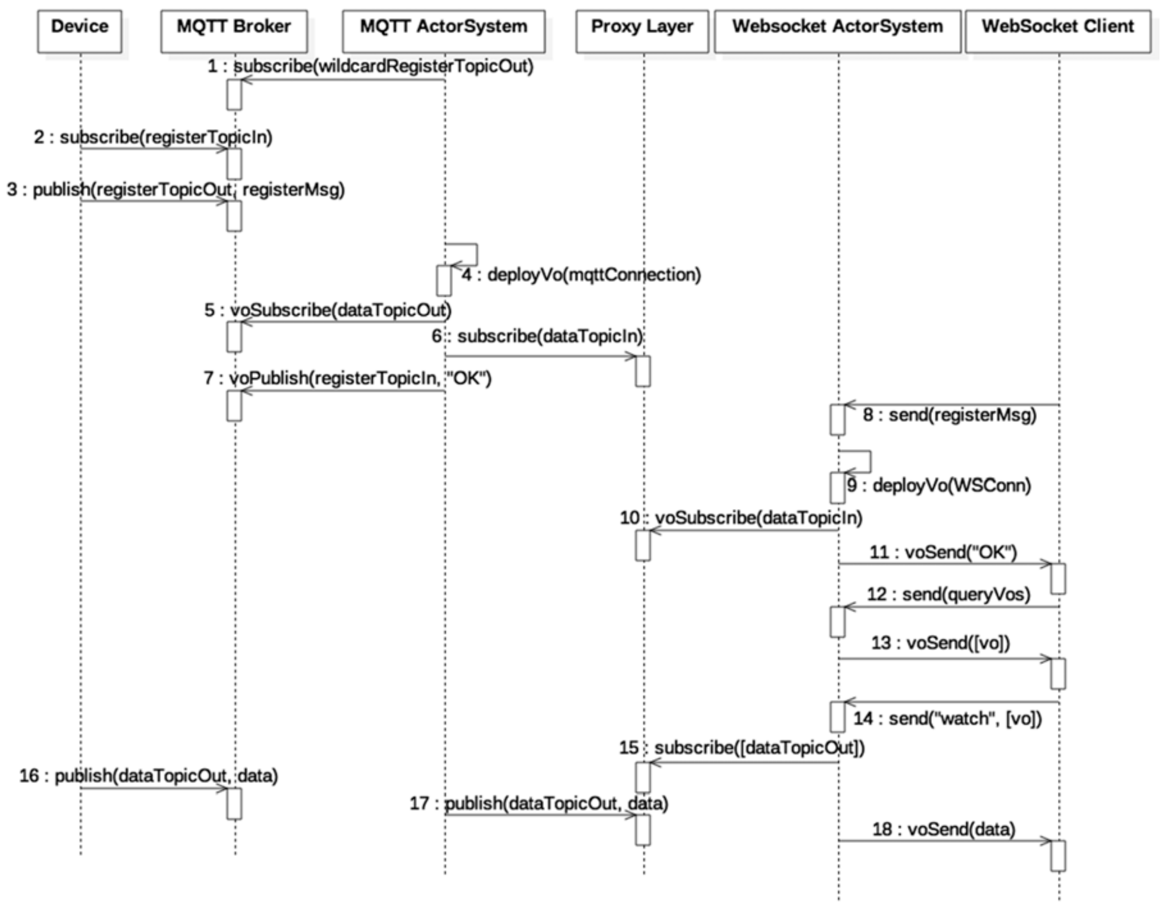

Device
MQTT Broker
MQTT ActorSystem
Proxy Layer
Websocket ActorSystem
WebSocket Client
1 : subscribe(wildcardRegisterTopicOut)
2 : subscribe(registerTopicIn)
3 : publish(registerTopicOut, registerMsg)
4 : deployVo(mqttConnection)
5 : voSubscribe(dataTopicOut)
6 : subscribe(dataTopicIn)
7 : voPublish(registerTopicIn, "OK")
8 : send(registerMsg)
9 : deployVo(WSConn)
10 : voSubscribe(dataTopicIn)
11 : voSend("OK")
12 : send(queryVos)
13 : voSend([vo])
14 : send("watch", [vo])
15 : subscribe([dataTopicOut])
16 : publish(dataTopicOut, data)
17 : publish(dataTopicOut, data)
18 : voSend(data)

Figure 7. Entity interaction diagram.

Entities perform actions by sending asynchronous messages. Actions between entities are represented in Figure 7 by directed arrows and the associated label gives information about the action performed and the involved and exchanged information in those actions. The information shown in parentheses can be seen as variables whose name gives semantic hints about their content. The schematic is explained as follows:

1. In step 1, an actor named MQTT Master that has been initially allocated in the MQTT ActorSystem subscribes to a wildcard topic such as *+/config/out* to receive incoming information about new devices. The plus sign in the wildcard topic means that the subscription is to all topics that match a *string* prepended by */config/out*.
2. A Device which communicates via MQTT initiates a registration process with the MQTT ActorSystem via the MQTT broker. As shown in steps 2 to 7, the Device subscribes to a topic dedicated to configuration purposes, which includes the registration process, and sends a registration request to the MQTT Master. If the registration process is successful, which is the flow shown, the MQTT Master deploys a dedicated actor responsible for handling the messages sent over the configuration topic and the data topic (*dataTopicOut*). The actor also subscribes to the Proxy Layer to receive messages directed to the device (step 6). Finally, the actor responds with an ACK ("OK" message) to the Device communicating that the registration process was successful. The virtual object that represents the Device and associated information is registered in the system's database (action not shown in Figure 7). Note that the actions prepended with the prefix "vo" (virtual object), (e.g., voSubscribe) correspond to actions performed by a dedicated actor, thus the voSubscribe action is performed by the actor deployed by MQTT Master in step 4. Also, the topic or message used in a "vo" action contains the ID of the virtual object and therefore messages and topics do not collide between Virtual Objects.
3. The WebSocket Client sends a message to the WebSocket ActorSystem, concretely to an actor named WebSocket Master, to initiate a registration process that starts at step 8 and ends at step 10. If successful, the WebSocket Master deploys a new actor which will be responsible for handling the messages sent over the previously established WebSocket connection (WSConn). The subscription process ends at step 11 when the dedicated actor sends and ACK ("OK" message) meaning that the registration process was successful. As with the MQTT registration process, the associated information is saved into the database.
4. Steps 12 and 13, exemplify a WebSocket Client query requesting for available virtual objects the dedicated actor response with appropriate data. The process of the actor querying the database is not shown in Figure 7. Then, in step 14, the WebSocket Client sends a message indicating what virtual objects to watch or subscribe to. Once the dedicated actor receives this information, it subscribes to the corresponding virtual object topics to receive updates of them in step 15. Variables in square brackets "[]" indicate a list of those variables.
5. Finally, steps 16 to 18 show how messages produced by the Device are consumed by the WebSocket Client.

Note that the process explained can be reversed. Once the devices (Device and WebSocket Client) connect to the architecture, both can perform the same actions. In fact, at the software level, the functionalities of both virtual objects are encoded equally (classes in Object Oriented Programming).

4.3. Emulation Setting

To implement the model represented in Figure 6:

- Mosquitto has been used as the MQTT server [54].
- Akka [52] has been used through the Scala API as an implementation of the Actor Model in Regions A and B. Note that the Erlang Virtual Machine allows actors to be hot deployed and that the characteristics of the Java Virtual Machine make this functionality harder to implement. Nevertheless, the OSGi framework provides the necessary means to accomplish this task, which could be useful for implementing Software Defined Utilities in the same way as used in Software Defined Networks [1,16]. Both in Erlang and Akka, an actor does not correspond to a system thread, but instead, a few threads, usually as many as the number of cores, are scheduled to process each actor mailbox when new messages arrive in order to optimize resources.

- For the Proxy Layer with the publish/subscribe function, a publish/subscribe cluster implementation has been used with Akka-Cluster.
- The Play Framework [55] has been used as WebSockets and HTTP server.
- MongoDB [56] has been used as the database to store available virtual objects.

In order to emulate the maximum number of network hops and distributed resources to approximate a real scenario, we have used the maximum number of lab resources that are at our disposal. We have used a MacBook Pro to simulate a large number of Devices and two Virtual Machines (VM). Each Virtual Machine is allocated in a different physical computer. Table 2 shows the specifications for each lab resource and the entities/services (Figure 7) allocated in them.

Table 2. Resources used in the emulation setting and service allocation.

Resource	Specifications	Services
VM 1	2 × 2.4 GHz Intel Xeon (4 cores per processor), 4 GB RAM	Mosquitto, MQTT ActorSystem
VM 2	3 × 2.4 GHz Intel Xeon (4 cores per processor), 6 GB RAM	Proxy Layer, HTTP/WebSocket Server and ActorSystem
MacBook Pro	2.7 GHz Intel Core i7 (2 cores per processor), 8 GB RAM	MQTT Devices, WebSocket Client

5. Experimentation and Results

In this section, we present an analysis of the successfully implemented prototypes that are governed by the techniques analyzed above. Our goal is to analyze the real-time performance of our system and to study how it behaves under different types of load. Concretely, the experiments measure the time interval from the moment the sensor data is sent by the Device or Smart Gateway until it arrives at the WebSocket Client.

The minimum message size includes the message id, the virtual object id, the payload, the unit of measure and the time the value was sensed. With payload or value equal to 0 bytes, the minimum message size is 132 bytes. The message id and virtual object id are raw UUIDs (Universally Unique Identifier) (36 bytes per UUID with hyphens) but an extra compression could be applied by encoding them to Base64 (22 Bytes) or Base85 (20 bytes), although the latter encoding is not URL-safe. The timestamp is represented as a 13-byte string. Finally, the unit of measure is represented as 1 byte and the remaining bytes account for JSON-encoding (JavaScript Object Notation) characters. Extra compression can be achieved using binary formats [57] but it was not within the scope of the project.

The legend in each experiment shows the mean, minimum, and maximum difference from the sending of a message from the Device to the reception of the same message by the WebSocket Client. From now on, we will refer to this difference as ∂t (*timedelta*). Therefore, we define $\partial t = rt - st$ for a message, where rt is the reception time and st is the sending time. We also define:

$$mean = mean(\partial t_i)$$

$$max = max(\partial t_i)$$

$$min = min(\partial t_i).$$

5.1. Experiment 1

The Dataport-PecanStreet project [58] provides a huge dataset of smart meter data. We have extracted the data from January 2015. The dataset contains data from 633 households and records of energy consumption and production at 15 min interval. Each record contains information about a maximum of 66 smart appliances per household but each household has no more than 15 smart appliances generating smart meter readings. A total of 124,331,328 individual smart meter records are available and 28,257,120 contain energy usage readings. For this reason, the payload size has been fixed to 4 bytes to fit a smart meter reading in each message and load tests are performed until total transmitted messages per burst resemble a scenario in which each household owns 15 smart appliances. As the experiments will be performed in an emulation setting and the goal is to perform stress tests, the interval of 15 min per smart meter reading has not been taken into account. We also perform 3 bursts where each device sends a message. The interval between bursts is of 1 s. Bursts are performed in order to analyze how actor mailboxes behave. Table 3 summarizes the description of the experiment. Figure 8 shows the results.

Table 3. Experiment 1 description setting.

Variable	Value
Devices	{1k, 2k, 3k, 4k, 5k, 6k, 7k, 9k}
Payload Size (bytes)	{4}
Burst Interval (ms)	{1000, 5000}
Bursts	{3}

As shown in Figure 8, the mailboxes of the actors tend to saturate at each sample, but the timedelta of the first message is less than 500 ms. A high percentage of messages are affected by the queue size, as the mean shape is closer to the maximum.

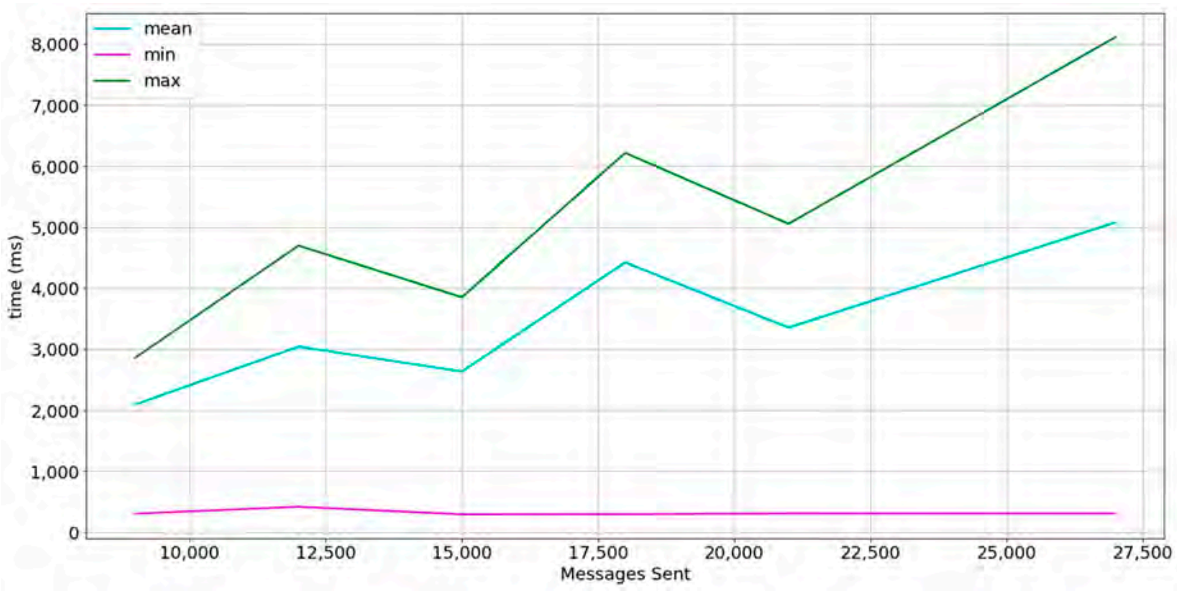


Figure 8. Results in experiment description setting 1 at a 1-s interval between bursts. The line in cyan represents the mean timedelta, the line in magenta represents the minimum timedelta, and the line in green represents the maximum timedelta.

Increasing the burst interval to 5 s, smooths the timedelta mean as shown in Figure 9. This means that a higher percentage of timedeltas compared with Figure 8 is closer to the minimum. It also shows that the maximum is 1.3 s approximately, compared to 5 s in Figure 8. The results obtained conform to time needs in non-critical Smart Grid applications [4,16,59].

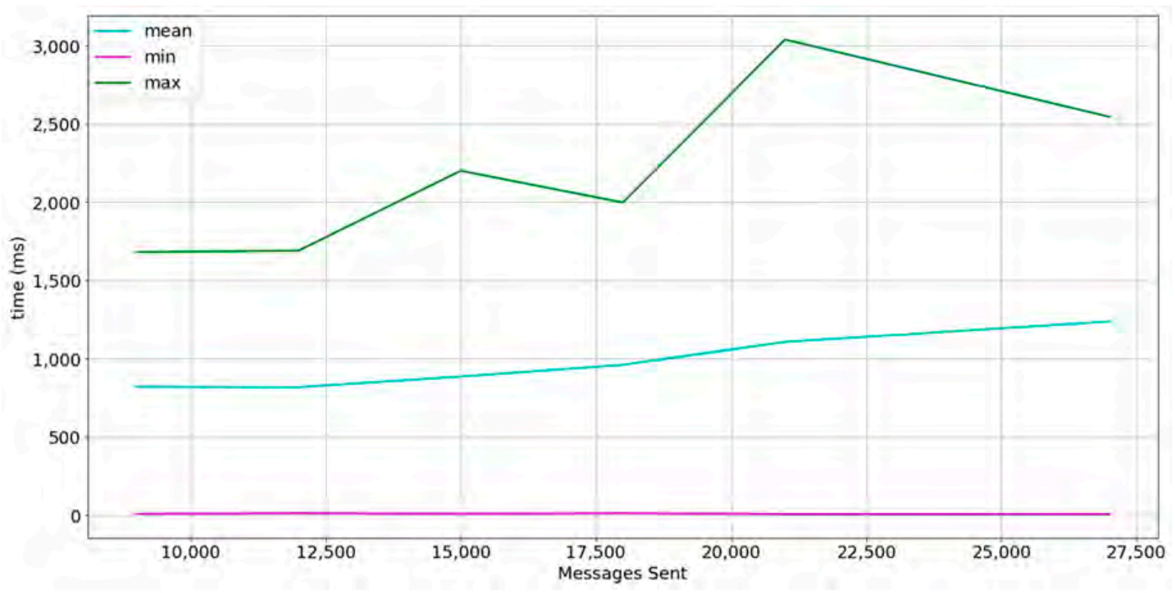


Figure 9. Results in experiment description setting 1 at a 5-s interval between bursts. The line in cyan represents the mean timedelta, the line in magenta represents the minimum timedelta, and the green line represents the maximum timedelta.

5.2. Experiment 2

The experiment description setting 2 is similar to the experiment description setting 1 but instead of increasing the number of simulated devices, it emulates the aggregation of messages in the payload field. The experiment shows 1000 devices sending messages with payloads that aggregate messages, emulating that each device is a Smart Gateway that aggregates messages. Table 4 summarizes the experiment description.

Table 4. Experiment 2 description.

Variable	Value
Devices	{1000}
Payload Size (bytes)	{272, 408, 544, 680, 816, 952, 1224}
Burst Interval (ms)	{1000, 5000}
Bursts	{3}

The results of the experiment are shown in Figure 10. Note that the x-axis shows aggregated messages instead of sent messages. For example, the message size of a message that aggregates messages is calculated as:

$$\begin{aligned} &messageSize(aggregatedMessages) \\ &= baseSize + aggregatedMessages * (baseSize + payload) \end{aligned}$$

where $payload = 4$ bytes, $baseSize = 132$ bytes. If a Smart Gateway sends two aggregated messages, the total payload is calculated as:

$$messageSize(2) = 132 + 2 * (132 + 4) = 132 + 272 = 404 \text{ bytes.}$$

The difference of the mean, minimum, and maximum parameters between samples at 1 s interval and 5 s interval are not abrupt, which means that reducing the interval between bursts from 5 s to 1 s does not have a great impact on performance, even with messages that have a payload equal to 1224 bytes. We conclude that our system is capable of delivering 1000 messages aggregating nine messages in their payload within a $\partial t < 1.42$ s at an interval of 1 s between 1000 messages, which is equal to 9000 aggregated messages delivered in less than 1.5 s and at a mean rate of 600 ms. The experiments start when 2 messages are aggregated, since the aggregation of 1 message is nonsense.

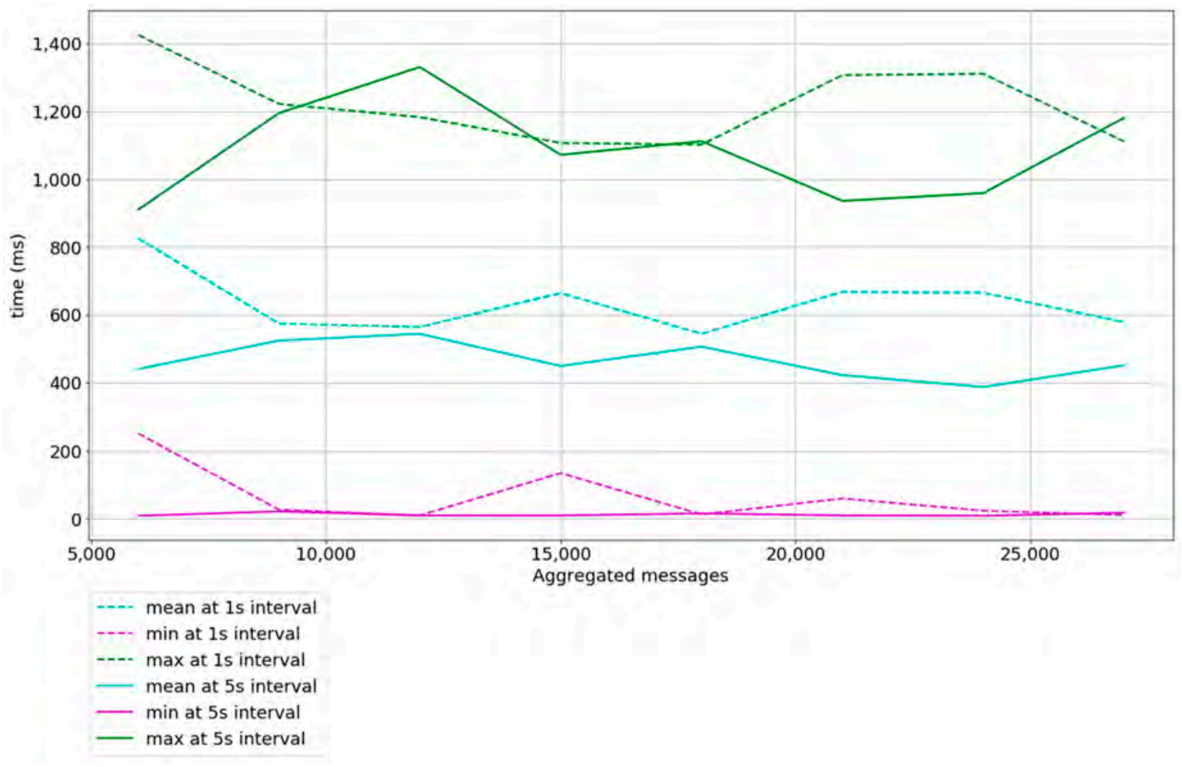


Figure 10. Results in experiment description setting 2 at 1 and 5-s interval between bursts. Dashed lines represent experiments at a 1-s interval between bursts and continuous lines represent experiments at a 5-s interval between bursts. Cyan lines represent the mean timedelta, magenta lines represent the minimum timedelta, and lines in green represent the maximum timedelta.

5.3. Discussion between Experiments 1 and 2

In Figure 11 a comparison between experiments 1 and 2 with the description setting where the time between bursts is 5 s is shown. The mean timedelta obtained in both experiments shows that a better performance can be achieved by aggregating messages (experiment 1). The slope of the mean function in experiment 1 is approximately 0 and the slope of the mean function in experiment 2 is positive. No messages were lost in either experiment 1 nor 2. While results in experiment 1 conform to time needs of non-critical Smart Grid applications, results in experiment 2 conform to some of time needs in critical Smart Grid applications [4,59,60]. We also conclude that message aggregation is preferred when possible, as it reduces the overheads of the entire system.

5.4. Web of Energy, Proof of Concept

To conclude the experiments, we directed our efforts to developing a web portal to visualize the data available through the HTTP and the WebSocket protocol, which prototypes the WoE. 1000 devices have been simulated, emitting synthetic smart meter data every 5 s. These sensors are distributed throughout the city of Barcelona. Figure 12 shows one of the functionalities of the system, a heat map updated every 5 s in near real-time. However, Figure 12 only shows a few of those sensors because the web browser performance is rapidly decreased as more and more objects need to be painted. A possible solution could be to aggregate smart meter readings and show the aggregation value in the heat map.

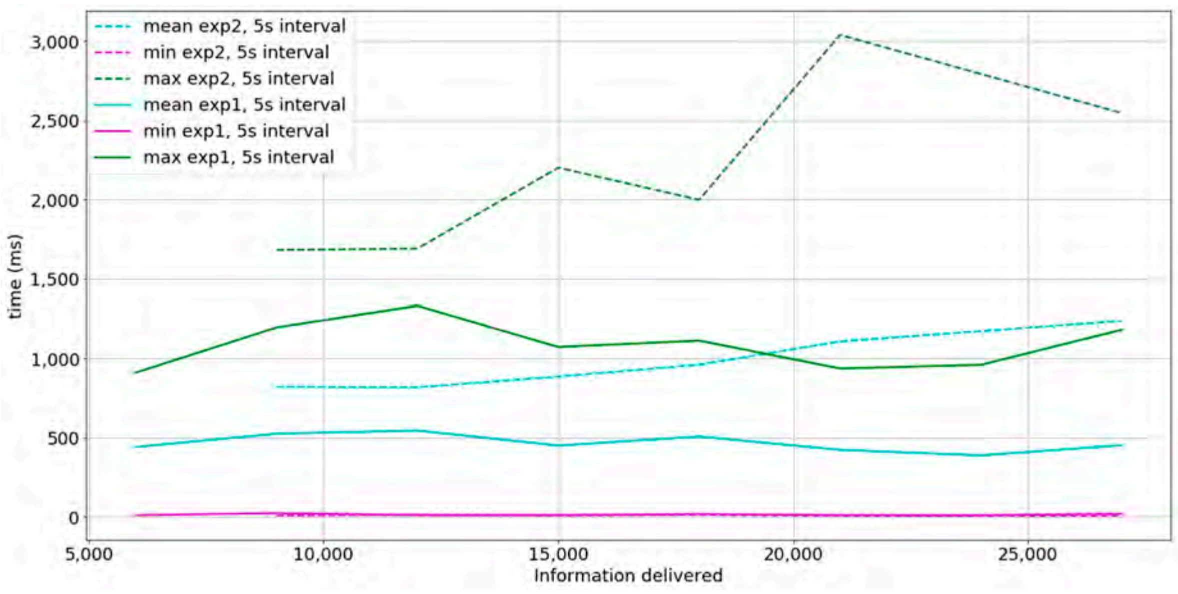


Figure 11. Comparison between experiment 1 and experiment 2 at a 5-s interval between bursts. x-axis equals to the total information delivered and refers to both individual messages at experiment 1 and aggregated messages at experiment 2. Dashed lines represent measurements in experiment 2 at a 5-s interval between bursts. Continuous lines represent measurements in experiment 1 at a 5-s interval. Cyan lines indicate the mean timedelta, magenta lines indicate the minimum timedelta, and green lines represent the maximum timedelta.

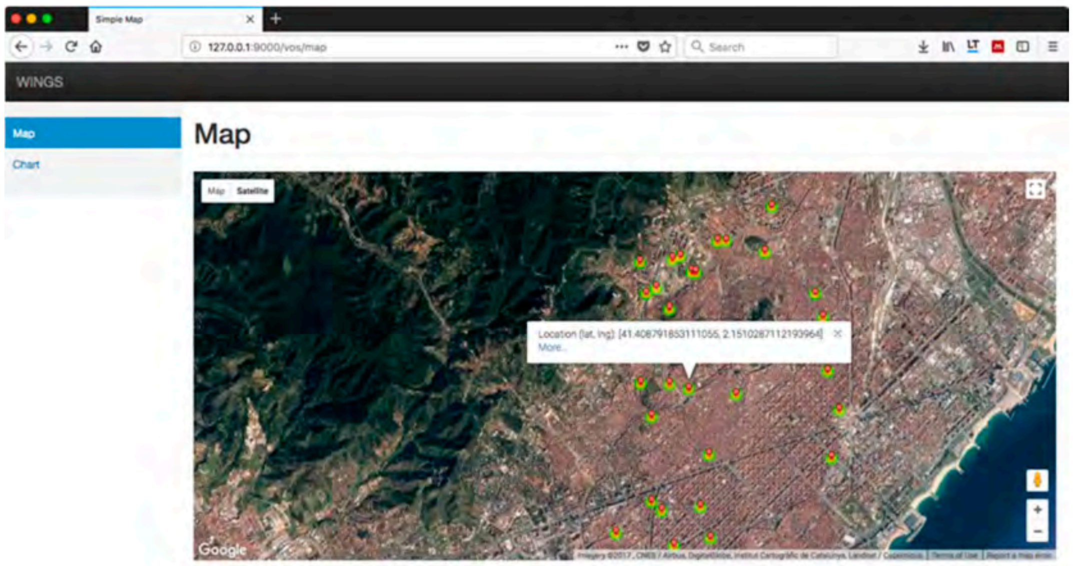


Figure 12. Graphic representation of sensor location.

Figure 13 shows the result of consulting the value of one of these sensors by clicking in the "More . . . " link shown in Figure 12. The chart label corresponds to the universal unique identifier (UUID) of the sensor. The values in the vertical axis correspond to the measurements obtained by the sensor. This simulated sensor reads voltage, so a "V" inside the parentheses is displayed. The values in the horizontal axis show the time at which the voltage reading is obtained, every 5 s. The data shown updates in near real-time.

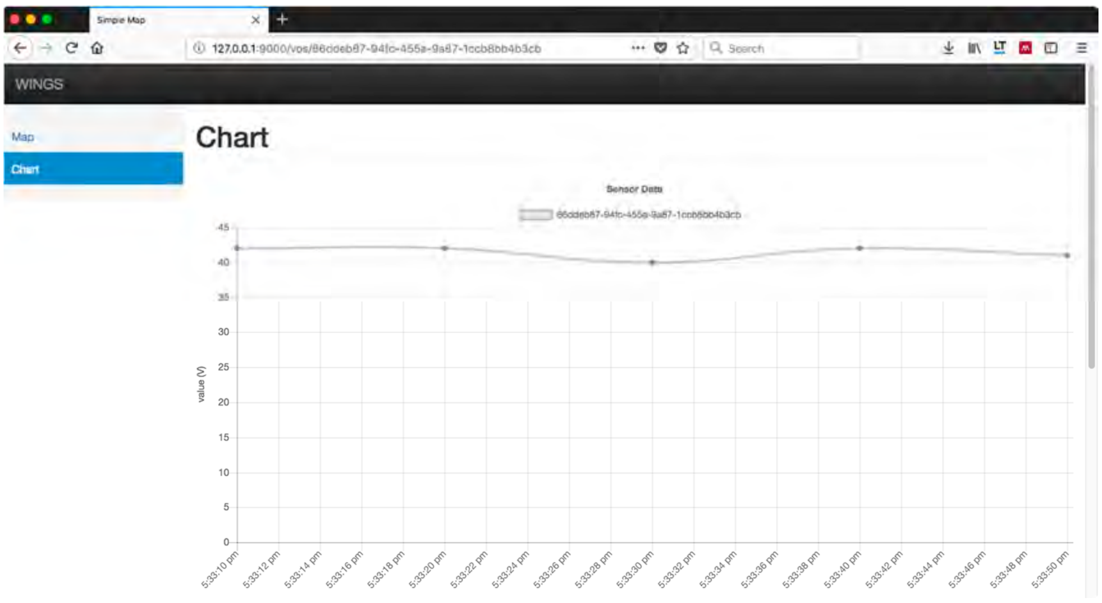


Figure 13. Chart showing the synthetic values emitted by a sensor every 5 s.

The application is designed with the ability to interact with each of these sensors, although in this initial version only the query of values is allowed. All queries are executed in near real-time and without delays, despite the large volume of data. This confirms that the design of the prototype is consistent and allows the extension of the rest of the WoE.

5.5. Discussion and Conclusions

A WoE prototype based on the Actor Model has been implemented. Our goals were: (i) to address some of the key issues related to ICTs in Smart Grid systems while demonstrating the benefits of the Actor Model in such ICT—Smart Grid use cases and (ii) to implement a prototype for further research. The Smart Grid can be seen as a specific Internet of Things application, with its own constraints and requirements. Specifically, we have addressed the heterogeneity of protocols by means of the WoT and some of the near real-time requirements of Smart Grid ICT systems. We have created Virtual Objects, a common abstraction in IoT scenarios, by means of the Actor Model. Each Virtual Object is represented by at least one actor, a lightweight reactive agent. Heterogeneity is addressed in the prototype system by the capacity of each implemented actor to translate heterogeneous protocols (MQTT, for example) to web protocols (HTTP or WebSocket). Performance tests show that the initial model meets some of the time needs of time-critical Smart Grid ICT applications when Virtual Objects aggregate several smart meters (messages that aggregate messages). In such cases, the timedelta mean slope remains neutral even at bursts of 9000 aggregated messages at a 1-s interval between bursts. Tests involving more devices have to be carried out in order to explore the scalability possibilities of the system. Note that system is flexible in that the actors are allocated/deployed when needed. Experiments also show that the system is suitable for non-time-critical Smart Grid traffic when messages can't be aggregated before reaching the prototype system. Finally, a simple web-based interface to visualize sensor location and sensor readings in near real-time has been implemented as the WoT interface to the Smart Grid, the Web of Energy.

The results encourage us to keep investigating in this direction. We acknowledge that our system and experiments do not cover all of the scalability and flexibility requirements in Smart Grid ICT systems but we have presented the benefits of the Actor Model for such systems and developed an initial prototype for further research and development.

6. Conclusions

While the overestimations on the initial prediction of the volume of internet connections were acknowledged and duly rectified, the fact that our society is becoming increasingly connected to the internet is undeniable. This acceleration has been promoted thanks to the advances of both silicon, which makes it possible to embed increasingly smaller computing units in everyday devices, and advances in low-power wireless protocols. At the same time, internet and web applications targeted at controlling and managing internet-connected devices are being continually improved. We are being encouraged to integrate internet-connected devices into our lives, where social network services serve as the easy-to-use interface between people and devices.

It is obvious that the Web of Things is continually extending to different application domains. Nevertheless, in order to continue this trend, it is essential that the software applications involved are built following guidelines that guarantee their performance, accessibility, and high availability. In this sense, an overall development model agreed upon and approved by different authors is becoming increasingly specific.

In this paper, the term Web of Energy is used to refer to the critical features that an architecture of the Web of Things must fulfill to be applied to the domain of Smart Grids. We highlight the requirements related to the implementation of smart functions, such as the distribution, resilience and self-healing of the system as well as the difficulties of renewing the traditional energy generation and management systems, both in terms of device and the software, which are generally interrelated. Given those challenges, the use of the Actor Model is proposed to overcome them. This paradigm is designed specifically to perform the modeling of concurrent and distributed systems, thus directly improving the inherent characteristics of distributed systems. In this sense, a proposal of middleware architecture using this paradigm for the creation of a ubiquitous sensor network has been presented in this paper.

7. Future Work

The future work to be done is focused on two different aspects. On the one hand, we want to expand the WoE application, providing it with more functionalities and allowing us to interact with the devices. We also want to integrate different types of sensors, with unequal capacities and performances.

On the other hand, characteristics of scalability, discoverability, and interoperability are only partially tackled by the WoT as it serves as the abstraction layer that comprises technologies that enable homogeneous communication among devices by means of HTTP and semantic web ontologies. However, a global IoT or WoT, the latter being an IoT empowered by web technologies, still has scalability and discoverability issues. As the number of Internet-connected devices grows, the lack of global scalability and discoverability functionalities will become more evident. In this regard, a novel IoT paradigm, the Social Internet of Things (SIoT) is gaining momentum.

In this novel paradigm, communication between social virtual objects becomes mandatory and the amount of messages exchanged between devices is expected to increase. Several works aim to provide example scenarios where SIoT can be leveraged [61–63], to name a few. In this sense, we highlight [63], where the authors use thing-to-thing social relationships to encourage communication to optimize energy usage by the Heat, Ventilation, Air Conditioning (HVAC) system in their laboratory. Results show that the comfort-aware HVAC system ensures users' thermal comfort, while at the same time reducing energy costs with respect to static and traditional methods. An ongoing challenge for us is the study of the implications of reactive, asynchronous communication together with computation distribution properties of the Actor Model in a SIoT system, in terms of performance and how these abstractions facilitate the development of this kind of applications. Further work on this subject encompasses the creation of proper simulation tools or modules to facilitate the simulation of SIoT enabled nodes and networks at multiple network layers (e.g., applications installed on top of constrained devices and fog or cloud applications).

Acknowledgments: Part of this research has been funded by the SUR of the DEC of the Generalitat de Catalunya and by the European Social Funds 2017 FI_B 00583. In addition, we would like to thank La Salle (Universitat Ramon Llull) for their support.

Author Contributions: V.C. and D.V. conceived the idea and decided the technology involved in the experiments; V.C. developed the prototype; D.V. and A.Z. supervised the work, and A.Z. contributed with relevant materials about Smart Grids domain; G.C. contributed in the related work and in the revision of the paper. All authors contributed equally to writing the paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Martín de Pozuelo, R.; Ponce de León, M.; Howard, J.; Briones, A.; Horgan, J.; Sánchez, J. Software defined utility: A step towards a flexible, reliable and low-cost smart grid. In Proceedings of the International Conference on Smart Grid Systems, Barcelona, Spain, 7–9 September 2016.
2. Gungor, V.; Sahin, D.; Kocak, T.; Ergut, S.; Buccella, C.; Cecati, C.; Hancke, G. Smart grid and smart homes: Key players and pilot projects. *IEEE Ind. Electron. Mag.* **2012**, *6*, 18–34. [CrossRef]
3. Rodríguez-Molina, J.; Martínez-Núez, M.; Martínez, J.-F.; Pérez-Aguiar, W. Business models in the smart grid: Challenges, opportunities and proposals for prosumer profitability. *Energies* **2014**, *7*, 6142–6171. [CrossRef]
4. Navarro, J.; Zaballos, A.; Sancho-Asensio, A.; Ravera, G.; Armendariz-Inigo, J.E. The information system of INTEGRIS: Intelligent electrical grid sensor communications. *IEEE Trans. Ind. Inform.* **2013**, *9*, 1548–1560. [CrossRef]
5. Shah, G.A.; Gungor, V.C.; Akan, O.B. A cross-layer QoS-aware communication framework in cognitive radio sensor networks for smart grid applications. *IEEE Trans. Ind. Inform.* **2013**, *9*, 1477–1485. [CrossRef]
6. Khan, A.A.; Rehmani, M.H.; Reisslein, M. Cognitive radio for smart grids: Survey of architectures, spectrum sensing mechanisms, and networking protocols. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 860–898. [CrossRef]
7. Maziku, H.; Shetty, S. Software Defined Networking enabled resilience for IEC 61850-based substation communication systems. In Proceedings of the 2017 International Conference on Computing, Networking and Communications (ICNC 2017), Santa Clara, CA, USA, 26–29 January 2017; pp. 690–694.
8. Zaballos, A.; Vallejo, A.; Selga, J. Heterogeneous communication architecture for the smart grid. *IEEE Netw.* **2011**, *25*, 30–37. [CrossRef]
9. Guinard, D. A Web of Things Application Architecture—Integrating the Real-World into the Web. Ph.D. Thesis, ETH Zurich, Zurich, Switzerland, 2011.
10. Caballero, V.; Vernet, D.; Zaballos, A.; Corral, G. Web of Energy: Hacia la integración inteligente para las redes de sensores en Smart Grids. In Proceedings of the XIII Jornadas de Ingenieria Telematica—JITEL2017, Valencia, Spain, 27–29 September 2017; Universitat Politècnica València: Valencia, Spain, 2017; pp. 30–39. (In Spanish)
11. Navarro, J.; Sancho-Asensio, A.; Zaballos, A.; Jiménez-Ruano, V.; Vernet, D.; Armendáriz-Iñigo, J.E. The management system of INTEGRIS. In Proceedings of the 4th International Conference on Cloud Computing and Services Science, Barcelona, Spain, 3–5 April 2014; pp. 3–5.
12. Vernet, D.; Zaballos, A.; Martín de Pozuelo, R.; Caballero, V. High performance web of things architecture for the smart grid domain. *Int. J. Distrib. Sens. Netw.* **2015**, *11*, 347413. [CrossRef]
13. Miorandi, D.; Sicari, S.; De Pellegrini, F.; Chlamtac, I. Internet of things: Vision, applications and research challenges. *Ad Hoc Netw.* **2012**, *10*, 1497–1516. [CrossRef]
14. W3C Member Submission 24 August 2015. Available online: https://www.w3.org/Submission/wot-model/ (accessed on 7 December 2017).
15. Shadbolt, N.; Berners-Lee, T.; Hall, W. The semantic web revisited. *IEEE Intel. Syst.* **2006**, *21*, 96–101. [CrossRef]
16. Martín de Pozuelo, R.; Navarro, J.; Corral, G.; Zaballos, A. Prototyping a software defined utility. *Energies* **2017**, *10*, 818. [CrossRef]
17. Iwata, J. Making Markets: Smarter Planet. Available online: https://www.ibm.com/investor/events/investor0512/presentation/05_Smarter_Planet.pdf (accessed on 12 November 2017).
18. Gartner Gartner Says the Internet of Things Installed Base Will Grow to 26 Billion Units By 2020. Available online: https://www.gartner.com/newsroom/id/2636073 (accessed on 7 December 2017).

19. Gartner Gartner Says 6.4 Billion Connected "Things" Will Be in Use in 2016, up 30 Percent from 2015. Available online: http://www.gartner.com/newsroom/id/3165317 (accessed on 7 December 2017).
20. Fielding, R.T. Architectural Styles and the Design of Network-Based Software Architectures. Ph.D. Thesis, University of California, Oakland, CA, USA, 2000.
21. Trifa, M.V. Building Blocks for a Participatory Web of Things: Devices, Infrastructures, and Programming Frameworks. Ph.D. Thesis, ETH Zurich, Zurich, Switzerland, 2011.
22. MQTT V3.1 Protocol Specification. Available online: http://public.dhe.ibm.com/software/dw/webservices/ws-mqtt/mqtt-v3r1.html (accessed on 7 December 2017).
23. Shelby, Z.; Hartke, K.; Bormann, C. *The Constrained Application Protocol (CoAP)*; Internet Engineering Task Force: Fremont, CA, USA, 2014. [CrossRef]
24. Trifa, V.; Wieland, S.; Guinard, D.; Bohnert, T. Design and implementation of a gateway for web-based interaction and management of embedded devices. In Proceedings of the 2nd International Workshop on Sensor Network Engineering, (IWSNE 09), Marina Del Rey, CA, USA, 8–10 June 2009; pp. 1–14. [CrossRef]
25. ThingWorx Enterprise IoT Solutions and Platform Technology. Available online: https://www.ptc.com/en/products/iot (accessed on 7 December 2017).
26. IBM. IBM Watson IoT—IoT Platform. Available online: https://www.ibm.com/internet-of-things/platform/watson-iot-platform (accessed on 7 December 2017).
27. Octoblu Octoblu | Integration of Everything. Available online: https://octoblu.github.io/ (accessed on 7 December 2017).
28. EVRYTHNG EVRYTHNG IoT Smart Products Platform. Available online: https://evrythng.com/ (accessed on 7 December 2017).
29. Bizer, C.; Heath, T.; Idehen, K.; Berners-Lee, T. Linked data on the web (LDOW2008). In Proceedings of the 17th International Conference on World Wide Web (WWW '08), Fira, Greece, 8–10 June 2008; ACM Press: New York, NY, USA, 2008; p. 1265.
30. World Wide Web Consortium. *RDF 1.1 Concepts and Abstract Syntax*; World Wide Web Consortium, 2014. Available online: https://www.w3.org/standards/techs/rdf#w3c_all (accessed on 7 December 2017).
31. Adida, B.; Birbeck, M.; McCarron, S.; Pemberton, S. *RDFa in XHTML: Syntax and Processing*; Recommendation. World Wide Web Consortium, 2008. Available online: https://www.w3.org/TR/2008/REC-rdfa-syntax-20081014/ (accessed on 7 December 2017).
32. Bechhofer, S. OWL: Web ontology language. In *Encyclopedia of Database Systems*; Springer: New York, NY, USA, 2009; pp. 2008–2009.
33. Guinard, D.; Fischer, M.; Trifa, V. Sharing using social networks in a composable Web of Things. In Proceedings of the 2010 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), Mannheim, Germany, 29 March–2 April 2010; pp. 702–707.
34. Gregorio, J.; de hOra, B. *The Atom Publishing Protocol*; Network Working Group, 2007. Available online: https://bitworking.org/projects/atom/rfc5023.html (accessed on 7 December 2017).
35. 1Worx ThingWorxComposer^{TM}—1Worx. Available online: http://www.1worx.co/the-thingworx-platform/thingworx-composer/ (accessed on 7 December 2017).
36. IBM Node-RED. Available online: https://nodered.org (accessed on 7 December 2017).
37. Ur, B.; Pak Yong Ho, M.; Brawner, S.; Lee, J.; Mennicken, S.; Picard, N.; Schulze, D.; Littman, M.L. Trigger-Action Programming in the Wild. In Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16), San Jose, CA, USA, 7–12 May 2016; ACM Press: New York, NY, USA, 2016; pp. 3227–3231.
38. Dijk, E.; Rahman, A.; Fossati, T.; Loreto, S.; Castellani, A. *Guidelines for HTTP-CoAP Mapping Implementations*; CoRE Working Group, 2016. Available online: https://tools.ietf.org/html/rfc8075 (accessed on 7 December 2017).
39. Collina, M.; Corazza, G.E.; Vanelli-Coralli, A. Introducing the QEST broker: Scaling the IoT by bridging MQTT and REST. In Proceedings of the 2012 IEEE 23rd International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), Sydney, Australia, 9–12 September 2012; pp. 36–41.
40. Nitti, M.; Pilloni, V.; Colistra, G.; Atzori, L. The virtual object as a major element of the internet of things: A survey. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 1228–1240. [CrossRef]
41. Khan, M.; Silva, B.; Han, K. A web of things-based emerging sensor network architecture for smart control systems. *Sensors* **2017**, *17*, 332. [CrossRef] [PubMed]

42. Cass, S. The 2016 Top Programming Languages. Available online: https://spectrum.ieee.org/static/interactive-the-top-programming-languages-2016 (accessed on 25 January 2018).
43. Usage Statistics and Market Share of Server-Side Programming Languages for Websites. Available online: https://w3techs.com/technologies/overview/programming_language/all (accessed on 7 December 2017).
44. Reactphp/React. Available online: https://github.com/reactphp/react (accessed on 7 December 2017).
45. Get Performance Insight into the Upcoming Release of PHP 7. Available online: http://www.zend.com/en/resources/php7_infographic (accessed on 7 December 2017).
46. Hewitt, C.; Bishop, P.; Steiger, R. Session 8 formalisms for artificial intelligence a universal modular ACTOR formalism for artificial intelligence. In *Advance Papers of the Conference*; Stanford Research Institute: Menlo Park, CA, USA, 1973; Volume 3, p. 235.
47. Agha, G. *Actors: A Model of Concurrent Computation in Distributed Systems*; MIT Press: Cambridge, MA, USA, 1986; ISBN 0-262-01092-5.
48. Sutter, H. The free lunch is over: A fundamental turn toward concurrency in software. *Dr. Dobb's J.* **2005**, *30*, 202–210.
49. Diaz Sanchez, D.; Simon Sherratt, R.; Arias, P.; Almenarez, F.; Marin, A. Enabling actor model for crowd sensing and IoT. In Proceedings of the 2015 International Symposium on Consumer Electronics (ISCE), Madrid, Spain, 24–26 June 2015; pp. 1–2.
50. Hiesgen, R.; Charousset, D.; Schmidt, T.C.; Wählisch, M. Programming actors for the internet of things. *ERCIM NEWS*, 2015; p. 25.
51. Armstrong, J. *Programming Erlang: Software for a Concurrent World*; Pragmatic Bookshelf: Raleigh, NC, USA, 2007.
52. Lightbend Akka. Available online: https://akka.io/ (accessed on 7 December 2017).
53. Fowler, M. Event Sourcing. Available online: https://martinfowler.com/eaaDev/EventSourcing.html (accessed on 13 January 2018).
54. An Open Source MQTT v3.1 Broker. Available online: http://mosquitto.org/ (accessed on 7 December 2017).
55. Lightbend Playframework. Available online: https://playframework.com/ (accessed on 7 December 2017).
56. MongoDb. Available online: https://www.mongodb.com/ (accessed on 13 January 2018).
57. Bormann, C.; Hoffman, P. *Concise Binary Object Representation (CBOR)*; Internet Engineering Task Force: Fremont, CA, USA, 2013; Available online: https://tools.ietf.org/html/rfc7049 (accessed on 7 December 2017).
58. DataPort—PecanStreet. Available online: https://dataport.pecanstreet.org (accessed on 13 January 2018).
59. Selga, J.M.; Zaballos, A.; Navarro, J. Solutions to the computer networking challenges of the distribution smart grid. *IEEE Commun. Lett.* **2013**, *17*, 588–591. [CrossRef]
60. Silos, Á.; Señís, A.; de Pozuelo, R.; Zaballos, A. Using IEC 61850 GOOSE Service for Adaptive ANSI 67/67N Protection in Ring Main Systems with Distributed Energy Resources. *Energies* **2017**, *10*, 1685. [CrossRef]
61. Atzori, L.; Girau, R.; Martis, S.; Pilloni, V.; Uras, M. A SIoT-aware approach to the resource management issue in mobile crowdsensing. In Proceedings of the 2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN), Paris, France, 7–9 March 2017; pp. 232–237.
62. Girau, R.; Martis, S.; Atzori, L. Lysis: A platform for iot distributed applications over socially connected objects. *IEEE Internet Things J.* **2017**, *4*, 40–51. [CrossRef]
63. Marche, C.; Nitti, M.; Pilloni, V. Energy efficiency in smart building: A comfort aware approach based on Social Internet of Things. In Proceedings of the 2017 Global Internet of Things Summit (GIoTS), Geneva, Switzerland, 6–9 June 2017; pp. 1–6.

sensors

MDPI

Article

Ontology-Defined Middleware for Internet of Things Architectures

Víctor Caballero *, Sergi Valbuena, David Vernet and Agustín Zaballos

Engineering Department, Universitat Ramon Llull (URL), La Salle, 08022 Barcelona, Spain; sergi.valbuena@students.salle.url.edu (S.V.); david.vernet@salle.url.edu (D.V.); agustin.zaballos@salle.url.edu (A.Z.)

* Correspondence: victor.caballero@salle.url.edu; Tel.: +34-93-290-2436

Received: 15 January 2019; Accepted: 26 February 2019; Published: 7 March 2019

Abstract: The Internet of Things scenario is composed of an amalgamation of physical devices. Those physical devices are heterogeneous in their nature both in terms of communication protocols and in data exchange formats. The Web of Things emerged as a homogenization layer that uses well-established web technologies and semantic web technologies to exchange data. Therefore, the Web of Things enables such physical devices to the web, they become Web Things. Given such a massive number of services and processes that the Internet of Things/Web of Things enables, it has become almost mandatory to describe their properties and characteristics. Several web ontologies and description frameworks are devoted to that purpose. Ontologies such as SOSA/SSN or OWL-S describe the Web Things and their procedures to sense or actuate. For example, OWL-S complements SOSA/SSN in describing the procedures used for sensing/actuating. It is, however, not its scope to be specific enough to enable a computer program to interpret and execute the defined flow of control. In this work, it is our goal to investigate how we can model those procedures using web ontologies in a manner that allows us to directly deploy the procedure implementation. A prototype implementation of the results of our research is implemented along with an analysis of several use cases to show the generality of our proposal.

Keywords: heterogeneity; middleware; semantic; ontology; behaviour; web-of-things

1. Introduction

The recent growth of the Internet has fostered the interaction between many heterogeneous technologies. The term Internet of Things (IoT) [1] has been coined as the umbrella term that refers to this new reality. The heterogeneity of the IoT encompasses both hardware and software. At the hardware level, an amalgamation of devices from multiple vendors take part in the IoT. At the software level, those devices use multiple Internet protocols and data exchange formats. Device manufacturers choose what seems to fit their needs in describing and operating their devices. Hence, the current IoT scenario is filled with devices/Things that are not fully interoperable. Even stronger lack of interoperability can be found in vertical domain specific silos of IoT. In order to facilitate the communication between devices and the integration of them into systems and systems of systems, researchers have conducted work that aims at providing a common communication interface. The Web of Things (WoT) [2] provides a common interface by translating the heterogeneity into the homogeneity of the Web 3.0. To achieve a common language, the WoT refers to Hyper-Text Transfer Protocol/Secure (HTTP/S) as the standard communication protocol and to Semantic Web [3] as the common information exchange structure.

An example that takes advantage of both IoT and WoT technologies is as follows. In the context of a Smart Home, there are several electrical appliances which consume or produce energy. On the one hand, home appliances that consume energy can be an air conditioner, vacuum cleaner, washing machine, clothes dryer, dishwasher, oven, stove, mobile charger, lights and so on. On the other hand, devices that produce energy usually take advantage of Renewable Energy Sources (RESs) like photo-voltaic panels. In order to take advantage of both production and consumption, those devices need to exchange information between themselves. A course of action would be, for example, to autonomously turn on the clothes dryer during the day when there is enough sunlight to produce enough energy for it to function. It is very unlikely, though, that they all use the same communication protocol and data exchange format. Some will use wireless protocols like Bluetooth, ZigBee, Z-Wave or 6LoWPAN, or wired ones like Ethernet, RS-232, or USB, to name a few. The WoT integration pattern proposes to use gateways as translators between those protocols (some are proprietary) and HTTP/S, along with data exchange format translation, which occurs from a variety of formats such as EXI, JSONB or unstructured data to a semantically annotated Web 3.0 standards, such as JSON or XML. Scenarios like the one presented are very likely to occur. Each scenario, however, will comprehend different types of devices that use different protocols and different data exchange formats. The simple scenario of pairing a light bulb from brand A with a switch from brand B can be different in each household. In such a simple use case, the task of describing each specific scenario can be tedious. Therefore, there is a need to describe the execution flow or behaviour of those devices in an abstract manner, thus the description can be used for multiple and diverse devices with the same capabilities.

1.1. Motivation

A large number of heterogeneous devices share the same capabilities and concrete services, with concrete, shared behaviours can be requested from them. Normally, the description of the services and the execution process that they follow are specified and implemented in multiple manners, which pose an obstacle for the shareability and interpretation of their execution flow. A great effort in providing a common abstraction and interoperability layer is being done in the framework of the IoT/WoT. The World Wide Web Consortium (W3C), for example, is leading an effort in order to provide a common communication and abstraction layer. There is also a growing tendency to describe services using Semantic Web standards such as OWL-S [4], W3C Web of Things (WoT) Thing Description (TD) [5] or the oneM2M Base Ontology [6]. Our vision is that we can share and reuse a concrete flow of execution among multiple and heterogeneous IoT deployments using existing standards. We can leverage Semantic Web Ontologies to model, and execute by means of a specialized engine, the behaviour of the devices that lay in the lower layer of the IoT, by providing a common template (abstraction) that defines such behaviour. This template can be reused by as many Things and services as needed in a variety of heterogeneous deployments.

1.2. Contributions

This work aims to address the challenge of describing and executing the same behaviour in multiple and heterogeneous IoT scenarios. In heterogeneous IoT scenarios, where devices from different manufacturers are aimed at providing the same service, it is often found that their heterogeneity prevents developers from reusing the same solution for each device. We address that challenge by using Finite-State Machines (FSMs) to model their behaviour, translating the FSM into in abstract description framework, namely Resource Description Framework (RDF) [7], and developing an engine capable of interpreting and executing the RDF model. The main contributions of this work are:

1. Abstraction of flow of control or behaviour over the heterogeneity of the IoT using existing standards such as RDF, SPARQL Protocol and RDF Query Language (SPARQL), FSM and HTTP/S. This abstraction brings reusability.
2. A prototype implementation of an engine and web interface that allows direct deployment, interpretation and execution of a concrete instance obtained by materializing the abstraction.
3. Analysis of several use cases that demonstrates the generality of the abstraction.

This article is organized as follows. Section 2 gives a brief overview of IoT/WoT standards relevant to our goal and analyses related work on control of flow/behaviour modelling using Semantic Web technologies. Section 3 gives an overview of the architecture that supports our approach. Section 4 introduces and models the use case that is used to explain our solution and Section 5 enhances the model using ontologies. Section 6 explains how we use the template (abstraction) and execute the behaviour described in it. Section 7 analyses several use cases in relation to our approach and finally Section 8 concludes the paper and presents future work.

2. State of the Art

Given the massive number of services and processes that IoT enables, it has become almost mandatory to describe such services and processes to enhance interoperability, allowing them to be automatically discovered and called by multiple, heterogeneous systems [8]. Ontologies such as OWL-S [4] and low-level specifications such as the TD [5] or the oneM2M Base Ontology [6] can be used together to describe IoT/WoT systems, fostering interoperability. OWL-S helps software agents to discover the web service that will fulfil a specific need. Once discovered, OWL-S provides the necessary language constructs to describe how to invoke the service. It allows describing inputs and outputs. Thanks to the high-level description of the service, it is possible to compose multiple services to perform more complex tasks. In OWL-S, service description is organized into four areas: the *process model*, the *profile*, the *grounding* and the *service*. Specifically, the process model describes how a service performs its tasks. It includes information about inputs, outputs, preconditions and results. Similarly, the oneM2M Global Initiative [9] defines a standard for machine-to-machine communication interoperability at the semantic level, the one M2M Base Ontology, which is a high-level ontology designed to foster interoperability among other ontologies using equivalences and alignments. The TD "is a central building block in the W3C Web of Things (WoT) and can be considered as the entry point of a Thing [...]. The TD consists of semantic metadata for the Thing itself, an interaction model based on WoT's Properties, Actions, and Events paradigm, a semantic schema to make data models machine-understandable, and features for Web Linking to express relations among Things". Both oneM2M Base Ontology and the ontology defined in TD strive for interoperability among multiple IoT applications and platforms, each one covering a large set of use cases, so there is also a work in progress to align the oneM2M ontology with the TD ontology [10]. Orthogonal to these ontologies, the SOSA/SSN ontology is an ontology for describing sensors and their observations. Among other concepts, it defines the class sosa:Procedure, which "is a reusable workflow, protocol, plan, algorithm, or computational method that can be used, among others, to specify how an observation activity has been made [...], (the sosa:Procedure) can be either a record of how the actuation has been performed or a description of how to interact with an actuator (i.e., the recipe for performing actuations)" [11]. How much detail is provided to define such procedures is beyond the scope of SOSA/SSN. It is our vision that detailed and deployable procedure descriptions could fit into and be orthogonal to the models just presented. This work aims at modelling the flow of control for a given service. Therefore, our focus is on modelling the procedure/behaviour of Things using ontologies in a manner that allows us to directly deploy the behaviour implementation.

Work aimed at modelling the behaviour of Things using FSMs and Web Ontology Language (OWL) exists in the literature. On [12], they aim to represent Unified Modeling Language (UML) FSMs using OWL, performing an almost one-to-one translation between UML concepts and OWL classes. Although their mapping from UML to OWL allows for a more machine-readable information structure, its complexity makes it unpractical to use for our objective. The work done in [13] presents a simpler FSM model. UML is used to specify platform independent navigation guides in web applications. They use OWL to describe a model for FSMs which serves as a meta-model for semantic web descriptions of the navigation guides on the Semantic Web. There also exists some scientific literature devoted not only to create a model to express the behaviour of a service but also to interpret and execute the behaviour. In [14], the aim is to develop an FSM that a special server can read and translate to executable entities. These executable entities are executed later by robots. They successfully satisfy their objective, their OWL FSM is domain specific and it includes properties that solve the complexity of their use case but make the FSM too complex for our goal. The goal in [15] is to model an FSM that can be easily translated into Programmable Logic Controller (PLC) code. This model is domain-agnostic as it is not directly merged with the PLC domain. Works [14,15] show how a semantic FSM can be translated and executed by machines, although their application field is industrial and not on the WoT. In our approach, we do not want to express or translate the model in languages other than OWL so it maintains the benefits that OWL gives. Our aim is to build a model that is directly interpreted by different machines.

Our solution aims to be as middleware-agnostic as possible, being a building block for middleware that implement other technical challenges. Therefore, we rely on standard and well-known architectures to contextualize our proposal. We have considered two big standardization and recommendation bodies devoted to promoting interoperability among IoT verticals. An international partnership project for IoT standards, the oneM2M Global Initiative [9] aims to minimize M2M service layer standards by consolidating standardization activities among several member organizations. The W3C Web of Things Working Group [5] aims to develop standards for the Web of Things; their goal is also to reduce fragmentation between IoT silos. For this work, we are interested in the architecture standards and recommendations that oneM2M and W3C WoT Working Group propose. As our objective is to contextualize and provide a prototype implementation for our solution, the W3C WoT Architecture recommendation seems more suitable for that purpose for its simplicity. The architecture is explained in Section 3.

3. Architecture

The objective of this paper to provide an abstraction mechanism to describe and execute the behaviour of Things in any WoT architecture. This section explains the architecture used in our proposal. There is a common trend on IoT architectures to represent physical devices with virtual representations or agents that enhance their capabilities. Such virtual representations are called Virtual Object (VO) [16]. Virtual Objects (VOs) are usually placed in fog or cloud middleware [16–18]. The behaviour templates developed in this work are interpreted and executed where those VO reside, thus more resources are available to execute the behaviour that the templates describe. The first layer in the seminal work of the WoT [2] defines an Accessibility Layer that enables each physical device to the web by translating each IoT protocol to HTTP/S and providing a common RESTful (REpresentational State Transfer) Application Programming Interface (API) to interact with them. As seen in Figure 1, we assume that physical devices are exposed via HTTP/S in order to interact with the VOs. A basic division between layers can be made. The first layer is composed of physical devices, which is the Perception layer. The second layer is responsible for enabling physical devices to the web, the Accessibility Layer. The third layer, the Virtual Object layer, is composed of VOs that interact with the Perception layer via the Accessibility layer. As explained in [2] and in the W3C WoT Architecture [5], Things can become Web Things (web-enabled Things) by three mechanisms. If they have enough computing resources to implement the HTTP/S protocol, they can be directly connected to the web. Otherwise, they can use Gateways (GWs) or cloud/fog infrastructures that translate their proprietary protocols or non-standard data exchange formats into web standards. The template developed in this paper is embodied by a VO. Each VO is given the capability to interact with physical devices through the Accessibility layer. Therefore, actions encoded by an HTTP/S call the Accessibility layer, which in turn informs the physical device to actuate in the real world. Thanks to this action abstraction, relationships can be made between one VO and multiple physical devices. It is only necessary to change the Uniform Resource Identifier (URI) actions refer to; this is explained in more detail in Section 5.3. The architecture is shown in Figure 1 where different settings can be appreciated. The following lists the settings, from left to right:

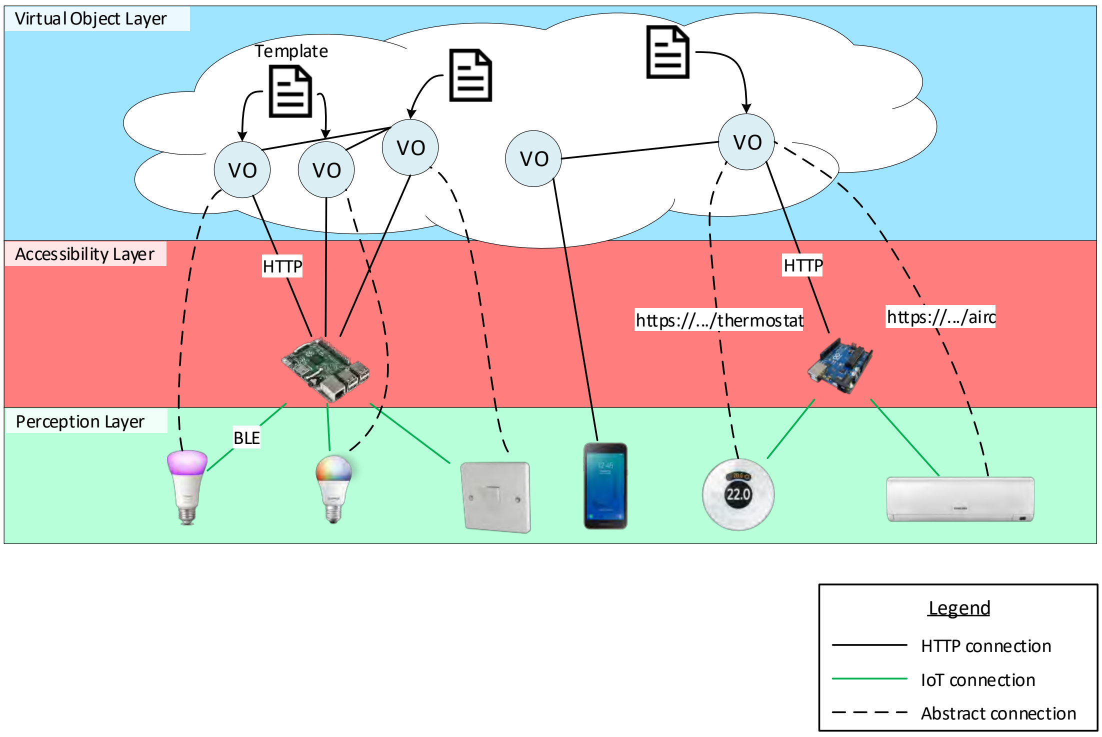


Figure 1. Network architecture used in our proposal. Three layers are depicted: (1) Perception layer, where physical devices reside; (2) the translation between Internet of Things (IoT) protocols and Hyper-Text Transfer Protocol/Secure (HTTP/S) is made at the Accessibility layer and; (3) the Virtual Object layer is comprised of virtual representations of physical devices, each Virtual Object (VO) embodies a template that describes the behaviour for physical devices. Straight lines represent direct connections between nodes, which means that no intermediate protocol or data exchange format is needed (nodes at the Accessibility layer are responsible for performing such actions). Dashed lines represent that the VO at one end of the line handles the physical device at the other end of the line. Finally, black straight lines represent that nodes are connected through HTTP/S and that there exists an HTTP/S Application Programming Interface (API) compatible with nodes at both ends of the line. Green, straight lines represent that nodes are connected through IoT protocols, such as Bluetooth Low Energy (BLE).

1. The first setting is composed of two smart light bulbs from different brands (they use different communication protocols and/or data exchange formats) and a light switch at the Perception layer. There is a Raspberry Pi at the Accessibility layer that translates IoT protocols from the physical devices to an HTTP/S API. Finally, three VOs dwell in the VO layer and each one embodies a template. Note that the same template is used for each one of the VOs that communicate with the smart light bulbs, which means that the same template is used to control both of them. The template for the light switch describes the behaviour for controlling (turn on/off) both smart light bulbs. The control is made at the VO layer. More precisely, the switch VO sends commands and receives messages from the two VOs that represent the smart light bulbs.
2. The second setting is composed of a smartphone at the Perception layer that directly connects to a VO at the VO layer through HTTP/S. To lighten the image, the template the VO embodies is not drawn. The interaction is made possible via an app installed at the smartphone.
3. The third setting comprehends a thermostat and an air conditioner machine at the Perception layer. They both connect to an Arduino board at the Accessibility layer, which exposes their functionality as an HTTP/S API. In this scenario, a single VO is responsible for managing

the thermostat and the air conditioner. This can be achieved by simply using both Uniform Resource Locators (URLs), *https://.../thermostat* and *https://.../airc*, as the target for the template actions. Finally, this VO communicates with the VO that represents the smartphone, enabling for controlling the temperature of the room.

The model architecture depicted in Figure 1 shows the interaction patterns between physical devices and our proposal (templates) in a WoT architecture. One of the features of interest in our proposal is that it is middleware-agnostic. Therefore, critical challenges such as full-fledged security and privacy are implemented by concrete middleware. For example, the VICINITY project [19] defines a Virtual Neighbourhood and is a platform that links various ecosystems providing interoperability as a service for infrastructures in the IoT. It tackles privacy by allowing the owner of Things to define sharing access rules of devices and services. The communication between VICINITY Nodes occurs between a Peer-to-Peer (P2P) network based on those rules. This network supports VICINITY Nodes with security services such as end-to-end encryption and data integrity [20]. The IoT ecosystems are connected to VICINITY through the VICINITY Node. Hence, security and privacy outside VICINITY is handled by the manager of the IoT ecosystem. Security in that ecosystem could be an exchange of identifiers and encryption keys between VOs and physical devices or gateways, representing the weakest phase when security could be compromised.

Integration with other middleware involves three main efforts: (1) to implement an engine to interpret the flow of execution defined by the FSM template (which can be reused by middleware developed with the same programming language); (2) to adapt the input and output channels of the FSM engine with an HTTP/S interface provided by the middleware; and to provide an RDF and SPARQL API to the FSM engine.

As shown in Figure 2, our approach complements third-party middleware by providing several desired properties. By combining RDF and the model of execution flow that FSMs provide, we enable a programming abstraction. The programming abstraction can be shared among multiple engine implementations in different middleware. The ease of deployment is facilitated by the fact that behaviour and executor are decoupled. Behaviours for web-enabled physical devices can be created ad hoc when needed and deployed, as the necessary software to execute the behaviour already exists. A Triplestore Database (TDB) or RDF API and SPARQL endpoint is also needed for the FSM engine in order to store the template and save the data received during runtime. Guards, conditions and actions use these data (see Section 5). As seen in Figure 2, our approach relies on HTTP/S in order to enable communication with the Perception Layer. Therefore, it is mandatory for third-party middleware to have an HTTP/S interface. Two pathways are shown in Figure 2. The first pathway assumes a sensor (thermostat/thermometer) with an HTTP/S client that pushes data to the middleware. The HTTP/S server in the middleware receives this request and sends it to the input interface of the FSM engine and triggers the FSM logic, which is read from the template instance. The FSM does not return any response with data as it may not be available at that time. However, the middleware is still able to respond to the HTTP/S request. The second pathway is the actuator pathway. There, the actuator is directly addressable as it is an HTTP/S server. Following the process triggered by the sensor, the FSM may generate an action to turn on the actuator. For example, if the temperature is higher than 27 °C, the course of action would be to turn on the air conditioner. This logic and action are written in the FSM template instance. The FSM generates an action which is sent to the HTTP/S client provided by the middleware. An HTTP/S request is sent to the actuator and the actuator may respond with valuable data, which is then used as the input of the FSM.

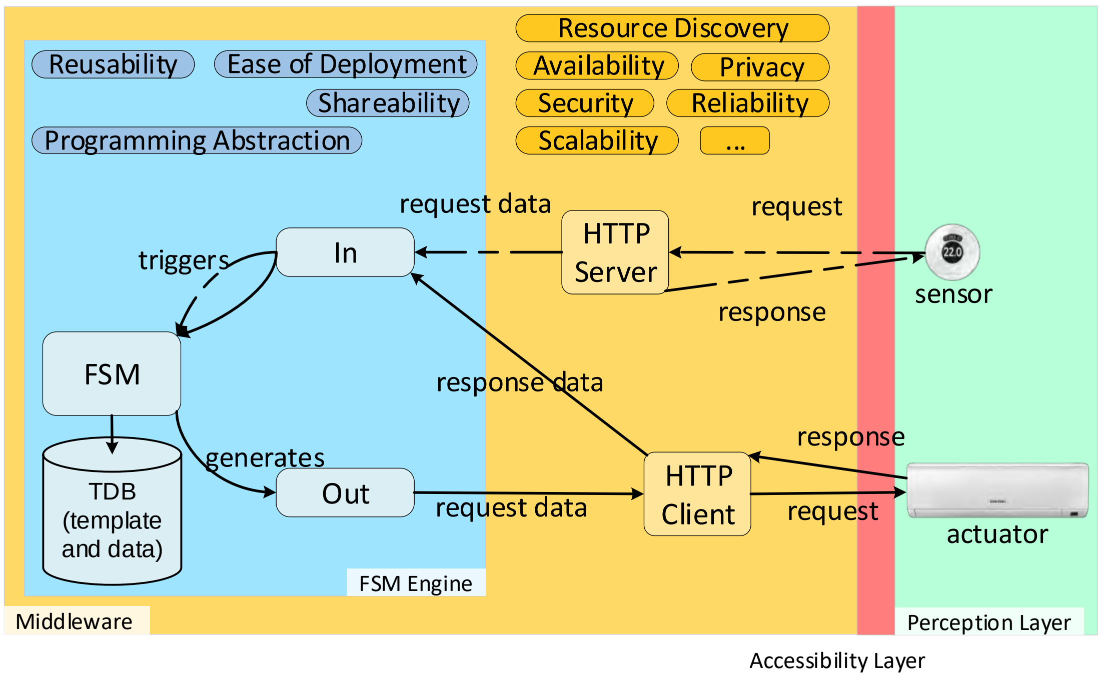


Figure 2. Functional blocks for third-party middleware integration. The layers are: Finite-State Machine (FSM) Engine and Third-party middleware (Middleware), Accessibility Layer and Perception Layer. The FSM Engine complements the middleware. Only two components are mandatory in the middleware layer to enable our approach. Arrows indicate the flow of information between components in each layer. Dashed lines represent the sensor pathway and straight lines represent the actuator pathway.

4. An Example to Model

This section introduces the use case that is modelled throughout the paper. Our objective is to model a template for a VO that allows users to exchange their energy profile information and check if they are compatible. For the sake of simplicity, we define "compatibility" of energy profiles as "similarity" of energy profiles. The use case is framed in the context of the Social Internet of Things (SIoT), which endows devices with the ability to belong to social networks of devices. The Lysis [21] platform is an implementation that enables the SIoT. There, VOs are extended to become social and create relationships; they become Social Virtual Objects (SVOs). Several types of relationships can be created. We are interested in Ownership Object Relationship (OOR) and Social Object Relationship (SOR) relationships. OORs are established among heterogeneous objects that belong to the same user. SORs are defined when objects come into contact (close range) because their owners come in touch with each other during their lives. The overall context of the use case we want to model is as follows:

1. A user interested in being part of a prosumer community to benefit from the community and achieve certain goals installs a discovery app on his/her smartphone. The smartphone becomes an SVO [21].
2. The potential Prosumer Community Group (PCG) user installs apps or allocates apps in the cloud that enable retrieving data from his/her energy-related devices (devices that produce or consume energy), enabling social capabilities for each one (SVO).
3. Devices and Distributed Energy Resources (DERs) that consume and/or produce energy owned by an owner establish an OOR relationship (Figure 3, 1). This relationship is then leveraged to

obtain energy-related data from DERs that belong to the same owner. These data can be both aggregated and stored or stored individually for each device.

4. The data obtained is used to build an energy user profile during a time period. The profile is then stored as part of the discovery app, and thus enabling the smartphone and other social devices to retrieve such information. A social device is a device that the user (prosumer) usually carries with himself/herself.
5. When the user establishes a social relationship with another potential user (Figure 3, 2), both social devices (i.e., smartphones) also establish social relationships (Figure 3, 3) and exchange energy profiles and goals. Then, the SVOs determine if they are compatible or not.

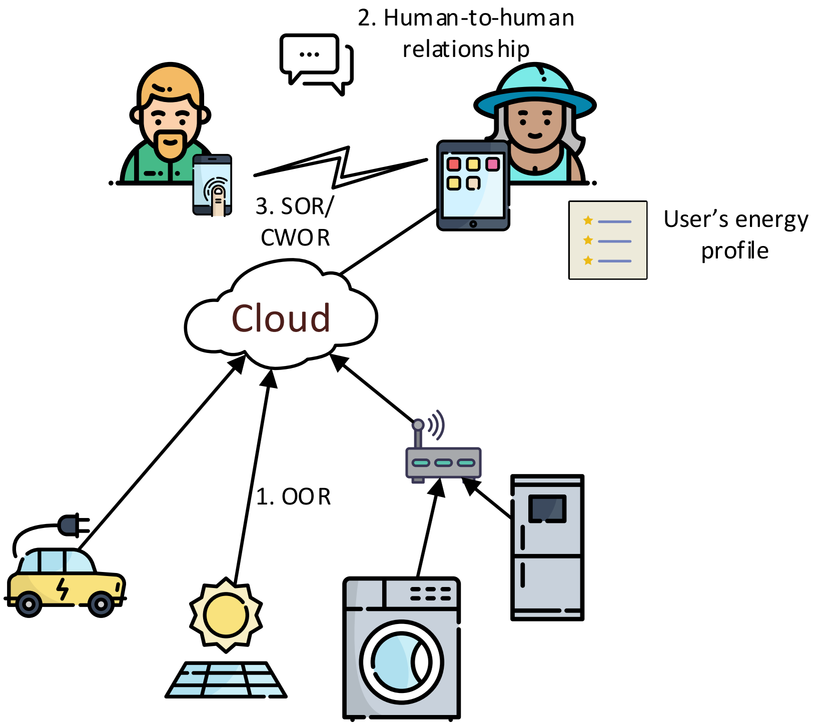


Figure 3. Prosumer Community Group formation leveraging Social Internet of Things. Firstly, prosumers and appliances establish an Ownership Object Relationship (OOR) via the augmented Social Virtual Object (SVO) (1). When prosumers come close together and establish and interact (2), their devices establish Social Object Relationship (SOR) or Co-work Object Relationship (CWOR) relationships and exchange the energy profiles of the prosumers (3).

Our objective is to model the behaviour expressed in Item 5. The model starts describing the steps that take place when one SVO meets another SVO and ends when the compatibility results between them have been exchanged. The initial peering process is assumed to be done by a service implemented in Lysis [21]. In addition, the energy profile is supposed to exist in the device. The process is described as:

1. Wait for a peer to be available.
2. Send the energy profile to the peer.
3. Wait until the peer's energy profile is received.
4. Check the compatibility between the device's and peer's profiles.
5. Inform of the compatibility result to the peer. The results are exchanged because the compatibility check of each device could be different.
6. Wait until the peer's compatibility result has been received.

To model the behaviour, we have chosen an FSM because it enables us to describe a sequence of actions to be executed and the use of conditions to modify the execution's flow. In order to visualize the FSM, we first model the behaviour using StarUML [22] following UML standards. Figure 4 shows the execution flow using states and conditions. The aim of UML is to provide a standard manner to model the design and behaviour of a system; it is made to be comprehended by humans and not by machines. However, our interest is to enable machines to interpret and execute this behaviour. OWL provides explicit meaning, making it easier for machines to automatically process and integrate information. From the options discussed in Section 2, we have chosen the FSM ontology developed in [13]. Some properties and classes have been added to integrate it with the web, others have been renamed to follow W3C naming conventions on ontologies. The modified ontology is used to describe the system's behaviour. Note that the FSM is designed in a manner that both VOs (each one interacting with one of the physical devices shown in Figure 3) share the same behaviour.

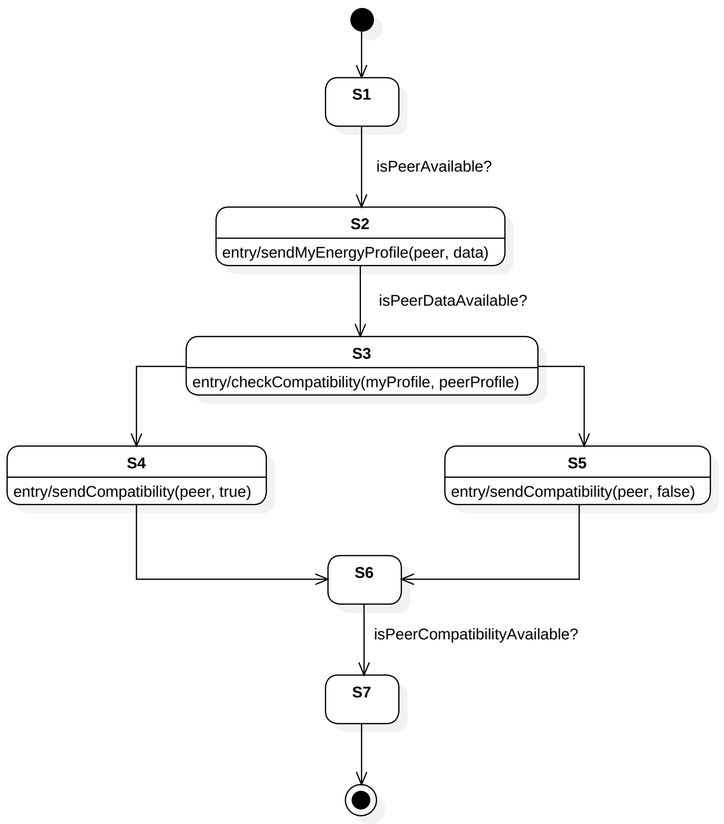

S1
isPeerAvailable?
S2
entry/sendMyEnergyProfile(peer, data)
isPeerDataAvailable?
S3
entry/checkCompatibility(myProfile, peerProfile)
S4
entry/sendCompatibility(peer, true)
S5
entry/sendCompatibility(peer, false)
S6
isPeerCompatibilityAvailable?
S7

Figure 4. Finite-State Machine representation using Unified Modeling Language.

5. Behaviour Modelling Using Ontologies

Throughout the rest of the paper, the prefix fsm: is used to shorten its URI. The siot: prefix represents some classes and properties specific to the domain of the use case. Finally, the: prefix is used to reference the file or ontology where the own model is stored.

5.1. Skeleton of the FSM

The first step is to define the skeleton of the FSM, it contains the instance of the machine, its states and its transitions.

As shown in Listing 1, an FSM is defined by the class fsm:StateMachine and the states as fsm:State. The initial state is also defined as fsm:InitialState, the final state as fsm:FinalState and intermediate states as fsm:SimpleState, these three classes extend from fsm:State. An fsm:StateMachine instance should have their states associated with the property fsm:hasStateMachineElement. Each state can have some exit or entry actions; they are associated with the property fsm:hasEntryAction or fsm:hasExitAction.

The transitions between states are described by the class fsm:Transition. As seen in Listing 2, a transition has a source state with the property fsm:hasSourceState and a target state with fsm:hasTargetState. A transition can also contain none or more guards associated with the property fsm:hasGuard.

```
### The FSM
:siot_fsm rdf:type owl:NamedIndividual ,
                   fsm:StateMachine ;
          fsm:hasStateMachineElement :S1_WatingForPeerState ,
                                     :S2_InformationExchangeState ,
                                     :S3_CheckCompatibilityState ,
                                     :S4_IsCompatibleState ,
                                     :S5_NotCompatibleState ,
                                     :S6_WaitingForPeerCompatibilityState ,
                                     :S7_CompareCompatibilitiesState .


### States
:S1_WatingForPeerState rdf:type owl:NamedIndividual ,
                                fsm:InitialState ,
                                fsm:State .

:S2_InformationExchangeState rdf:type owl:NamedIndividual ,
                                      fsm:SimpleState ,
                                      fsm:State ;
                             fsm:hasEntryAction :sendMyData .

:S3_CheckCompatibilityState rdf:type owl:NamedIndividual ,
                                     fsm:SimpleState ,
                                     fsm:State ;
                            fsm:hasEntryAction :checkCompatibility .

:S4_IsCompatibleState rdf:type owl:NamedIndividual ,
                               fsm:SimpleState ,
                               fsm:State ;
                      fsm:hasEntryAction :sendImCompatible .

### ...

:S7_CompareCompatibilitiesState rdf:type owl:NamedIndividual ,
                                         fsm:FinalState ,
                                         fsm:State .
```

Listing 1. Finite-State Machine and states.

5.2. Guards and Conditions

The purpose of a guard, defined by the class fsm:Guard, is to provide a method to evaluate if the transition has to be travelled. The guard can contain none or more conditions to express what has to be evaluated. It can also have none or more actions to be executed if the guard evaluates true.

A condition is defined with the class fsm:Condition; it contains a body object marked as fsm:Body that represents the condition's content. This body has a string associated with fsm:hasContent that contains the actual condition represented as a SPARQL query. The body should also include its type associated with fsm:hasBodyType to indicate the type of the body's content. Even though it is possible to use different types, an ASK SPARQL query is needed when used on conditions, as the main focus is to work directly with RDF data and ontologies. The reason to use the type property is that the body object is also used as the body and the value for the header "Content-Type" to be sent on HTTP/S requests, where it can take other formats such as RDF or JavaScript Object Notation (JSON). Some conditions are modelled on Listing 4. The procedure to evaluate the transitions is:

- Retrieve all the state's exit transitions. Note that only one of these transitions should be true at the same time, otherwise different executions of the same FSM may lead to different results if the order of the evaluation of transitions changes.
- For each transition, retrieve its guards (Listing 3). If it has no guards, then that transition is considered feasible. If any guard is true, then the transition is also feasible, which means that an OR condition is applied between all the guards. The utility of having multiple guards is to have different actions that are executed only under certain cases and not each time the transition is travelled.
- For each guard, retrieve its conditions. Another OR operation is applied between these conditions, and if any of them is true, the guard also evaluates to true. It is important to evaluate all the guards and to not stop when one of them is true, as all the guards' actions must be executed if their guard is true.
- The body of each condition should be an ASK query. An ASK query is a type of query that contains a pattern of triples, it searches for a set of triples that match the pattern. If any matching set is found, the query returns true, and it is false otherwise. The query is evaluated against the RDF database. The condition evaluates the same as the ASK query.

```
:S1_to_S2_Transition rdf:type owl:NamedIndividual ,
                              fsm:Transition ;
                     fsm:hasSourceState :S1_WatingForPeerState ;
                     fsm:hasTargetState :S2_InformationExchangeState ;
                     fsm:hasTransitionGuard :S1_to_S2_T_Guard .

:S3_to_S4_Transition rdf:type owl:NamedIndividual ,
                              fsm:Transition ;
                     fsm:hasSourceState :S3_CheckCompatibilityState ;
                     fsm:hasTargetState :S4_IsCompatibleState ;
                     fsm:hasTransitionGuard :S3_to_S4_T_Guard .

```

Listing 2. Transitions.

```
:S1_to_S2_T_Guard rdf:type owl:NamedIndividual ,
                           fsm:Guard ;
                  fsm:hasGuardCondition :IsPeerAvailable .

:S3_to_S4_T_Guard rdf:type owl:NamedIndividual ,
                           fsm:Guard ;
                  fsm:hasGuardCondition :IsPeerCompatible .
```

Listing 3. Guards.

```
:IsPeerCompatible rdf:type owl:NamedIndividual ,
                           fsm:Condition ;
                  fsm:hasContent
        """
            PREFIX siot: <file:///D:/projects/ontologies/siot/siot.owl#>
            PREFIX owl: <http://www.w3.org/2002/07/owl#>
            PREFIX fsm: <file:///D:/projects/ontologies/fsm/fsm.owl#>

            ASK {
                self: siot:hasPeer ?peer .
                self: siot:hasCompatibility ?compatibility .
                ?compatibility siot:withPeer ?peer .
                ?compatibility fsm:hasContent \"true\" .
            }
        """ .
```

Listing 4. Conditions.

5.3. Actions

An action requests or sends data via HTTP/S to another node in the architecture. The HTTP Vocabulary in RDF 1.0 [23] ontology by W3C is used to define actions. An action is defined with the class fsm:Action and http:Request at the same time. An action has a body, associated with fsm:hasBody, which is sent with the request. The body follows the same structure used in the conditions; it can have different types like RDF (Listing 5), JSON and others. Moreover, a timeout can be specified to limit the amount of time that an action waits for the response, it is specified in milliseconds with the property fsm:hasTimeoutInMs. A request method like GET or POST is indicated with the property http:mthd. This property references an object and not a literal, W3C has already defined the standard method objects in [23], these objects are the preferred ones to use. Therefore, as templates are not bound to perform actions using the same URI, a template can perform actions on multiple nodes. This, in conjunction with the FSM model, enables the templates or, more precisely, their instantiation, the VOs, to interact with multiple physical devices and services, such as other VOs. This is the case of the third scenario described in Figure 1, where a single VO is able to orchestrate two different physical devices. The VO also communicates with the smartphone described in the second scenario via its VO.

The action has a URI to express where the request is done. This URI can be expressed in different ways; our two approaches are:

- Use an absolute URI with the property *http:absoluteURI* to express the final address as in Listing 6. This is used when we know in advance where the request is done and that the address will not change.
- Use a prototype URI with the property fsm:hasPrototypeURI, shown in Listing 7. A prototype is an object of the class fsm:Prototype that defines the structure of a URI with one or more placeholders that are replaced during runtime. A schema of the classes and properties involved is presented in Figure 5. This is used when we do not know in advance the final address or when it changes frequently. An absolute URI is computed from the prototype each time the action needs to be executed. The replacements of parameters are not done on the ontology; this newly computed URI is built separately in program memory. A prototype defines the URI's structure with a string linked with fsm:hasStructure, for example *http://localhost:9000/[fsm_id]/send_data*, where *[fsm_id]* is the placeholder. The prototype has an fsm:Parameter for each placeholder, linked via the property fsm:hasParameter. The purpose of the parameter is to indicate what placeholder needs to be replaced and to provide a function that generates the value to fill the placeholder. Each parameter has the placeholder itself as a string (like *[fsm_id]*) linked with fsm:hasPlaceholder and a SPARQL

query as a string linked with fsm:hasQuery. The query must be a SELECT query that returns only one value. The value returned by the query is used to replace on the URI the placeholder with the same parameter key.

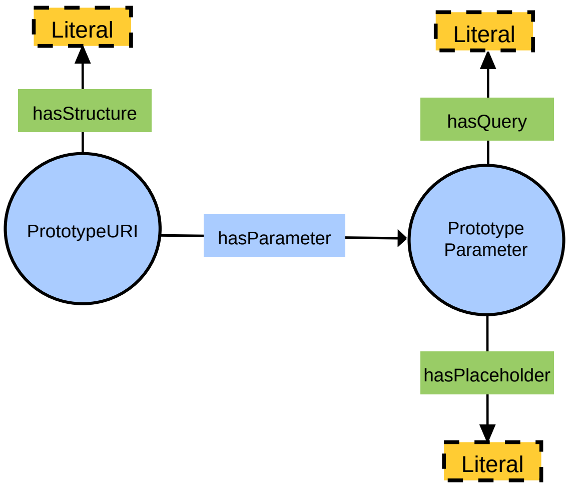


Figure 5. Schema of PrototypeURI.

```
:sendImNotCompatibleBody rdf:type owl:NamedIndividual ,
                                  fsm:Body ;
                         fsm:hasBodyType fsm:rdf ;
                         fsm:hasContent
        """
            @prefix siot: <file:///D:/projects/ontologies/siot/siot.owl#> .
            @prefix owl: <http://www.w3.org/2002/07/owl#> .
            @prefix fsm: <file:///D:/projects/ontologies/fsm/fsm.owl#> .

            self: siot:hasCompatibility ?compatibility .
            ?compatibility siot:withPeer self:myPeer .
            ?compatibility fsm:hasContent \"false\" .
        """ .
```

Listing 5. Bodies.

```
:checkCompatibility rdf:type owl:NamedIndividual ,
                             fsm:Action ,
                             http:Request ;
                    fsm:hasBody :checkCompatibilityBody ;
                    http:absoluteURI <http://localhost:9000/check_compatibility> ;
                    http:mthd <http://www.w3.org/2011/http-methods#POST> .
```

Listing 6. Action with absolute Uniform Resource Identifier.

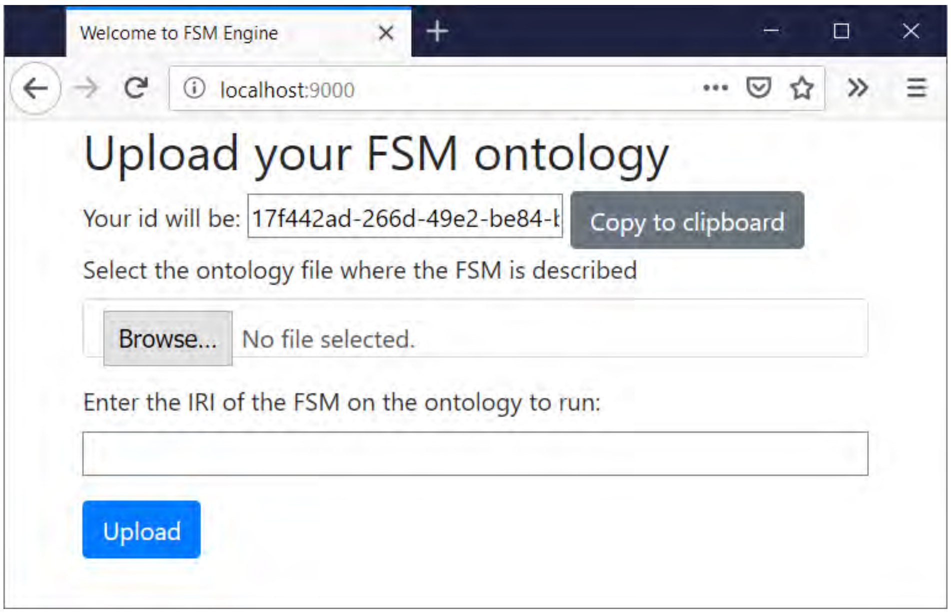
```
:sendImNotCompatible rdf:type owl:NamedIndividual ,
                              fsm:Action ,
                              http:Request ;
                     fsm:hasBody :sendImNotCompatibleBody ;
                     fsm:hasPrototypeURI :sendDataPrototype ;
                     http:mthd <http://www.w3.org/2011/http-methods#POST> .

:sendDataPrototype rdf:type owl:NamedIndividual ,
                            fsm:PrototypeURI ;
            fsm:hasParameter :peerUriParameter ;
            fsm:hasStructure "[peer_uri]/send_data" .

:peerUriParameter rdf:type owl:NamedIndividual ,
                     fsm:PrototypeParameter ;
            fsm:hasPlaceholder "[peer_uri]" ;
            fsm:hasQuery
        """
            PREFIX siot: <file:///D:/projects/ontologies/siot/siot.owl#>

            SELECT (str(?peer) as ?peer_uri)
            WHERE
            {
                self: siot:hasPeer ?peer
            }
        """ .
```

Listing 7. Action with prototype Uniform Resource Identifier.

6. Execution

This section explains how the proposed ontology-defined behaviours can be executed. We have implemented the template interpreter and embedded it in a web server. We have used Java as the programming language to implement the template interpreter and executor. Apache Jena [24] has been used as the TDB and SPARQL engine. Play Framework 2 [25] has been used to implement the web service. Akka [26], an Actor Model toolset, has facilitated the task of deploying the two VOs used in the example and enabled us to endow the engine with a reactive programming model. This means that the engine only "wakes up" when there is a new request for that VO, that is when new data is available. Both Play Framework 2 and Akka ease the task of implementing asynchronous actions. As explained in Section 5.3, the amount of time an action waits for a response (HTTP/S response) can be specified. Nevertheless, actions are performed sequentially and asynchronously. If actions A_1 and A_2 with fsm:hasTimeoutInMs 1 s and 5 s respectively are to be performed when entering a state, both actions will execute at the same time, without A_2 waiting for A_1 response. No other FSM operation like checking guards will be performed until the actions finish. An action finishes if it receives a response within its response timeout or it has not received any response during that time. We have made this implementation decision to prevent the VO from being blocked while waiting for the response.

As shown in Figure 6, users are provided with a graphic interface where they can upload their models. Users are asked to upload the file with the FSM ontology and the Internationalized Resource Identifier (IRI) that identifies the instance of fsm:StateMachine to be executed. When accessing the web the user is given an ID that identifies the FSM instance. This ID is also used to identify the VO's unique endpoint that it will be listening HTTP/S requests from. The VO is also able to send HTTP/S requests to both web interfaces of physical Things and other VOs. Each VO is provided with a RDF database that stores all the data received and generated during the execution of the FSM. The RDF database uses a base prefix ":" that uniquely identifies the data. The IRI of this prefix is built from the server URL (e.g., *http://localhost:9000/*) and the previous auto-generated ID. The IRI has a # appended at the end to refer to objects inside this domain. The final IRI looks like *http://localhost:9000/[fsm_id]#*.


Welcome to FSM Engine

localhost:9000

Upload your FSM ontology

Your id will be: 17f442ad-266d-49e2-be84-l

Copy to clipboard

Select the ontology file where the FSM is described

Browse... No file selected.

Enter the IRI of the FSM on the ontology to run:

Upload

Figure 6. Graphic interface where users can upload their models of Finite-State Machines. Users are requested to load their Finite-State Machine (FSM) ontology and the Internationalized Resource Identifier (IRI) that identifies the concrete instance of the FSM. Users are also provided with an ID that uniquely identifies the Virtual Object (VO) that will use the FSM.

In order to ease how conditions and actions are built, the VO automatically adds the RDF database's ":" prefix to all RDF data and SPARQL queries that are written on the ontology or sent to the server. The VO also adds a prefix called "self:" that refers to the same IRI that ":" but without the final #, so the FSM instance is able to reference itself. The IRI of self: is then *http://localhost:9000/[fsm_id]*.

The endpoint of the VO provides an API with three generic actions that enable communication with the FSM instance:

- Get data: it returns the result of a SPARQL SELECT query. The query is sent to the server on the request's body. It is a GET request and the URI is *http://localhost:9000/[fsm_id]/get_data*.
- Send data: it saves RDF data on the RDF database. The RDF data to be stored is sent on the request's body. It is a POST request and the URI is *http://localhost:9000/[fsm_id]/send_data*.
- Execute operation: it executes a SPARQL query like INSERT on the RDF database. The query to execute is sent on the request's body. It is a POST request and the URI is *http://localhost:9000/[fsm_id]/execute_operation*.

After the user sends the data via the initial form, a VO is deployed. During the deployment process, the FSM ontology is read and pre-loaded. The process is as follows:

1. Load the FSM ontology into Apache Jena.
2. Search on the model and read the FSM instance of type fsm:StateMachine that has the IRI specified by the user.
3. Read all the states and states' actions. All entry and exit actions are also read.
4. For each state, read the transitions that have the state as the source are searched. For each transition, retrieve their guards, conditions and guards' actions.

In the initialization process, the VO applies default values to some properties if they are not specified on the ontology:

- Action timeout: if no timeout is specified for an action, the VO sets a default timeout of 1 second for that action.
- Action method: if no method is specified for an action, the VO sets the method to GET.
- Action body: if the body is not specified, it is set to an empty string.
- Body type: if no type is specified for a body, it is treated as plain text.

After the initialization process, the VO starts to execute the FSM. The model explained in Section 4 and modelled through the paper has been executed using the prototype implementation. Two VOs are deployed and instantiated using the same template. Conditions *isPeerAvailable?*, *isPeerDataAvailable?* and *isPeerCompatibilityAvailable?* are fulfilled by external services. Those services are mocked as the main purpose of the prototype implementation is to validate the feasibility and consistency of our approach. They return true or the necessary data so the FSM can transition to the next state. In order to visualize the execution flow of the FSM in real time via a web browser, we have endowed the VO with a WebSocket interface.

7. Use Case Analysis

This section analyses several use cases to assess the suitability and to identify the strengths and weaknesses of our solution.

7.1. Use Cases

We have considered seven use cases with different qualitative characteristics. Those use cases are gathered or related to European projects where the authors participate. Three of them relate to the Healthcare IoT for equipment tracking, personal monitoring and medicine dispense, they are related to the Advanced Training in Health Innovation Knowledge Alliance (ATHIKA) project. One of them considers logistics in Industry 4.0, gathered from the project Strategic Programme for Innovation and Technology Transfer (SPRINT). The aim of both projects is to transfer knowledge, and thus a learning enhancement system is analyzed. The Smart City use case involving intelligent trash bins is related to the ENVISERA project. Finally, an analysis is performed over the use case explained in Section 4 which is related to the Smart Grid [17,27], it is referred to as the "Prosumers" use case.

We also provide a qualitative analysis of the effort needed to integrate our approach in different types of middleware. In order to do that, we classify middleware in use cases according to the types described in [28,29]. We assume that they support the HTTP/S protocol both as server and client. Healthcare middleware is considered service-based. Service-based middleware is more secure than cloud-based middleware in that cloud security and privacy are managed by the cloud provider. However, end-to-end encryption is still challenging in IoT middleware. Actor-based middleware provide the capability of embedding lightweight actors in resource-constrained, thus enabling security capabilities to end devices. Actor-based middleware is not considered for healthcare use cases as many challenges are still to be solved [28]. An event-based middleware is considered for the use cases of logistics in Industry 4.0, intelligent trash bins and prosumers. Concretely, an event-based and actor-based (no embedded actors in the physical device) middleware is assumed [17]. Event-based architectures are recurrent in Smart City, Industry 4.0 and Smart Grid systems as they allow massive data ingestion. Finally, the prototype implementation presented in Section 6 is considered to be used between web-enabled devices and a Learning Management System (LMS) system.

7.1.1. Hospital Equipment Tracking

Hospital equipment, like stretchers and wheelchairs, can be tracked to obtain its position and whether they are being used or not. The tracking is used to optimize the utilization of resources, to keep track of the inventory and to reserve the equipment. Our VO can be used to represent each piece of equipment. The VO will be asking the sensors for the current position and for other values like the weight they are carrying. For instance, when the VO detects a significant change on the weight it can send a message to a centralized service.

7.1.2. Hospital Personal Monitoring

On a hospital, patients identity, status or position can be monitored through IoT to increase their safety. The medical staff can also be tracked to optimize their workflow. The VO used for the patient and the staff are different and there is a one-to-one relationship between any user (patient or staff) and a VO. The VO can be used to retrieve the current position and patients' vital signs. If the patient exits their room or some value of a vital sign gets out of a healthy range, the VO can send a notification to the medical team. The staff position is also useful, for example, to alert the professional closer to the patient.

7.1.3. Intelligent Medication Dispenser

An intelligent medication dispenser is a machine that gives patients the correct medication at the right time and also checks for incompatibilities between medicines. This is a perfect tool to avoid problems like the ones explained by the authors in [30] such as a lethal combination of medicines, recalled medicines, not taking medicines on the right time or taking the same dose twice. The VO can implement the logic of deciding which medicines to dispense. First, it will wait for the identification of the patient (for instance, with an Radio-Frequency IDentification (RFID) card). Second, the VO will retrieve the information about the medication assigned to the patient by a doctor (this information has been previously added). Then, it will select the medication that needs to be taken at that time and verify that it had not been taken already. The VO also checks the compatibility with other medicines. It uses an external database that describes medicine compatibility in RDF format. Then, the VO executes a SPARQL query with the necessary reasoning logic. If any incompatibility is found, the patient is notified. The VO can be also programmed to alert patients at the time they have to take the medication through a smart-alarm or a smartphone.

7.1.4. Logistics on Industry 4.0

Logistics on warehouses can be made more efficient through automatic processing of the incoming items. The author in [31] introduces a use case where all pallets of items have an RFID tag. The RFID tags are read on their arrival by RFID readers. Multiple tags can be read at the same time as their position on the pallet is not important as long as they are in the range of the reader. The tag can provide information like the product identifier, the number of items in the pallet or additional information like fragile content or notes written by the sender or the driver. The stock of the warehouse is updated through an Enterprise Resource Planning (ERP) application. A VO can implement this process. For each RFID tag processed, the VO sends the information of the tag to the ERP system. The VO can ask the ERP for the best place to store the incoming item or it can even implement the logic of deciding the best place to store it via queries and reasoning. Finally, the VO will notify a worker or robot to carry the item to its storage place.

7.1.5. Learning Enhancement

The learning performance of students can be improved by monitoring the environment of the classroom and the status of the student. The work done in [32] explains a way to achieve this; they use the technology of Wireless Sensor Networks (WSNs) to integrate temperature, humidity, illumination, the concentration of CO_2 sensors and human emotions detection cameras on wireless networks. Some parameters like temperature are used to automatically adjust the room temperature, but others like the students' mood may need the attention of the professor. A VO can be used to actively monitor all the parameters and adjust the conditions of the room to optimal values; it can also alert the professor about the students that are losing focus. We assume that the VO has a one-to-one relationship with the classroom (temperature, humidity, CO_2, ...) and one-to-many with the students in a classroom, but it is a relationship with a reduced and limited set of entities (students).

7.1.6. Intelligent Trash Bin

The intelligent trash bin is an already used technology on Smart Cities that monitors the state of trash bins and enables optimization on the waste collection. Authors in [33] propose an architecture that implements this technology; the bin contains an Arduino and attached sensors that collect information about waste level, temperature, humidity, motion and weight. The information of the sensors is sent to the cloud where patterns to optimize the collection of waste are extracted. A previous step can be achieved using edge computing to analyze the physical state of the bin to prevent heavy damage, as the analysis on the cloud may not be real time. The one-to-one or one-to-many relation between a VO and an intelligent trash bin or set of trash bins (if they are placed close to one another) can be established. The VO is placed on the edge cloud. Each time, the VO receives the report from the Arduino, before sending it to the cloud, it will analyze the values of humidity, toxicity (with additional sensors), motion or weight to detect potential problems like fire, vandalism/bad use or a possible toxic hazard. If any of these problems are detected, the VO sends an alert to the trash management system.

7.2. Analysis and Discussion

This section provides a qualitative analysis of the described use cases. A summary of the analysis is provided in Table 1. The following metrics are considered to provide the analysis:

- Time reliability. This metric considers the limitations of the technologies that have been used to develop our solution, namely RDF, SPARQL and HTTP/S. They allow interoperability and shareability, the main focus of our proposal. However, they may be computationally heavy, especially SPARQL. This solution may not be suitable for time-critical applications. Specialized systems are better suitable to run time-critical applications. The speed of the network due to congestion can be mitigated with a dedicated Local Area Network (LAN). A Low score means that the VO cannot keep up with the timing requirements of the use case. A High score means that the use case does not need accurate timing and that some delay is acceptable.
- Scalability. Scalability represents how the VO performance scales with the number of Things that it manages. The total performance of the system degrades when the number of Things controlled increases. Usually, the performance degradation is linear and the biggest impact will be at time reliability because the available resources will be shared between more Things. Having more Things to control usually means that there will be more states and actions to perform, and that will translate in more delays. A Low score means that the manner in which the system is designed provokes a performance degradation as more Things are integrated into the system.
- Reusability. We consider reusability as the degree of how easy it is to reuse the generated template or templates for other similar use cases. For example, a template used by a VO that senses the room temperature and actuates accordingly to increase the comfort of the inhabitants can be easily used on other rooms or buildings. It may be only necessary to change the temperature thresholds. The temperature is represented as the value to be queried and not hardcoded in the execution flow or template. This example has a High reusability. On the other hand, a template that describes the behaviour of a robot that assembles car pieces has Low reusability. There will be substantial differences in the assembly logic of different models of cars and between companies.
- Suitability. Suitability is the degree of fitness between the behaviour design that the current FSM ontology allows and the desired functionalities/behaviour of the use case. If none or few modifications are needed, the suitability is considered High. If major modifications are needed, then suitability is Low.
- Incompatibilities (with domain-specific ontologies). The degree of incompatibilities between the base ontologies used by the proposed solution and the ontologies of the use case domain.

A score of High means that it is hard to integrate our solution with domain-specific ontologies. It is possible to have a score of None if there are no incompatibilities.

- Outsourcing. Outsourcing refers to the need to use external services that provide operations that the FSM cannot do. If the VO is independent or uses few external services, then the outsourcing is considered Low. VOs that act as service aggregators have high outsourcing. The value of this metric does not have an explicit positive or negative meaning, its purpose is to describe a characteristic of the VO behaviour.
- Query complexity. Query complexity is an estimation of how heavy the queries of the use case are. Heavy means that the number of returned triples is very high or/and the query execution time is expected to be long (more than half a second). A High complexity means that heavyweight queries will be performed during the execution flow.
- Knowledge complexity. Knowledge complexity refers to how much knowledge about ontologies, SPARQL and behaviour representation with FSMs the user requires to implement the system for a concrete use case. A High score means that the user necessitates a High degree of knowledge to represent the behaviour of the system. A Low score denotes that few experiences are needed as it is easy to represent the behaviour.
- Integration Effort. Integration effort considers the required effort to integrate our system in the middleware/architectures assumed for each use case. If the middleware supports actors or VOs, the effort is considered Low or Medium.

Table 1. Use cases benchmark.

	Hospital Equipment Tracking	**Hospital Personal Monitoring**	**Intelligent Medication Dispenser**	**Logistics on Industry**	**Learning Enhancement**	**Intelligent Trash Bin**	**Prosumers**
Time reliability	High	Medium	High	Medium	High	High	High
Scalability	High	Medium	High	High	High	High	High
Reusability	High	Medium	High	Medium	High	High	High
Suitability	High	High	High	High	High	High	High
Incompatibilities	None	None	None	None	None	None	None
Outsourcing	Low	Low	Medium	Medium	Low	Low	Medium
Query complexity	Low	Low	Medium	Low	Low	Low	Low
Knowledge complexity	Low	Low	Medium	Low/Med.	Low	Low	Low
Integration Effort	Medium	Medium	Medium	Medium	Low	Medium	Medium

Time reliability is High in the cases where it is not important if the system has delays greater than one second, for instance, it is not critical if the VO of the medicine dispenser has a delay of two seconds. It is Medium where a second of difference is important like monitoring the health of a patient (a fast alert could save the patient's life). In the case of Logistics, it also critical if, for each item to be unloaded, there is a delay of some seconds because a lot of items are expected to be processed fast and that can sum up to a big delay.

Scalability gets a Medium score on hospital personal monitoring because adding more Things to the VO may cause bigger delays that are not desired. This is because if more constants are to be monitored on the patient, the tasks and logic that the VO performs increase.

Reusability is Medium on the logistics use case as the desired behaviour about how the items are stored and processed may change for each company or warehouse. The idea behind templates is to define a behaviour abstracted from the physical implementation to provide reusability. It is also Medium in the use case of hospital personal monitoring as each hospital may have different workflows to optimize the performance of the staff. Reusability is High in other use cases as we consider that the behaviour/execution flow can be defined generically.

Suitability is High in all use cases as the VO is able to execute the desired behaviour. There are no incompatibilities between our proposed solution and domain-specific ontologies. Our solution relies on ontologies in order to be as compatible as possible with other ontologies. Domain-specific ontologies are expected to extend higher level ontologies such as SOSA/SSN in order to promote template reusability.

Outsourcing is Medium where the system has some operations that rely on external processes. The medication dispenser relies on an external catalogue in order to check medicine compatibility. The logistics use case scores Medium at outsourcing because it relies on the ERP to perform its tasks; some operations are too complex to be implemented by a FSM or need specialized software.

Finally, query complexity is Low in almost all cases as the queries are usually with local data and are simple. It is Medium on the medication dispenser because there may be some heavy queries that check the compatibility between all the medications of the patient. Knowledge complexity is Medium on the dispenser as the queries for medicine compatibility check can be complex to write. On the logistics, the queries can be also complex if the ERP does not provide the location to store the items; queries of searching the optimal place should be written in this case.

Our approach is easily applicable in learning enhancement, intelligent trash bin and prosumers use cases. We consider that they have a generic structure (High reusability) and that the modelling of their behaviour is straightforward with Low query and knowledge complexity. They are not time-critical use cases so they score High in time reliability. Their score is High on scalability, as each VO has a relation one-to-one or one to a reduced and limited set of entities to control. Prosumers use case has a Medium score on Outsourcing as it uses external services to get informed if a peer is available or for the complex operation of checking the energy profile compatibility.

The integration effort with any concrete middleware comprehends (1) implementing the FSM engine in the programming language used by the middleware; (2) implementing a software adapter to connect the FSM inputs and outputs to the HTTP/S server and client interfaces of the middleware; and (3) providing an RDF and SPARQL API to the FSM engine. If the middleware provides a TDB, it is only necessary to connect the FSM engine to the database. Otherwise, it needs to be implemented. For that reason, the integration effort is considered Medium in almost all use cases and respective type of middleware. Once this initial effort is done, the effort remains to create the template of the desired behaviour using the ontologies and considerations described in this manuscript. Regarding the learning enhancement use case, a simple web service like the one presented in Section 6 can be used as the orchestrator between web-enabled devices and the LMS. Note that the purpose of the prototype is only to provide a web execution environment to the FSM engine. It lacks common web security and privacy mechanisms such as authentication and authorization. If the integration between our approach and the middleware is not possible or requires too much effort (e.g., in middleware that does not have the concept of VO/(Web-)Thing), our approach can be used at the application layer as a microservice that composes with the middleware's HTTP/S API.

8. Conclusions and Future Work

Given the heterogeneity that characterizes the IoT, a novel middleware-agnostic approach that allows for describing and executing the behaviour of devices has been proposed and implemented. The objective is to allow the reusability and shareability of the execution flow among multiple and heterogeneous IoT deployments. The approach relies heavily on existing standards to promote interoperability and reusability. FSMs are used as the model to create the execution flow using ontologies. Our work is contextualized in a reference architecture recommended by W3C, the W3C WoT Architecture. We have used the concept of VO, which are virtual counterparts of physical devices as the computational entities that run the concrete instances of the templates. Several use cases have been analyzed to asses the viability and suitability of our solution. Relying on standards such as RDF, SPARQL and HTTP/S has some drawbacks. They tend to be more heavyweight than an ad hoc solution. For example, HTTP/S can be replaced by Constrained Application Protocol (CoAP) (a lightweight, IoT version of HTTP/S) and the data model can be replaced by a SQL or NoSQL database. For that reason, our approach is not well suited for time-critical applications such as monitoring and reasoning over patients' vital signs in a hospital. However, our approach is well-suited for IoT scenarios that are non-time-critical and with a low level of variability between each deployment. It allows for reusing behaviours for heterogeneous physical devices with the same set or subset of capabilities.

In future works, we expect to fully apply a TD interface to the VOs. Our aim is to extract and generate part of the TD definition from the FSM instance. This will also enable the alignment with other ontologies such as the oneM2M Base Ontology and enable service composition. Given that each state has its own inputs and outputs, research is needed to identify which Properties, Actions and Events (according to the TD model) should be exposed. SPARQL performance is computationally heavy and our solution only allows to send SPARQL queries to external services. We plan to add the capability of sending non-SPARQL requests to external services described using ontologies such as TD ontology or oneM2M Base Ontology. Our goal was to facilitate deployments in similar IoT scenarios. Nevertheless, the task of creating the template ontology can be tedious, especially if the template has multiple states and actions with prototype URIs. Our approach has some restrictions imposed by the FSM model and the WoT architecture (we assume that physical devices expose an HTTP/S API). Therefore, we plan to create a visual tool to hasten the creation of FSM templates using visual building blocks like in StarUML. The IDE will be used to create the template in a visual manner and to automatically translate the visual representation to the ontological one.

Author Contributions: V.C. and D.V. conceived the idea and decided the technologies involved in the prototype; V.C. and S.V. implemented the prototype; V.C. contributed with insightful knowledge about the technologies involved; A.Z. and D.V. supervised the execution of the work and contributed with insightful discussions and ideas to improve the overall quality of the work; A.Z. is also the link with the European projects ATHIKA and SPRINT. All authors contributed equally to writing the paper.

Funding: This work received funding from the "Agència de Gestió d'Ajuts Universitaris i de Recerca (AGAUR)" of "Generalitat de Catalunya" (grant identification "2017 SGR 977") and ENVISERA project, funded by Spanish Ministerio de Economía y Competitividad (Plan Estatal de Investigación Científica y Técnica y de Innovación 2013–2016, grant reference CTM2015-68902-R). Part of this research has also been funded by the "SUR" of the "DEC" of the "Generalitat de Catalunya" and by the European Social Funds (grant reference 2018FI_B1_00171). In addition, we would like to thank La Salle—Universitat Ramon Llull for their support.

Acknowledgments: Icons used in Figure 3 made by Freepick and Gregor Cresnar at www.flaticon.com

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Atzori, L.; Iera, A.; Morabito, G. Understanding the Internet of Things: Definition, potentials, and societal role of a fast evolving paradigm. *Ad Hoc Netw.* **2017**, *56*, 122–140, doi:10.1016/j.adhoc.2016.12.004. [CrossRef]
2. Guinard, D. A Web of Things Application Architecture—Integrating the Real-World into the Web. Ph.D. Thesis, University of Fribourg, Fribourg, Switzerland, 2011.
3. Shadbolt, N.; Hall, W.; Berners-Lee, T.; Hall, W. The Semantic Web Revisited. *IEEE Intell. Syst.* **2006**, *21*, 96–101, doi:10.1109/MIS.2006.62. [CrossRef]
4. Martin, D.; Burstein, M.; Hobbs, J.; Lassila, O.; McDermott, D.; McIlraith, S.; Narayanan, S.; Paolucci, M.; Parsia, B.; Payne, T. OWL-S: Semantic Markup for Web Services. Available online: https://www.w3.org/Submission/OWL-S/ (accessed on 17 February 2019).
5. World Wide Web Consortium. W3C Web of Things Working Group. Available online: https://www.w3.org/WoT/WG/ (accessed on 17 February 2019).
6. OneM2M. oneM2M Base Ontology. Available online: www.onem2m.org/technical/published-drafts (accessed on 13 February 2019).
7. World Wide Web Consortium. RDF 1.1 Concepts and Abstract Syntax. Available online: https://www.w3.org/TR/rdf11-concepts/ (accessed on 18 January 2019).
8. Charpenay, V.; Kabisch, S.; Kosch, H. Semantic data integration on the web of things. In Proceedings of the 8th International Conference on the Internet of Things—IOT '18, Santa Barbara, CA, USA, 15–18 October 2018; pp. 1–8.
9. OneM2M. oneM2M Global Initiative. Available online: http://www.onem2m.org/ (accessed on 18 February 2019).
10. OneM2M. WoT Interworking. Technical Report. 2018. Available online: http://www.onem2m.org/ (accessed on 18 February 2019).

11. Haller, A.; Janowicz, K.; Cox, S.J.; Lefrançois, M.; Taylor, K.; Le Phuoc, D.; Lieberman, J.; García-Castro, R.; Atkinson, R.; Stadler, C. The modular SSN ontology: A joint W3C and OGC standard specifying the semantics of sensors, observations, sampling, and actuation. *Semant. Web* **2018**, 1–24, doi:10.3233/SW-180320. [CrossRef]
12. Belgueliel, Y.; Bourahla, M.; Brik, M. Towards an Ontology for UML State Machines. *Lect. Notes Softw. Eng.* **2014**, *2*, 116–120, doi:10.7763/LNSE.2014.V2.106. [CrossRef]
13. Dolog, P. Model-Driven Navigation Design for Semantic Web Applications with the UML—Guide. In Proceedings of the Workshops in Connection with the 4th International Conference on Web Engineering, ICWE 2004, Mubich, Germany, 28–30 July 2004; pp. 1–12.
14. Haage, M.; Malec, J.; Nilsson, A.; Stenmark, M.; Topp, E.A. Semantic Modelling of Hybrid Controllers for Robotic Cells. *Procedia Manuf.* **2017**, *11*, 292–299, doi:10.1016/j.promfg.2017.07.108. [CrossRef]
15. Pessemier, W.; Deconinck, G.; Raskin, G.; Saey, P.; van Winckel, H. Developing a PLC-friendly state machine model: Lessons learned. *Proc. SPIE* **2014**, *9152*, 915208, doi:10.1117/12.2054881. [CrossRef]
16. Nitti, M.; Pilloni, V.; Colistra, G.; Atzori, L. The Virtual Object as a Major Element of the Internet of Things: A Survey. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 1228–1240, doi:10.1109/COMST.2015.2498304. [CrossRef]
17. Caballero, V.; Vernet, D.; Zaballos, A.; Corral, G. Prototyping a Web-of-Energy Architecture for Smart Integration of Sensor Networks in Smart Grids Domain. *Sensors* **2018**, *18*, 400, doi:10.3390/s18020400. [CrossRef] [PubMed]
18. Negash, B.; Westerlund, T.; Tenhunen, H. Towards an interoperable Internet of Things through a web of virtual things at the Fog layer. *Future Gener. Comput. Syst.* **2019**, *91*, 96–107, doi:10.1016/j.future.2018.07.053. [CrossRef]
19. Guan, Y.; Vasquez, J.C.; Guerrero, J.M.; Samovich, N.; Vanya, S.; Oravec, V.; Garcia-Castro, R.; Serena, F.; Poveda-Villalon, M.; Radojicic, C.; et al. An open virtual neighbourhood network to connect IoT infrastructures and smart objects—Vicinity: IoT enables interoperability as a service. In Proceedings of the 2017 Global Internet of Things Summit (GIoTS), Geneva, Switzerland, 6–9 June 2017; pp. 1–6.
20. The Vicinity Consortium. 2nd Open Call. Technical Details. Technical Report; Vicinity Consortium. 2019. Available online: https://vicinity2020.eu/vicinity/ (accessed on 18 February 2019).
21. Girau, R.; Martis, S.; Atzori, L. Lysis: A platform for iot distributed applications over socially connected objects. *IEEE Internet Things J.* **2017**, *4*, 40–51, doi:10.1109/JIOT.2016.2616022. [CrossRef]
22. MKLab. StarUML. Available online: http://staruml.io (accessed on 18 January 2019).
23. World Wide Web Consortium. HTTP Vocabulary in RDF 1.0. Available online: https://www.w3.org/TR/HTTP-in-RDF10/ (accessed on 10 February 2019).
24. Apache. Apache Jena. Available online: https://jena.apache.org/ (accessed on 16 February 2019).
25. Lightbend. Play Framework 2. Available online: https://www.playframework.com/ (accessed on 16 February 2019).
26. Lightbend. Akka. Available online: https://akka.io/ (accessed on 16 February 2019).
27. Martín de Pozuelo, R.; Zaballos, A.; Navarro, J.; Corral, G. Prototyping a Software Defined Utility. *Energies* **2017**, *10*, 818, doi:10.3390/en10060818. [CrossRef]
28. Ngu, A.H.H.; Gutierrez, M.; Metsis, V.; Nepal, S.; Sheng, M.Z. IoT Middleware: A Survey on Issues and Enabling technologies. *IEEE Internet Things J.* **2016**, *4*, 1–20, doi:10.1109/JIOT.2016.2615180. [CrossRef]
29. Razzaque, M.A.; Milojevic-Jevric, M.; Palade, A.; Cla, S. Middleware for Internet of Things: A Survey. *IEEE Internet Things J.* **2015**, *3*, 70–95, doi:10.1109/JIOT.2015.2498900. [CrossRef]
30. Gomes, C.E.M.; Lucena, V.F.; Yazdi, F.; Gohner, P. An intelligent medicine cabinet proposed to increase medication adherence. In Proceedings of the 2013 IEEE 15th International Conference on e-Health Networking, Applications and Services, Healthcom 2013, Lisbon, Portugal, 9–12 October 2013; pp. 737–739.
31. Gilchrist, A. *Industry 4.0—The Industrial Internet of Things*; Apress: Berkeley, CA, USA, 2016.

32. Chiou, C.K.; Tseng, J.C. An Intelligent Classroom Management System based on Wireless Sensor Networks. In Proceedings of the 2015 8th International Conference on Ubi-Media Computing, UMEDIA 2015, Colombo, Sri Lanka, 24–26 August 2015; pp. 44–48.
33. Gutierrez, J.M.; Jensen, M.; Henius, M.; Riaz, T. *Smart Waste Collection System Based on Location Intelligence*; Procedia Computer Science; Elsevier: Amsterdam, The Netherlands, 2015; Volume 61, pp. 120–127.

Social Internet of Energy - A new paradigm for Demand Side Management

Víctor Caballero[1], David Vernet[1] and Agustín Zaballos[1]

***Abstract*—The creation of prosumer communities leveraging the Smart Grid domain to improve energy production and consumption patterns has been proposed in the literature. We propose to use a novel Internet of Things paradigm, namely the Social Internet of Things, to enable appliance-level metering (sub-metering) and to create prosumer communities dynamically. We use the term Social Internet of Energy to refer to the integration between prosumers and devices via social relationships. This paper explores the diverse technologies that will enable the Social Internet of Energy. On the one hand, we explore the different aspects and value propositions of the Social Internet of Things, applying this novel paradigm to the creation of prosumer communities. On the other hand, we explore Demand Side Management, its synergy with Social Internet of Things and the properties and challenges that may arise. Concretely, in this paper we aim at grouping prosumers by their energy load shape compatibility and then reschedule intra-group load shapes. We propose a clustering model and we develop two clustering heuristics to distribute prosumers among Prosumer Community Groups and one Peak-to-Average Ratio prediction heuristic in order to reduce the Peak-to-Average Ratio and the amplitude of energy peaks per cluster. Then, on the basis of a real dataset, we perform simulations and analyze and discuss the results, paving the way for further research.**

Keywords*—social internet of things, internet of things, smart grid, prosumer, community, prosumer community groups.

I. Introduction

Energy demand in the world is increasing rapidly. Worldwide energy consumption is expected to increase by 29.7% from 2017 to 2050 [1]. The majority of the energy demand is met by non-renewable energy resources which are starting to be insufficient and produce undesirable climate changes that harm the world we live in [2]. Hence, society is moving towards the use of renewable and sustainable energy sources. In order to meet worldwide energy demand using Renewable Energy Sources (RESs), it has been proposed to group energy users (prosumers) to optimize generation, demand-side or storage resources and increase individual Distributed Energy Resource (DER) visibility, while allowing these groups to work autonomously to some extent. Those features are and will be possible thanks to the underlying Smart Grid (SG) infrastructure. The SG can be seen as a hybrid system composed of the electric power system and a heterogeneous Information and Communication Technology (ICT) infrastructure [3]. Several solutions have emerged, aiming to increase the visibility of individual DERs and optimize system operation. Proposed solutions for the Communityware Smart Grid [4] aim to create prosumer clusters to make them gain market visibility and reduce system operation costs. Three main clustering solutions have emerged: Virtual Power Plants, Microgrids and Goal-Oriented Prosumer Community Groups. A brief description of each one is given in the following list.

1) Microgrid: A Microgrid (MG) is an integrated energy system consisting of distributed energy resources and multiple electrical loads operating as a single, autonomous grid either in parallel or "islanded" from the existing utility power grid [5].
2) Virtual Power Plant: Virtual Power Plants (VPPs) rely upon software systems to remotely and automatically dispatch and optimize generation, demand-side or storage resources in a single, secure Web-connected system. VPPs represent an "Internet of Energy" tapping existing grid networks to tailor electricity supply and demand for a customer, maximizing value for both end-user and distribution utility through software innovations. [5].
3) Goal-Oriented Prosumer Community Group: In contrast, Prosumer Community Groups (PCGs) virtually interconnect prosumers that may not be technically connected. According to [6], a Goal-Oriented PCG is defined as "a network of prosumers having relatively similar energy sharing behaviors and interests, which make an effort to pursue a mutual goal and jointly compete in the energy market". As a result, PCGs can create a dynamic ecosystem of cooperating prosumers [6], [7].

The work presented in this article emerges from the concept of Smart Grid flexibility and Software Defined Utility (SDU), presented in [3], from knowledge gathered in the FP7 European projects INTEGRIS (INTelligent Electrical GRId Sensor Communications) [8] and FINESCE (Future INternEt Smart Utility ServiCEs) [9]. Thus, the proposed approach aims to be a technological solution to achieve the so-called Smart Grid flexibility by some European utilities such as ENEL, ESB or Iberdrola. In order to provide increased Smart Grid flexibility, there are some works related to the Future Smart Grid [10] carried out by the authors [8], [9].

PCGs excel for their dinamicity at managing resources, leveraging a SG infrastructure. The SG can be considered a particular case of the Internet of Things (IoT) [11], referred as the Internet of Energy (IoE) [12]. IoE aims at optimizing the efficiency of the energy infrastructure by creating a distributed

[1]V. Caballero, D. Vernet and A. Zaballos are with Faculty of Engineering, La Salle Barcelona - Ramon Llull University, 08022 Barcelona, Spain
*corresponding author: Víctor Caballero (email: victor.caballero@salle.url.edu).

network of sensors and energy generators. In this situation, a novel paradigm, namely the Social Internet of Things (SIoT) can contribute in many manners, enabling a Social Internet of Energy.

The Social Internet of Energy (SIoE) aims at enhancing the connectivity and relationships established among the users and devices in the IoE (and Future Smart Grid) by leveraging the technology offered by the SIoT. In this work, we will focus on explaining how PCGs can be dynamically allocated and then how those groups can be allocated in a hierarchical/fractal style. The main contributions of this article are:

- Explore how the SG can harness the SIoT technology to improve energy management at the demand side (Demand Side Management).
- Analyze which technologies are involved when combining Demand Side Management (DSM) and SIoT in a SG.
- Advance in the research of the combination of DSM and SIoT through experimentation and evaluation using cluster-based segmentation.

This article is organized as follows. Section II gives a brief overview of how the SIoT emerged and its characteristics. Section III proposes to apply the SIoT paradigm to the SGs and explains the benefits of such approach to create communities of prosumers. Section IV presents a use case: to apply SIoT to DSM and analyzes the manners in which SIoT-enabled devices can cooperate. Section V presents the DSM model that will be used to create communities of prosumers. Section VI explains the tests that have been performed to evaluate the presented approach and analyzes and discusses the results. Section VII proposes a new hierarchical clustering algorithm that nourishes from the conclusions drawn in Section VI. Finally, the article is concluded in Section VIII, where future work is also presented.

II. Background: Social Internet of Things

The Social Internet of Things (SIoT) is one of the last and novel configurations proposed in the IoT framework and, according to a generational analysis of the IoT [13], it belongs to the third generation of IoT. The SIoT envisions interactions among humans and things, besides human-to-human and simple object-to-object interactions, the former enabled by Social Network Services (SNSs) and the latter already tackled by the IoT vision. By leveraging Social Networks (SNs) new logical configurations involving both humans and things or networks of things can be created. In the future SIoT, humans and things are not logically separated and instead, can be represented as nodes of the same network, both providing services [14]. For example, in [15] a Social Web of Things (SWoT) is proposed, where users can access physical devices through SNSs authorization mechanisms and physical devices (things) can publish to the SNS by the same mechanisms: SNSs delegated access mechanisms such as OAuth. The work done in [15] is also relevant as SNS friend relationships are used as the trust model in order to authorize users to perform actions that involve physical devices (web-enabled things) owned by another entity. The realization of the SIoT paradigm regarding object communities will bring desired properties to the IoT:

1) scalable navigability via relationships;
2) flexibility, as each node identifies its friends and uses those connections to perform scalable service discovery and matchmaking and
3) trustworthiness that, given a specific goal, will allow nodes to use their experience (individual or collective) to decide which service is most reliable to serve that goal, based on the level of trust.

The novel SIoT paradigm is evolving rapidly, and thus appropriate reviews have emerged. As explained, the SIoT first steps where in integrating SNSs with the IoT. In [15], the Web of Things (WoT), a web-enabling paradigm for the IoT, and SNSs where merged together, enabling the interaction between things (mainly human-to-thing interactions). The work done in [16] provides a good review on the evolution from the IoT to the SWoT, and reviews main architectures and work related to SWoT. For example, several works use SNS to authenticate and authorize owners' friends to access their devices [17], [18]. The review article [16] presents early work on blending IoT, Web technologies and SNSs; thus, work related to SIoT understood as networks and communities of objects is excluded. The integration between SNs and IoT continues with the idea of objects creating networks and communities among themselves [19]. Researches in this direction aim to describe how social relationships between objects or things are established and what mechanisms are used to establish those connections. In fact, a review on the cluster between IoT and SIoT [20] highlighted the benefits of integrating objects with social relationships, the emerging architectures of SIoT and the challenges and open research issues for this new paradigm.

As in human social relationships, object relationships are also based on "feelings" about each other. In the SIoT paradigm, those feelings are referred as trust properties and serve as the computation parameters to decide in which degree an object can be trusted, and thus to measure the healthiness of the relationship [21], [22]. In this sense, trust management mechanisms and social network structures in SIoT are reviewed in [23] and [24], along with several works contributing to the SIoT. Also, in the same manner as human relationships, trust relationships among SIoT objects can be vulnerable to some attacks such as bad-mouthing or whitewashing attacks [25]. The following sections briefly describe the SIoT properties that will be leveraged in the remainder of this paper and give appropriate references for further reading:

1) Relationships: According to [26], an object can establish five types of relationships or friendships:

- Parental Object Relationship (POR): Defined among similar objects, built at the same period by the same manufacturer and can be used, for example, to share optimal configuration.
- Co-location Object Relationship (CLOR): Established among heterogeneous objects used always in the same place. Some of those relationships will be established regardless of the need for object cooperation, but they are useful to create short links in the network for discoverability purposes.
- Co-work Object Relationship (CWOR): Created among heterogeneous objects that need to collaborate to provide

a service. For example: sensors in a city to obtain temperature measurements.

- Ownership Object Relationship (OOR): This relationship is established among heterogeneous objects that belong to the same user.
- Social Object Relationship (SOR): Defined when objects come into contact (close range) because their owners come in touch with each other during their lives. Those relationships, which usefulness varies from friend to friend, enable the creation of networks and communities of objects, aiming to improve IoT scalability and navigability and to provide the user with the desired service or service composition.

2) Trust management: Trustworthiness management in the SIoT involves the formulation of trust mechanisms to understand how to build a reliable system based on the behavior of other objects. Trust involves, at the basic level, two objects, the trustor and the trustee in a relationship context, such as its goal. Trust properties are reviewed in [21], where the authors present several ways in which trust can be computed. The computation of trust can involve multiple parameters. It is straightforward to think that the computation only involves the trustor and the trustee. While this is true, it can also involve opinions and recommendations from other nodes or context in which the trust relationship is created, including past experiences. Two types of trust levels are analyzed in [21], objective trustworthiness and subjective trustworthiness.

- Objective trustworthiness: The information about each node is distributed and stored using a Distributed Hash Table (DHT) structure. This information is visible to every node but is only managed and stored by Pre-Trusted Objects (PTOs).
- Subjective trustworthiness: Each node computes the trustworthiness of its friends on the basis of its own experience and the experience of its friends. If the node providing a service is not a direct friend of the one requesting it, the trustworthiness is calculated by word of mouth through a chain of friendships.

3) Service Discovery: Regarding the scalability of the discovery process and navigability, each thing is cognitively able to query a service through its established relationships and evaluate the trustworthiness of the returned service. This form of tackling scalability and navigability from a social community perspective is based on the conclusions that people are tied by short chains of acquaintances [27] and that there are structural clues in a social network that help people find a short path even without a global knowledge of the network [28]. For example, a novel Resource Discovery Mechanism based on Preference and Movement Pattern Similarity with the aim to build sub-communities of similar preference and movement nodes is proposed in [29]. Then, virtual global communities are formed based on the sub-community similarity as depicted in Fig. 1, improving the search performance. In addition, the authors also design a resource discovery algorithm that dynamically adjusts the search radius to balance the performance and communication overhead.

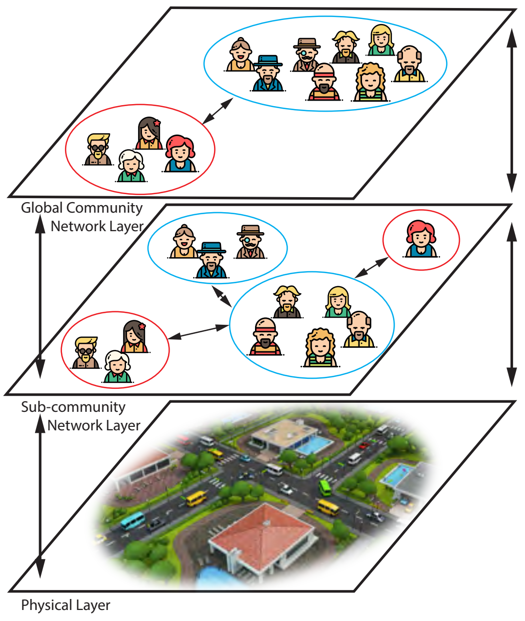


Fig. 1. Three-layer network structure. Sub-communities are created on the basis of preference and movement similarity. Then, virtual global communities are created on the basis of sub-community similarity.

III. Social Internet of Things and Goal-Oriented Prosumer Community Groups

We envision that SIoT technologies can bring goal-oriented PCGs closer to reality. The following subsections explain the synergy that can be achieved by applying SIoT paradigm to PCGs:

A. Homogeneity

The SIoT envisions the creation of social networks of things, where they can intercommunicate with each other. To achieve this goal, every thing or Virtual Object (VO) [30] must be capable of "speaking" the same language. The WoT paradigm uses the Hyper-Text Transfer Protocol / Secure (HTTP/S) to achieve communication protocol inter-operability among things and Semantic Web technologies [31] to achieve message format homogeneity and ease discoverability. For example, authors in [32] implement a Semantic Search Engine to discover available world wide web services. The Lysis platform [33] implements the WoT paradigm and only requires physical devices (for example, smartphones) to be registered in the cloud platform. Once registered, they become a social-enabled thing. Thanks to the relations a thing can establish with other things, such thing can be notified by a "friend" (another thing) to install a specific application that provides the desired functionality. The example in [33] involves a user's goals (to arrive at the hotel where she has made a reservation), a smartphone and an airport multimedia infopoint. They establish a Social Object Relationship (SOR). Thanks to the SOR, the airport multimedia infopoint suggests the user to install a SocialMobility App, which, in turn, leverages the Co-location Object Relationship

(CLOR) relationships established by the airport multimedia infopoint and its surrounding services, like taxi service and bus station service. Finally, the user is able to receive information about the transportation services she needs to arrive at the hotel.

Goal-Oriented Prosumer Community Groups as explained in [6] make use of energy consumption and production data to perform statistical analysis and assign each prosumer to a group. Users (prosumers) usually do not own devices from the same brand. This means that devices, which could be renewable generators or smart meters, cannot easily interconnect with each other; neither from a local perspective (i.e by means of a local area network or short-range protocol) nor from a virtualization perspective (i.e cloud Application Programming Interfaces (APIs)), as they are vendor specific. IoT technologies and platforms allow devices to connect to each other and expose a standardized API to ease and homogenize access. Therefore, in the sense of homogenization, PCG can greatly benefit from IoT and WoT technologies such as the one presented in [33]. An energy-related use case that leverages SIoT is given in [34], where the authors use thing-to-thing relationships to optimize energy usage by the Heat, Ventilation, Air Conditioning (HVAC) system in their laboratory. Results show that the comfort-aware Smart HVAC system ensures users' thermal comfort, while at the same time reduces energy costs with respect to static and traditional approaches.

B. Leveraging object relationships for Prosumer Community Groups

Regarding social relationships between objects, two straightforward use cases emerge related to Prosumer Community Groups.

1) Prosumer Ownership: The use case deals with the aforementioned heterogeneity. This use case is similar to the one presented in [34] and is as follows. Prosumers are users that own devices that produce/consume energy. To create a PCG, the basic social network that is needed to create is the one composed of all devices that produce or consume energy, owned by the same prosumer. After enabling those devices to be part of the homogeneous IoT they will, according to the SIoT relationships, establish an Ownership Object Relationship (OOR). The just established OOR will enable them to easily find and communicate with each other and will allow the creation of Micro Engines (MEs) [33] that aggregate multiple individual functionalities. This will allow data aggregation whenever it is necessary through MEs and ad-hoc feedback, as smart appliances will be able to be the target of a supervision strategy. For example, according to the user's goals and the PCG goals, a smart appliance can show a warning to the user if he/she wants to turn the appliance on (i.e, a washing machine) in a condition unfavorable to attain such goals.

2) Goal-Oriented Prosumer Community Group formation: This second use case relates PCGs dynamic formation and SIoT. The authors in [35] propose to create a Social Network Service to assist in the creation of Goal-Oriented Prosumer Community Groups. It is our belief that the first step, PCG formation, can be greatly assisted by SIoT technologies. Several sub-steps are involved in PCG formation. The following list enumerates them and highlights how they can be completely o partially achieved thanks to SIoT technologies.

1) Prosumer's energy behaviour: Current energy behaviour can be implicitly analyzed by observing quantitative energy consumption and production. An IoT infrastructure for data aggregation can be enough to query these measurements. However, the trustworthiness of the obtained data can be compromised. For example, the prosumer could change consumption/production data to obtain a benefit by joining a PCG with pre-qualification criteria that otherwise he/she couldn't have applied for. The SIoT offers two trust mechanisms: objective trustworthiness and subjective trustworthiness [21]. In the current setting, where it is of great importance to avoid SIoT-related attacks to prevent community deterioration, subjective trustworthiness mechanisms are preferred, as they are able to isolate malicious nodes, even with 70% of malicious nodes [21].
2) Prosumer Community Group dynamic formation: Ad-hoc networks are the basis of SIoT and, as such, some of the SIoT basic relationships and discovery mechanisms are based on social interactions such as SOR and Co-work Object Relationship (CWOR). Therefore, prosumers that belong to a social group are more likely to share the same energy prosumption behaviours and energy-related goals. Hence, ad-hoc social relationships between objects can have a beneficial impact on PCG formation, but it will not replace human intervention. For example, a goal-oriented PCG formation mechanism has to be aware of energy-related goals that prosumers want to achieve in the future. Prosumers can input their desired energy-related goals via a web-service or a mobile app. This information is then shared among and via "friends" via SIoT relationships, as it is done in [33], such process is highlighted in Section III-A. Human-to-human interaction between prosumers and authorized agents is tackled in [35], which we envision that could be nourished by a lower layer, the SIoT, leveraging social relationships between objects to exchange prosumer's energy consumption and production data and their goals, optimizing potential PCG target prosumers and network traffic.

As demonstrated in [36], a network with a giant component can be generated by only using three types of relationships: OOR, SOR and CWOR, the same ones we propose to create PCGs. This means that, eventually, there will exist a path between all SIoT-enabled devices. Even if CWOR relationships are excluded, the creation of the giant component is still possible. The authors also conclude that new SIoT search mechanisms can take advantage of external metrics (distance metrics between SIoT objects characteristics that are external of the network) to improve their performance. Sections V to VII are devoted to find object distance in the context of PCGs.

C. Goal-Oriented Prosumer Community Group dynamic formation

This section illustrates the envisioned process of dynamic group formation that leverages the SIoT paradigm. Assuming that a user is able to perform actions in a SIoT cloud platform such as Lysis [33]:

1) A user interested in being part of a PCG to benefit from the community and achieve certain goals installs PCG discovery app on his/her smartphone. The smartphone becomes a Social Virtual Object (SVO) [33].
2) The potential PCG user installs apps or allocates apps in the cloud that enable retrieving data from his/her energy-related devices (devices that produce or consume energy), enabling social capabilities for each one (SVO).
3) Devices and DERs that consume and/or produce energy owned by an owner establish an OOR relationship (Fig. 2, 1). This relationship is then leveraged to obtain energy-related data from DERs that belong to the same owner. These data can be both aggregated and stored or stored individually for each device.
4) The data obtained is used to build an energy user profile along a time period. The profile is then stored as part of the PCG app, and thus enabling the smartphone and other social devices to retrieve such information. A social device is a device that the user (prosumer) usually carries with himself/herself. We specifically refer to social devices as the key enablers to create SORs between users, as demonstrated in [36].
5) When the user establishes a social relationship with another potential user (Fig. 2, 2), both social devices (i.e, smartphones) also establish social relationships (Fig. 2, 3) and exchange energy profiles and goals. Then, the SVOs determine if a PCG needs to be created or one of the users is eligible to be part of an existing PCG (i.e, the PCG the other user is part of).

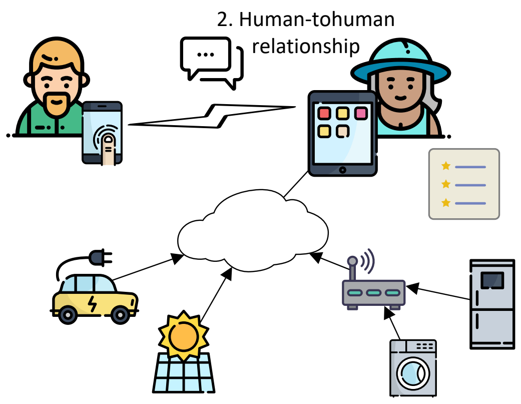


Fig. 2. Prosumer Community Group formation leveraging SIoT. Firstly, prosumers and appliances establish an OOR via the augmented SVOs (1). When prosumers come close together and establish and interact (2), their devices establish SOR or CWOR relationships and exchange the energy profiles of the prosumers (3).

IV. Use case: Demand Side Management and Social Internet of Things

This section presents a use case where a SG-enabled feature, namely DSM, is complemented and improved by leveraging SIoE.

The SG improves reliability performance and allows customers' responsiveness by encouraging better efficiency decisions. DSM represents an integral part of SG. DSM is the modification of consumer's demand for energy through various methods to better match the demand for power with the supply. Some examples of DSM programs are Direct Load Control (DLC) and Real Time Pricing (RTP). "DLC programs for residential load management are based on an agreement between the utility company and the customers. The utility, or an aggregator, can remotely control the operations and energy consumption of certain appliances such as lighting, thermal comfort equipment, refrigerators, and pumps" [37]. "According to RTP programs, the price of electricity varies at different hours of the day and each user is expected to individually respond to the time-differentiated prices by shifting its own load from the high-price hours to the low-price hours" [37]. In this context, Interruptible Loads (ILs) are consumers who agree to be interrupted, as required and within constraints, to maintain system security or reduce market prices, and are usually compensated by paying reduced tariffs.

We present an example scenario using SIoE for DSM as follows: a user utilizes a specific mobile application to enable SIoT interactions. The application uses the energy profile gathered by querying the user's home appliances. The application also asks the user to set time periods where the smart appliances can function; for example, a user is usually at home at 17:00 and he can have the clothes in the washing machine at 17:30. He only needs to run the washing machine once, and it can be from 17:30 to 19:00. In that way, the user sets the time period from 17:30 to 19:00. Finally, the application shares the energy profile with other users using SIoT relationships, for example, using SORs. When two applications detect that both user's energy profiles are compatible, they create a small PCG. Given two or more prosumers $a, b, c, \ldots$, we define their "compatibility" as how much they contribute in reducing their join Peak-to-Average Ratio (PAR). Their compatibility increases as $par(a + b + c + \ldots)$ gets colser to 1. The process can be repeated for each new user that meets a user in the PCG via SIoT relationships. This process aims to create multiple PCGs that try to autonomously reduce overall energy peaks and improve energy generation efficiency. In case a user needs to start an appliance out of his autonomously scheduled time, the application leverages SIoT relationships within its PCG to find a "friend" able to re-schedule the time when his appliances will start functioning to stabilize load curve as much as possible between both users. Which will, in turn, stabilize the load curve of the PCG they are in. This process is repeated until the overall load curve is optimized for the situation. The process just described can harness the potentialities that integration with the WoT provides. Related work in merging the WoT and SG for web-enabled, energy-aware smart homes describes a similar DSM use case. The authors in [38] enable smart appliances to the web by providing an HTTP/S API that allows the interaction with them by sending queries and commands. It is proposed that the SG Utility informs about variable electricity tariffs to a Home Controller, which in turn tries to reschedule the working period of the smart appliances. Hence, the WoT is used as a set of standards to create an energy-aware smart home.

A. Properties

In the presented scenario, we can identify properties and issues to be addressed. The following is a list comprehending those properties and next sections analyze each one in more detail.

- The SIoT paradigm enables local rather than global search and navigability, although this does not mean that distant nodes cannot establish a relationship between themselves. What is to be tackled in this regard is the eligibility of a node to be a member of a cluster (PCG) and how to search for new members.
- There are several clusters of users (PCGs). This means that users are distributed into groups according to some variables. By splitting users into groups, energy management can be greatly improved as in MG, VPP and PCG.
- A scheduling algorithm is needed in order to schedule the times when electrical appliances should start in order to reduce energy consumption peaks. The scheduling algorithm can be based on a global knowledge assumption or agent-based knowledge, meaning that each agent/user contributes to the scheduling process without sending all information to a ”global” entity that executes the algorithm.
- In order to evaluate scenarios involving combinations of the items presented before, we can assess two properties: communication efficiency and overall energy efficiency.

1) Cluster organization: As it has been already mentioned, we expect to create multiple PCG using the SIoT paradigm (Fig. 3). Yet, the inter- and intra-cluster organization and interaction has to be explored. Once the appropriate members to form the PCG have been found, two main intra-cluster member organization can be analyzed.

- Master-slave organization: The cluster has a cluster leader or master node. The master node is responsible for running the scheduling algorithm with the information of its slave nodes. The master node has to be capable of running a scheduling algorithm within the time constraints set to achieve an optimal energy management, it should have enough resources for achieving such a task. A master-slave organization also supposes a Single Point of Failure (SPOF). The SPOF problem can be partially alleviated as the SIoT paradigm assumes the use of VOs to represent the virtual counterparts of physical devices. Those SVOs are usually allocated in the cloud as in [33], which means that devices are augmented with virtual resilience and higher computing power. Otherwise, if a complete cloud model is followed, a model where the functionalities of the master node are replaced by a component at the aggregation layer can be implemented. Following the approach implemented in Lysis [33], each SVO is able to perform its own optimization process. This information is then aggregated per PCG at the aggregation layer via a ME. This ”aggregation layer” model is not strictly a master-slave organization as understood in the literature. However, both models follow the same structure. For the sake of simplicity, and to avoid adding more classifications, we have considered this “aggregation layer” approach as a master-slave organization. Disambiguation and clarifying explanations are added when necessary.

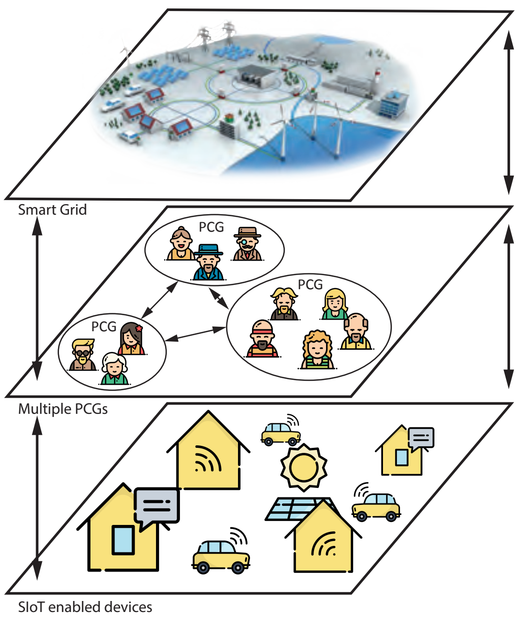


Fig. 3. Multiple PCG structure.

- Fully distributed: The cluster is organized without a cluster leader and all the nodes participate in the scheduling process. They do not share their information with all the nodes in the cluster but only with the ones they need to set their own scheduling. Therefore, algorithms that come into play in this fully distributed organization are mainly agent-based, asynchronous and distributed algorithms.
- Distributed computation: Another cluster organization which is not to be confused with the master-slave organization explained before (although it can be part of it) is that the nodes involved in the cluster are also part of the computation resources used by the master node. Therefore, this cluster organization refers to the parallelization of the computation and the nodes’ organization to attain such goals. For example, in [39] nodes are used to distribute the computation of a parallel genetic algorithm to optimize energy consumption in a building.

2) Scheduling algorithms: In order to distribute energy consumption processes over time in a way that reduces the PAR a scheduling/planning algorithm is needed. A first division of the type of algorithms considers how the knowledge is distributed among all nodes. Thus, global-knowledge algorithms and agent-based algorithms can be considered.

- Global-knowledge algorithms need all the information gathered by the PCG to extract an optimal energy plan for the members of the PCG. Therefore, there should be a component that aggregates the information, either a cluster leader (master node) or an aggregator component at the aggregation layer. Global-knowledge algorithms align with the master-slave cluster organization as all information is eventually conveyed to the master node or at the aggregator. Note that although one can think of the utility as the master node, a further subdivision into PCGs can be applied. Then, slave nodes can convey the information needed to the master node to run the algorithm and master nodes can send the information to the utility when needed.
- Agent-based and distributed algorithms account for keeping the core information in each node where each node only shares the information needed to schedule a proper plan. Those algorithms align with the fully-distributed cluster organization.

A second division considers the type of algorithm to be used. They usually consider energy efficiency and/or user comfort as a stochastic optimization problem. The explanation that follows pretends to give an overview of the types of algorithms used in DSM but is far from an exhaustive analysis of the state of the art. The reader can refer to [40] for an in-depth analysis. Appropriate references to revisions of such algorithms are provided when available. Some work analyzed in this section does not focus on load shifting *per se*, but it is relevant in that the reduction in electricity price has a relation with energy saving.

Global-knowledge DSM techniques for load shifting need the information of all ILs before running. Evolutionary Algorithms (EAs) (which include genetic algorithms) redistribute shiftable loads in order to achieve the best solution based on the objective function of load shifting and user comfort [41]–[43]. Particle Swarm Optimization (PSO) algorithms also prove to be successful in load shifting and scheduling of ILs [44]–[46].

Other works present in the literature focus on game theory to solve these optimization problems. Game-theoretic algorithms can be distributed easily. Indeed decision theory fields such as game-theory or evolutionary game-theory combined with Multi-Agent Systems (MASs) is recurrent in modern literature [47]. Moreover, MASs applied to SGs and, more concretely, to MG management and DSM is gaining momentum as a solution to give autonomicity in the management of such complex systems [48], [49]. References [47]–[49] support the idea that a mid-large-scale SG is likely to be managed in a cooperatively and more autonomously manner with multiple agents involved. Therefore, the SIoT paradigm presents itself as a dynamic, flexible and scalable organization solution to such approach.

3) A structural perspective on Demand Side Management and Social Internet of Things: So far, EA, PSO algorithms and game-theoretic algorithms in DSM contexts have been presented in order to introduce the reader to global- and local-knowledge algorithms. It is obvious that other types of algorithms have been used for DSM apart from those. The focus of this section is to analyze the implications of global-knowledge algorithms and local-knowledge algorithms or, rather, the implications of different cluster organizations in merging DSM and SIoT.

If the PCG is organized in a master-slave structure:

- The master node is responsible for running the scheduling algorithm and thus should have enough resources to compute the result.
- As the information of each member of the cluster will be conveyed to the master node, each slave node should have a direct relationship to the master node.
- Considering a possible master node failure and in order to avoid a SPOF, slave nodes must form a strongly connected component (graph theory). If the current master node goes down, a new master node can be chosen by means of a distributed quorum.
- If a similar structure is followed but with an aggregator component instead of a master node, the aggregator component may go down due software or underlying hardware malfunction. Since it is a component in the cloud, it can be automatically re-instantiated.

If the PCG is organized as a fully distributed cluster with no master node:

- Each node cooperates with other nodes in the PCG. Nodes themselves are responsible for computing their rescheduling on the basis of the information of other nodes.
- Each node sends their predicted scheduling information to a subset of nodes in their PCG. Given that load consumption can be predicted [50], [51], some nodes are more likely to be compatible with each other, meaning that they are more likely to achieve a lower PAR.
- Although the intra-cluster rescheduling computation is not done in a master-slave structure, PCGs need to communicate in order to optimize overall PAR. A master node should be elected for this purpose, but note that the failure of the node and consequently, the election of a new one, is not as resource-consuming as in the master-slave organization when it requires a master node.

4) Solution evaluation: Resulting solutions when evaluating the application of SIoT to the SG can be assessed in two manners. The first one involves communication-related aspects and deals with:

- The searchability of new members to include in a PCG.
- The navigability among the members that pertain to a PCG and between different PCGs.
- The performance of the communication. Special attention must be paid to the relationships between nodes of PCGs and the number of messages exchanged.
- There is also the possibility of moving the SVOs from the cloud to the edge cloud in order to reduce communication overhead. The work presented in [52] analyzes this possibility and associated challenges and shows that moving 75% of SVOs to the edge cloud reduces network overhead considerably.

The second manner of evaluation concerns the energy efficiency of the proposed solutions. As already pointed out, there are several methodologies or algorithms that help in predicting energy consumption loads and even adapt to changes. The fitness of an algorithm will be ultimately measured by its capacity to optimize energy efficiency while being able to leverage the scalability, flexibility and navigability provided by the SIoT paradigm.

V. Clustering Model

In this section, we will explain the model we propose in this work and future ones. According to [40], there are six (6) basic DSM services (Fig. 4):

1) Peak Clipping: Reduction of grid load.
2) Conservation: Reduction of the load during the day by reducing overall consumption. For example, by using more energy-efficient appliances.
3) Load Building: Increase the load by increasing the overall consumption.
4) Valley Filling: Improvement of system load factors such as PAR by increasing load during off-peak periods.
5) Flexible Load Shape: Specific contracts and tariffs that allow the control of prosumer's equipment.
6) Load Shifting: Reduction of grid load during peak demand and load building across off-peak periods.

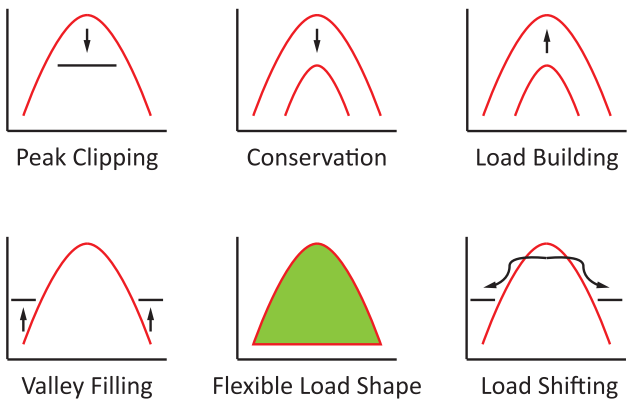


Fig. 4. Load shifting effects. Vertical axis is power and horizontal axis is time.

As the purpose of DSM is to achieve those effects when desired, the granularity of the model allows the representation of each consumer's appliance as a load. Also, as the number of residential DERs increases, energy generation and storage must also be represented. Each load type should abide by certain rules, imposed by their characteristics and prosumer preferences. Some loads cannot be shifted and allow to reduce the energy they demand by changing their working mode; other loads should be treated as non-shiftable loads because the user does not allow the system to shift the load, or simply they cannot be shifted; energy generated by DERs may not be storable and a load building effect should take place to leverage the extra energy produced; and so on. By rescheduling positive loads (energy consumption) and negative loads (energy generation) better system load factors can be achieved. The DSM model that follows is the simplest one, it considers only electricity consumption and the PAR as the metric to measure the achieved optimization after clustering and rescheduling the loads. We acknowledge, however, that the DSM challenge is a multi-objective challenge. For example, only aiming to reduce the PAR may lead to an increase in the amplitude of all energy peaks without the proper policy; we address this problem in Algorithm 2.

In order to create PCGs or clusters, our model aims to create clusters of compatible prosumers to reduce the inter-cluster PAR by reducing the intra-cluster PAR. Let $\mathcal{U}$ be the set of users and $\mathcal{T}$ the set of time slots, where $U \triangleq |\mathcal{U}|$ and $T \triangleq |\mathcal{T}|$. The symbol $\triangleq$ indicates that letter in the left hand side is the cardinality of the set on the right hand side. For each $u \in \mathcal{U}$, we define the power consumption vector as

$$\vec{l_{u,T}} = [l_u^1, ..., l_u^t, ..., l_u^T] \tag{1}$$

where l_u^t is the energy consumption of user u at time slot t. As the model allows for appliance granularity, the set of appliances that belong to a user u are represented by $\mathcal{A}_u$ and $A_u \triangleq |\mathcal{A}_u|$. Then, l_u expands to the matrix

$$lm_{u_{A_u,T}} = \begin{bmatrix} lm_{1,1} & \dots & lm_{1,t} & \dots & lm_{1,T} \\ lm_{2,1} & \dots & lm_{2,t} & \dots & lm_{2,T} \\ \vdots & & \vdots & & \vdots \\ lm_{A_u,1} & \dots & lm_{A_u,t} & \dots & lm_{A_u,T} \end{bmatrix} \tag{2}$$

Total load per user $u \in \mathcal{U}$ during T time slots is denoted by L_u

$$L_u = \sum_{t=1}^{T} l_u^t \tag{3}$$

the PAR is then denoted by

$$par(u)_{u \in \mathcal{U}} = \frac{\max_{t \in \mathcal{T}} l_u^t}{\frac{1}{T} L_u} \tag{4}$$

and is defined in a similar manner by each appliance at a more granular level and by each cluster in a less granular level. In fact, intra-cluster consumption can be represented by a vector similar to the one presented in Eq. (1) and then dis-aggregated as per consumer in a similar manner as in Eq. (2).

The objective is to create clusters of prosumers as much compatible as possible and with the capability of self-managing their energy needs as a community. For the sake of simplicity, the time subindex T in $\vec{l_{u,T}}$ is omitted in the descriptions that follow. The set of load shapes is defined as $\mathcal{L}$ where $\mathcal{L} = \{\vec{l_{u1}}, ..., \vec{l_{u_i}}, ..., \vec{l_{uU}}\}$. We define the set of clusters as $\mathcal{C}$ and its cardinality as $K \triangleq |\mathcal{C}|$, where $\mathcal{C} = \{C_1, ..., C_k, ..., C_K\}$ and $C_k \subset \mathcal{L}$. All members in $\mathcal{C}$ are in the same microgrid. Moreover, members of a cluster k cannot pertain to other clusters: $\forall (C_k, C_m) \in \mathcal{C}; k \neq m : C_k \cap C_m = \varnothing$. Given two or more prosumers $l_{ux}, l_{uy}, l_{uz}, ...$ where $x \neq y \neq z$, we define their "compatibility" as how much they contribute in reducing their join PAR ($par(l_{ux} + l_{uy} + l_{uz} + ...)$). For each cluster C, it is desirable to maximize their compatibility, which means to minimize their PAR to be as closer to 1 (lowest value) as possible. Given that each cluster is a subset of $\mathcal{L}$, the compound PAR of all clusters is equal to $par(\mathcal{L})$, Eq. (5).

$$\forall C_k \in \mathcal{C} : par(\bigcup C_k) = par(\mathcal{C}) = par(\mathcal{L}) \quad (5)$$

In the same manner as $\vec{l_u}$ expands to $lm_{u_{A_u},T}$ (Eq. (2)), a cluster $C \in \mathcal{C}$ can be collapsed by calculating the summation of all profile shapes $\vec{l_u} \in C$, that is the cluster profile shape. We will refer to the latter as $\vec{C} = \sum_{i=1}^{|C|} \vec{l_{u_i}}$ for each $\vec{l_{u_i}} \in C$. Also $\vec{\mathcal{C}} = \sum_{k=1}^{K} \vec{C_k} = \sum_{i=1}^{U} \vec{l_{u_i}}$.

A. Algorithm

This section presents the algorithm used to create consumer communities and reschedule their consumption in order to minimize the PAR. Our objective is to assess the viability of reducing the PAR per community and experiment how PAR is reduced after consumption rescheduling per community. It is, however, not in the scope of this article to simulate inter-community communication to reduce global PAR. Also, no contribution from SIoT relationships is expressed in this algorithm. The rationale is that the algorithm and the analysis of the results will serve as the reference measurements given a concrete dataset. We rely on the giant component (Section III-B2) as the enabler to eventually create PCGs that will include all the members of the dataset. Once reference measures and external metrics are known, we will be able to study the formation of new meaningful relationships to improve navigability for this specific use case (Demand Side Management). Future improvements are explained in Section VIII. Note that the word *cluster* is used to refer to PCGs as this word is commonly used in algorithms, but we do not use a conventional clustering algorithm.

Given a dataset $Dataset$ containing user load profiles, create K clusters ($C_k \in \mathcal{C}$) that minimize $dist(\mathcal{C})$ to make each cluster similar to one another in terms of PAR while minimizing the PAR of each one (Algorithm 1). We will refer to $dist(\mathcal{C})$ as the clustering heuristic. Then, for each cluster, reschedule its elements so that $par(C_k) > par(C_k^{'})$ (Algorithm 2). In order to get an optimal solution, Algorithm 1 is run $len(D) * K$ times and the solution with the lowest $dist(\mathcal{C})$ is taken. Each time Algorithm 1 is run, K points/seeds are randomly chosen from the initial dataset, and each seed is the initial element of the cluster C_k. Note that operations such as $\mathcal{C}$ += C_k represent the replacement of element C_k in set $\mathcal{C}$. On the contrary, operations such as c = $C_k + d$ represent adding the point d to the cluster C_k and assigning the result to c, without modifying C_k nor d (notice that there is not an equal sign in the operation).

In Algorithm 2, instead of applying the rescheduling process to the cluster at once using a resource-consuming algorithm, we choose to reschedule one dis-aggregated appliance ($lm_{a_u,t}$) at a time. Doing so, we pursue to avoid consuming too many resources using stochastic methods for planning and rescheduling. We envision each cluster to be able to apply load rescheduling of all of its members, and the node applying the algorithm could be a commodity hardware. The "rescheduleOnce" algorithm works as follows:

```
Data: $Dataset$
Result: $\mathcal{C}$
$\mathcal{C}$ = $\mathcal{S}$ = seeds($K$, $Dataset$)  ▷ Randomly chooses
  $K$ points from dataset $Dataset$
foreach $d$ in $Dataset - \mathcal{S}$ do
    c = $C_1 + d$
    cSet = $\mathcal{C}$ + c
    bestDistance = $dist(cSet)$
    bestCSet = cSet
    foreach $C_k \in \mathcal{C}$ do
        c = $C_k + d$
        cSet = $\mathcal{C}$ + c
        if $bestDistance > dist(cSet)$ then
            bestDistance = $dist(cSet)$
            bestCSet = cSet
        end
    end
    $\mathcal{C}$ = bestCSet
end
```
Algorithm 1: First step of the algorithm.

1) Given a cluster C, find which member has the highest contribution in increasing $par(C)$.
2) For all the appliances of that member, reschedule when each of them should be functioning using Algorithm 3 in order to reduce $par(C)$. We assume that each appliance can be freely rescheduled. Then, given an appliance, permutations for each of its elements are tried. A permutation is implemented if Algorithm 3 evaluates "true". The arguments passed to Algorithm 3 are: $\vec{C}$ as the original cluster load shape and $\vec{C'}$ as the original load shape but with a permutation in the load shape of one of its contained appliances.
3) Finally, repeat from Item 1 until there is no significant change in the value of the intra-cluster PAR.

```
Data: $C_k$
Result: $C_k^{'}$
$C_k^{'}$ = $C_k$
clusterSize = $|C_k^{'}|$
memory = Memory(clusterSize)
 ▷ An empty list of clusterSize elements
push(memory, $par(C_k^{'})$)
i = 0
do
    if $i$ == clusterSize then
        i = 0
        reset(memory)
          ▷ Delete all elements on the list
    else
        i = i + 1
    end
    $C_k^{'}$ = rescheduleOnce($C_k^{'}$)
    push(memory, $par(C_k^{'})$)
while $i$ != clusterSize or significantChange(memory)
```

Algorithm 2: Second step of the algorithm. This algorithm is applied to each $C_k \in \mathcal{C}$ and produces a new set $\mathcal{C}^{'}$ where each cluster has a lower PAR.

```
Data: $\vec{C}, \vec{C'}$
maxIsLowerThanBefore = $max(\vec{C'}) < max(\vec{C})$
maxIsEqualAsBefore = $max(\vec{C'}) == max(\vec{C})$
lessMaxsThanBefore =
  $count(\vec{C'}, max(\vec{C'})) < count(\vec{C}, max(\vec{C}))$
if lessMaxsThanBefore and maxIsEqualAsBefore or
  maxIsLowerThanBefore then
    true
else
    false
end
```

Algorithm 3: Predicts if the permutation will reduce the cluster PAR. $max(\vec{c})$ is an element-wise operation that returns the maximum value in $\vec{c}$. $count(\vec{c}, n)$ is an element-wise operation that counts how many elements equal to n are in $\vec{c}$.

VI. Implementation and Results

A. Implementation

As our objective is to develop a strong and consistent model for future extension, the programming language chosen has been Scala [53]. Scala offers a strong and static type system, which reduces the chances of errors during runtime, as opposed to dynamically typed languages such as Python [54]. Scala also offers algebraic and statistical libraries, although the ones found in the Python community have more functionality. Therefore, Scala has been used to implement the model and algorithms explained in this article.

B. Tests and results

To test the clustering and rescheduling method proposed in Section V, part of the large dataset provided by Dataport [55] is used. We have selected households which have a smart meter installed with individual circuits for each appliance. 189 enrolled households in 2015-01-01 had a smart meter (eGauge device) with individual circuits for each appliance. eGauge readings are per hour, meaning that each record contains power readings for up to 12 circuits during 24 time spans.

The objectives of the following tests are to discover how well perform Algorithms 1, 2 and 3. Different heuristics are tested in Algorithm 1. A good solution is one that distributes prosumers among several clusters and keeps the cluster PAR as low as possible. Algorithm 2 is then run using the output of Algorithm 1 as the input. The metrics used to measure how well is a solution are explained along with the meaning of each column of the table that presents the results. Tests have been run for several K, using two different $dist(\mathcal{C})$ metrics (from now on, $dist(\mathcal{C})$ metrics are referred as *aggregate metrics*). Results are presented in tables. In order to refer to each row of the table, we will take the value of K as the row index. The values in each column are the mean of several runs. The meaning of the different columns are (for consistency, the same names and associated meaning are used throughout the text):

- points per cluster: is an array enclosed by square brackets ("[]") and each value separated by a comma (",") that represents the number of points or $\vec{l_u}$ assigned to each cluster. The values of this column contain the *best* cluster configuration. Our criteria to choose the *best* cluster configuration is that it has the lowest "s1. max m" among all runs. The rationale for choosing "s1. max m" over "s1. agg m " is that the variations in the latter are almost negligible.
- s1. agg m (Step 1 Aggregate Metric): the values correspond to the result after applying the chosen $dist(\mathcal{C})$ in Algorithm 1.
- s1. max m (Step 1 Maximum Metric): each value corresponds to the $\max par(C_k)$ after Algorithm 1 is applied.
- s1 peak (kW) (Step 1 peak (kW)): the values in this column correspond to the aggregate maximum energy peak $\max l_C^t$ after Algorithm 1 is applied.
- s2. agg m, s2. max m, s2 peak (kW) have the same meaning as their "s1" counterparts except that they are calculated after Algorithm 2 is applied.
- total m (Total Metric) is calculated as the $par(Dataset')$ where $d^{'} \in Dataset^{'} | d^{'} \in \mathcal{C}^{'}$ after Algorithm 1 and Algorithm 2 have been applied. That is, the PAR of all points (load shapes) as if there was only one big cluster. Note that Algorithm 2 only operates per cluster, so it is the case that this value increases or decreases without following any pattern.

The first aggregate metric tested is:

$$dist(\mathcal{C}) = \frac{\sum_{k=1}^{K} par(C_k)}{K} \tag{6}$$

That is, the average of PARs of each cluster. Results are presented in Table I. The values for $K = 1$ are the same as those without applying any clustering algorithm, as there is only one big cluster, which we can take as a reference case. It can be appreciated that the PAR is 1.37 at the beginning and it decreases down to 1.28 after applying Algorithm 2. Also, after applying Algorithm 2 and due to Algorithm 3, energy peak lowers from 676 kW to 633 kW. However, for $K \neq 1$, the aggregate metric used is not fair when distributing points across clusters. It tends to create one big cluster and then several small ones or clusters with zero (0) elements. The following inequations hold true: "s1. max m" > "s2. max m", "s1. peak (kW)" > "s2. peak (kW)" for any number of K, which means that Algorithm 2 improves the PAR and reduces the energy peak of each cluster. "s1. max m" and "s2. max m" also tend to increase as K increases. As the column "total m" indicates, the real PAR remains almost the same and it improves the value for $K = 1$ sometimes. However, our objective is to create multiple PCGs and the aggregate metric used does not serve for that purpose, as it tends to group users in one single cluster.

In order to address the unfairness of the first experiment, we propose to use the following aggregate metric:

$$dist(\mathcal{C}) = par(\vec{\mathcal{C}_{par}})|\vec{\mathcal{C}_{par}} = \forall C_k \in \mathcal{C} \rightarrow par(C_k) \quad (7)$$

$\vec{\mathcal{C}_{par}}$ is a vector whose components are the PAR of each one of the clusters (for each $C_k \in \mathcal{C}$, calculate $par(C_k)$). Then, $par(\vec{\mathcal{C}_{par}})$ is the PAR of that vector. PAR is calculated using Eq. (4). The intention of the aggregate metric just presented is to equally distribute the PAR of each cluster by reducing the difference between $par(C_k)$, which also better distributes the number of users assigned to each cluster. Results are shown in Table II. An inspection of the column "points per cluster" shows that the users are more equally distributed across clusters. The results in "s1. agg m" show how equal are the PARs of each cluster. In fact, for $K \neq 1$, "s1. agg m" is one (1), which means that the value present in the column "s1. max m", which is the highest PAR value among the clusters, is equal for all clusters. Again, thanks to Algorithm 2, PAR and the maximum energy peak is reduced for each K after its application, as shown in Figs. 5 and 6. Fig. 5 shows how the maximum aggregate PAR tends to increase as K increases as there are fewer elements per cluster and therefore, it is less likely that the elements in each cluster are able to compensate the per-cluster metric. It is, however, possible to reverse this tendency after the rescheduling process, as less rescheduling movements are able to achieve a better metric or worsen it. Fig. 6 shows how peak power decreases after applying Algorithm 3, but it does not follow a specific pattern, as the main goal is to reduce the PAR. "total m" column shows that the real PAR remains almost the same. Note also that for $K = 2$ there is an improvement regarding $K = 1$, which we can take as the reference solution as it has the lowest PAR ("s1. max m" tends to increase as K increases). Clusters contain almost the same number of points, 95 and 94 respectively, PAR for both clusters is the same as for $K = 1$ and, after applying Algorithm 2, "total m" is lower than for $K = 1$. This solution is not only better in terms of the values presented, it is also better because Algorithm 2 can be run in parallel for each one of the clusters and thus alleviating the centralization of computing resources needed to apply DSM, which is one of the main objectives of this work.

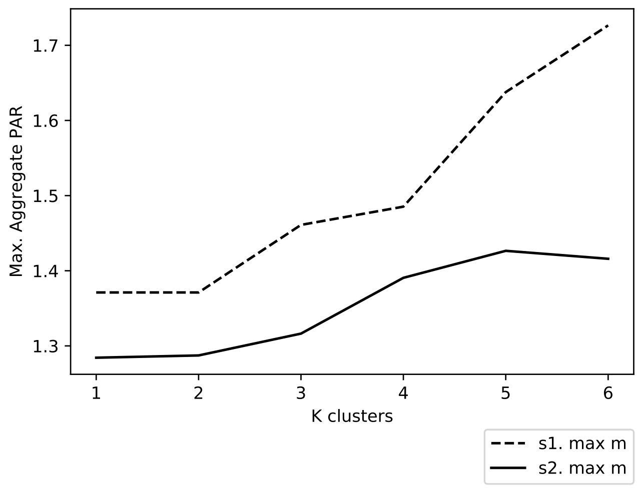


Fig. 5. Comparative between Maximum Aggregate PAR before (dashed line) and after (solid line) applying the rescheduling algorithm. Values (Y-axis) per each K (X-axis) are computed using Eq. (7).

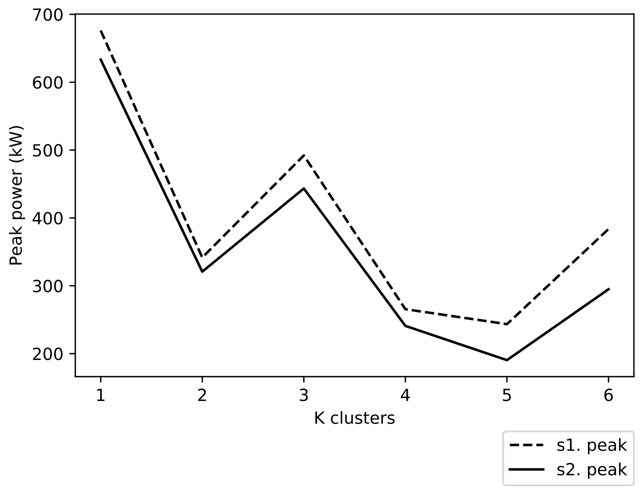


Fig. 6. Comparative between Peak power (kW) before (dashed line) and after (solid line) applying the rescheduling algorithm. Values (Y-axis) per each K (X-axis) are computed using Eq. (7).

C. Discussion

Two heuristics (metrics) have been applied to distribute consumers (points) across several clusters and then their energy load profiles have been rescheduled to improve the PAR per cluster. The first proposed heuristic (Eq. (6)) tends to group the majority of consumers into one single cluster, whereas the second heuristic (Eq. (7)) distributes consumers more fairly, meaning the number of consumers per cluster is more egalitarian. We choose the latter heuristic (Eq. (7)) as better because our objective is to create PCGs that are able to

TABLE I. SIMULATION RESULTS: AVERAGE AGGREGATE

K	points per cluster	s1. agg m	s1. max m	s1. peak (kw)	s2. agg m	s2. max m	s2. peak (kw)	total m
1	[189]	1.371114	1.371114	676.160150	1.284301	1.284301	633.348667	1.284301
2	[0, 189]	1.185557	1.371114	676.160150	1.142150	1.284301	633.348667	1.284301
3	[18, 171, 0]	1.141561	1.386598	657.846300	1.108997	1.289023	611.553717	1.279029
4	[20, 0, 4, 165]	1.126272	1.394957	641.170733	1.096543	1.278069	587.445167	1.262169
5	[0, 8, 12, 153, 16]	1.125646	1.419435	609.907067	1.099749	1.302855	559.814567	1.271343
6	[0, 25, 11, 138, 11, 4]	1.122694	1.430808	586.851050	1.088956	1.255731	515.042300	1.224344

TABLE II. SIMULATION RESULTS: PAR AGGREGATE

K	points per cluster	s1. agg m	s1. max m	s1. peak (kw)	s2. agg m	s2. max m	s2. peak (kw)	total m
1	[189]	1.371114	1.371114	676.160150	1.284301	1.284301	633.348667	1.284301
2	[95, 94]	1.000000	1.371114	341.647200	1.009352	1.287300	320.762883	1.275193
3	[42, 34, 113]	1.000007	1.461032	492.058050	1.021967	1.316365	443.335850	1.303961
4	[57, 63, 41, 28]	1.000127	1.485290	265.469917	1.049070	1.390578	240.844383	1.320364
5	[50, 21, 40, 46, 32]	1.000355	1.637695	243.410700	1.077833	1.426521	190.383117	1.308784
6	[33, 13, 71, 29, 24, 19]	1.010122	1.726447	383.596300	1.063379	1.415940	294.851850	1.298971

perform an auto-rescheduling process in order to reduce both the intra-cluster PAR and maximum energy peak (kW). Using the dataset provided by Dataport to evaluate the algorithms, we found that the best solution is obtained when $K = 2$ and Eq. (7) is used as the aggregate metric, which not only improves the reference case for $K = 1$ but also allows to distribute computation resources. The prediction heuristic (Algorithm 3) used to decrease the PAR and maximum energy peak in Algorithm 2 has proven to work successfully. However, the rescheduled loads obtained are not optimal due to the implementation, as it only uses a linear heuristic to choose the elements to reschedule, avoiding brute force and thus reducing computation costs.

VII. HIERARCHICAL CLUSTERING

From the conclusions drawn in Section VI-C and in order to improve Algorithm 1, we have developed a hierarchical clustering algorithm that supersedes Algorithm 1. Next sections explain the rationale of the algorithm and perform some experiments.

We are interested in finding and grouping $(\vec{l_{u_x}}, \vec{l_{u_y}}) \in \mathcal{L}; x \neq y : \vec{l_{u_x}} + \vec{l_{u_y}} = 0$. The justification is that $par(\vec{l_{u_x}} + \vec{l_{u_y}}) = 1$, which is the lowest possible value of PAR. In order to do so, we first need to define the *centroid* operation.

Given a generic set $\mathcal{S}$ and $\vec{s} \in \mathcal{S}$, the centroid operation is defined as:

$$centroid(\mathcal{S}) = \frac{1}{|\mathcal{S}|} \sum_{i=1}^{|\mathcal{S}|} \vec{s_i} \qquad (8)$$

Our objective is now to find a virtual mirror image $\vec{l'_{u_y}}$ for $\vec{l_{u_x}} \in \mathcal{L}$. Without losing generality, the mirror image for $s \in \mathcal{S}$ is calculated as shown in Eq. (9). Eq. (9) reads as: given the direction and length from $centroid(\mathcal{S})$ to s, find $mirror\_image(s)$ which is at the same length from $centroid(\mathcal{S})$ but in opposite direction.

$$mirror\_image(s) = \vec{l'_{u_y}} = 2 * centroid(\mathcal{S}) - s \qquad (9)$$

To find a real mirror image $\vec{l_{u_y}} \in \mathcal{L}$ for $\vec{l_{u_x}} \in \mathcal{L}$, we apply the k-nearest-neighbour algorithm to $\vec{l'_{u_y}}$ in the set $\mathcal{L}$ where $k = 1$ (1-nearest-neighbour). Also, in order to limit the size of the cluster in terms of total energy consumption, we have added the following constraint: only add $\vec{l_{u_y}}$ the to cluster if the total energy consumption of the cluster when $\vec{l_{u_y}}$ is added is below a threshold. We expect this threshold to be set by the utility. As we do not have any specific constraint, we have calculated the threshold value as the mean energy consumption per member in C. Given that C is a cluster where $\vec{l_{u_y}} \notin C$ and C' is the same cluster with $\vec{l_{u_y}}$, add $\vec{l_{u_y}}$ to C only if the average energy consumption per member in C' is less or equal to the average energy consumption per member in C. Stop adding members to the cluster otherwise. After obtaining the clusters, they are grouped using the same procedure without the total energy consumption restriction. The grouping process stops when the desired number of clusters K is met.

A. Tests and results

This section presents the results obtained when applying the hierarchical clustering heuristic just presented. The objective is to compare the results with the ones obtained in Table II. However, we do not apply a rescheduling algorithm. The rescheduling algorithm implemented in this work still does not use inter-cluster communication. This means that when rescheduling is applied, it achieves a local solution, rather than a global one and thus it is possible that it worsens the $par(Dataset')$. In this new hierarchical clustering, the number of leaf clusters is higher than in Algorithm 1. The lack of inter-cluster communication produces a larger negative impact than in the experiments performed in Section VI. We have performed experiments taking samples from 50% to 100% of the total dataset at 10% steps. The sampling procedure from 50% to 100% shuffles the entire dataset and grabs the desired number of samples. Then the algorithm is applied. The procedure is repeated multiple times and the results presented are the mean of all runs. The results presented in Table III are using the 100% of the dataset. They clearly

TABLE III. SIMULATION RESULTS: HIERARCHICAL CLUSTERING

K	points per cluster	s1. agg m	s1. max m	s1. peak (kw)
1	[189]	1.371114	1.371114	676.160150
2	[111, 78]	1.000665	1.374143	408.915451
3	[90, 60, 39]	1.006628	1.397967	277.239788
4	[35, 51, 51, 52]	1.015163	1.429992	226.177060
5	[81, 53, 33, 17, 5]	1.026647	1.462162	201.312466
6	[27, 23, 21, 24, 35, 59]	1.034796	1.492908	173.663762

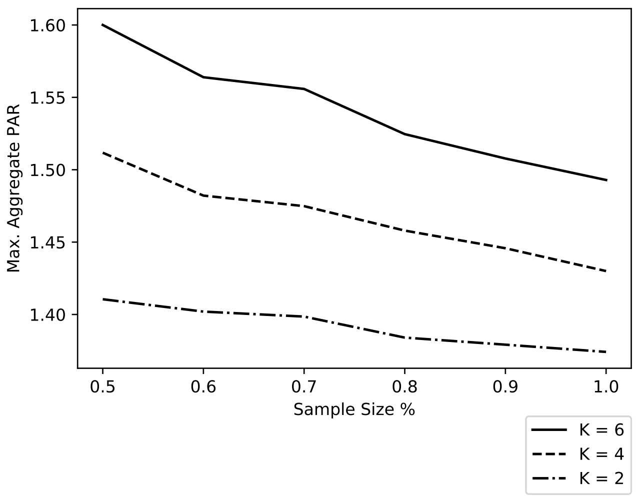


Fig. 7. Comparative between "s1 max m" for different sample sizes when K is 2, 4 and 6.

outperform the results obtained in Table II. If we look at the column "s1. agg m", it increases slightly more rapidly using hierarchical clustering, but the increases are negligible. "s1 max m" and "s1. peak" both outperform the results in Table II. The increment of "s1. max m" from $K = 1$ to $K = 6$, for Table II is 26% and for Table III is 8%. This means that the degree on which "s1. max m" increases related to the increase in the number of clusters is lower using the proposed heuristic. Also, if we inspect "s1. peak (kw)", values in Table II don't show any specific pattern, while values in Table III decrease as the number of clusters grows. This occurs as expected: if the number of clusters grows, members per cluster decrease and thus the peak power of the cluster decreases.

In Fig. 7 it can be appreciated that as the number of available prosumers grows, "s1. max m" decreases. This occurs as expected: if there are less available prosumers to cluster, there are fewer chances of finding a prosumer closer to $\vec{l'_{u_y}}$. This reasoning is also reflected in "s1. agg m", as it slightly decreases as the sample size grows.

VIII. CONCLUSION AND FUTURE WORK

Electrical energy demand is increasing rapidly and is usually met by non-renewable energy resources which are starting to be scarce, and thus novel solutions empowered by the integration of ICT and the electric power system, the SG, are emerging. The SG challenge can be studied as a particular case of the wider and broader IoT challenge, with its own specific needs. It is envisioned that DERs will be an integral part of the new electrical grid. In this work, we have presented our vision of the integration of the SG and a novel IoT paradigm, the SIoT. The term Social Internet of Energy (SIoE) is used to refer to this synergy. It allows physical devices that consume or produce energy to create social relationships to improve overall scalability. At the same time, it allows prosumers in a SG to create social relationships in order to optimize their energy usage. It has been explained how SIoT relationships can help in integrating multiple energy consuming devices both at the owner level, by leveraging Ownership Object Relationship, and at the community level, by leveraging Social Object Relationship and Co-work Object Relationship. We have started to develop a model in order to test algorithms and heuristics that aim at evaluating the viability of the proposal. Evaluation tests conducted in this regard have the objective of finding a heuristic to create PCGs and then reschedule those groups individually. A publicly available dataset has been used to perform the tests. From two proposed heuristics to assign prosumers to each PCG, one has been found to distribute them in a fair manner. A PAR prediction heuristic has been used in order to perform the rescheduling process, which has been proven to reduce both PAR and the amplitude of energy peaks. From the conclusions drawn in those experiments, we have developed a new hierarchical clustering algorithm that outperforms the previous one. Having more than one PCG implies that the rescheduling process, which is an individual process for each PCG, can be performed in parallel. Although there is a long pathway in order to evaluate the proposal explained in this article, the results and the associated discussion presented in Sections VI and VII will serve as the basis for our future work. Those experiments are valuable in that they have served to obtain mechanisms to evaluate object similarity (compatibility) and egalitarian cluster distribution. They also show, given a specific scenario, the values expected for different metrics regarding energy efficiency. Once this is known, and using network modelling tools, we can simulate and explore when are those groups and networks created and how their structure is. Those analyses will serve to analyze and improve navigability and search performance by creating new connections if necessary. They will also be used to include dynamic allocation and modification of clusters/PCGs. Future work aims at:

- Improving the model, not only theoretically, but the implementation (code) that supports the algorithms, by adding: ILs, generation, and storage models and the capacity to add rules per prosumer and appliance.
- Improving network structure by including energy profile compatibility metrics during its creation and modification.
- Developing new dynamic clustering algorithms to update clusters on the basis of changes on energy profiles and prosumer-related information.
- Once the proper algorithms and heuristics are found, the next step will be to apply MAS theory and game theoretic algorithms to distribute the computations among prosumer

devices (SVOs).

Acknowledgments

This work received funding from the "Agència de Gestió d'Ajuts Universitaris i de Recerca (AGAUR)" of "Generalitat de Catalunya" (grant identification "2017 SGR 977") and ENVISERA project, funded by Spanish Ministerio de Economía y Competitividad (Plan Estatal de Investigación Científica y Técnica y de Innovación 2013-2016, grant reference CTM2015-68902-R). Part of this research has also been funded by the "SUR" of the "DEC" of the "Generalitat de Catalunya" and by the European Social Funds (grant reference 2019 FI_B2 00164). In addition, we would like to thank La Salle - Universitat Ramon Llull for their support.

Icons used in Figs. 2 and 3 made by Freepick and Gregor Cresnar at www.flaticon.com.

References

[1] "International Energy Outlook 2018," p. 21, Sep. 2018. [Online]. Available: https://www.eia.gov/outlooks/ieo/.

[2] X. Yu, C. Cecati, and T. Dillon, "The New Frontier of Smart Grids," *IEEE Industrial Electronics Magazine*, vol. 5, no. 3, pp. 49–63, 2011, ISSN: 1932-4529. DOI: 10.1109/MIE.2011.942176.

[3] R. Martín de Pozuelo, A. Zaballos, J. Navarro, and G. Corral, "Prototyping a Software Defined Utility," *Energies*, vol. 10, no. 6, p. 818, Jun. 2017, ISSN: 1996-1073. DOI: 10.3390/en10060818. [Online]. Available: http://www.mdpi.com/1996-1073/10/6/818.

[4] K. Stamatis, "Communityware Smartgrid," *21st International Conference on Electricity Distribution*, no. June, pp. 6–9, 2011.

[5] P. Asmus, "Microgrids, Virtual Power Plants and Our Distributed Energy Future," *Electricity Journal*, vol. 23, no. 10, pp. 72–82, 2010, ISSN: 10406190. DOI: 10.1016/j.tej.2010.11.001. [Online]. Available: http://dx.doi.org/10.1016/j.tej.2010.11.001.

[6] A. D. Rathnayaka, V. M. Potdar, T. Dillon, and S. Kuruppu, "Framework to manage multiple goals in community-based energy sharing network in smart grid," *International Journal of Electrical Power & Energy Systems*, vol. 73, pp. 615–624, Dec. 2015, ISSN: 01420615. DOI: 10.1016/j.ijepes.2015.05.008. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0142061515002136.

[7] A. J. D. Rathnayaka, V. M. Potdar, T. Dillon, O. Hussain, and S. Kuruppu, "Goal-Oriented Prosumer Community Groups for the Smart Grid," *IEEE Technology and Society Magazine*, vol. 33, no. 1, pp. 41–48, 2014, ISSN: 02780097. DOI: 10.1109/MTS.2014.2301859.

[8] *FP7 INTEGRIS Project Website*. [Online]. Available: https://cordis.europa.eu/project/rcn/93726_en.html (visited on Dec. 13, 2018).

[9] *FP7 FI-PPP FINESCE Project Website*. [Online]. Available: http://www.finesce.eu/ (visited on Dec. 13, 2018).

[10] G. Kariniotakis, L. Martini, C. Caerts, H. Brunner, and N. Retiere, "Challenges, innovative architectures and control strategies for future networks: the Web-of-Cells, fractal grids and other concepts," *CIRED - Open Access Proceedings Journal*, vol. 2017, no. 1, pp. 2149–2152, Oct. 2017, ISSN: 2515-0855. DOI: 10.1049/oap-cired.2017.1287. [Online]. Available: http://digital-library.theiet.org/content/journals/10.1049/oap-cired.2017.1287.

[11] V. Caballero, D. Vernet, A. Zaballos, and G. Corral, "Prototyping a Web-of-Energy Architecture for Smart Integration of Sensor Networks in Smart Grids Domain," *Sensors*, vol. 18, no. 2, 2018. DOI: 10.3390/s18020400. [Online]. Available: http://www.mdpi.com/1424-8220/18/2/400.

[12] N. Bui, A. Castellani, P. Casari, and M. Zorzi, "The Internet of Energy: A Web-Enabled Smart Grid System," *IEEE Network*, vol. 26, no. 4, pp. 39–45, 2012, ISSN: 0890-8044. DOI: 10.1109/MNET.2012.6246751. [Online]. Available: http://ieeexplore.ieee.org/document/6246751/.

[13] L. Atzori, A. Iera, and G. Morabito, "Understanding the Internet of Things: definition, potentials, and societal role of a fast evolving paradigm," *Ad Hoc Networks*, vol. 56, pp. 122–140, Mar. 2017, ISSN: 15708705. DOI: 10.1016/j.adhoc.2016.12.004. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1570870516303316.

[14] A. M. Ortiz, D. Hussein, S. Park, S. N. Han, and N. Crespi, "The cluster between internet of things and social networks: Review and research challenges," *IEEE Internet of Things Journal*, vol. 1, no. 3, pp. 206–215, 2014, ISSN: 23274662. DOI: 10.1109/JIOT.2014.2318835.

[15] D. Guinard, "A Web of Things Application Architecture - Integrating the Real-World into the Web," en, PhD thesis, 2011, p. 220. DOI: 10.3929/ethz-a-006713673. [Online]. Available: http://webofthings.org/dom/thesis.pdf.

[16] T.-Y. Chung, I. Mashal, O. Alsaryrah, V. Huy, W.-H. Kuo, and D. P. Agrawal, "Social Web of Things: A Survey," *2013 International Conference on Parallel and Distributed Systems*, pp. 570–575, 2013, ISSN: 1521-9097. DOI: 10.1109/ICPADS.2013.102. [Online]. Available: http://ieeexplore.ieee.org/document/6808239/.

[17] D. Guinard, M. Fischer, and V. Trifa, "Sharing using social networks in a composable Web of Things," in *2010 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, IEEE, IEEE, Mar. 2010, pp. 702–707, ISBN: 978-1-4244-6605-4. DOI: 10.1109/PERCOMW.2010.5470524. [Online]. Available: http://ieeexplore.ieee.org/document/5470524/.

[18] A. Kamilaris and A. Pitsillides, "Social networking of the Smart Home," *21st Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, pp. 2632–2637, 2010. DOI: 10.1109/

PIMRC.2010.5671783. [Online]. Available: http://ieeexplore.ieee.org/document/5671783/.

[19] L. Atzori, A. Iera, and G. Morabito, "SIoT: Giving a social structure to the internet of things," *IEEE Communications Letters*, vol. 15, no. 11, pp. 1193–1195, 2011, ISSN: 10897798. DOI: 10.1109/LCOMM.2011.090911.111340.

[20] ——, "The Internet of Things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010, ISSN: 13891286. DOI: 10.1016/j.comnet.2010.05.010. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1389128610001568.

[21] M. Nitti, R. Girau, and L. Atzori, "Trustworthiness Management in the Social Internet of Things," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 5, pp. 1253–1266, May 2014, ISSN: 1041-4347. DOI: 10.1109/TKDE.2013.105. [Online]. Available: http://ieeexplore.ieee.org/document/6547148/.

[22] M. Nitti, R. Girau, L. Atzori, A. Iera, and G. Morabito, "A subjective model for trustworthiness evaluation in the social Internet of Things," *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC*, pp. 18–23, 2012, ISSN: 2166-9570. DOI: 10.1109/PIMRC.2012.6362662.

[23] W. Abdelghani, C. A. Zayani, I. Amous, and F. Sèdes, "Trust management in social internet of things: a survey," in *Conference on e-Business, e-Services and e-Society*, Springer, 2016, pp. 430–441, ISBN: 978-3-319-45233-3. DOI: 10.1007/978-3-319-45234-0_39.

[24] Z. Yan, P. Zhang, and A. V. Vasilakos, "A survey on trust management for Internet of Things," *Journal of Network and Computer Applications*, vol. 42, pp. 120–134, Jun. 2014, ISSN: 10848045. DOI: 10.1016/j.jnca.2014.01.014. [Online]. Available: http://dx.doi.org/10.1016/j.jnca.2014.01.014.

[25] F. Bao, I. R. Chen, and J. Guo, "Scalable, adaptive and survivable trust management for community of interest based Internet of Things systems," in *2013 IEEE Eleventh International Symposium on Autonomous Decentralized Systems (ISADS)*, IEEE, Mar. 2013, pp. 1–7, ISBN: 978-1-4673-5070-9. DOI: 10.1109/ISADS.2013.6513398. [Online]. Available: http://ieeexplore.ieee.org/document/6513398/.

[26] L. Atzori, A. Iera, G. Morabito, and M. Nitti, "The social internet of things (SIoT) - When social networks meet the internet of things: Concept, architecture and network characterization," *Computer Networks*, vol. 56, no. 16, pp. 3594–3608, 2012, ISSN: 13891286. DOI: 10.1016/j.comnet.2012.07.010.

[27] J. Travers and S. Milgram, "An Experimental Study of the Small World Problem," *Phychology Today*, vol. 32, no. 4, pp. 179–197, 1977, ISSN: 0038-0431. DOI: 10.1016/B978-0-12-442450-0.50018-3. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/B9780124424500500183.

[28] J. Kleinberg, "The small-world phenomenon: An algorithmic perspective," in *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, ACM, 2000, pp. 163–170.

[29] Z. Li, R. Chen, L. Liu, and G. Min, "Dynamic Resource Discovery Based on Preference and Movement Pattern Similarity for Large-Scale Social Internet of Things," *IEEE Internet of Things Journal*, vol. 3, no. 4, pp. 581–589, 2016, ISSN: 23274662. DOI: 10.1109/JIOT.2015.2451138.

[30] M. Nitti, V. Pilloni, G. Colistra, and L. Atzori, "The Virtual Object as a Major Element of the Internet of Things: A Survey," *IEEE Communications Surveys and Tutorials*, vol. 18, no. 2, pp. 1228–1240, 2016, ISSN: 1553877X. DOI: 10.1109/COMST.2015.2498304.

[31] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," *Scientific American*, vol. 284, no. 5, pp. 34–43, May 2001, ISSN: 0036-8733. DOI: 10.1038/scientificamerican0501-34. [Online]. Available: http://www.nature.com/doifinder/10.1038/scientificamerican0501-34.

[32] A. Kamilaris, S. Yumusak, and M. I. Ali, "WOTS2E: A search engine for a Semantic Web of Things," in *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, IEEE, Dec. 2016, pp. 436–441, ISBN: 978-1-5090-4130-5. DOI: 10.1109/WF-IoT.2016.7845448. [Online]. Available: http://ieeexplore.ieee.org/document/7845448/.

[33] R. Girau, S. Martis, and L. Atzori, "Lysis: A platform for iot distributed applications over socially connected objects," *IEEE Internet of Things Journal*, vol. 4, no. 1, pp. 40–51, 2017, ISSN: 23274662. DOI: 10.1109/JIOT.2016.2616022.

[34] C. Marche, M. Nitti, and V. Pilloni, "Energy efficiency in smart building: A comfort aware approach based on Social Internet of Things," in *2017 Global Internet of Things Summit (GIoTS)*, IEEE, Jun. 2017, pp. 1–6, ISBN: 9781509058730. DOI: 10.1109/GIOTS.2017.8016267. [Online]. Available: http://ieeexplore.ieee.org/document/8016267/.

[35] A. Rathnayaka, V. Potdar, and S. Kuruppu, "Design of Smart Grid Prosumer Communities via Online Social Networking Communitie," *International Journal for Infonomics*, vol. 5, no. 1, pp. 544–556, 2012. [Online]. Available: https://scholar.google.com/scholar?hl=en&q=design+of+smart+grid+prosumer+communities+via+online+social+networking+communities&btnG=&as_sdt=1,5&as_sdtp=#0.

[36] C. Marche, L. Atzori, A. Iera, L. Militano, and M. Nitti, "Navigability in Social Networks of Objects: The Importance of Friendship Type and Nodes' Distance," in *2017 IEEE Globecom Workshops (GC Wkshps)*, Singapore: IEEE, Dec. 2017, pp. 1–6, ISBN: 978-1-5386-3920-7. DOI: 10.1109/GLOCOMW.2017.8269111. [Online]. Available: http://ieeexplore.ieee.org/document/8269111/.

[37] P. Siano, "Demand response and smart grids—A survey," *Renewable and Sustainable Energy Reviews*, vol. 30, pp. 461–478, Feb. 2014, ISSN: 13640321. DOI:

10.1016/j.rser.2013.10.022. [Online]. Available: http://dx.doi.org/10.1016/j.rser.2013.10.022.

[38] A. Kamilaris, Y. Tofis, C. Bekara, A. Pitsillides, and E. Kyriakides, "Integrating Web-Enabled Energy-Aware Smart Homes to the Smart Grid," *International Journal On Advances in Intelligent Systems*, vol. 5, no. 1, pp. 15–31, 2012, ISSN: 1942-2679. DOI: 10.1.1.476.3440. [Online]. Available: http://www.cs.ucy.ac.cy/ResearchLabs/netrl/papers/files/kamilaris_gridHome.pdf.

[39] C. Yang, H. Li, Y. Rezgui, I. Petri, B. Yuce, B. Chen, and B. Jayan, "High throughput computing based distributed genetic algorithm for building energy consumption optimization," *Energy and Buildings*, vol. 76, pp. 92–101, 2014, ISSN: 03787788. DOI: 10.1016/j.enbuild.2014.02.053. [Online]. Available: http://dx.doi.org/10.1016/j.enbuild.2014.02.053.

[40] B. P. Esther and K. S. Kumar, "A survey on residential Demand Side Management architecture, approaches, optimization models and methods," *Renewable and Sustainable Energy Reviews*, vol. 59, pp. 342–351, Jun. 2016, ISSN: 13640321. DOI: 10.1016/j.rser.2015.12.282. [Online]. Available: http://dx.doi.org/10.1016/j.rser.2015.12.282.

[41] M. AboGaleela, M. El-Sobki, and M. El-Marsafawy, "A two level optimal DSM load shifting formulation using genetics algorithm case study: Residential loads," in *IEEE Power and Energy Society Conference and Exposition in Africa: Intelligent Grid Integration of Renewable Energy Resources (PowerAfrica)*, IEEE, Jul. 2012, pp. 1–7, ISBN: 978-1-4673-2550-9. DOI: 10.1109/PowerAfrica.2012.6498651. [Online]. Available: http://ieeexplore.ieee.org/document/6498651/.

[42] V. Jayadev and K. Swarup, "Optimization of microgrid with demand side management using Genetic Algorithm," pp. 1–6, 2013. DOI: 10.1049/ic.2013.0124. [Online]. Available: http://ieeexplore.ieee.org/ielx7/6712970/6718582/06718595.pdf?tp=&arnumber=6718595&isnumber=6718582%5Cn.

[43] C. Bharathi, D. Rekha, and V. Vijayakumar, "Genetic Algorithm Based Demand Side Management for Smart Grid," *Wireless Personal Communications*, vol. 93, no. 2, pp. 481–502, Mar. 2017, ISSN: 0929-6212. DOI: 10.1007/s11277-017-3959-z. [Online]. Available: http://link.springer.com/10.1007/s11277-017-3959-z.

[44] M. Pedrasa, T. Spooner, and I. MacGill, "Scheduling of Demand Side Resources Using Binary Particle Swarm Optimization," *IEEE Transactions on Power Systems*, vol. 24, no. 3, pp. 1173–1181, Aug. 2009, ISSN: 0885-8950. DOI: 10.1109/TPWRS.2009.2021219. [Online]. Available: https://ieeexplore.ieee.org/document/4914742.

[45] A. Sepulveda, L. Paull, W. G. Morsi, H. Li, C. P. Diduch, and L. Chang, "A novel demand side management program using water heaters and particle swarm optimization," in *2010 IEEE Electrical Power & Energy Conference*, IEEE, Aug. 2010, pp. 1–5, ISBN: 978-1-4244-8186-6. DOI: 10.1109/EPEC.2010.5697187. [Online]. Available: http://ieeexplore.ieee.org/document/5697187/.

[46] P. Faria, Z. Vale, J. Soares, and J. Ferreira, "Demand Response Management in Power Systems Using Particle Swarm Optimization," *IEEE Intelligent Systems*, vol. 28, no. 4, pp. 43–51, Jul. 2013, ISSN: 1541-1672. DOI: 10.1109/MIS.2011.35. [Online]. Available: http://ieeexplore.ieee.org/document/5744065/.

[47] Panait, Luke, L. Panait, and S. Luke, "Cooperative Multi-Agent Learning: The State of the Art," *Autonomous Agents and Multi-Agent Systems*, vol. 3, no. 11, pp. 387–434, 2005, ISSN: 13872532. DOI: 10.1007/s10458-005-2631-2.

[48] R. B. Smith, L. R. Phillips, H. E. Link, and L. Weiland, "Agent-based control of distributed infrastructure resources," Sandia National Laboratories (SNL), Albuquerque, NM, and Livermore, CA, Tech. Rep. January 2006, 2006, p. 118. DOI: 10.2172/883135. [Online]. Available: http://www.sandia.gov/ccss/documents/sand_2005_7937.pdf.

[49] C.-S. Karavas, K. Arvanitis, and G. Papadakis, "A Game Theory Approach to Multi-Agent Decentralized Energy Management of Autonomous Polygeneration Microgrids," *Energies*, vol. 10, no. 12, p. 1756, Nov. 2017, ISSN: 1996-1073. DOI: 10.3390/en10111756. [Online]. Available: http://www.mdpi.com/1996-1073/10/11/1756.

[50] N. Fumo, "A review on the basics of building energy estimation," *Renewable and Sustainable Energy Reviews*, vol. 31, pp. 53–60, Mar. 2014, ISSN: 13640321. DOI: 10.1016/j.rser.2013.11.040. [Online]. Available: http://dx.doi.org/10.1016/j.rser.2013.11.040.

[51] H.-x. Zhao and F. Magoulès, "A review on the prediction of building energy consumption," *Renewable and Sustainable Energy Reviews*, vol. 16, no. 6, pp. 3586–3592, Aug. 2012, ISSN: 13640321. DOI: 10.1016/j.rser.2012.02.049. [Online]. Available: http://dx.doi.org/10.1016/j.rser.2012.02.049.

[52] I. Farris, R. Girau, L. Militano, M. Nitti, L. Atzori, A. Iera, and G. Morabito, "Social Virtual Objects in the Edge Cloud," *IEEE Cloud Computing*, vol. 2, no. 6, pp. 20–28, Nov. 2015, ISSN: 2325-6095. DOI: 10.1109/MCC.2015.116. [Online]. Available: http://ieeexplore.ieee.org/document/7397053/.

[53] M. Odersky, "The Scala Experiment-Can We Provide Better Language Support for Component Systems?" *Conference Record of the Acm Symposium on Principles of Programming Languages*, vol. 33, p. 166, 2006. [Online]. Available: http://www.springerlink.com/index/NB7DVNKMMK4Q233D.pdf.

[54] E. Meijer and P. Drayton, "Static typing where possible, dynamic typing when needed: The end of the cold war between programming languages," *Revival of Dynamic Languages*, 2004.

[55] *Pecan Street Inc. Dataport*, 2019.

UNIVERSITAT RAMON LLULL

Esta Tesis Doctoral ha sido defendida el día ____ d________________ de 201__

En el Centro____

de la Universidad Ramon Llull, ante el Tribunal formado por los Doctores y Doctoras abajo firmantes, habiendo obtenido la calificación:

Presidente/a

____

Vocal

____

Vocal *

____

Vocal *

____

Secretario/a

____

Doctorando/a

(*): Sólo en el caso de tener un tribunal de 5 miembros