Toward organic ambient intelligences ? : EMMA

Clément Duhart

Normandie Université

THESE

Pour obtenir le diplôme de doctorat

Spécialité Informatique

Préparée au sein de l'Université du Havre

Toward Organic Ambient Intelligences ?
EMMA

Présentée et soutenue par

Clément DUHART

	Thèse soutenue publiquement le (date de soutenance) devant le jury composé de		
Mme Salima HASSAS	Professeur Université Claude Bernard Lyon 1	CNRS	Rapporteur
Mr Frederic WEIS	Maître de Conférences Université de Rennes 1 HDR	IRISA	Rapporteur
Mr Francis ROUSSEAUX	Professeur Université de Reims	IRCAM	Examinateur
Mr Joseph PARADISO	Professeur MediaLab	MIT	Examinateur
Mr Laurent GEORGE	Professeur ESIEE Paris	INRIA	Examinateur
Mr Cyrille BERTELLE	Professeur Université du Havre	LITIS	Directeur de thèse
Mr Jean-Marc LACROIX	Chercheur à Thalès Securité		Invité
Mr David MENGA	Chercheur à Électricité de France (EDF)		Invité

Cyrille BERTELLE, LITIS

UNIVERSITÉ LE HAVRE

SPMI ECOLE DOCTORALE

litis

PhD prepared at
LITIS – EA 4108
Laboratoire d'Informatique, du Traitement de l'Information et des Systèmes
Université du Havre
25 Rue Philippe Lebon,
76600 Le Havre, FRANCE

PhD in collaboration with
LACSC
Laboratoire d'Analyse et Contrôle des Systèmes Complexes
ECE Paris
37, Quai de Grenelle
Paris, 75725, FRANCE

PhD in collaboration with
MIT Medialab
Media Art and Sciences
Massachusetts Institute of Technology
75 Amherst St,
Cambridge, MA 02139, USA

Les pages se tournent et ne se ressemblent pas, mais quand celle-ci est la dernière, celle qui clôt un chapitre, on y respire à travers le temps passé. Certains instants dans la vie d'un doctorant peuvent ressembler à une traversée du désert où essoufflé, seul et désorienté, on oublie de lever les yeux pour voir la forêt qui nous entoure. Tout a commencé dans le sous-bois de l'ECE Paris, où deux chênes protecteurs, Brouaye et Rouyres ont formé l'étudiant devenu enseignant.

C'est alors que dans un coin ensoleillé, le discret marronnier Laurent George, par une coque tombée, m'indiquait la voie de la Recherche. Au loin se distinguait le cyprès Cyrille Bertelle qui me montrait l'entrée de cette forêt où j'allais pendant quatre ans cheminer. J'avançais en quête d'idées, quand David Menga, le plus haut des peupliers, m'orienta sur le sentier de l'internet des objets. Depuis la canopée, ce guide ne m'a jamais quitté.

Perdu dans mes pensées, il m'est souvent arrivé d'être rappelé à la réalité lorsque mon pas décidé croisait le destin d'une Pierre, tantôt Courbin tantôt Sauvage dans la mousse dissimulée. Quand certains jours la thèse était trop lourde à porter, j'ai pu échanger avec les Ents Faust, Worms et Fauberteau toujours enclins à m'accompagner. Plongé au cœur de cette forêt en perpétuelle régénération, j'y ai puisé des graines auprès des noisetiers Sylvain Leroy ou encore Cherrier. Ces dernières tout juste plantées ont permis aux jeunes arbustes Mardegan et Sonti de pousser.

C'est en préparant ma sortie de forêt que la liane Xiao Xiao m'a hissé vers cette ancienne contrée dont les mythes m'ont toujours hanté. J'y ai fait la rencontre de JoeP, le Baobab atypique, qui nourrit sa communauté au rythme incessant des promeneurs égarés.

Quand le vent a trop soufflé, j'ai toujours su où m'harnacher auprès des arbres Sensei Lecouvé, Zerhat et Taieb et de leurs protégés Khalifa, Carl et bien d'autres ...

À tous ces arbres.

À mes amis qui ont su être là.

À ma famille qui a su me faire pousser droit.

À ma douce fiancée qui a su ne pas être jalouse d'Emma.

Author's publication list

Peer Review Conferences and Journal

1. [DCB13] **Clement Duhart**, Michel Cotsaftis, and Cyrille Bertelle. "Lightweight Distributed Adaptive Algorithm for Voting Procedures by Using Network Average Consensus". English. In: *PRIMA 2013: Principles and Practice of Multi-Agent Systems*. Volume 8291. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, pages 421–428. ISBN: 978-3-642-44926-0. DOI: 10.1007/978-3-642-44927-7_30

2. [DCB14] **Clement Duhart**, Michel Cotsaftis, and Cyrille Bertelle. "Wireless Sensor Network Cloud Services: Towards a Partial Delegation". In: *Proceedings of 5th International Conference on Smart Communications in Network Technologies 2014 (IEEE SaCoNeT 2014)*. Vilanova i la Geltru, Spain, June 2014

3. [DB14] **Clement Duhart** and Cyrille Bertelle. "Methodology for Artificial Neural controllers on wireless sensor network". In: *IEEE Conference on Wireless Sensors (ICWiSE)*. 2014, pages 67–72. DOI: 10.1109/ICWISE.2014.7042663

4. [DB15] **Clement Duhart** and Cyrille Bertelle. "Toward Organic Computing Approach for Cybernetic Responsive Environment". In: *International Journal of Ambient Systems and Applications (IJASA)* 3.4 (2015). DOI: DOI:10.5121/ijasa.2015.3401

Submissions

1. [DSB] **Clement Duhart**, Pierre Sauvage, and Cyrille Bertelle. "A Resource Oriented Framework for Service Choreography over Wireless Sensor and Actor Networks". In: *Submission in International Journal of Wireless Information Networks (IJWI)* ()

2. [May+16] Brian Mayton, Gershon Dublon, Spencer Russell, Evan F. Lynch, Vasant Ramasubramanian, Donald Derek Haddad, **Clement Duhart**, Qiansheng Li, Glorianna Davenport, and Joseph A. Paradiso. "Deploying the Living Observatory: From Environmental Sensor Network to Networked Sensory Landscape". In: Submission in ACM. 2016

Contents

List of Figures

List of Tables

List of Algorithms

Listings

Part I

Introduction

Chapter 1

General introduction

L'homme est une corde tendue entre l'animal et la Surhomme une corde au-dessus d'une abîme.

Man is a rope stretched between the animal and the Superman–a rope over an abyss.

Friedrich Nietzsche [Nie17]

Contents

1.1 Internet of Things

The term *Internet of Things (IoT)* is used in very different ways. Its common sense nature refers to the set of physical devices connected to the Internet. They are mainly composed of the Responsive Environments (RE) and wearable devices. On the one hand, the Internet provides its services within physical environments in which people are more and more connected. On the other hand, the Internet has become a huge database of knowledge for human being in which Artificial Intelligence (AI) already has an important place. The underlying idea is that everything will be connected to Internet, including people, things and environments, in order to help people in their daily tasks, while macro issues, such as resource sharing, should be self-managed by intelligent systems in the general interests of society.

1.1.1 Business Opportunities

1.1.1.1 Economy, Society and Technologies

There are a lot of possible uses, such as environmental preservation, residential protection, assistance for elderly people and augmented reality. Major technological firms predict an explosion in the development of the IoT, such as Cisco's estimation that 50 billion objects will be connected by 2020 [1]. Several social aspects will be affected by the incorporation of the Internet into daily environments. Moreover, this must be discreetly managed in order to facilitate its acceptance by people before intelligent systems are able to provide some kind of education in response to irrational human behaviour. For example, on a human scale, people are not conscientious in relation to their daily actions within the global environment. The promises are significant, however, while there are a lot of technological and theoretical issues still to be solved. For example, the huge number of connected objects in the future is already forcing engineers to consider new designs in order for global energy consumption to be sustainable. This constraint has a large impact on the future connectivity between the IoT and the Internet core. Therefore, new paradigms for future applications and services using the Internet must be investigated to prepare for the arrival of the cybernetic world in the future.

1.1.1.2 Many Visions Converging

This technology will affect all society domains, such as private residences, transportation, cities and industries. Therefore, a lot of different visions will appear, reflecting different interests. Aggarwal et al. [AAS13] summarize the classification into three major groups. The *Things-Oriented Vision* assumes that connected objects in the future will be intrinsically heterogeneous because of the important difference in their application requirements. Meanwhile, the *Internet-Oriented Vision* supposes that this heterogeneity must be abstracted by a communication layer, such as the Internet Protocol (IP). Finally, the *Semantic-Oriented Vision* is interested in the management of the heterogeneity at a data level without any consideration for the network specificities. During the last decade, new theoretical and technical frameworks have been investigated to provide new materials relating to these different visions. Nowadays, it is necessary to determine how they can be integrated into the same framework.

1.1.1.3 Current and Future Applications

Historically, the development of this technology has been promoted by the monitoring of structural health. Kim et al. [Kim+06] present a complete architecture

[1] Evans, D. The Internet of Things: How the Next Evolution of the Internet Is Changing Everything. Cisco Internet Business Solutions Group, IBSG (2011)

to monitor the vibrations of the Golden Gate Bridge. This kind of application requires numerous sensors that are spatially distributed in the region of interest. They use wireless networks, which are less expensive and much easier to deploy. Another application concerns the preservation of heritage buildings by monitoring their evolution over time [Cer+09]. Hence, the main requirements involve the collection of time-lined data and the definition of easy deployment solutions in large spaces at low-cost. The development of smart grids represents an important challenge in terms of optimizing energy consumption around the world. The increase in world energy consumption has caused a lot of problems regarding its management in large-scale networks [YLD07]. The storage of significant amounts of energy is difficult, even if hydraulic dams temporarily address this problem. Hence, energy production should be adapted in real-time so that consumption is facilitated through the use of renewable energies. Energy providers require detailed information about local energy consumption and production through smart grids. In addition, some regulation strategies, such as preventing energy drops, require some home appliances to be turned off remotely to avert a peak in energy consumption. Finally, energy providers need real-time information to regulate energy used in multi-scale architecture at a national level as well as home appliances. This technology has been progressively applied in human environments, such as hospitals for healthcare applications [Ko+10]. The traceability of drugs, patients and personnel is a permanent issue in hospitals with possible dramatic consequences. Hence, monitoring and traceability are improved by collecting real-time information to facilitate medical coordination by central information systems. This application uses sensitive data, which must be protected to maintain patient and staff privacy. Moreover, this application is itself sensitive regarding the impact upon patient health in cases of malfunctions. Therefore, the system must be safe and secure in order to protect human integrity.

Nowadays, daily environments are being studied in order to improve comfort, safety and accessibility. The first type of environment concerns houses and other buildings where people spend most of their time. Moreover, these buildings consume a lot of energy. These places require large-scale infrastructures, which must be secure, energy efficient and evolutionary. Some domotic systems are available to address these different issues. Traditionally, a central computer manages a domotic system. Most of the time, these systems are not designed to be interoperable with other ones. Hence, the systems are limited in their evolutionary possibilities; moreover, they are dependent on manufacturers, who can stop their production and maintenance at any time. But, these systems must be deployed and maintained during the entire lifetime of the building. Therefore, applying the IoT in relation to important infrastructures requires heterogeneity to be supported at the hardware and application levels in order to ensure the sustainability of the system. The definition of standards for IoT technologies and protocols is a primary requirement.

1.1.2 Technology Challenges

Gubbi et al. [Gub+13] present a vision, different architectural elements and future directions for the IoT. The architecture requires a lot of different technologies, from the design of devices and the network to the implementation of different services. Moreover, the emergence of new software paradigms, such as ubiquitous computing, is profoundly changing the vision of the Internet.

1.1.2.1 A Meeting of Engineering

The IoT is the meeting of different engineering areas. The design of connected objects by manufacturers requires experts in electronic and embedded systems. The establishment of a common network must be performed by security and telecommunication experts, while service applications for end users should be designed by software engineers. Although these domains seem naturally compartmentalized, they are strongly connected; for example, they must be embedded in a single device. Hence, an important technology challenge is the definition of abstraction layers to limit the requirement of multiple expertise. Without easy solutions for designing devices, networks and applications, IoT would not exist.

1.1.2.2 Ubiquitous Computing

Ubiquitous Computing (UC), which appeared at the same time as the development of the IoT, is a concept within software engineering relating to pervasive services. Such software has the capacity to communicate with users and other software applications through any interface and in any place with any format. The *leitmotif* of UC is *everywhere and anywhere*. The services follow the users in their daily activities in order to assist and provide them with information. Hence, the software design evolves towards high-level languages in order to be not only accessible through the Internet but also proactive with regard to multiple interfaces: wearable, RE and others.

1.1.2.3 Towards Web 3.0

The extension of the Internet into the physical environment will drastically increase the number of networks around the world known as the *Capillary Internet*. As this huge network will collect data from appliances around the world, this raises the issue of data access. In previous decades, a similar discussion about the browsing data content concluded that the Web is very adaptable in terms of linking data together. Its concept of hypertext associates a new remote resource to a word contained in the text. This simple feature has continued with the development of Web 2.0 which adds the concept of semantic links. Nowadays, a word can refer to a set of remote resources, while browsing is dependent on the semantic context of interests, such as the Wikipedia platform. Web 3.0 could provide the features needed to link the digital and physical worlds through the concept of cross-reality.

1.2 Motivations of the Thesis

The growing interest in the IoT is promising in terms of improving the protection of people and the environment. It should help to improve accessibility, security and comfort in the daily environment, while saving energy and other consumable resources. The term itself is new, but the referent concepts exist in science fiction from several years ago. Nowadays, the technologies exist, but:

What kind of Ambient Intelligence (AmI) is required and how can it be done?

The motivations of the thesis are based on the observation that several theoretical and technological activities are converging within autonomous systems. On the one hand, AI techniques are incorporated in relation to Multi Agent System (MAS) in order to design role-based agents for adaptive systems. On the other hand, the Embedded System (ES) has become more and more powerful to the extent that the entire Operating System (OS) can be hosted in a single chip. To this extent, this thesis focuses on the study of their combination in the direction of the distributed AmI. The bio-inspired proposal considers systems, such as an Artificial Organism (AO), which are composed of the different appliances in a managed environment. This approach focuses on interaction mechanisms in order to model system intelligence instead of a programming approach. The main assumption is that the programming approach cannot support evolving features in the same way as a biological system. It depends of an initial coding style to validate the operations at a compilation step; meanwhile, an interaction model evaluates the feasibility of an operation during the process runtime, as is the case in chemical systems, in which evolving reactions are produced according to molecular interactions. More exactly, the investigations focus on the AmI design, such as a cyberbrain composed of several interacting Artificial Neural Network (ANN). Each of them corresponds to a particular cognitive function, including context learning, controlling appliances and cognitive synchronization. This autonomous cyberbrain is composed of distributed neuron groups across different appliances in order to minimize the network communications.

The EMMA framework has not only been developed for the purpose of cyberbrain design. It offers a complete solution for the IoT infrastructure to appliance manufacturers, network administrators and service providers. A set of interfaces and tools independently facilitates the development of different system parts in order to integrate them easily into an existing IoT infrastructure. The differently used technologies are standard formats of the Internet Engineering Task Force (IETF) and the Institute of Electrical and Electronics Engineers (IEEE), allowing the EMMA system to be interconnected with Internet services and other frameworks at the appliance, network and application layer level. In addition, it is assumed that the user privacy and data security can be only preserved within fully distributed architectures in which data stays located in the managed environment.

1.3 Contributions and Content

- Part I introduces the concept of the Internet of Things (IoT) and Ambient Intelligence (AmI) in Chapter 1. Chapter 2 provides more details of the different aspects of WSANs, AmIs and Organic Computings (OCs).
- Part II presents a framework contribution entitled the EMMA framework, with an introduction in Chapter 3 on the technologies used for WSANs and discussions about the impact made by centralized versus decentralized software architecture. Chapter 4 details the developed middleware based on Resource Oriented Architecture (ROA) for the choreography of Web services. Each appliance has its hard-coded services within containers, which publish data through resources. Based on an interaction model, mobile agents allow the data flows to be routed between the different appliance services in order to build Service Choreography (SC). A graphic model, based on a Petri network, allows the system to be analyzed and composed in order to design distributed AmI. Chapter 5 presents the deployment methodology, which is bio-inspired by the DNA-RNA process. A set of mobile agents forms Dynamic Network Agent (DNA) in order to deploy an Residual Network Agent (RNA) graph, according to its reactions with appliance resources.
- Part III presents two algorithms for AmI with theoretical studies and practical implementations over the distributed EMMA middleware. Chapter 6 presents an Artificial Neural Controller (ANC) to automatically drive the appliances based on empirical learning and logical rule-based descriptions. The knowledge transfer between the logical space of rules and the statistical space of data allows experts to understand, at a global level, the system operations that are encoded in Artificial Neural Network (ANN). Chapter 7 presents a Voting Procedures (VP), which synchronizes decisions between different AmI components. This VP is fully distributed and fault-tolerant regarding time delays, switching topology and packet loss.
- Part IV presents the contribution made to the Tidmarsh Project at the MIT Media Lab. An ambient sound recognition engine entitled TidZam has been designed in order to recognize animal calls from outdoor microphones. Based on deep learning technology, the system acts as event detector to locate animals on a three-dimensional virtual environment. An additional application involving speaker recognition is also presented.
- Part V brings the current state of work contributions, with regards to the EMMA organic framework and the first implementations of components for future neural-based Ambient Intelligence (AmI), to a close. Several perspectives are discussed in relation to general discussions about the IoT, the OC and the AmI, while some framework improvements are proposed in terms of their current limitations.

Chapter 2

State of the Art

Se méfier des penseurs dont l'esprit ne fonctionne qu'á partir d'une citation.

Beware of thinkers whose minds function only when they are fueled by a quotation.

Emil Michel Cioran [Cio95]

Contents

2.1 Introduction

The state-of-the-art technology discussed in this chapter encompasses the different research domains relating to Wireless Sensor and Actor Networks (WSANs), Ambient Intelligences (AmIs) and Organic Computings (OCs). This chapter's main framework is organized by a set of papers, which are briefly summarized by a single paragraph, in order to present the Internet of Things (IoT) in a bottom-up approach from hardware to intelligence systems. The papers have been selected according to their impact in the relevant research community, their originality or their relevance to the main concepts presented in this thesis.

The first two sections focus on parallel works between a WSAN and an AmI. The first section looks at the hardware, network and programming paradigm pertinent to the construction of the IoT, whereas the second section considers the information system needed to control it intelligently. The last section presents the new paradigm of OC in order to discuss its application to the IoT. The main framework is focused on the use of distributed AmI techniques to address the challenge posed by the requirements in WSANs.

2.2 Wireless Sensor and Actor Networks

Wireless Sensor and Actor Network (WSAN) is a technological term referring to a group of sensors and actuators connected by a wireless medium. Sensors collect data from the physical world and the actuators operationalize actions within an environment. The control loop is composed of sensing, controlling and acting.

[Mio+12] *Miorandi et al.* present current and future applications of WSANs, such as in smart homes/smart cities, environmental monitoring, healthcare, smart business/inventories and security/surveillance with current enabling technologies. The authors propose a taxonomy of WSAN technology at the intersection of the embedded systems, the distributed system and the distributed intelligence. WSANs have features and issues in different technological domains. New engineering paradigms must be invented to uncouple problems associated with electronic design, networking and information system development.

[AIM10] *Atzori et al.* extend the discussion around the concept of middleware. It is a piece of software inside the connected object to create a bridge between the hardware, the network and the applications. The object and network functionalities are encapsulated inside service containers in order to facilitate application design. Middleware is responsible for providing mechanisms for service composition and its management, as well as object abstractions and system sustainability (trust, privacy, integrity and security).

[AK04] *Akyildiz et al.* has identified several problems in the information system in relation to the coordination of actuators according to sensing events. Authors have focused on distributed models in which actor-actor communications are performed. In some situations, the action plans are dependent on successive events in different sensors, whose execution can require several time-lined actions performed by different actors. The information should be transmitted to all concerned actors and sensors to allow them to evaluate the situation. They have to communicate together in order to synchronize their decisions. Authors emphasize the issues around such distributed coordination to ensure the consistency of the respective action plan.

The next section presents the features and constraints of connected objects in order to understand network requirements. Connected object, as a term, refers to an autonomous device that communicates with other ones in order to manage an environment. The literature on networking focuses on the major technologies enabling the extension of the Internet inside the physical world. The last section presents some of the different approaches to software architecture and application protocols, which will need to coexist in the future Internet of Things (IoT).

2.2.1 Connected Objects

A connected object is composed of a wireless communication module, a computation unit and an electronic application layer of sensors or actuators. They are spatially distributed in the environment to form a large-scale network. Given their significant prevalence, they must be energy efficient and remotely maintainable in order to ensure a long lifetime for the system.

2.2.1.1 Identification, Sensing and Acting

[Mio+12] *Miorandi et al.* identify four main roles for connected objects:

- *Acting*: these actively change the environment and inform the system about manual controls performed by people
- *Sensing*: these gather information from the environment and people's behavior in order to synchronize their representation in the system
- *Information*: these inform the user about current system parameters and the state of the managed environment
- *Identification*: these are used to identify an unconnected object or people using Radio Frequency IDentification (RFID) technology

2.2.1.2 Energy, Computation and Communication

[SM11] *Soua et al.* discuss techniques used to save energy in a WSAN. Most applications use a large number of nodes, which are spatially distributed and function with a small battery. Given their number leads, it can be very difficult to change their battery. Therefore, energy harvesting techniques are used to reload batteries, while software has been designed to decrease energy consumption. Energy is mainly consumed by the transceiver radio during data transmissions. Hence, numerous proposals have focused on the network stack in terms of reducing the protocol overhead and improving packet routing, as well as proposing protocols for the avoidance of communication collisions or highlighting the ongoing development of mechanisms relating to duty cycles for sleeping modes.

[KEW02] *Krishnamachari et al.* study the impact of application layers on energy consumption. They evaluate global energy-saving using data aggregation. Several producers of data are randomly distributed over a WSAN using different models. Their data are aggregated before reaching the data gathering stage. The conclusions show that an energy gain is possible when data producers are clustered. Hence, data-centric architecture is another leverage to save energy.

2.2.1.3 Lifetime, Design and Maintenance

[RDT07] *Rubio et al.* discuss programming approaches for connected objects. Most WSANs are only deployed once and are intended to operate for a long time. Their initial programming would require update mechanisms for maintenance or functionality extension. Therefore, the design of their software must include reprogramming features. Moreover, their hardware must support remote deployment in term of networking and memory capacity because they are distributed in a large-scale network. The approaches made by components should be preferable in order to partially update the software. The network bandwidth is used less than for entire node reprogramming during data transmission and consumes less memory for the update storage.

[Bel+05] *Bellis et al.* proposes an implementation of a wireless sensor module based on Field-Programmable Gate Array (FPGA) technology. Instead of executing the program by a Central Processing Unit (CPU), it is hard-coded at the physical layer. The program is event-driven by its input pins, which allows the system to sleep most of the time and, therefore, save energy. Their research team is interested in the miniaturization of connected objects at the scale of a 10 mm cube. They have successfully built a 25 mm platform, which is compared with other manufacturer platforms, such as those shown in Table 2.1. It appears that this technology significantly decreases energy consumption and platform size, while increasing the data rate of a radio transceiver. This is promising for the next generation of miniaturized connected objects.

The software technology for connected objects is still not defined. The remote reprogramming is essential to maintain a long lifetime for the system. The implementation technology is evolving with the aim of reducing the size of connected objects and saving energy.

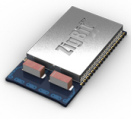
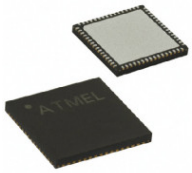
				
Reference	TelosB Mote	Raven Platform	ZigBit Module	Atmega128rfa1 SoC
CPU	8 MHz	8 MHz	8 MHz	16 MHz
RAM	10 Kb	16 Kb	8 Kb	16 Kb
Flash	48 kB	128 Kbytes	128 Kbytes	128 Kbytes
AES	No	No	No	Yes

Table 2.1 – Examples of platform miniaturization progress for WSAN

2.2.2 IP-Based Network

[VD10] *Vasseur et al.* describe an overview of WSAN developments in the last decade. At the beginning, academics and industries were developing their own protocol solutions. This exploration phase was helpful for the identification of usages, constraints and approaches. Therefore, the future IoT will be composed of different technologies, networks and applications. A common stack of network communications has to be determined, with Internet Protocol version 6 (IPv6) as the most likely candidate.

[RN10] *Rodrigues et al.* present a survey of IP-based solutions for WSANs. Initially, finding a solution was considered inappropriate according to the complexity of the TCP/IP protocol and its overhead. However, the use of a proxy to translate the protocol into a heterogeneous infrastructure raises more problems regarding the single point of failure, network congestion, maintenance and scalability. Therefore, the research community has already started work on adapting IP technology for WSANs. After several implementations of IP solutions, a new standard has appeared: 6LoWPAN [SB11]. Nowadays, intensive efforts are being undertaken focused on using or adapting standard protocols in line with the technological development of the Internet.

[Hog+12] *Höglund et al.* present an example of a 6LoWPAN application for the monitoring of energy quality in the supplier chain. Their solution monitors the amount of energy produced by the supplier and consumed by customers. Several parameters are collected in the process, such as frequency, voltage and power. Data are sent periodically to the energy management system through a 6LoWPAN mesh network. The network is auto-established with IP connectivity, with the communications performed over the Internet without an intermediate proxy. This application validates the use of such technology in industrial applications.

2.2.2.1 Low Power ZigBee Technologies

[Lay13] *IEEE 802.15.4 norm* is the major radio frequency technology used in relation to 6LoWPAN. It is a standard defined for low power wireless technology. All its features have been designed to save energy during listening and transmission. It consumes 1,600 % less energy with a cover range of 25% compared to Wi-Fi. This norm defines the radio frequency and the Media Access Control (MAC) layer in an OSI model when building a scalable mesh network. Nowadays, numerous products use this norm, such as ZWave, Sensinode and ThingSquare. The solutions used in their software vary according to the use of the 6LoWPAN standard or a proprietary protocol.

2.2.2.2 Network, Routing and Security

[Dun03] *Dunkel* proposed, in 2003, an innovative implementation of 6LoWPAN over IEEE 802.15.4, known as uIP. As well as being very efficient in terms of memory usage, its event-driven kernel, the Contiki OS, is very reactive with a low energy consumption. Many industrialists and researchers are still reliant on it to evaluate new algorithms, applications or platforms.

[JZS12] *Jara et al.* proposed, in 2012, another IPv6-based protocol that does not follow the 6LoWPAN standard, known as the Glowbal IP. It was implemented using the Contiki OS to replace the uIP network stack. Their motivations concerned overhead protocol reduction and easy curve learning to build WSANs and their applications. As it is not standardized, it requires a proxy to translate network exchanges between traditional networks and the Glowbal WSAN.

[RN10] *Rodrigues et al.* conduct a survey of previously published solutions, 6LoWPAN implementations and operating systems for WSANs. The authors conclude that the two main approaches, namely, sensor stack-based and proxy-based architecture, must be merged in future WSANs. They expect that nodes will have auto-configuration mechanisms that need to be partially autonomous. Meanwhile, network functionalities, such as routing and security, must be under the responsibility of a proxy-based gateway to save program memory for the node applications.

[YF04] *Younis et al.* propose an energy efficient routing protocol for WSANs. They propose a cluster-based approach called the HEED protocol. Particular nodes are elected to route network traffic according to the residual energy of their battery. They demonstrate the efficiency of their hierarchical routing network and open a discussion about a trade-off between energy savings, accuracy and latency.

[Gun+07] *Gungo et al.* investigate the possibility of defining a routing protocol that considers mixed metrics. Routing tables are dynamically established according to the link quality between nodes and their residual energy. This proposal validates the possibility of simultaneously addressing the objectives of the connected objects and those of networks using metric aggregation.

[SSS+10] *Singh et al.* present a survey on main routing protocols according to their approach and applications, such as mobile networks, hierarchical protocols, Quality of Service (QoS) protocols and multi-path protocols. They conclude that the routing protocol is inherently dependable of network topology, application and routing objectives (energy-saving, bandwidth optimization etc.).

[TED10] *Tsiftes et al.* present an implementation on the Contiki OS of the IETF candidate for a routing protocol in 6LoWPAN. The Routing Protocol for Low power and Lossy Networks (RPL) protocol has been designed to provide a flexible protocol that addresses, as much as possible, common use cases. It uses mixed metrics, customizable at the application level, to define routing path orientation over WSANs. The results of their implementation show several years of network lifetime on Tmote Sky motes. It should be noted that the Contiki OS has been implemented to easily change any layer of the uIP network stack, including the routing protocol.

[BBB08] *Bojkovic et al.*, [PLH06] *Pathan et al.*, [CY05] *Camtepe et al.* and [BN08] *Boyle et al.* discuss the major security issues relating to WSANs. The physical protection of systems is an intractable problem because communications are wireless and spatially distributed in an open-system environment. Therefore, the security aspects that are addressed by these authors concern communication protection through encrypted communications. As most attacks are based on the insertion of false information, communications must be encrypted and an intruder detection policy should be defined. Several approaches are introduced concerning the scale of the key cryptography distribution scheme: network-keying, group- or node-keying, or pair-wise keying (between each node). All of these schemes can be applied to different network stack layers: data link, network or application layers. According to these security schemes, the design of a secured WSAN depends on the node capacities and on the addressed threats.

[KRM13] *Krentz et al.* present a security framework for 6LoWPAN at the link layer. Their pair-wise key scheme, using Advanced Encryption Standard (AES), is energy efficient and has a low memory footprint, in the same way that most platforms for WSANs have a dedicated AES chip. Their proposal focuses on the security protocol to protect the network against Denial of Service (DoS) attacks. As the link layer protection requires all nodes to have the same security protocol, a heterogeneous network with secured and unsecured nodes cannot coexist.

[Uki+13] *Ukil et al.* propose another pair-wise key scheme using AES for 6LoWPAN encryption, but one that is applied to the application layer. The authors propose a secured Constrained Application Protocol (COAP) with double authentication. It should be noted that the COAP is a candidate for the application layer according to the IETF, which increases the interest in this proposal. It is energy efficient with a low memory footprint and allows the WSAN to be composed of heterogeneous nodes. Network establishment is performed identically by secured and unsecured nodes. However, the proposed protocol cannot protect against DoS attacks.

2.2.2.3 Heterogeneity and Internet Integration

[Del+06] *Delicato et al.* are interested in the architecture of WSAN applications. They warm about the risk of designing rigid WSANs in which applications are strongly coupled with hardware and the network. They insist that future usages of a WSAN cannot be predicted, as new devices and services can be added during its runtime. Moreover, in the case of failures, services must be partially maintained. This constraint implies self-adaptivity of the system. They proposes the use of Web technologies at the application level of nodes in order to introduce application flexibility. Instead of designing hard-linked applications for nodes, Web services should be used on nodes to interface and share services. The data are exchanged from one Web service to another, which facilitates the management of application heterogeneity. This proposal is consistent with IP-based proposals for the network stack, while extending the scope of earlier ideas through the use of ontologies to homogenize data semantics between the nodes and other services on the Internet.

2.2.3 Software Architectures

[RDT07] *Rubio et al.* resume previously presented requirements about software architecture and programming approaches for WSANs. They identify that nodes and the network must have auto-configuration and security mechanisms available for large-scale networks. The node software must be lightweight with reprogramming mechanisms to ensure its maintenance and evolution. WSAN applications must be coordinated and executed in real time to manage the environment. Finally, software architecture has to support application heterogeneity and a huge amount of data. The authors present several programming paradigms along with their advantages and drawbacks. There are two main kinds of architecture: data centralization and distributed applications. Both of them have to respect previously presented requirements to be practical in WSANs. Authors classify programming approaches into three categories. The *node-centric* approach consists of independently implementing each connected object behavior. This approach is very efficient in terms of program optimization and analysis. Meanwhile, the macro behavior of a WSAN is difficult to anticipate. Conversely, *macro-programming* determines the set of tasks assigned to each node. The data exchanges involved are fully determined to manage WSAN coordination. Although the system consistency is easily maintainable, the system overhead is important and the WSAN has to be homogeneous. The third concerns the use of a *middleware*, which is a trade-off between the node-centric and macro-programming approaches. The middleware provides the information and the functionalities of a WSAN to each connected object, such that they can adapt their behavior and synchronize information according to the WSAN state.

2.2.3.1 Data Formatting and Application Protocols

The establishment of an homogeneous connectivity between WSANs and the Internet has implied the necessity to define a standard for a network stack for communications inside a WSAN. The data, their formatting and their semantic, however, are strongly dependent on the applications. To this extent, the following papers discuss the management of these heterogeneities in relation to the current investigation about the application protocol for 6LoWPAN.

[JMS05] *Jammes et al.* propose service-oriented architecture for WSANs. Each node provides a Web service interface using Devices Profile for Web Service (DPWS). This protocol is a technology used on the Internet to interconnect heterogeneous Web services in order to form a distributed application. It manages data formatting and its meta-description with Extensible Markup Language (XML), its transport through Simple Object Access Protocol (SOAP) and the security layer. However, this protocol is too heavy for tiny target platforms.

[DH+10] *Dawson et al.* propose an Simple Measurement and Actuation Profile (sMAP) to describe and format data. It is based on JavaScript Object Notation (JSON) formatting language, which is lighter than XML. sMAP is a data profile that defines a set of attributes and objects to exchange data in a WSAN. The main idea is to uncouple the data format and its semantics from the data transfer protocol. Hence, according to the application, and independently of the data, the application protocol can be selected.

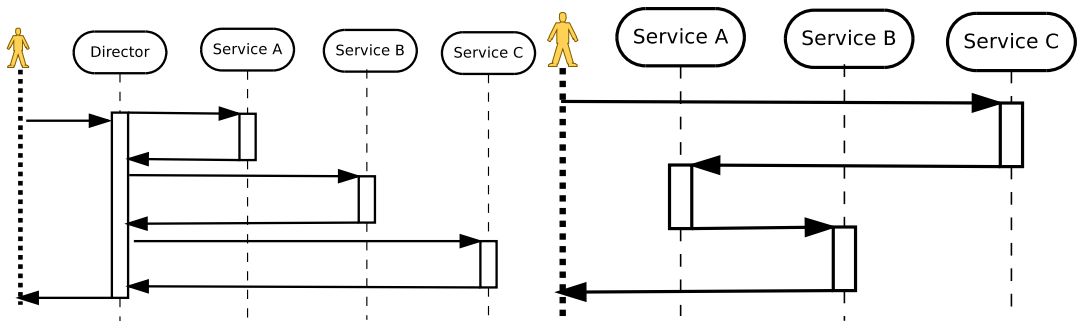
[KDD11] *Kovatsch et al.* present their implementation of a COAP protocol on the Contiki OS. It is the application protocol candidate of IETF for 6LoWPAN. Similar to the Hyper Text Transfer Protocol (HTTP), it is used in REpresentational State Transfer (RESTFUL) architecture to facilitate Web service management. Each set of data is encapsulated in a resource with its own representation profile. Hence, traditional HTTP requests retrieve data or manage them directly on the node. Additional mechanisms are added in COAP, such as data block fragmentation or a publish-subscribe engine to facilitate Web service composition.

[HJ08] *Huang et al.* present their designed architecture for managing data heterogeneity using ontologies. All data are collected on a gateway to be stored in an Resource Description Framework (RDF) semantic database. Therefore, all data can be managed similarly and independently of their formatting and meta-description language.

2.2.3.2 Web Service Orchestration and Choreography

[KLD12] *Kovatsch et al.* present their gateway platform to manage a WSAN known as Actinium. Each object has a proxy object inside the framework, which is synchronized with the real connected object through the WSAN. In this orchestration architecture, each object communicates only with the gateway, as illustrated in Figure 2.1.1. This kind of architecture has significant limitations regarding network congestion on routers around the gateway in large-scale networks. However, this proposal is very innovative due to its use of mobile and scripted applications. Instead of developing hard-coded applications executed on the gateway, they use Node.JS scripts to build applications. They are executed in an independent container in order to manage applications through resource utilization. The authors consider applications, such as any data, which are managed in the same way as any connected object providing a service.

[Che+13; Che+11] *Cherrier et al.* have worked on choreography architecture in which services of connected objects communicate directly together, as illustrated in Figure 2.1.2. The gateway configures node services to send requests directly to the concerned service. When an event occurs on a sensor, the node processes a small treatment locally to send a request to another mote. Successive requests are sent over the WSAN to execute operations on actuators or treatments on other nodes. This event diffusion over the network is stopped when the system reaches a static state. The authors demonstrate the efficiency of the choreography approaches in comparison with orchestration for large-scale networks. However, the stability of such system is an important challenge, while applications are limited by node capacities to perform complex operations. In this sense, authors propose a logic model to validate distributed application consistency, known as DL-Lite, and an associated protocol called SALT.



2.1.1: Orchestration architecture 2.1.2: Choreography architecture

Figure 2.1 – Service-oriented architecture: orchestration and choreography

2.2.3.3 Object Abstraction and Middleware

[Ban+11] *Bandyopadhyay et al.* present the main reasons for the use of middleware in IoT applications. Firstly, it is necessary to abstract hardware and networking regarding the huge number of diverse devices and application domains. It is conducted in a uniform manner to manage the heterogeneity of data and software components using an adaptation layer. They provide an Application Programming Interface (API) to hide the particularities of the connected objects. Hence, they facilitate application implementation.

[MA06] *Molla et al.* compare six examples of middleware to conclude that there is significant trade-off between the challenge of resource constraint and middleware scalability and adaptability. Moreover, they emphasize that, most of the time, QoS, security and context awareness are either partially or not at all supported by middleware.

Hadim et al. [HM06], Rubio et al. [RDT07], Wang et al. [Wan+08] and Rahman et al. [Rah06] propose a middleware classification according to the programming approach taken:

- **Application-driven middleware** provides the applications with access to network and system configurations.

  [Dun03; Dun+06] *Dunekls et al.* have developed the Contiki OS for WSANs. It includes a uIP 6LoWPAN stack, which is manageable by the applications, and a mechanism for remote installations as well as updates of applications over wireless communications.

  Advantage: Program optimization.
  Drawback: No control on global WSAN behavior.

- **Message-oriented middleware** is based on a publish-subscribe mechanism to provide an abstraction layer between producers and consumers of data. It facilitates asynchronous communications over a WSAN.

  [Sou+04] *Souto et al.* have developed Mires middleware based on the Tiny OS. The node applications subscribe to topics that are filled with sensor data by the operating system. The authors highlight the potential energy gain by only transmitting messages that refer to subscribed topics.

  Advantage: Easy design of applications.
  Drawback: Low network scalability.

- **Database middleware** provides an abstraction in order to consider WSANs, such as a distributed database. Queries are diffused over the network and the responses are aggregated in order to generate the final one on the sink.

[Mad+05] *Madden et al.* have developed TinyDB middleware based on the Tiny OS. The authors provide an advanced SQL-like language to request data over the WSAN. It allows requests to create local storage, aggregate temporal data, and trigger data collection and others mechanisms. They use a duty cycle mechanism to save energy during data aggregation and request resolution.

Advantage: WSAN abstraction.
Drawback: Only for data collection.

- **Service-oriented middleware** facilitates service choreography in providing features of service discovery and binding.

  [Kus+07] *Kushwaha et al.* have developed an OASIS framework, which is composed of middleware and a set of tools to design and deploy choreography over a WSAN. The mapping of communication flows and tasks are performed offline on dedicated platforms in order to be deployed on the connected objects when online .

  Advantage:: Management of WSAN heterogeneity
  Drawback: Requires offline supervisors.

- **Tuple space middleware** is based on the concept of associative memories. The applications exchange data through a distributed and shared memory space, which is fragmented over the WSAN.

  [Cos+07] *Costa et al.* have developed TeenyLIME middleware based on the Tiny OS. Applications are fully uncoupled from connected object functionalities using a tuple space. The authors explain that this model is very efficient for one-to-many and many-to-many communications thanks to the use of common tuples. Moreover, the concurrent access of functionality by several applications is easily manageable by mutex and semaphors.

  Advantage:: Global system management.
  Drawback: Network bandwidth consumption.

- **Virtual machine middleware** interprets the applications through a runtime engine. They are implemented in separate and small modules with a high level language in order to be distributed over the WSAN.

  [Kwo+06] *Kwon et al.* have developed ActorNet based on the Tiny OS. Applications are designed similarly to agents, which have the ability to migrate to other connected objects. This feature offers new techniques for application design. Instead of collecting data to be moved to the applications, they are moved to the data.
  Advantage: Advance programming.
  Drawback: Virtual machine overhead.

2.3 Ambient Intelligence

[ANA10] *Augusto et al.* reflect on the historical development of Ambient Intelligence (AmI) since the advent of technology: at the beginning of the previous century, computers occupied an entire room, whereas, nowadays, they can be camouflaged in the human environment. The term AmI was introduced at the end of the previous century to name the system that manages an environment through a physical infrastructure, which has henceforth been called Smart-X. This multidisciplinary area and the huge number of possible applications have resulted in numerous visions. The system design is oriented differently according to the environment, the connected object, the network, human-computer interfaces or artificial intelligence. In all visions, the system has to adapt its behavior according to the evolution of the managed environment and the applications. However, calculating the societal implications is still at the experimental stage, in which its impact on human behavior in a world of hidden and permanent technological assistance is being evaluated:

- [Aug09] *Augusto et al.* present a survey on smart classroom experimentations. Several uses have been developed with different kinds of collected data. The main idea is to increase interactivity between students and teachers through digital supports, such as tablets and Web interfaces. The authors argue that the post-analysis and the online capture of student behavior and annotations can improve course content and teacher pedagogy. Moreover, AmI can help to adapt the course online to the students through the use of alternative pedagogical material and e-learning. Finally, not enough experimentations have been conducted to enable conclusions about the AmI framework for education, although it looks very promising for the future.
- [PBG07] *Paganelli et al.* present a context-aware e-tourism application. Additional information is provided to visitors according to their location, profile, and their preferences and those of other tourists. When a tourist arrives at a destination, the system evaluates who else is in the same place and encourages the exchange of their opinions through a chat messaging. Context awareness is a feature allowing a system to adapt its decisions according to a multi-modal and sequential analysis of a situation. The system does not make a decision according to the local states of variables, but according to their correlation in a global scheme. Hence, the system is managed globally, which guarantee its consistency. The authors use a combined object-oriented data model to collect and manage information with an ontology-based model to implement a reasoning engine. Their e-tourism application facilitates interactions between visitors and highlights the best places at the respective destination according to their recommendations.

- [AE10] *Alemdar et al.* present a survey on AmI requirements, architecture and technologies for healthcare applications. They identify three scales of infrastructure. The body area network is composed of sensors located on patients, nurses and doctors, as well as on pills, in order to improve care traceability. The personal area network refers to the set of sensors to manage the residential environment, such as video cameras, bed tracking and environment monitoring (brightness, pressure, temperature), and provide rich contextual information to the AmI applications. Meanwhile, the wide area network is the communication backbone needed to manage the system remotely through human-computer interfaces over the Internet. Security issues and enabling technologies are listed in different studies. The conclusion emphasizes the necessity to develop multi-modal sensor systems to address the challenge of context-aware and pervasive applications in healthcare, while also considering security and privacy issues.
- [RM13] *Rashidi et al.* present a survey on AmI challenges and techniques for assisting older adults. People are increasing living longer, which in turn affects their mobility and autonomy. While robotics has offered promise in recent decades, its usage with older people is difficult because robots are not accepted by this population. As such, a smart environment seems to be more practicable because intelligence and operations are camouflaged within the environment for assisted people. The authors refer to enabling technology and its usage, but also the main issues surrounding this topic. The identification of human activity is the principal challenge. Context-aware features allow a system to consider itself in its global state, whereas human activity identification tends to predict what humans need next. However, older people can be inconsistent in terms of their actions according to their current needs because of their gradual physical and cognitive decline. Hence, a system must have enough knowledge not to be influenced by such behavior in the course of providing assistance.

[Sad11] *Sadri* presents a comprehensive survey on AmI. The development of such technology has become urgent regarding the aging population. It can help older people with independent living, as well as other handicapped people, such as those who are blind. The whole of society may possibly benefit from applications in spaces for the elderly, healthcare, business, commerce, leisure and smart homes. The author lists an important number of projects implemented in real experimentations using different kinds of technology. In doing so, she identifies two main approaches, one of which is based on technological automation for elementary decision mechanisms, while the other is based on adaptive intelligence with complex reasoning engines and knowledge representation. Furthermore, the author studies the role of affective computing and human emotions in AmI, as well as its social impact on a group of people. Finally, while the technology is available, its acceptance by humans is not still guaranteed.

2.3.1 Responsive Environment

2.3.1.1 Event Condition Action Rules

[Aug+08; AN04] *Augusto et al.* have developed a framework based on Event-Condition-Action (ECA) rules to model complex patterns of appliance interactions in smart homes. According to an event occurring on a sensor, the system must respond in order to assist in human activity or adapt the environment. The authors propose a new language that is capable of handling spatiotemporal and uncertainty conditions for smart homes. Its semantics facilitate the description of the events needed to formulate the conditions to operate an action. These rules are used to drive the environment from a knowledge database that is filled by experts. The belief-rule-base component takes into account the degree of belief in order to ensure the consistency of decisions when several condition rules are satisfied. An evidential reasoning approach is used to generate the proper output state for actuators. Conversely, when none of the rules can be applied, the system must learn by extracting rules from human activity. Although this symbolic space for describing system behavior is powerful because the internal state of the system cannot be understood by the expert, the use of a description language is difficult in order to describe complex states and actions.

2.3.1.2 Action Planning

[Sim+06] *Simpson et al.* provide a survey on planning techniques for a smart home. This problem consists of determining a path of actions to reach a goal state from an initially known one. A plan is a sequential ordered collection of actions determined by analyzing the current situation. Firstly, the system has to identify the current context in a smart home according to the sequence of previous events. Then, it has to select the corresponding action plans to control the smart home. An important issue appears when the subsequent states can be derived from the expected state because of actuator errors, external events or human activity. Hence, the planning engine must consider the probability of fails in order to avoid an intractable situation.

2.3.1.3 Decision Tree

[ST06] *Stankovski et al.* discuss the use of decision tree in smart home applications. A decision tree is a graphical representation in which possible actions are represented by a rectangle, while the probability of reaching the next state is represented by a circle. Hence, a decision does not guarantee that the desired next state is reached. It takes into account the possibility of derivation during the planning phase. Its design requires a lot of experimental data to determine the model, but the planning simply uses the expected utility function and classical inference algorithms.

2.3.2 Reasoning Engine

2.3.2.1 Fuzzy Rule-Based Engine

[DHC05] *Doctor et al.* present the project known as iDorm, which is an AmI based on the fuzzy logic model. Each piece of information has an associated degree of certainty in order to compute operations with a tolerance interpretation. For example, the term *the door is a little open* can be represented mathematically by the value 'open' and a certainty degree of $\frac{1}{2}$. Hence, the reasoning engine deduces action plans in terms of the door being neither completely open nor closed. The fuzzy logic model introduces the inaccuracy of language semantics inside the model. It facilitates the modeling of continuous functions in a symbolic space, which avoids the determination of a strict system state according to the sensor values. The authors present results in which their engine of fuzzy logic produces fewer decision errors than other logic systems.

2.3.2.2 Context-Awareness System

[Mag+06] *Magerkurth et al.* present the project known as Amigo, which is an ontology model based on AmI. It is used to separate the language logic from its semantics. The keywords are described in terms of a relationship between two other keywords. This set of descriptions defines a graph called an ontology. This technique is used to facilitate knowledge transfer between heterogeneous devices, such as several types of AmI that do not share a common description language. The authors use ontologies to establish advanced relationships between the different pieces of informations used to model the system context. Hence, the system focuses on the relationship between the pieces of information instead of their content, which improves the efficiency of the context analysis.

2.3.2.3 Learning Techniques

[AIA10] *Aztiria et al.* present a survey on learning techniques for AmI. The learning process consists of extracting a general function to associate an output with an input from empirical data. For example, the state of the actuators is determined according to the current context of the system. There are two main approaches: the extraction of a statistical model, such as one involving ANC or a Bayesian network; and symbolic inference, which tries to built associative rules between input and output, such as fuzzy logic, sequence discovery, instance learning and pattern matching. The trade-off for learning techniques is their ability to learn quickly with the best possible accuracy using a minimal set of data. Meanwhile, the result must be understandable by a human expert. The authors conclude that no learning techniques currently exist that enables AmI to simultaneously address all of these requirements.

2.3.3 Ambient Intelligence Architectures

2.3.3.1 Autonomic Computing

[ATK11] *Andrushevich et al.* discuss the application of autonomic architecture within AmI. This kind of architecture was introduced by IBM in 2001 in order to build self-adaptive systems [Com+06]. Their architecture is based on a MAPE-K control loop, as illustrated in Figure 2.2, for AmI applications. It is composed of four main components, which share a common knowledge space representing system parameters, user preferences and extracted system information:

- *Monitoring* consists of aggregating and filtering the data collected from the sensors and the actuator states.
- *Analysis* determines the system context by evaluating the human and environment activities in order to select an action plan.
- *Planning* generates or updates the action plan according to possible future scenarios and system goals.
- *Execution* deploys and executes the selected plan on the actuators.

This control loop has been designed for self-x properties (self-optimization, self-healing and others). At each iteration, the different components create new knowledge that they learn from and exploit in order to improve their results.

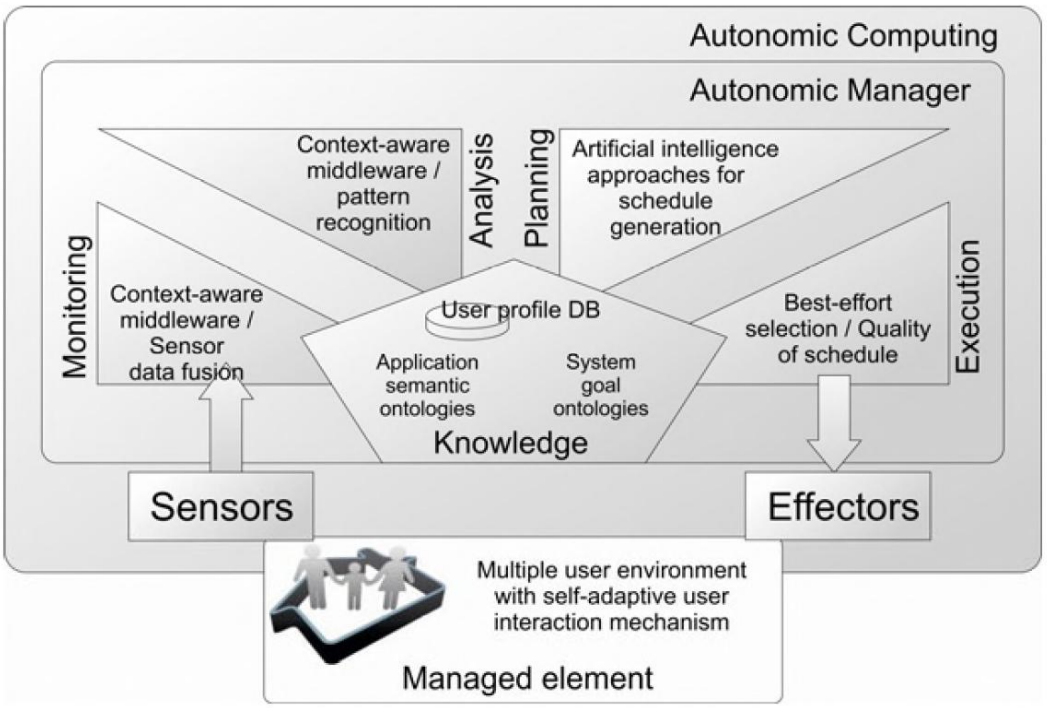


Figure 2.2 – MAPE-K control loop instantiated for AmI.

2.3.3.2 Multi-Agent System

[SV05] *Da Silva et al.* present architecture for MAS in order to design a form of AmI. It is composed of a set of software agents representing the physical devices or residents in a digital environment, as illustrated in Figure 2.3. Each agent is synchronized bidirectionally with the status of the real device. Hence, the agents can communicate together without any network costs. Within their MAS architecture, the agents exchange data and negotiate the shared resources through a tuple space, which represents the physical world. The authors emphasize the interest in MAS for designing complex interactions in AmI. Instead of designing a global artificial intelligence, which drives everything, each agent has its own intelligence that collaborates with others' intelligence to achieve its own goals.

[SM06] *Spanoudakis et al.* present another form of architecture of MAS for the purposes of designing AmI. The authors are interested in the interoperability of their work with other approaches. Firstly, they use the Agent Communications Language (ACL) defined by the Foundation for Intelligent Physical Agents (FIPA). This language defines a set of protocols and performatives to implement agent dialogues. Hence, the addition of new agents does not require particular knowledge about their specific platform. Secondly, they integrate their MAS platform into an Open Services Gateway initiative (OSGi) platform. This kind of platform offers a set of interfaces and functionalities to interconnect heterogeneous components and devices. Finally, this platform can be easily extended by adding a new agent without having particular knowledge and being interconnected with other platforms.

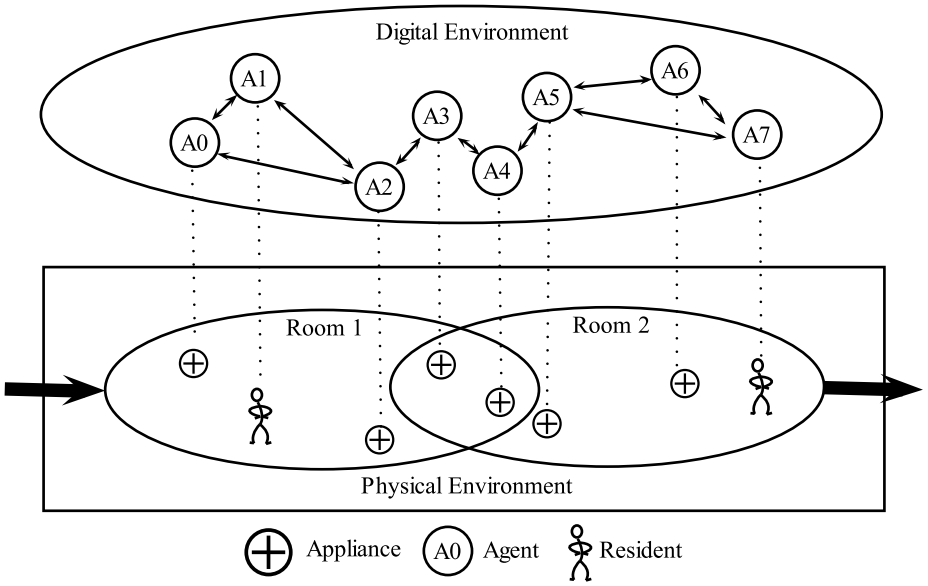


Figure 2.3 – Architecture for a multi-agent system for ambient intelligence.

2.4 Organic Computing Approach

[MSSU11] *Müller-Schloer et al.* present a compilation of papers focused on Organic Computing (OC). This new paradigm is motivated by the necessity to define a framework in order to control self-organization processes in large and complex networks. Although numerous contributions in Multi Agent System (MAS) have proposed frameworks with self-x properties, there are no well-defined definitions that are commonly used. In this book, the authors propose a framework to mathematically define the terms of autonomy, organization, adaptivity, robustness and others with their relationships. The OC framework focuses on methodology to develop self-organized MAS with trustworthy responses.

2.4.1 Principles and Challenges

2.4.1.1 Trustworthy Systems

[Ste+10] *Steghöfer et al.* discuss the challenges and perspectives concerning trustworthy OC systems. Behind the idea of the self-organized system is the reduction in control parameters for a managed system. A system increases its autonomy when the number of parameters is reduced to cover the same number of different tasks. Therefore, the removed parameters are self-established by the system itself. A trustworthy system provides a correct response independently of internal or external disturbances through the self-adaptation of its internal parameters. The authors define *trust* in terms of a multi-faceted concept composed of functional correctness, safety, security, reliability, credibility and usability. They derive these concepts in relation to *trust models*, *trust metrics* and *trust algorithms*. Finally, they conclude that these kinds of technology will be only available in the industrial sector once they are trusted, controlled and comparable.

2.4.1.2 Self-X Properties

[Naf+10] *Nafz et al.* present a formal framework for the compositional verification of self-x properties in OC systems. The authors use a temporal logic to guarantee that the system outputs stay inside a specified acceptance space for the functioning mode. They split the problem into two parts: the validation of the functional system and the self-x algorithms. Based on these two analyses, the authors provide theorems to determine the robustness of the entire system. They evaluate their proposal using an application of the self-organized data flow problem. Their future work will be interested in extending their formal framework to liveness properties.

2.4.1.3 Design Methodology

[KB05] *Kasinger et al.* propose a framework and a methodology for designing Multi Agent System (MAS) that is extendible into OC architecture by adding self-x properties. The authors present a metamodel that can be instantiated through 14 design phases until reaching the final operational OC system. They use an extended Unified Modelling Language (UML) model to represent the different components of the OC system. Their methodology is illustrated on a manufacturing control system, in which a traditional MAS is designed beforehand to add self-x properties. Their contribution is a first step towards software engineering methodology for OC using standard tools.

2.4.2 Models and Architectures

2.4.2.1 Observer-Controller Model

[Ric+06] *Kasinger et al.* present the generic observer-controller model for OC systems, as illustrated in Figure 2.4. The component observer evaluates the current situation of the System under Observation and Control (SuOC) according to an observation model, which is selected by an expert. This model configures the observer engine by selecting the proper analyzers and event predictors to provide context information to the controller. This latter evaluates the possible actions according to the constraints, then selects the best one according to its history database. The constraint model is updated by a simulation engine, which adapts the human requirement to the SuOC.

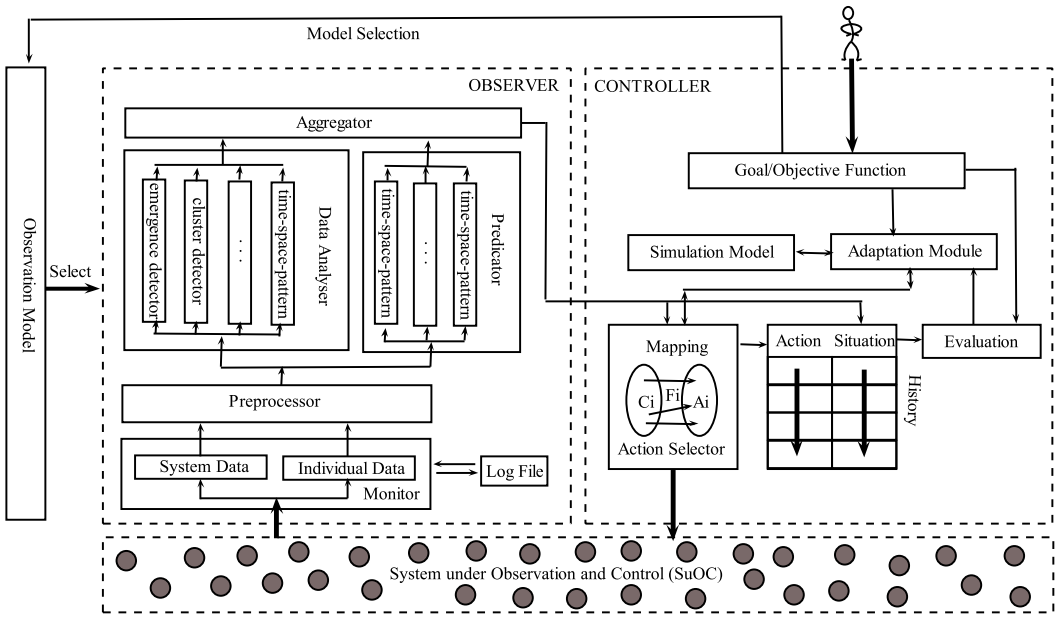
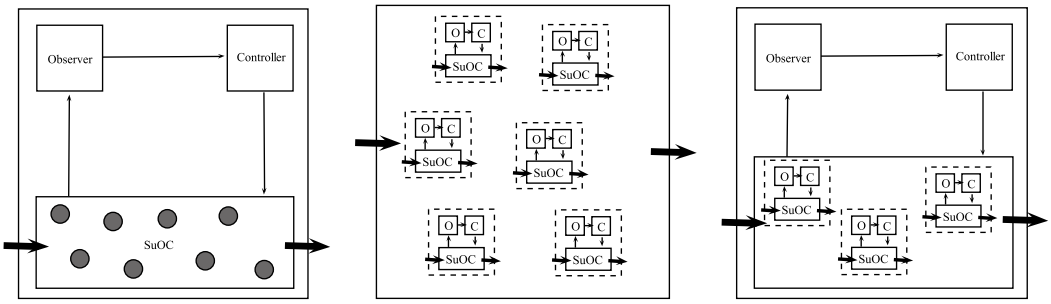


Figure 2.4 – Generic observer-controller model for organic computing.

2.4.2.2 Multi-Scale Architecture

[Bra+06] *Branke et al.* present an application of organic traffic control on city roads. Informed by the generic observer-controller model discussed above, they introduce its application on a different scale, as illustrated in Figure 2.5. The SuOC can be considered differently according to its application or architectural choices. If the observer and controller engines are located in an independent system, the data are centralized and the orders are remotely distributed over SuOC, as shown in Figure 2.5.1. Conversely, each component of the managed system can have its own observer-controller engines in order to form a Multi Agent System (MAS), as in Figure 2.5.2. The combination of both approaches is a MAS managed by another observer-controller engine in a multi-scale architectural style, as in Figure 2.5.3.



2.5.1: Centralized 2.5.2: Decentralized 2.5.3: Multi-scale

Figure 2.5 – The different scales of observer-controller architecture.

2.4.2.3 Evolutionary Computation

[MKD11] *Matsumaru et al.* present a paper on organization-oriented chemical programming (OOCP). In an Organic Computing (OC) system, the observer-controller model supervises an SuOC, which is mostly a complex system, such as one found in a physical environment or a complex network. This SuOC, however, can be also implemented intentionally in the same way as a complex system. The authors present the OOCP framework in an OC system. The resolution of a problem is performed by a set of interactions between artificial chemical elements. The problem is defined as a set of possible interaction rules: positive or negative reactions. At the end of the process, the result is encoded in terms of final concentration of the artificial chemical components. Hence, the observer-controller engine prepares the initial conditions for the SuOC analysis and controls the emergence phenomena in order to collect the results. Such an approach has a very strong robustness with regard to failures and system disturbances, which is due to problem resolution not being dependent on hardware computations but the result of an emergency at the software level.

2.4.3 Application Examples

2.4.3.1 System on Chip

[BKK08] *Buchty et al.* have developed an innovative self-configuring bio-inspired architecture known as the *Digital on-Demand Computing Organism* (DodOrg). Their project involves fully distributed middleware in order to distribute tasks over multi-core architecture. They present a complete framework composed of three levels: an application test bed, organic middleware and the Organic Processing Cells (OPC). The tasks are implemented with a high level language in order to be distributed over the middleware to minimize global chip temperature. Each computation unit, which is implemented on FPGA, embeds its own observer-controller model and communicates with others by an hormonal interaction system. The authors argue that their system is extremely robust, self-organizing, flexible, real-time, large-scale and self-healing system.

[SHB09] *Schuck et al.* present, in detail, the architecture of OPC technology. The authors work focuses on the remote reprogramming of cells through the *internal configuration access port* of Xilinx Virtex-FPGA technology. The DodOrg project offers a new generation of computers, in which tasks are dynamically distributed over the cell according to the computation requirements of the application, as well as factors in the computer environment, such as temperature. This contribution adds the possibility of changing FPGA programming of the DodOrg system in order to deploy new functionalities inside the chips.

[Bou+06] *Bouajila et al.* present another type of System on Chip (SoC) architecture, based on OPC technology. SoC technology is highly sensitive to environmental variations, which can introduce bit and timing errors. Hence, the authors have developed a system on FPGA to control execution pipeline consistency for Metal Oxide Semiconductor Field Effect Transistor (MOSFET) technology. This is a first step towards self-correction properties for SoC technology at the hardware execution level.

[FS05] *Fey et al.* present OPC architecture for image preprocessing tasks in future Complementary Metal Oxide Semiconductor (CMOS) camera chips. The system architecture is fully distributed over the FPGA. Instead of using a sensor matrix connected to the process unit, each pixel has its own observer-controller system. Hence, the camera chip is composed of a set of OPC. The authors propose a concept of marching pixels, which uses mobile agents located in each OPC to detect an object. The proposed algorithm is similar to an algorithm with attractive and repulsive mechanisms to help the agents become fixed in an interest area. Therefore, either the detection or the tracking of an object involves the analysis of the position of agents on the OPC. The authors expect to extend their work for three-dimensional cameras, which are strongly limited by traditional computation approaches. Indeed, they require much more computations and, in turn, powerful chips that are difficult to miniaturize.

2.4.3.2 Traffic Lights in a Smart City

[Pro+08; Pro+11] *Prothmann et al.* present an application of OC principles and architecture for traffic control in a smart city. The system regulates the vehicle flows by managing the traffic lights in order to reduce network congestion and the average delay time at the intersections. The system is composed of two observer-controller layers. The first one determines the best policy for the traffic lights according to the observations on current vehicle flows. The second one creates offline classifiers in order to predict the traffic according to the period and the events in the surrounding area.

2.4.3.3 Energy Management in Smart Homes

[Bec+10] *Becker et al.* present a smart home project based on OC technology. The system is composed of multi-scale architecture. Each appliance has its own observer-controller engine in order to locally monitor its usage and its energy consumption, as well as to control them. A main observer-controller engine is responsible for managing all of the appliances. It has to manage the appliance scheduling according to the requirements of the residents and the energy grid operator. In the proposed scenario, the system has to even out the total energy consumption of the smart house. Some appliances have a flexible energy consumption, such as the dishwasher, which can be deferred to a later point in the day. Similarly, an electric car can be powered during the night, while it can also be used to supply power during high levels of demand for energy in order to reduce the energy provided by the provider grid. Hence, the system differs and load-balances energy consumption, as shown in Figure 2.6.

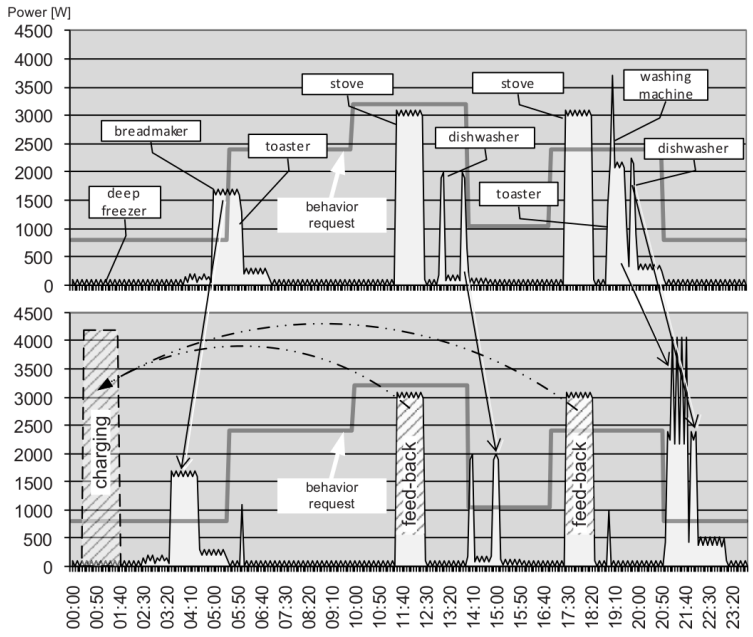


Figure 2.6 – Example of the energy management policy in a smart home.

2.5 Synthesis

This chapter has been presented state-of-the-art developments relating to IoT research activities. It is composed of three parts, from the technological requirements and constraints, to the Ambient Intelligence (AmI) techniques and forms of architecture, which have been extended due to the growing involvement of Organic Computing (OC).

Applications relating to the IoT require massive data produced by smart objects. Their huge number exposes several issues regarding their communications in a large-scale network, which has to be easily connected to the Internet. The constraint in having limited energy consumption can reduce their computation and memory capacities. Being at the intersection between the physical and digital worlds, they will be produced by known IoT experts. Therefore, middleware technology has been intensively developed to provide an efficient abstraction layer between the physical object and the digital world in order to build secure, reliable and heterogeneous architecture, which is composed of thousands of tiny nodes.

The IoT is developing jointly with AmI in order to build smart environments. The technological constraints on the IoT have to be considered when designing intelligent systems. On the one hand, new techniques of artificial intelligence have been investigated in order to address the challenge of smart environments, while, on the other hand, traditional approaches have to be redesigned to be used on existing IoT architecture. Historically, an intelligent system was located in a set of identified platforms in order to remotely manage a system. Nowadays, they must be distributed over a network of embedded systems in order to locally manage the environment, while the system consistency must be maintained at a macro level.

Some researchers have started a new initiative to develop a paradigm for such a complex system. The OC approach is a bio-inspired technology that is interested in designing a distributed and autonomic system, in which the system control is not operated by a central decision system, but the interactions of individual agents, in order to facilitate the emergence of the desired global behavior. Current developments focus on the definition of standard architecture and models, which can be evaluated in real use cases. However, there is still no generic framework for the design of IoT applications, such as an autonomic OC system.

The present thesis has been motivated by proposals for tools needed to design operational IoT systems that are compatible with OC requirements, as stated in Part II. At the same time, it relates to Part III, in which distributed algorithms for context learning and distributed decisions are proposed in relation to an OC-like approach.

Contribution: An Organic Ambient Intelligence[1]

[1]Summarized in Clement Duhart and Cyrille Bertelle. "Toward Organic Computing Approach for Cybernetic Responsive Environment". In: *International Journal of Ambient Systems and Applications (IJASA)* 3.4 (2015). doi: DOI:10.5121/ijasa.2015.3401

In Organic Computing (OC) community, the term of Organic refers to systems in which the degree of autonomy is increased thanks to a supervisor layer Observer/Controller which self-managed some internal parameters. Therefore such systems are able to change their configuration in order to reduce required human interventions during the dynamic of the managed environment. The reflexion is oriented on a new generation of software architecture based on stacked layers of different functional levels to adapt the system behaviour on runtime. However based on previous State of the Art, there is no major research on intrinsic properties of organic chemistry of self-adaptation which allows atomic elements to be self-organized to form functional materials at higher levels and multi-scales. In optimization problems based on neural networks, genetic and ant algorithms or others, the information is not stored in their atomic components of neurons, ants, genes but in the self-organization of their interactions. Because the current computers are based on sequential execution, such algorithms have to be adapted to simulate interaction computation with the issues of parallel executions, asynchronous communications and independent process life-cycle.

This thesis is interested in studying a new approach of computation bio-inspired by organic chemistry, cellular supports and neural architectures.

The background contribution is a reflexion and an experimentation on the design of Artificial Organism (AO) for autonomic computing. Obviously such position cannot be fully studied during a single thesis, but the developped softwares and theorical models provide first *Proof-of-Concepts* to design, deploy and execute Artificial Neural Network (ANN) over distributed support of Wireless Sensor and Actor Network (WSAN) based on interraction of chemistry material at an atomic layer. It is supposed that the next generation of intelligent systems would be more interested in new kinds of computation to have adapted hardware and architectures to Artificial Intelligence (AI) algorithms.

Environment Monitoring and Management Agent (EMMA) is a proposal of framework for distributed Organic Ambient Intelligence (AmI).

AmI is an interesting application challenge for this approach such as it is an open and distributed system at hardware, network, service and intelligent layers. Multi-levels of failures must be considered in addition of their integration with current and future technological evolution of the concerned research areas. In addition this kind of systems must be highly adaptable such as their purposes, constraints and ressources are evolving at each time and over the time. When AmI will be deployed with the hardware support,the software and the networking, it should not be possible to determine their future use over building life cycle. Hence the autonomy, scalability, security and Internet integration of the system are major concerns of EMMA framework.

The WSAN is considered such as an autonomous System under Observation and Control (SuOC) managed remotely by one or several supervisors which deploy or optimize the running services during runtime. Two applications of intelligent components are proposed: an actuator management by context learning with Artificial Neural Controller (ANC) and a distributed neural voting procedure for parameter selection. The SuOC is composed of the connected objects located in the managed environment. Each of this connected object is a System on Chip (SoC) with a wireless module executing hard-coded services such as sensor drivers or system tasks. The Web-based architecture is integrated with Internet services thanks to the use of last network standards of IPv6 LoW Power Wireless Area Networks (6LoWPAN). The EMMA middleware abstracts the WSAN to the AmI thanks to a Resource Oriented Architecture (ROA). Finally, the AmI is resumed by a set of Service Choreography (SC) which form a graph of interactions between the object WEB resources. The communication between the ressources of node services are operated by reactive mobile agents. Hence, the middleware is a distributed, autonomous and shared tupple space in which mobile agents are executed. The Service Choreography (SC) act as Observer/Controller component distributed over the nodes to manage the actuators according to the sensors through their respective resources. A security layer ensures that operational communications stay local and encrypted to protect data privacy and system security. Finally, the supervisors are responsible to manage remotely the SC. In the proposal, they are implemented according to the MAPE-K architecture. They compose and deploy the SC from a database according to WSAN requirements and update them according to the system feedbacks. The Figure 2.7 presents the architecture overview with the corresponding chapter colors.

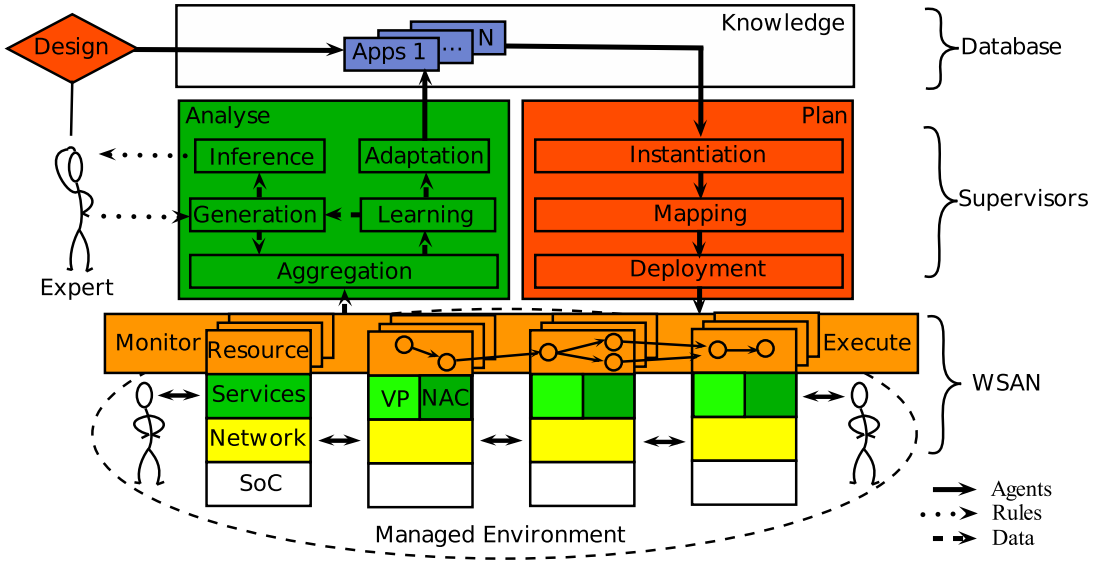


Figure 2.7 – Thesis Overview: EMMA Architecture.

Part II

- Chapter 3 - *Capillar Internet Network* presents the network architecture on which is running the EMMA framework. It does not contain direct contributions, however the different evaluated approaches to build an integrated WSAN with Internet Protocol version 6 (IPv6) networks are presented. Impacts of software architectures between service orchestration and choreography are evaluated in order to validate the use of a distributed model. Finally, the different tools developed during this thesis are introduced.
- Chapter 4 - *Active Resource Middleware* presents the Resource Oriented Architecture (ROA) of the middleware with its mobile agents. They are used to interconnect the resources in order to build the choreographies of services which are denoted SC. The specifications of the agents are detailed including their graphical representations which are based on an augmented Numerical Petri Network (NPN). The self-x properties of the mobile agents are discussed regarding to the deployment requirements. Finally, the SC partitioning for security aspects are presented.
- Chapter 5 - *Service Choreography Deployment* presents the deployment process of SC. They must be deployed according to the hosting capacities of nodes, the network constraints and the application requirements in order to minimize the network communications. The deployment process is itself considered such as a SC which must be mapped over the network. The mathematical formulation is resumed by a pseudo-Boolean optimization (PBO) to map the mobile agents and their temporal resources on nodes.

Part III

- Chapter 6 - *Artificial Neural Controller* presents the methodology used in EMMA system to create ANC. There are two learning loops to exchange information between a knowledge base composed of logical descriptive rules and the statistical classifiers of ANC. They are trained to control actuators according to sensor and system states. The ANC are deployed and executed directly on service components of the nodes. Mobile agents are responsible of their deployments and connections with the other system components through resources.
- Chapter 7 - *Neural Voting Procedure* presents a fully distributed algorithm of voting procedure to allow the nodes to take a common decision without the centralization of their choice preferences. The implementation of this algorithm is done by mobile agents deployed over the middleware to form a distributed ANN. The nodes publish their preferences through their resources which are connected by the mobile agents.

Part II

An Organic IoT Framework

Chapter 3

Capillar Internet Network

Notre progrès technique tant vanté et la civilisation en général peuvent être comparé á une hache mise dans les mains d'un psychopathe criminel.

Our entire much-praised technological progress and civilization generally could be compared to an axe in the hand of a pathological criminal.

Albert Einstein [Ein71]

Contents

3.1 Introduction

The Internet of Things (IoT) is the extension of the Internet into the physical world in order to collect real-time information about particular environments. For future applications, such environments should be controlled by intelligent systems from the Internet in order to improve their accessibility and comfort, as well as their safety and security, while optimizing global energy consumption. However, this will require large-scale infrastructure regarding the network, information systems and applications. Each of these parts are strongly dependent on physical characteristics. Given the number of objects involves implies that they should be designed such that their size, cost and energy consumption are limited, which in turn will have a significant impact on their communications. Their deployment in large environments, such as buildings, does not allow for the use of wires to connect them. Hence, wireless communications are preferred.

This chapter presents a proposal for network architecture for the IoT by combining current technologies to offer a maximum number of possibilities, in terms of addressability, packet routing and architectural flexibility, in order to establish end-to-end data flows from appliances to Internet services. Section 3.2 introduces the different technologies and their achievements in building a complete network solution, composed of appliances and their wireless network, which is connected to the Internet through gateways. This proposal is based on current technological standards based on an end-to-end IPv6 connectivity. Other proposals can be found in the literature, but they are mostly based on an intermediate multi-protocol proxy. However, such approaches are not compatible with the spirit of an homogeneous IoT based on the Internet Protocol (IP), which is the concern of this chapter.

Based on this network overview, an evaluation of Service Oriented Architecture (SOA) paradigms is investigated in relation to service orchestration or choreography in Section 3.3. The first paradigm involves architecture in which all data are collected on a central system, which controls everything. This approach facilitates the design and deployment of new services, but it has strong network limitations in large-scale networks. Conversely, service choreography locally distributes all communications between appliances, which reduces network congestion; however, the management of such system is a difficult task. Given that the results have shown that architecture based on service choreography improves network reactivity, this thesis has been motivated by the proposal of a framework for the orchestration of service choreographies.

Finally, Section 3.4 presents an overview of the different tools developed during this thesis to build and manage service choreography infrastructures. The following chapters, which together form Part II, detail theoretical contributions and implementation results, in order to provide a software suite using the appliance software and network management for service development.

3.2 Network Infrastructure

Network infrastructure refers to the set of devices required to establish communication links between appliances and the Internet. The development of large-scale wireless networks for IoT applications has raised the issue of energy consumption. On the one hand, some end-devices run on batteries because power lines are not accessible in all places; while, on the other hand, a critical number of network devices can dramatically increase general energy consumption.

Nowadays, standard protocols have been defined in order to improve network lifetime, while ensuring its sustainability through the use of IP-based protocols. The Internet Engineering Task Force (IETF) defines the IPv6 LoW Power Wireless Area Networks (6LoWPAN) protocol for low power and lossy networks. The energy consumption, memory and computation of network devices are considered in order to connect them to Internet over wireless communications.

Figure 3.1 presents three kinds of devices, which are considered in such an infrastructure: a node is an energy constraint device, which must avoid significant communication in order to prolong its lifetime; a router is an intermediate node with lower energy constraint needed to establish a mesh network between nodes; and the Border Edge Router (BER) is a gateway between the Internet and the Wireless Sensor and Actor Network (WSAN).

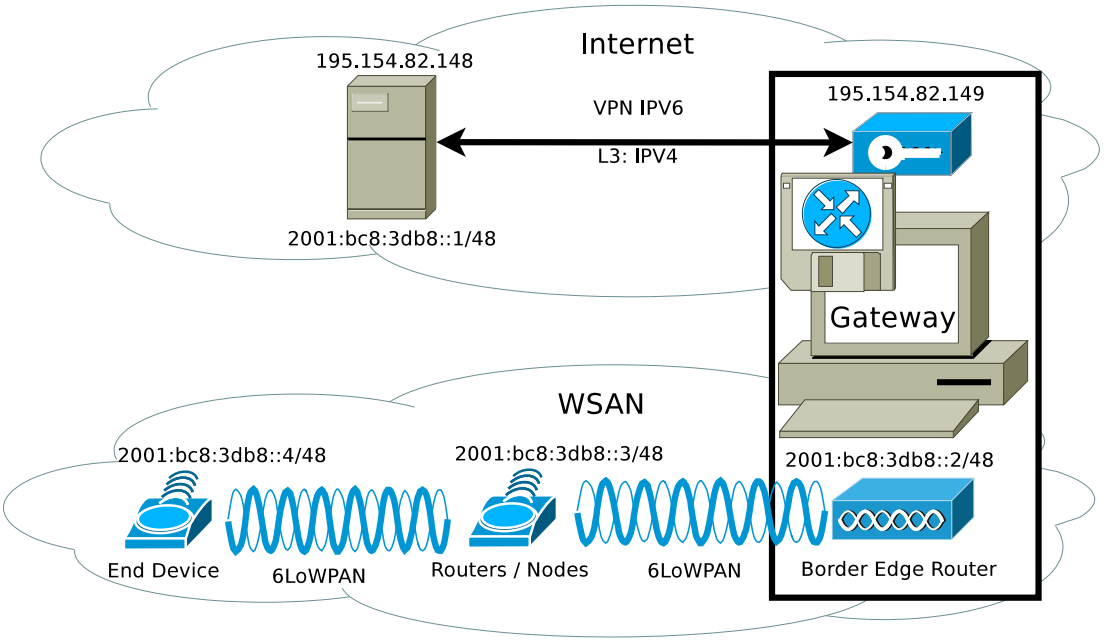


Figure 3.1 – Illustration of a classical IoT network infrastructure.

3.2.1 Wireless Sensor and Actor Networks

A Wireless Sensor and Actor Network (WSAN) is composed of distributed energy constraint devices, which communicate over a mesh network. The large number of devices involved and their mobility, in terms of failures or moving physically, require self-configuration mechanisms to establish and maintain networks.

3.2.1.1 IP Connectivity: 6LoWPAN

The 6LoWPAN protocol, defined in RFC 4944 for IEEE 802.15.4 wireless technology, has been designed to address issues concerning RFC 4919: IP connectivity, limited packet size, self-configuration and security aspects. Most IPV6 mechanisms are used in 6LoWPAN, in addition to adaptations for the reduction of energy consumption through the limitation of packet size by the IP and transport header compression.

3.2.1.2 Routing Protocol: RPL

The Routing Protocol for Low power and Lossy Networks (RPL) routing protocol, illustrated in the sequence diagram 3.2, allows WSAN to automatically and dynamically establish IP addresses and routing tables by using the IPv6 *Node Discovery* mechanism (for more details, see draft-ietf-6lowpan-nd and -rpl). Each node generates its own local IP address (fe80::) until receiving a network prefix disseminated from a Border Edge Router (BER). The nodes then subscribe to the first available router in order to maximize the data rate between them and the BER. This operation can be iterated periodically to maintain the best network connectivity between nodes and the BER.

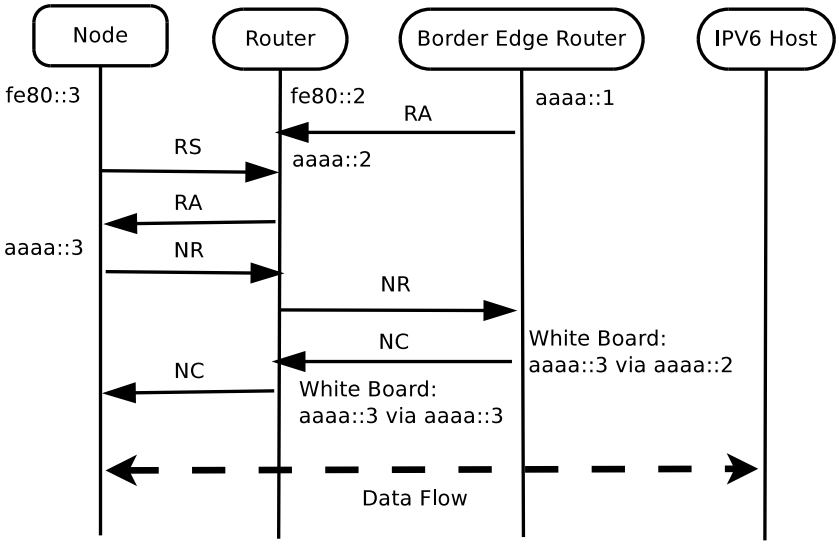


Figure 3.2 – Sequence diagram of 6LoWPAN-RPL network establishment.

3.2.1.3 Gateway: Border Edge Router

A gateway is a network device establishing connectivity between two different networks. A WSAN based on the 6LoWPAN has two major functions. Firstly, the IPv6 packets must be compressed or uncompressed between the WSAN and traditional network interfaces. Secondly, the gateway hosts the Border Edge Router (BER), which is responsible for the network prefix dissemination used during the establishment of network connectivity by the RPL protocol. In Figure 3.3, the network infrastructure is composed of two BERs (gateways), which leads to the construction of different overlapped routing trees. Indeed, the IPv6 protocol allows nodes to have several IPs, which can be used to select the best routing path according to the communication recipient.

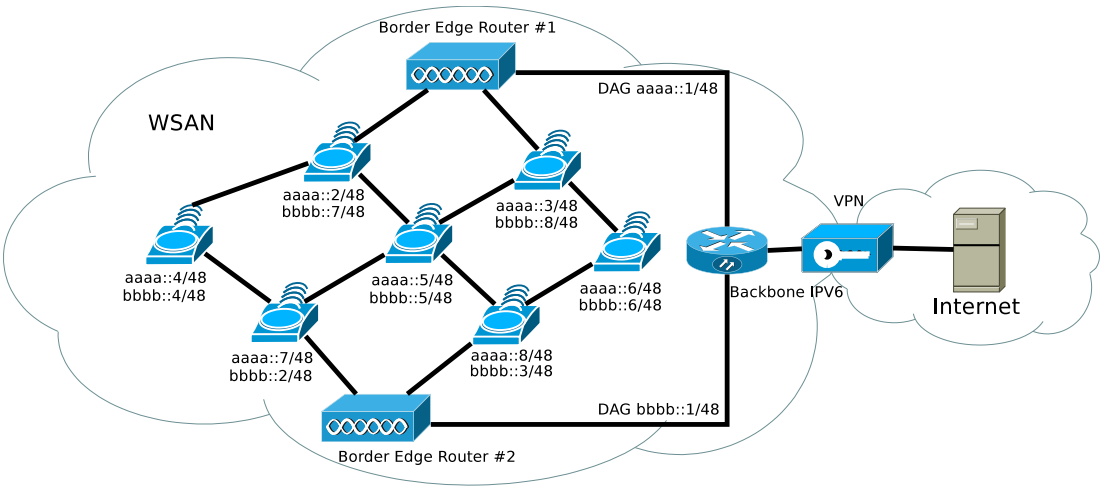


Figure 3.3 – WSAN Multiple Routing RPL DAG.

3.2.2 Home Information System

The Home Information System (HIS) refers to the set of connected appliances in a residence. Currently, it is mainly composed of multimedia devices, which are already connected through standard protocols, such as Universal Plug and Play (UPnP) and Bluetooth. Their development was not initially designed to be an extension of the Internet, even if more and more HISs are connected to the Internet because of the emergence of Wi-Fi enabling devices, such as Network Attached Storage (NAS), wireless printers and tablets.

The integration of Wireless Sensor and Actor Networks (WSANs) on the Internet should be easy to facilitate given current developments in HIS based on IP technology. The following section reviews the different devices enabling WSAN integration into HIS and technological aspects used in this thesis.

3.2.2.1 Permanent Gateway

Recently, plug computers have appeared in HIS for the purpose of installing local fanless servers at home. Even if their hardware architecture, which is based on ARM technology, is designed to limit energy consumption, most of these devices are able to execute a complete Operating System (OS) based on Linux. They are used in conjunction with NAS and multimedia platforms, which makes them an interesting access point for WSANs. On the one hand, the Linux kernel is natively supporting the IPv6 and includes a set of network management tools. On the other hand, Contiki, presented in Chapter 2, is an Operating System (OS) for nodes, which supports Remote Network Driver Interface Specification (RNDIS), thereby allowing it to communicate natively with the Linux kernel. Hence, the architecture in Figure 3.4 is a native integration of a 6LoWPAN WSAN in an HIS based on IPv6.

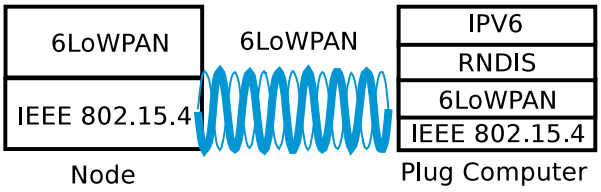


Figure 3.4 – 6LoWPAN integration on a plug computer based on GNU/Linux.

3.2.2.2 Mobile Gateway

The development of smartphones and the improvement in their Internet connectivity offer new perspectives for gateways in HIS. Firstly, they are platforms that already host applications and preferences of residents. Secondly, they enjoy a permanent Internet connection, which can be shared with the HIS instead of using a dedicated Internet Service Provider (ISP). Hence, they all have gateway characteristics for WSANs and are interested in privacy; for example, the HIS is physically disconnected from the Internet when there is no user present.

Figure 3.5 presents architecture for integrating a WSAN to the Internet through a mobile gateway, such as a tablet. A Tunnel Serial Line Internet Protocol (TunSLIP) is established between a Contiki OS node and the Linux kernel of an Android OS. This kind of connection is useful to bypass the lack of an RNDIS driver on a lightweight Linux kernel.

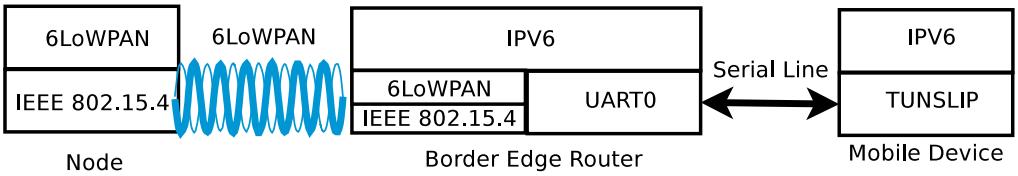


Figure 3.5 – 6LoWPAN integration on a mobile phone based on Android.

3.2.2.3 IPv6 Backbone

The deployment of IPv6 is still an experimental enterprise on a classical network infrastructure, even if significant efforts are made to promote it. Currently, the Internet does not fully support IPv6 routing; moreover, most network tools are not stable enough for this IP technology. Hence, its use is mainly concerned with a local network backbone, instead of a general Information Technology (IT) infrastructure. In such situations, the dissemination of the IPv6 network prefix can be operated locally from a Border Edge Router (BER) localized on previously presented gateways. Otherwise, it can be installed on a remote Internet server, which pushes IPv6 routes through an Internet Protocol version 4 (IPv4) tunnel.

Figure 3.6 presents a general infrastructure based on an IPv6 backbone, which ensures connection between different WSAN, Wireless Fidelity (WiFi) and traditional Home Information System (HIS) devices, such as NAS. In turn, each device has local access to a WSAN; in particular, tablets and mobile phones can operate like a remote control with personal user applications and preferences.

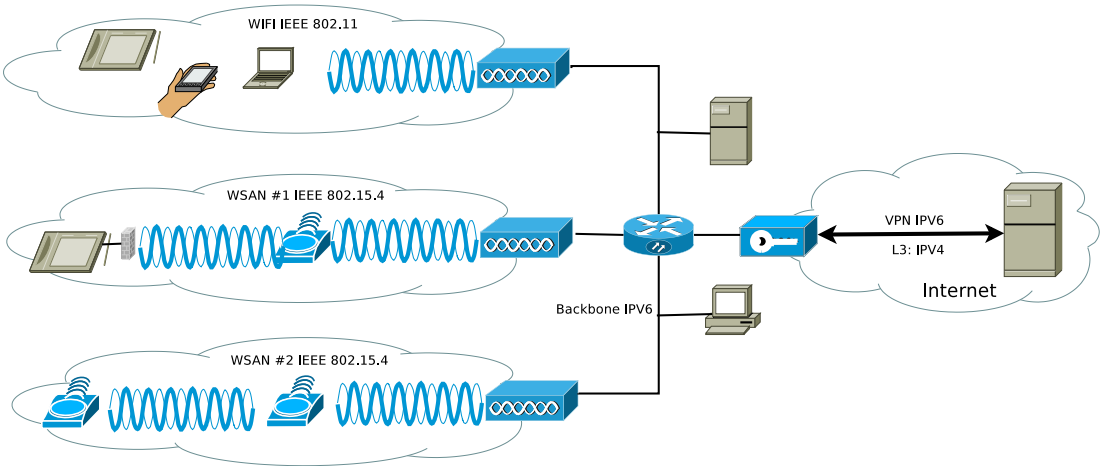


Figure 3.6 – WSAN Integration in Home Information System (HIS).

3.2.3 Internet Integration

The Internet integration of HIS is a still an open issue in IoT discussions. The IPv6 should permit direct access to local devices from the Internet; however, their vulnerability to remote attackers raises difficult security questions. Indeed, WSANs are very sensitive to Denial of Service (DoS) because of their strongly limited bandwidth. Even if WSANs are IP enabling, they must be protected. There are two major use cases for the Internet integration of HIS. On the one hand, the management of home appliances should be managed by external services, such as a security guardian company. On the other hand, an IPv6 backbone of

HIS could be distributed over several distant sites, such as on a campus composed of different buildings.

Figure 3.7, the two examples are represented in which three 6LoWPAN islands communicate with each other over the Internet and external IPv4 services.

IPv6 communications can be performed thanks to *6to4* tunneling. The 6LoWPAN packets are uncompressed on the gateway and encapsulated into IPv4 ones in order to transfer them between HIS islands over the Internet. This approach is natively supported by IP technology, while the communications can also be secured using a tunnel based on a Virtual Personal Network (VPN).

IPv4 services have two main solutions for communicating with an IPv6 HIS. They can translate their IPv4 packets to IPv6 ones thanks to a dual stack IPv4-IPv6. Hence, the communications are peer-to-peer between 6LoWPAN devices and external services. Otherwise, the HIS provides a proxy interface, which is responsible for collecting data and managing devices in the HIS, according to the requests of external services. As such, there is no direct communication between the IPv4 external services and the WSAN. This approach is preferred by the IoT community, as it is much easier to protect WSANs against attackers or other inappropriate usages on the Internet. However, the proxy interface must implement all possible uses of the WSAN, as it is the unique controller device.

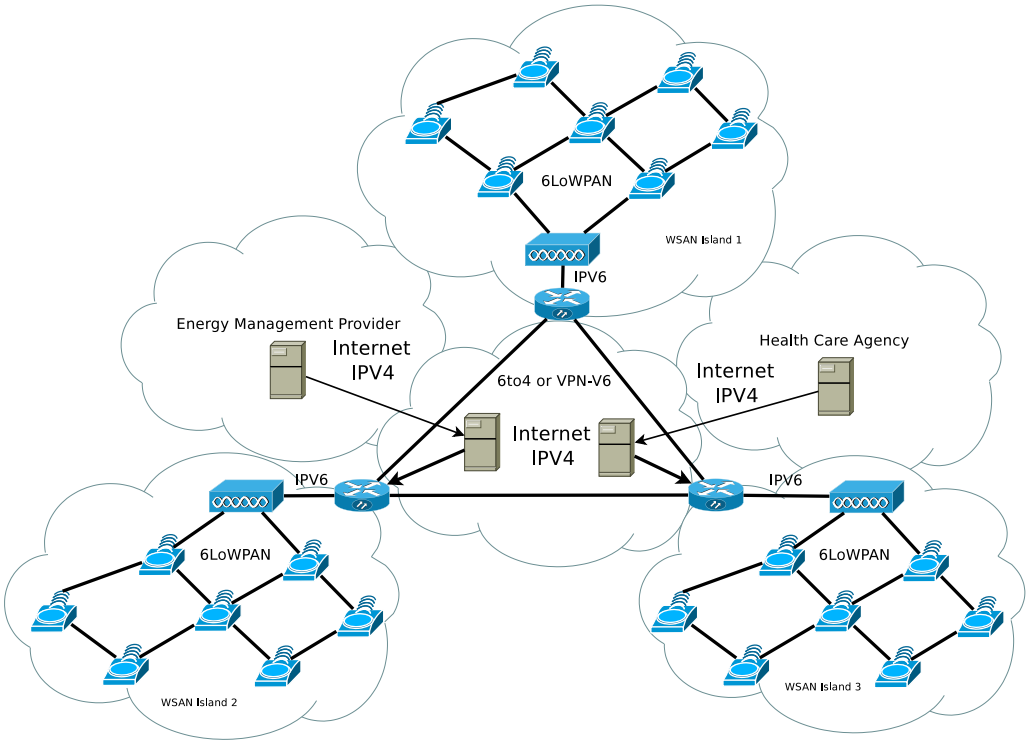


Figure 3.7 – HIS integration into an IPv4 heterogeneous multi-site infrastructure.

3.3 Service-Oriented Architectures

The typical network infrastructure for the IoT, as presented in the previous section, is composed of several network segments connected through gateways. Long path communications between the Internet and a WSAN are not desirable if network bandwidth, reactivity and security are to be maintained. Hence, this section evaluates the relevance of Service Oriented Architecture (SOA) in order to minimize external communications of a WSAN. Indeed, IoT applications are mainly composed of collaborations between different objects in the same physical place. SOA is a generic model in which applications invoke local or remote services to produce, process or manage data. The two major approaches are:

- Service Orchestration (SO): a central system collects all data and directly controls all appliances.

- Service Choreography (SC): different appliances are configured in order to directly exchange information with each other.

3.3.1 Resource-Oriented Architecture

Resource Oriented Architecture (ROA) is software architecture based on SOA in which an application is a set of software components interconnected at a Web resource level. Each service is encapsulated into a container, which publishes a REpresentational State Transfer (REST) interface of resources. Hence, an ROA application is a program in which operations and variables refer to Web resources. Its execution is a set of successive Hyper Text Transfer Protocol (HTTP) requests in order to perform data processing on remote services. The high flexibility of HTTP allows the requests to be adapted to the target services if they are constrained. Moreover, the ROA application can be executed from a central system, such as an SO, or it can be distributed over the services themselves, such as in the case of SC. In this case, an ROA application is a set of distributed publish-subscribe mechanisms that interconnect the resources of different services.

Its application on a Wireless Sensor and Actor Network (WSAN) abstracts the actuators, sensors and other operations by resource. In such situations, the application of a control loop between sensors and actuators is performed locally instead of across the differently presented network segments. Therefore, remote supervisors are responsible for configuring the different devices so that SC can be deployed between them, which means that only the data concerning the *service and network discovery* are collected remotely from the supervisor. As such, the management of SC is performed remotely, whereas the communications of the applications are executed locally. Supervisors are only responsible for monitoring and deploying SC in order to prolong the network lifetime.

3.3.2 Experimental Model Analysis

The previous sections have presented the standard protocols for Wireless Sensor and Actor Networks (WSANs) designed for centralized architecture, such as in cases where communications are transmitted along a routing tree. The discussions around SOA have emphasized the interest in Service Choreography (SC) in order to avoid long path communications. The following experimentations evaluate this apparent conflict in applying SC on WSANs. The experimental setup is composed of 16 nodes (AVR atmega128rfa1, see Table 2.1), executed in a COOJA simulator and distributed randomly in order to build a centric routing tree around the gateway. The longest path to the gateway is limited to 4-hops.

3.3.2.1 Service Orchestration

Orchestration experimentation sends Internet Control Message Protocol (ICMP) requests one-by-one from the external gateway to each node through an RNDIS interface. Figure 3.8 shows an important correlation between node activity and its relative position in routing RPL Directed Acyclic Graph (DAG). Routers forward more network flows when they are close to the gateway. Conversely, node response time increases according to the number of intermediate routers until the gateway is reached. RNDIS interface overcost is very important (more than 200 ms) in comparison with router reactivity (less than 50 ms).

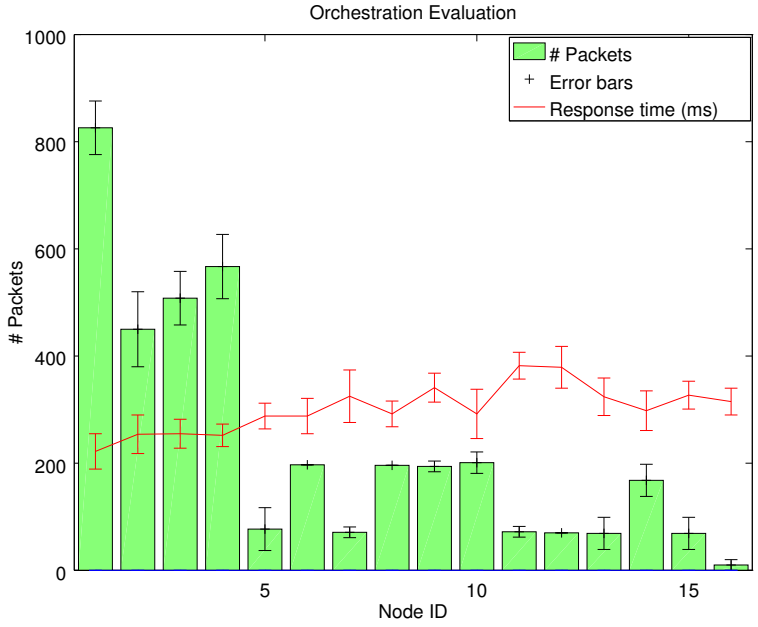


Figure 3.8 – Service Orchestration (SO) evaluation on a random network.

3.3.2.2 Service Choreography

In choreography experimentation, each node periodically sends an ICMP request to a random node in the WSAN. Figure 3.9 presents a similar form of results for orchestration experimentation because the network traffic follows the routing path built from the gateway. However, it is more distributed over all routers than during orchestration. Besides, the response time of all random communications is significantly reduced by a factor of 10 because there is no overcost due to 6LoWPAN and IPv6 translation on a gateway.

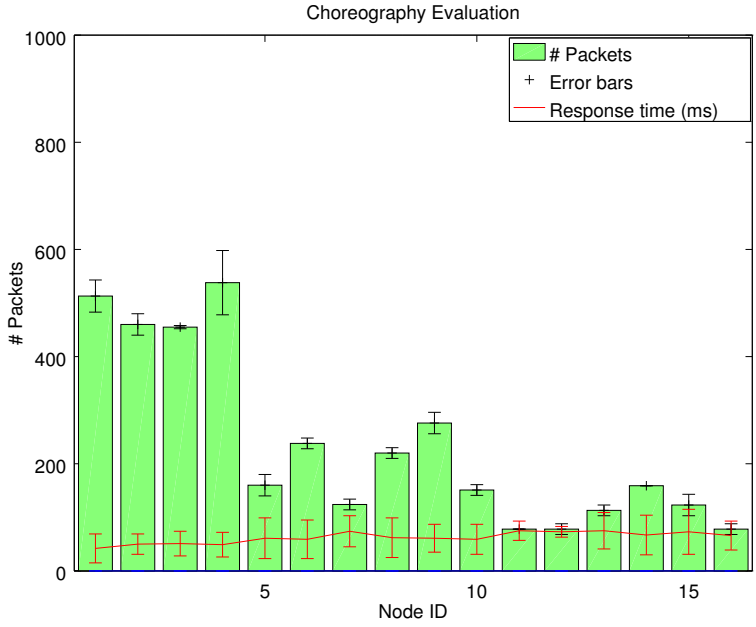


Figure 3.9 – Service Choreography (SC) evaluation on a random network.

Experimentations show that a decentralized model improves node response time and therefore network reactivity. This result is very interesting because the experimentation on SC has been produced in terms of its worst-case scenario. Indeed, the communications are randomly produced, whereas, in real application, they would be configured in order to minimize network load. The consumers of services should be located close to the device that produces the service. It should be noted that, in the below experimentation, the extreme multi-hop path is 8-hops, with an average response time between neighbours of 7 ms. Therefore, the design of efficient service mapping, according to network topology, is a major key to reducing network load and validating the application of SC on a WSAN.

3.4 Software Tool Contributions

This section presents the different software that has been developed for deploying SC over WSANs; more details are given in subsequent chapters.

3.4.1 Active Resource Middleware

emma-node is middleware based on ROA and implemented on the Contiki OS [Dun03] for 6LoWPAN networked applications.

3.4.2 Hybrid Network Simulator

The *emma-cooja* package contains a set of plug-ins for the COOJA simulator [Ost+06]. It provides a simulated radio environment in which WSAN nodes are emulated. Below is a list of the developed plug-ins:

- *emma-cooja-rndis* is a plug-in to execute hybrid simulation in which emulated and physical nodes take part in a common application.
- *emma-cooja-analysis* vis a plug-in for SC behaviour analysis. Resources are monitored and managed for each node (see Figure 3.10, in which the network communications are visible in real-time between the nodes, in the form of a circle with their IP, and the Unified Resource Identifier (URI) of their internal Web resources is shown in blue typeface).
- *emma-cooja-emma* is a plug-in to connect the COOJA simulator in the same way as an EMMA node such as the simulator is an actor of the simulation.

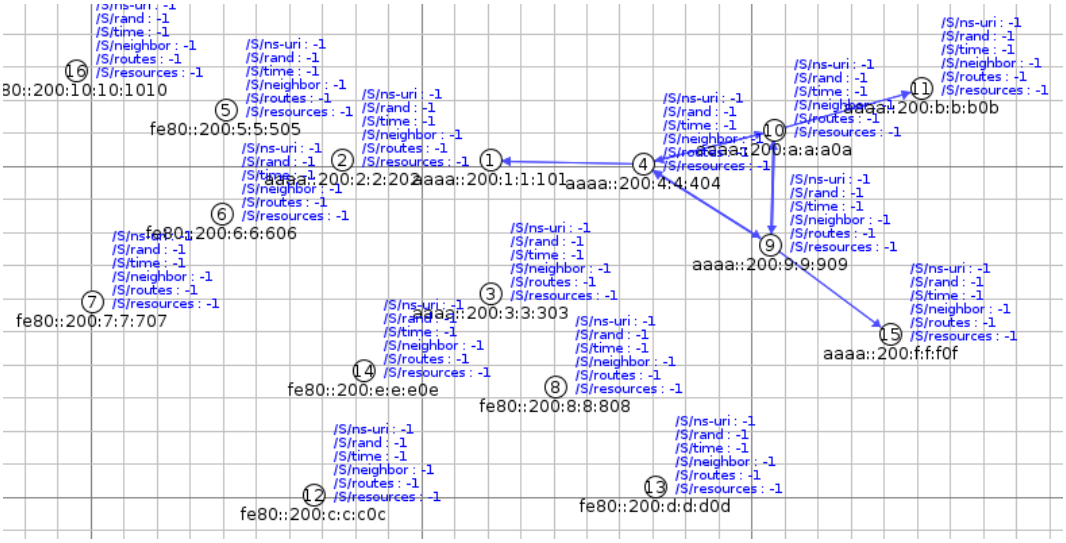


Figure 3.10 – Screenshot of the simulator plug-in *emma-cooja-analysis*.

3.4.3 Service Choreography Software

The *emma-design* package, as illustrated in Figure 3.11, contains two pieces of JAVA software to design, analyze and deploy SC:

- *emma-design-agent* is an application for the graphic design of SC based on an adapted Petri network model, as detailed in Chapter 4. In addition, a simulator engine is available to evaluate the logical behaviour of SC.
- *emma-design-mapper* is an application for instantiating designed SC according to a WSAN target. This software evaluates the best SC mapping and deployment process in order to minimize the network load.

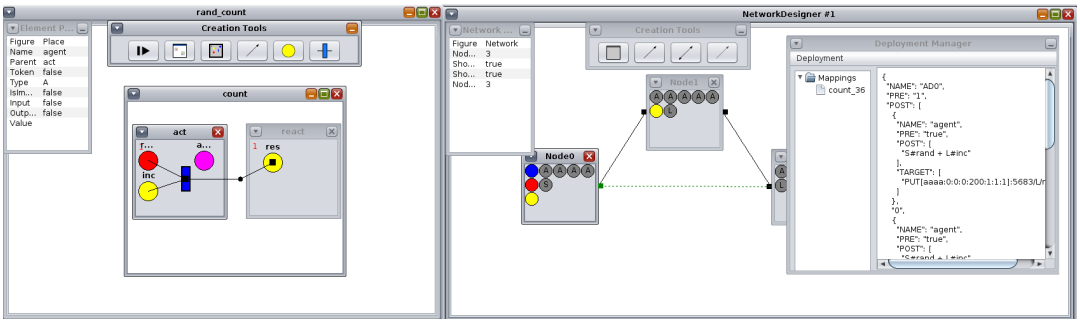


Figure 3.11 – Screenshot of the *emma-design* application.

3.4.4 Network Tools and Connectors

The *emma-network* package provides a set of tools to connect a WSAN to Internet gateways and IT administration tools:

- *emma-network-security* is a REDIS-based application to store AES security keys for the WSAN on a gateway (see Section 4.4.3).
- *emma-network-rndis* and *emma-network-tunslip* allow 6LoWPAN connectivity between the WSAN and Linux. It is used by gateways, as described in Section 3.2.2, or by a COOJA simulator for hybrid simulation, as described in Section 3.4.2.
- *emma-network-snmp* is an Simple Network Management Protocol (SNMP) agent, which monitors the WSAN. It allows classical IT network monitoring tools to monitor an EMMA WSAN.
- *emma-network-proxy* is a Node.js reverse proxy Constrained Application Protocol (COAP)-HTTP on a gateway, whose function is to exchange data between the Internet and the WSAN.

3.5 Summary

This chapter has introduced the major technologies relating to the development of Wireless Sensor and Actor Networks (WSANs) and their integration on the Internet. Since the last decade, numerous studies have permitted the establishment of network protocols in order to address the issue regarding the energy constraint of devices and its impact on the general network infrastructure. The proposed end-to-end infrastructure is based on the standard protocol of 6LoWPAN, which represents a first step towards the IPv6 Internet. The proposal is composed of different network segments, which underlie the necessity to propose hierarchical software architecture given that the direct access of devices from the Internet raises several issues regarding network bandwidth, reactivity and security.

Service Oriented Architecture (SOA) has a strong interest in Home Information System (HIS) because it facilitates the service composition of different appliances in an Responsive Environments (RE). The trivial experimentation between Service Orchestration (SO) and Service Choreography (SC) has concluded that the distributed approach is more efficient in terms of network reactivity that the centralized one. However, the management of a distributed system is a difficult task, which should be delegated to a supervisor located on a gateway. Hence, the data flows of applications are located inside the WSAN, which reduces network congestion around the routers and improves network reactivity and data privacy, whereas their management should be operated from the Internet.

During this thesis, a set of software has been developed in order to design, deploy and execute an Internet of Things (IoT) infrastructure, in which appliances collaborate according to SC applications. These different forms of software permit the deployment of real WSAN infrastructure, its simulation and the design of applications based on SC. Different plug-ins have also been developed to analyze system behaviour during the design, simulation and execution of such applications.

The following chapters in this part of the thesis present the theoretical models, problem formalization and resolution for the design, execution and deployment of SC on a Wireless Sensor and Actor Network (WSAN).

Chapter 4

Active Resource Middleware[1]

L'utopie est le rêve nécessaire et la réalité le défi permanent.

The utopia is the necessary dream and the reality is the permanent challenge.

Daniel Cohn Bendit [CB15]

Contents

[1] Published in Clement Duhart, Michel Cotsaftis, and Cyrille Bertelle. "Wireless Sensor Network Cloud Services: Towards a Partial Delegation". In: *Proceedings of 5th International Conference on Smart Communications in Network Technologies 2014 (IEEE SaCoNeT 2014)*. Vilanova i la Geltru, Spain, June 2014

4.1 Introduction

Wireless Sensor and Actor Networks (WSANs) are physically distributed systems; however, the implementation of their information systems can be centralized or decentralized. In the orchestration approach, central servers receive data from nodes and take decisions to control them. This involves an upstream that collects data and a downstream that manages the nodes. The main advantage of this approach is the easiness of application deployment and extension. Meanwhile, the communications between applications and nodes produce network congestion and response latency in large-scale networks, such as those previously presented. Service choreography involves decentralized architecture in which applications are located onto the nodes. Communications is performed locally between the nodes, which avoid long paths of communications. In turn, the issue is the deployment and execution of such applications on strongly constrained nodes.

The Active Resource Middleware (ARM) is a contribution of Service Choreography (SC) middleware applied on a WSAN. It is an abstraction layer between node services and their data exchanges at the choreography level. Its Resource Oriented Architecture (ROA) [GTW10], as presented in Section 4.2, facilitates the software implementation for motes by providing application interfaces, such as data structure, memory management, resource access, data processing, file systems and security layers. The EMMA middleware is implemented on the Contiki OS and designed to save the memory footprint.

The applications located on the nodes are considered as autonomous services that must be connected in order to design distributed applications through a Web-based choreography language presented in Section 4.3. This design of an SC consists of a configuration of distributed publish-subscribe mechanisms between services located on the nodes. This graphical and flexible high-level language is based on an augmented Petri network model, which exploits its theoretical background in a dynamic system analysis. Section 4.4 considers how they can be easily decomposed in order to design a complex SC.

The SC is performed thanks to reactive agents that model the publish-subscribe mechanisms. They are executed over the ARM in order to transmit requests between the node services. In turn, they can be used to manage the heterogeneity of application protocols with other Constrained Application Protocol (COAP) services. As they are also resources, they can be managed remotely from supervisors, as well as locally by themselves. An agent can create or delete other agents, including itself, which offers the agent the ability to rewrite itself. Hence, the SC can be designed with self-x properties in order to evolve during its execution.

4.2 Architecture

An Environment Monitoring and Management Agent (EMMA) node is an input-output processing module that receives requests, processes them and emits new requests to other nodes. It is composed of an File System (FS), a COAP client and a COAP server. Node functionalities can use or produce any type of data, such as numerical values (for sensors and actuators) and binary streams (for audio players). All of them are similarly processed in EMMA middleware.

4.2.1 System Components

This middleware is implemented on the Contiki OS by a set of standalone module applications, which communicate by event messaging in order to make use of the sleeping mode of microcontrollers. The global architecture is illustrated in Figure 4.1, while the memory footprints are presented in Table 4.1. It consists of:

- *emma-server* - for managing incoming COAP transactions and access rights
- *emma-client* - for forging packets and sending COAP transactions
- *emma-resource* - for storing resource files and managing node services

The interface between node services and the middleware core uses JSON files to allow the EMMA core to be independent of data types. Moreover, microcontrollers have a limited RAM memory (between eight and 16 kB); therefore, all resources are stored and executed from permanent flash memory, which allows large requests to be processed.

All services implement the *emma-resource-services* interface and are managed by the resource file system *emma-resource*. There are three default services: a system configuration interface, a numerical data store and an agent evaluator, which is used for service choreography.

Modules	RAM	Program memory
emma-client	381 B	8,267 B
emma-server	456 B	4,528 B
emma-resource	648 B	4,108 B
emma-JSONparser	0 B	382 B
emma-preprocessor	95 B	4,116 B
emma-service-system	60 B	2,845 B
emma-service-numeric	10 B	576 B
emma-service-agent	210 B	6,586 B
Total	1.9 kB	31.4 kB

Table 4.1 – Memory footprints of EMMA modules on the Contiki OS.

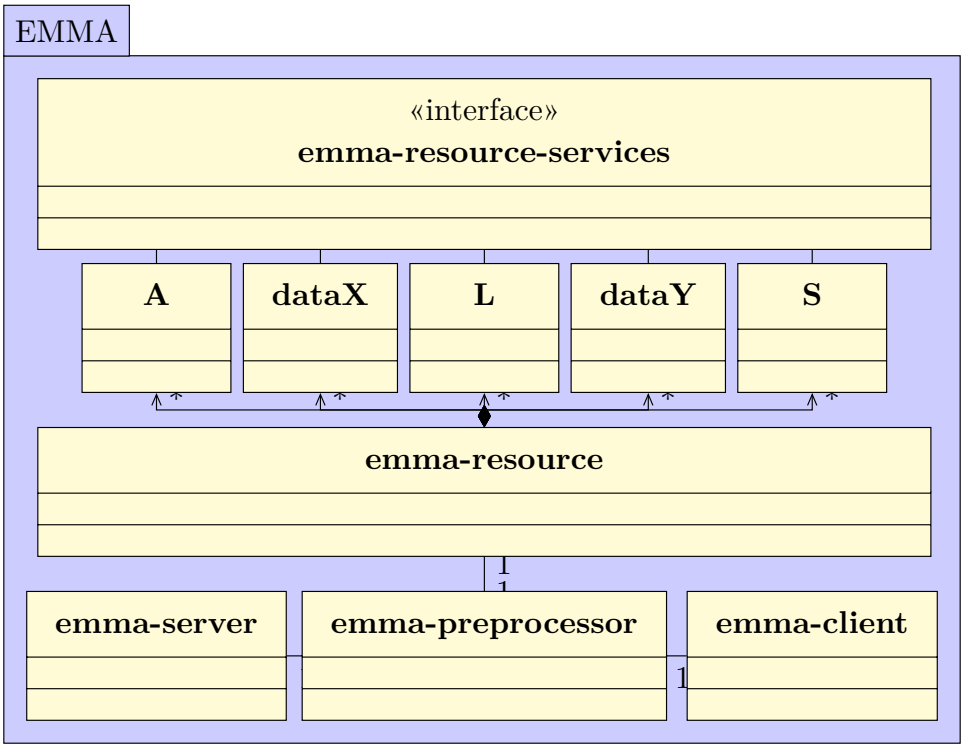


Figure 4.1 – EMMA middleware UML diagram.

4.2.2 Resource File System

While all resources have a type according to their responsible service, they are stored in permanent memory as JSON text files by the resource file system *emma-resource*. They are accessible by COAP requests with their Unified Resource Identifier (URI). To reduce memory costs, they are mapped in permanent memory by a tree-like organization to avoid being stored in an index table. Therefore, the resource file system has to navigate between resource files, as shown in Figure 4.2.

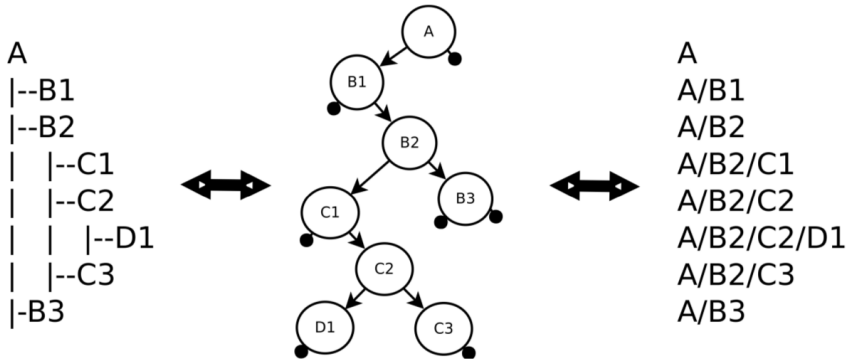


Figure 4.2 – EMMA middleware resource file system schema.

4.2.3 COAP Web service Interface

EMMA middleware is REpresentational State Transfer (REST) architecture in which COAP requests change resource states managed by node services. Its Web service implementation uses the Erbium COAP engine [KDD11] of the Contiki OS, as illustrated in Figure 4.3. The COAP protocol is an HTTP-like protocol, which is basically composed of a URI to define the resource target, a payload containing data and a method to perform an operation. With the right access, a request can read *GET*, modify *PUT*, create *POST* or delete *DELETE* a resource. Each service has its own processing callbacks to change JSON resource files stored in the permanent memory according to the request method. The IPv6 LoW Power Wireless Area Networks (6LoWPAN) protocol does not include a fragmentation mechanism that is the responsibility of the application layer. The *block-wise transfer* mechanism of the COAP protocol is designed to save memory buffer. Requests are processed online by a block of 32 bits to avoid requests being assembled in a large buffer. Moreover, the use of small blocks facilitates network packet scheduling at the Media Access Control (MAC) layer, such as in the use of Time Division Multiple Access (TDMA) protocol.

Finally, the set of resource data forms node context, which evolves according to processed requests. Node behavior depends of this context state to send internal or external requests in the same way as a cellular system.

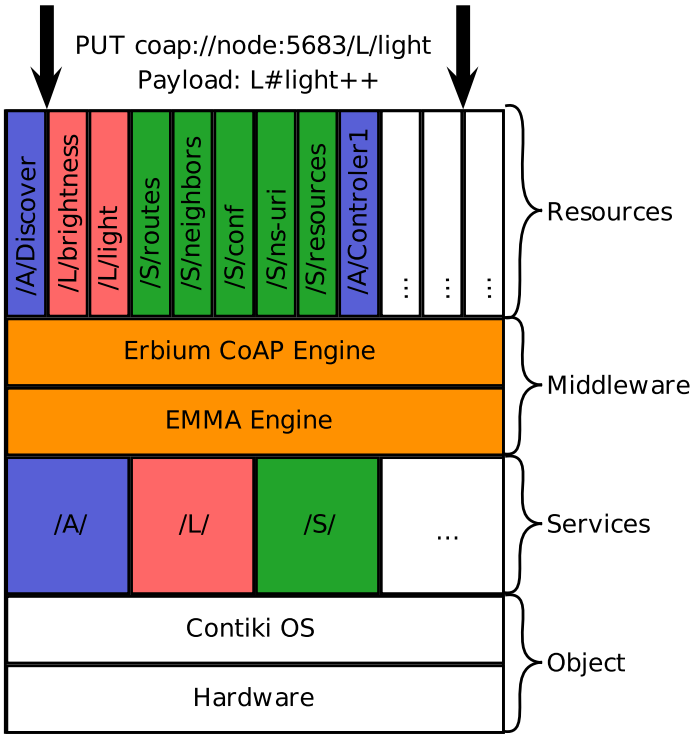


Figure 4.3 – EMMA node overview schema.

4.3 System Dynamic

Node services are provided to EMMA middleware through their resource publishing. By default, there are three types of resource: system, local and agent, which are presented in Section 4.3.1. The system dynamic is achieved by the agents, which generate COAP requests according to the current resource state of their hosting node. As requests change the internal resource state of the target node, a distributed computation flow over the WSAN is executed when an agent triggers other agents, as detailed in Section 4.3.2. Lastly, Section 4.3.3 presents an EMMA graphical model of an Service Choreography (SC) based on a Petri network.

4.3.1 Basic Services

The following services are implemented in the same way as any service and can be removed from EMMA middleware on the Contiki OS compilation to build simple Web services. However,they are necessary for service choreography over the Wireless Sensor and Actor Network (WSAN).

4.3.1.1 Local Service

The local service provides simple numerical data resources to store values. Agents require them during computation flows to store cache values, parameters or data exchanged between each Service Choreography (SC).

4.3.1.2 System Service

The system resources detailed in Table 4.2 encapsulate data or configuration parameters for the Contiki OS or EMMA middleware. They are used by agents to change system configurations or access sensor and actuator drivers.

Resource URI	Methods	Description
/S/conf	GET/PUT	EMMA & Network configuration
/S/time	GET	Node uptime
/S/rand	GET	Random number
/S/energy	GET	Energy battery level
/S/ns-uri	GET	URI to RDF node description
/S/routing	GET/PUT/POST/DEL	Routing tables
/S/neighbor	GET	List of neighbor nodes
/S/resources	POST/DELETE	List of all resources

Table 4.2 – List of system resources on EMMA node.

4.3.1.3 Agent Service

An agent is a configuration file for the augmented publish-subscribe mechanism, which specifies 'what, when and where' to send a COAP request. An EMMA agent a is a JavaScript Object Notation (JSON) file stored on node n that contains a set of resources denoted X_n. It is composed of three elements:

- **A Boolean activation function** $PRE_a(X_n)$
  Example: $/L/threshold < /S/brightness$
- **A list of denoted resource targets** Y_a
  Example: *PUT[IPv6]:port/S/light*
- **The associated payloads** $\forall y \in Y_a,\ POST_a^y(X_n, y)$
  Example: *{'value':'/S/light++'}*;

When Boolean activation function $PRE_a(X_n)$ is true, it sends COAP requests to target resource $y \in Y_a$ according to $POST_a^y(X_n, y)$, as summarized in Eq. (4.1).

$$\text{If } PRE_a(X_n) : \forall y \in Y_a, y \underset{method}{\longleftarrow} POST_a^y(X_n, y) \tag{4.1}$$

As illustrated in following agent examples, the PRE field specifies the firing condition in order to send a request to each resource target stored in the $TARGET$ field. A target is defined by a COAP method (GET/PUT/POST/DELETE) and a URI ([*IPv6*]:*port*/*resource*). The payload stored in the $POST$ field for each resource target is a template file, which is processed to replace variables with their resource value. If the payload contains mathematical operations, they are performed before transmitting the request. This payload can contain unresolved variables, which are replaced by the resource values of the target node.

4.3.1.3.1 Publish-Subscribe Agent The JSON Agent 4.1 is hosted on a brightness sensor and sends orders to a light to increase its value before transmitting the measured brightness to a database every 10 seconds if it is lower than 50.

```
{
   "NAME": "AgentSensor",
   "PRE": "L#brightness<50 && S#time%10 == 0",
   "POST": [
        "{'value':'R#light+1'}",
        "L#brightness"
    ],
    "TARGET": [
        "PUT[aaaa::2]:5683/L/light",
        "PUT[aaaa::1]:5683/database/light"
    ]
}
```

JSON Agent 4.1 – Example of a periodic publish-subscribe agent.

4.3.1.3.2 Composed Agent JSON Agent 4.2 locally contains and deploys other agents before deleting itself, following which the deployed agent transmits the list of resources to a supervisor.

```
{
    "NAME": "RelayAgent",
    "PRE": "A#RelayAgent",
    "POST": [
        {
            "PRE": "S#rand%5==0",
            "POST":[
                "{'resources':S#resources}"
            ],
            "TARGET": [
                "PUT[aaaa::3]:5683/L/Example"
            ]
        },""
    ],
    "TARGET": ["POST[aaaa::2]:5683/A/TargetAgent",
              "DELETE[0::1]:5683/A/RelayAgent"]
}
```

JSON Agent 4.2 – Example of a relay agent.

4.3.1.3.3 Self-X Agent The JSON agent 4.3 contains a reference to itself in the PRE condition, the POST field and the TARGET tables. It is a *self-deployer agent*, which sends itself to all its neighbors when it arrives on a node. It then locally deploys an agent to push the list of resources before deleting itself.

```
{
    "NAME": "DiscoverDeployer",
    "PRE": "A#DiscoverDeployer",
    "POST": [
        "A#DiscoverDeployer",
        {
            "PRE": "S#rand%5==0",
            "POST":["{'resources':S#resources}"],
            "TARGET": ["PUT[aaaa::1]:5683/NetworkInfo"]
        },""
    ],
    "TARGET": [
                "POST[ff02::2]:5683/A/DiscoverDeployer",
                "POST[0::1]:5683/A/DiscoverNotifier",
                "DELETE[0::1]:5683/A/DiscoverDeployer"
              ]
}
```

JSON Agent 4.3 – Example of a self-deployer agent.

4.3.2 Computation Flows

A computation flow is a set of agent activations produced by a domino effect. This occurs when agent activations depend on the target resources of the other ones. An illustration of this interaction chain is illustrated in Figure 4.4. Requests are emitted by service agents (blue resources) to change another remote or local resource, which can trigger another agent etc. There is no difference for the Active Resource Middleware (ARM) between local and remote resources due to the middleware abstraction layer between the Service Choreography (SC) and the nodes. A tuple space for agents is provided through the set of all available resources. The agents are executed independently over the Wireless Sensor and Actor Network (WSAN), although they are synchronized by events on their sensitive resources. This event chain moves on the event-driven kernel of the Contiki OS. Therefore, concurrent access to a resource cannot appear inside nodes, while parallel execution can only be used over several nodes. New events are stored in an event pool and processed when the current computation flow of the node is terminated. This execution constraint allows computation flows to be managed by the structural design of the Service Choreography (SC), such as in the case of mutex, semaphore, switching, parallel execution, and synchronization for load balancing, memory management and task scheduling.

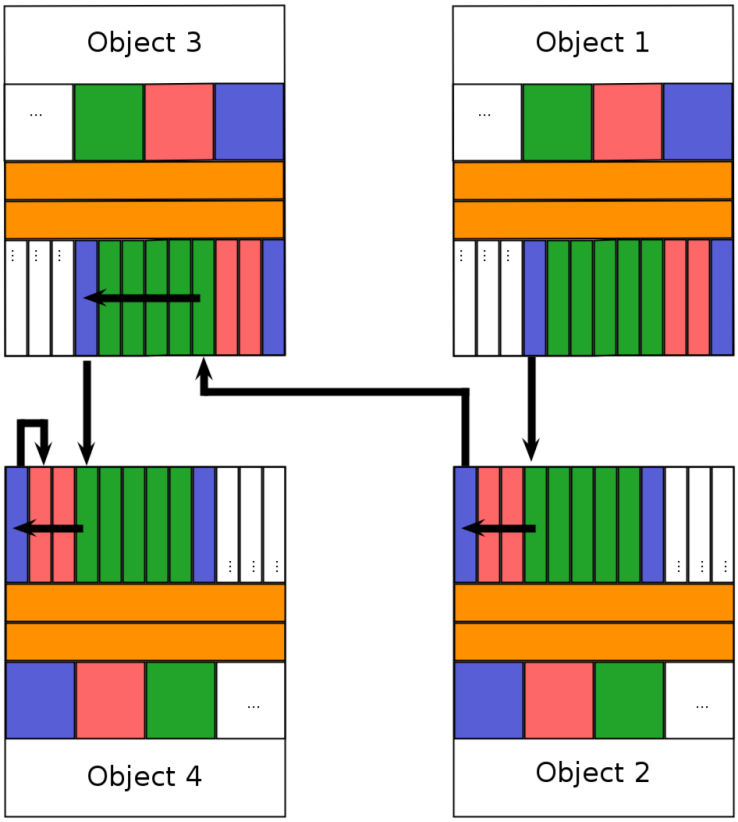


Figure 4.4 – Illustration of a computation flows by an event chain of requests.

4.3.3 Graphical Model

The EMMA graphical model uses a continuous [SR02] and numerical [JL93] Petri network [GS07] to design and analyze the logical behavior of the Service Choreography (SC). It models resource entities and system dynamics through agent activation conditions and requests.

In EMMA, the SC is modeled by an augmented Petri network in which requests are referred by tokens, agents by transitions and resources by places. A transition is fired if the following two conditions are satisfied: (1) a token appears in any input place and (2) the agent's Boolean condition is returned as true. This transition activation produces a token for each output place and changes target resource values in the corresponding preprocessed payload. Agents are also resources then, with each transition associated with a place. If this kind of place is deleted, the associated transition is destroyed, whether for creation or editing. Therefore, this Petri network model is dynamic and can change during its execution. This model illustrated in Figure 4.5 allows SC to be simulated independently of its execution supports. Its behavior is validated by classical algorithms found in the literature on Petri networks concerning safety, liveness, reversibility, determinism, termination, output correctness and input dependence [BWWH91]. Moreover, classical patterns can be reused directly, such as in cases of sequence, parallel splits, synchronization, exclusive choices, simple merges, multi-choices, structured synchronizing merges, multi-merges, arbitrary cycles and multiple instances [Rus+06].

This Service Choreography (SC) computes the differential value $p1(t) = p0(t-1) - p0(t)$. Agent t0 computes the differential value between a new value of p0 and its previous one stored in p2 by the agent t1. If p1 reaches the value 50, the agent t2 is fired and uninstalls the SC including itself.

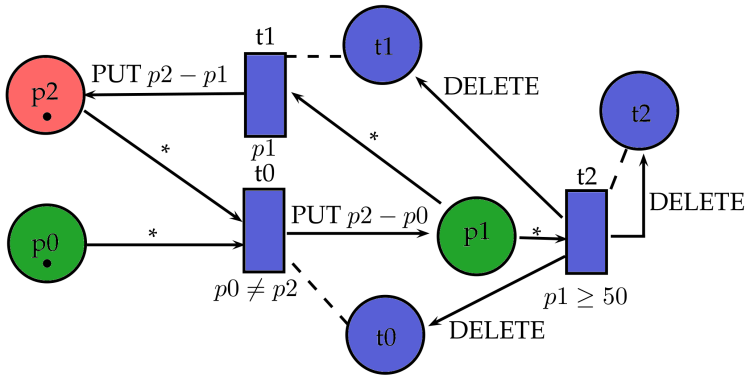


Figure 4.5 – Example of an EMMA graphical model created by a Petri network.

This Petri network adaptation, used in *emma-design-tools*, allows application behavior to be simulated. According to a set of input resource values, the final state of the Petri network is stored in a SC profile.

4.4 Service Choreography

4.4.1 Hierarchical Composition

A Responsive Environments (RE) is composed of several appliances that require a different SC in order to manage them. Moreover, each SC can require information to be exchanged with other ones; for example, an SC for a control loop of actuators should communicate with the SC responsible for energy management. Hence, the design of a general SC is not suitable, given that its general Petri network ought to be very complicated and, therefore, unmanageable.

The hierarchical composition of a Petri network is used in order to design and validate each different SC separately. Each Petri network for an SC has a set of input and output places in order to abstract the different subplaces and transitions into a general transition, as illustrated in Figure 4.6. The different SC are then connected together through their input and output places, as investigated in Kaaniche et al. [Kaa+12]. The general SC behavior is evaluated by analyzing those of the different sub-SC. A behavior is defined as a set of output place states according to a set of possible input stimuli. Hence, at the resource level (detail view 0), the Petri networks are evaluated by classical algorithms found in the literature, in order to build the transition function between input and output places. Then they are abstracted due to a general transition. This approach significantly simplifies the design and validation of an SC, in the same way as its sub-SCs are considered as black boxes, which can be reused with an already validated behavior. Hence, the validation process of SC behavior is resumed by correspondence analysis between the behavior transitions of its sub-SCs.

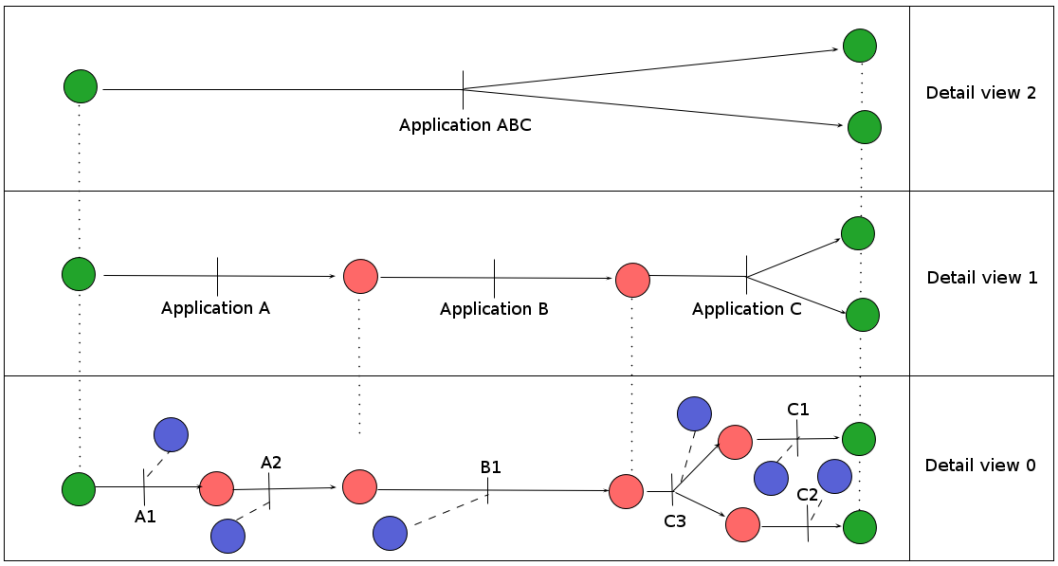
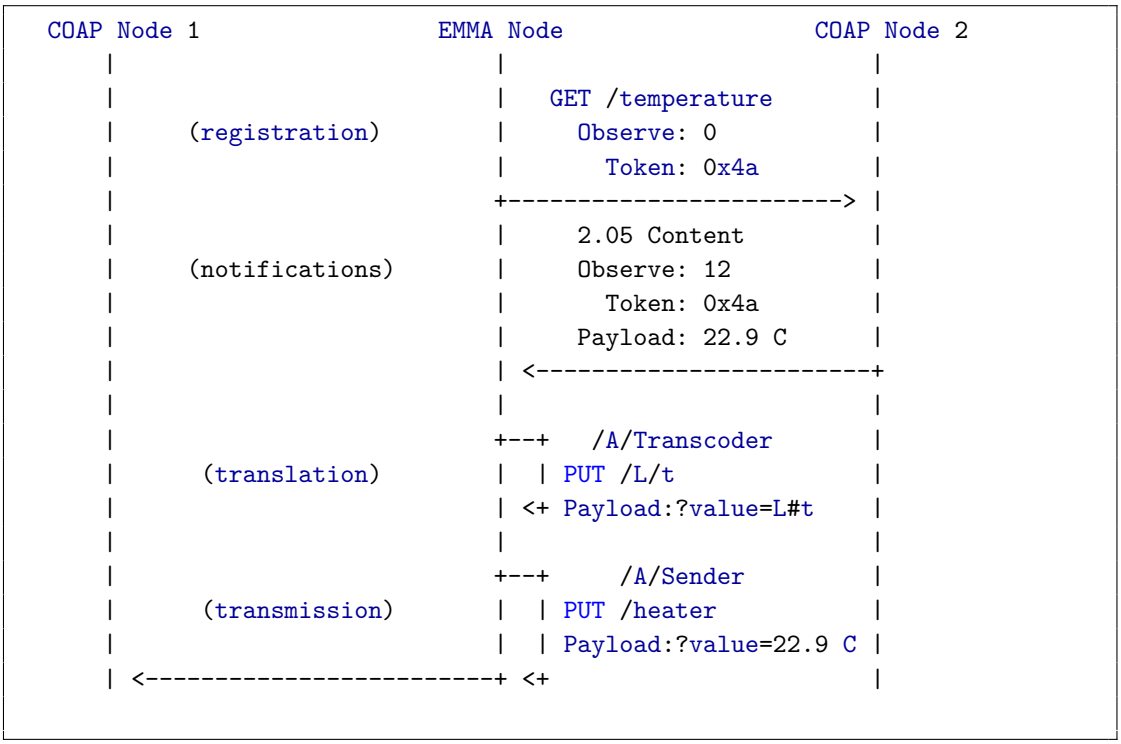


Figure 4.6 – Illustration of Service Choreography (SC) composition.

4.4.2 Web Service Heterogeneity

Internet of Things (IoT) devices are provided by several different manufacturers with their own information systems. The network protocol has been standardized, but the application layer has not, given that future usages are unpredictable. Hence, the management of application heterogeneity at the choreography level is required. Traditionally, this issue is managed by a proxy server, which translates the requests. In this sense, the COAP has an internal static publish-subscribe mechanism, which allows the proxy to collect data from all nodes in order to ensure translations.

The EMMA middleware allows this translation to be operated directly by the agents. As their requests are fully specified through the *POST* and *TARGET* fields, they can natively send any kind of payload using any URI. For example, if remote middleware uses the Extensible Markup Language (XML) formatting language, the *POST* field of the agent should contain the required XML template. Moreover, the agent can generate requests to subscribe to the COAP observer mechanism in order to collect the data from other middleware. The combination of these two types of agent allows an EMMA node to ascribe data to other middleware in order to generate COAP requests for others, as illustrated in the sequence diagram 4.4.



Sequence Diagram 4.4 – Sequence diagram of an SC for heterogeneity management.

4.4.3 Name Space Security

Data privacy and system security are important challenges in the IoT. The decentralized aspects and the possibility for different service providers to deploy SCs requires a new approach towards protecting the system. In the EMMA framework, the SCs have individualized protection through different security partitions in the namespace. An Name Space (NS) is a set of resources that shares a same security partition. Access is protected by an AES 128 bits security key *NSKey*, which is used to encrypt communications between the resources. Each secured agent adds an additional field *NS* to the request in order to indicate the *NSKey* ID used for payload encryption. The target resource then uses its corresponding *NSKey* located at the URI */L/X/aes-keys* to decode the payload. As a resource can be shared between several SCs in cases of composition, they have several *NSKeys*. The deployment process consists of the creation of the resources in which are included the different required *NSKeys*. This process is operated by the supervisor in relation to the different nodes by secure communications. Each node is accessible thanks to a *RootKey*, which is deployed manually by the supervisor (by Radio Frequency IDentification (RFID) or by writing it into the permanent memory during the compilation of node software). The *RootKey* is used by the supervisor to encrypt the payload used for the deployment of SC resources with their *NSKeys*. Figure 4.7 summarizes this chapter with three nodes, which have different resources that are accessible according to their secured namespace containing an SC implemented by hierarchical composition of three other ones that are modeled by Petri networks.

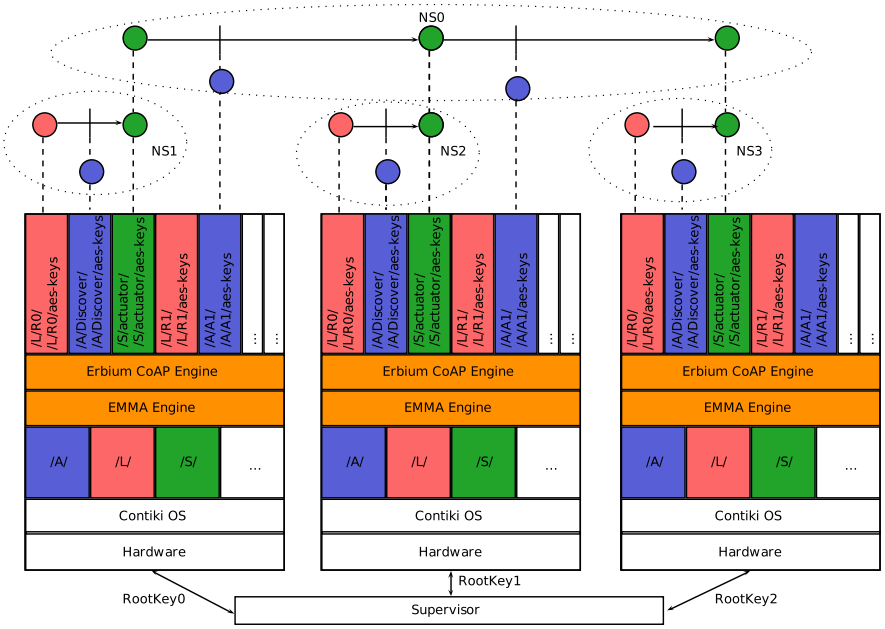


Figure 4.7 – Overview of the EMMA secured architecture.

4.5 Summary

This section has presented the Active Resource Middleware (ARM), which is the engine core of an EMMA system. It is an abstraction layer between the appliances with the network and a resource tuple space. These resources are provided by local node services and connected in order to form an Service Choreography (SC). The particular agent resources are used to model and deploy the communication rules by distributed publish-subscribe mechanisms. In addition, this proposed model offers the possibility of adding self-x properties to an SC, such as self-deployment. The SC design is totally abstracted to a Wireless Sensor and Actor Network (WSAN) by an augmented Petri network, which is used to analyze SC behavior before being deployed on appliances. The management of heterogeneity with other COAP middleware is natively addressed due to the proposed agent model in a fully distributed way. In addition, the protection of data privacy is based on a proposal of secured namespaces in order to partition the SC, which could be designed by different service providers.

The different classical challenges for WSANs are summarized in Table 4.3 with their referring components in relation to an EMMA framework. Those challenges concerning the network are not addressed, such as the EMMA framework, which focuses on the application layer. However, the quality of service should be possibly considered through a priority routing engine in the Contiki OS event stack. An additional field should be added to the agents in order to define the priority of the computation flows. Hence, the appliances should be able for them to be scheduled according to their priority.

Layer	Challenges	Entities
Hardware	Abstraction	Contiki OS
	Energy Efficiency	Mapping Deployment
	Hardware Constraints	Lightweight Services
Network	Addressability	6LoWPAN
	Routing Protocol	RPL
	Dynamic Topology	None
	Quality of Service	None
Application	Adaptability	Resource tuple Space
	Programming	Petri Network Model
	Knowledge	Petri Network Patterns
	Scalability	Choreography Composition
Data	Security	Name Space
	Heterogeneity	Agent

Table 4.3 – WSAN challenges addressed by active resource middleware

Chapter 5

Service Choreography Deployment[1]

Le tout est autre chose que la somme de ces parties.

The whole is different than the sum of its parts.

Kurt Koffka [Kof77]

Contents

[1]Submission in Clement Duhart, Pierre Sauvage, and Cyrille Bertelle. "A Resource Oriented Framework for Service Choreography over Wireless Sensor and Actor Networks". In: *Submission in International Journal of Wireless Information Networks (IJWI)* ()

5.1 Introduction

The previous chapter presented models, mechanisms and features of an Active Resource Middleware (ARM) to build a Service Choreography (SC). From the design to operational deployment, the different steps are outlined in Section 5.2. Given that Wireless Sensor and Actor Networks (WSANs) have several life cycles, a deployment process should be adapted. At the first time of starting, all node and network resources are available for the deployment process, which is no more possible during the running of services. Consequently, three deployment processes are considered according to their properties. Direct deployment between supervisor and nodes is only used for punctual mapping corrections. Meanwhile, composed deployment facilitates a SC upgrade by delegation to the remote nodes. Finally, at the initialization step, all WSANs can be used by self-deployment to map everything, everywhere.

The genericity of the EMMA resource model allows for SC mapping to be automatic. According to the node services and their resource hosting capacity, SCs are distributed over entire WSANs to minimize network communication load. Several constraints are considered in the course of the mapping process regarding node, network and application specifications. Its mathematical formulation is successfully resumed by the pseudo-Boolean optimization (PBO) problem in Section 5.3, which allows any PBO solver to be used. Finally, Section 5.4 presents results about the complexity of the mapping problem according to number of resources and WSAN sizes. Analysis shows that, in a large-scale network, it is preferable to use several deployment partitions instead of one big mapping process. Evaluation of deployment agents shows their execution time on an ARM for comparing them and proposing an example of best practice for SC deployment.

5.2 Network Mapping Process

The Service Choreography (SC) mapping process associates an empty resource space for each place of the SC. Such an association is performed according to the resource type and the hosting capacity of nodes for each service. The efficiency of a mapping process is defined by a cost function in Section 5.3, which minimizes, by default, the network communication load. This mapping process is composed of three specification stages: the *functional design*, the instantiated graph and the network mapping, as illustrated in Figure 5.1 and discussed in the following sections.

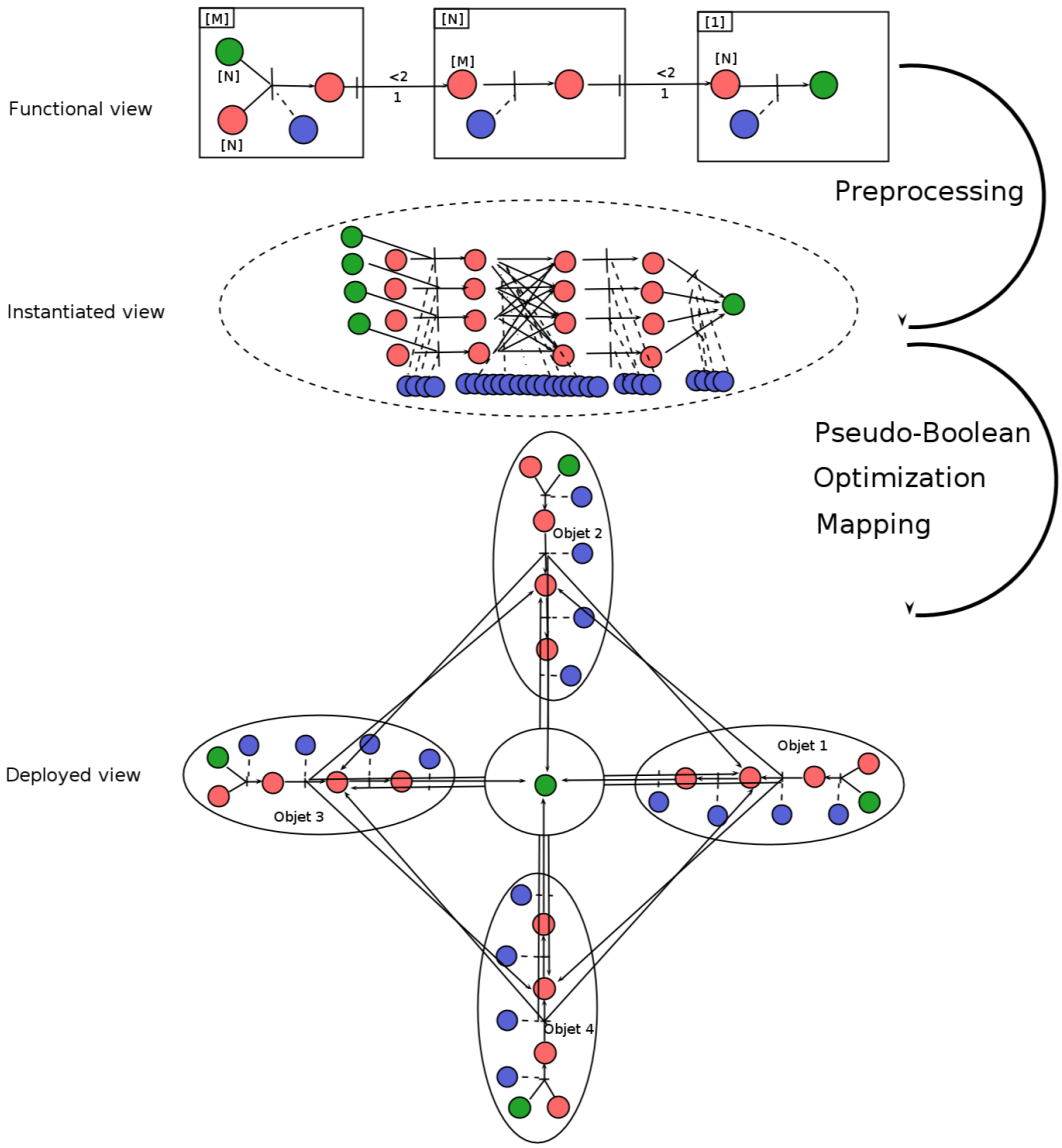


Figure 5.1 – Network mapping process for a multi-layer perceptron.

5.2.1 Stages Overview

5.2.1.1 Functional Design

The *functional design* establishes SC templates, which could be deployed over different Wireless Sensor and Actor Networks (WSANs). They are abstracted from the WSAN and node services as a result of the previously discussed Petri network. A Petri network in a SC connects a set of input places to output places in order to define their general transition functions. The input and output places are characterized in order to define compatible resources of node services, which could be used for this SC.

The different input-output resources produced by the node services are connected through agent resources A (modeled by transitions) in addition to temporary resources L (corresponding to places). This specification stage introduces the concept of scope, which manages structural dependencies, as illustrated at the top of Figure 5.1. The term *scope* alludes to the mapping specification, which refers to a group of resources that must be mapped on the same node. For implementation reasons, a transition resource a must be on the same node n, which the resources of its input places X_n required due to its activation function $PRE_a(X_n)$ in the Petri network. Otherwise, for efficiency reasons, such high frequency communication exchanges, a SC designer would seek to force several resources to be located on the same node. The mapping can be constrained by scopes because of the following:

- *Scope multiplicity:* A scope that requires several identical scopes is connected by a link of multiplicity M. For example, an agent of data aggregation requires M values produced by the sensor resources. Hence, the scope containing the agent is linked to the scope, which models the sensor resources by a multiplicity parameter equal to M.

- *Scope dependency:* The SC design is operated independently of the target network. However, it is possible to constrain the resource mapping in respect of network constraints, such as the maximal number of communication hops between two scopes.

5.2.1.2 Instantiation and Simulation

The *instantiation* step generates the complete graph of a Service Choreography (SC) based on the set of *functional design* templates for a target of WSANs. Based on the network information collected by the service discovery mechanism (see 4.2), all SC templates are duplicated for each couple of input-output scopes in order to satisfy their multiplicities. All final resources are defined and linked to their value range specifications. If a resource in a SC template has a different value range to those in a WSAN, an intermediate agent is automatically placed to adapt value

range. During this step, the SC designer will have to check and correct possible mistakes produced by this automatic process performed by *emma-tools-design*.

The *simulation* of the generated SC must validate its behavior. As previously discussed, the simulation of this final SC for each possible initial marking on input places produces a large set of tests. The use of composed general compositions, which have already been simulated, drastically reduces the number of transitions. All places and transitions contained in a uncorrected template is resumed by their composed transition. Based on their value ranges on output places, the input ranges of the next connected transition are reduced. Hence, all composed transitions, which are connected without an intermediate transition being manually added, can be reduced to a single one with a fixed value range of its input and output places. Even if the set of possible initial markings on input places is large, the simulation is resumed to simulate the manually added transition with the composed transitions.

Given that the *deployment process* is also a Service Choreography (SC), as presented in the previous chapter, it should be validated in the same way. The *deployment process* requires some resources, which have to be reserved during the mapping process. Therefore, the SC of deployment is added into the instantiated Petri network, although it uses some temporary resources.

5.2.1.3 Network Mapping

Network mapping determines the list of possible matchings between the instantiated SC graph and the resource distribution on the ARM under the *functional design* constraints. In addition, this process builds the *composed agents*, which install the resources along a specified path of deployment (by default, it follows the network routing path from the supervisor: the 6LoWPAN edge router). The mathematical formulation of this mapping problem is resumed by a PBO problem detailed in Section 5.3.

The process of deployment is based on a set of *composed agents* (see 4.2 defined in Section 4.3.1.3). They are deployed on a node in order to install the resources on the hosting node or on the neighbors. In addition, they send other agents of deployment to other nodes, such as Matroska containers. Given that an agent is also a resource, the number of resources that it can install is limited by its maximum size. Hence, several *composed agents* of deployment are generated according to the hosting capacity of nodes. The mapping of the deployment process determines the distribution of the deployment agents over the WSAN and their composition. This process is performed by backpropagation along the defined deployment path in order to include the deployment agents with the others until they reach their maximum size. This process is reiterated until the coverage for all of the resources to deploy is reached, after which each deployment agent is sent to its corresponding node.

5.2.2 Dynamic Deployment

Dynamic deployment is composed of two stages. Firstly, deployment agents are diffused over an Active Resource Middleware (ARM) to deploy locally resources. Then, deployment agents are deleted to allow only service resources. The deployment agents, called Dynamic Network Agents (DNAs), change the dynamic of WSAN behavior, which is produced by a set of SC agents called Residual Network Agents (RNAs). Both these categories are defined to facilitate the organization, management and understanding of the mapping dynamic.

5.2.2.1 Residual Network Agents

Residual Network Agents (RNAs) are the set of agents deployed by a Dynamic Network Agent (DNA). They can be considered, from the node perspective, as a set of interaction rules that define its behavior; from a system perspective, they can be considered as part of a Service Choreography (SC). The RNA produces the static system dynamic by diffusing computation flows over the ARM when an event occurs on input places. These agents do not modify the system dynamic by modifying other agents.

5.2.2.2 Dynamic Network Agents

Dynamic Network Agents (DNAs) are responsible for updating the system dynamic by modifying other agents. They are used during the installation procedures of a Service Choreography (SC), as well as by a SC with self-x properties. During a deployment, they are *composed agents* that contain a set of rules to install the resources (including other agents) on the nodes. They then delete themselves when the deployment process is terminated.

5.2.2.3 Self-X Agents

After the deployment process, the DNAs are normally deleted as they have deployed all the resources. The agent model, however, offers the possibility of allowing them in standby. Instead of starting the deployment under the condition that the agent has arrived on a new node, the field $PRE_a(X_n)$ can be defined in order to match it with a particular resource context of the node. The agent then deploys the resources when they are required. Hence, an agent can contain different SCs, which are deployed according to the online node requirements.

The first use example of this approach involves the establishment of the *self-repair* property. If a node resource is accidentally deleted, a sensitive DNA in its presence could deploy it again. Another example involves the *self-deployment* of a common SC, such as a *network and service discovery* mechanism. A DNA can contain a set of SCs, which must be deployed on all nodes. Hence, it broadcasts itself to all nodes in order to install the required resources for a common SC on them. The new connected node then automatically receives a basic SC.

5.2.3 Deployment Process

The deployment process deploys a different SC from the supervisor to the nodes. In the literature, different middleware perform this operation in a peer-to-peer dynamic between the supervisor and the nodes. However, in a large-scale WSAN, the deployment process produces significant network congestion around the routers, such as that evaluated in Chapter 3. The EMMA framework proposes two other strategies through the use of Dynamic Network Agents (DNAs): the *deployment container* and *self-deployment* in addition to *direct deployment*.

5.2.3.1 Direct Deployment

Direct deployment sends agents directly to each node from the supervisor. In a large-scale WSAN, this approach is not suitable when the supervisor is far away. However, as presented in Chapter 3, the supervisor can be localized on a mobile device. Hence, this communication path can be reduced by physically moving the supervisor on the region of interest. A technician can then apply local corrections with their tablet.

5.2.3.2 Deployment Container

A *deployment container* uses agents to carry other agents. These DNAs, which are illustrated in Figure 5.2, are sent to nodes and deploy locally required resources. Other DNAs can be contained within them and sent to other locally accessible nodes. This composition of a *deployment container* produces a deployment chain, such as in the case of Matroska containers. However, the overhead of this kind of DNA is very important, as evaluated in Section 5.4.

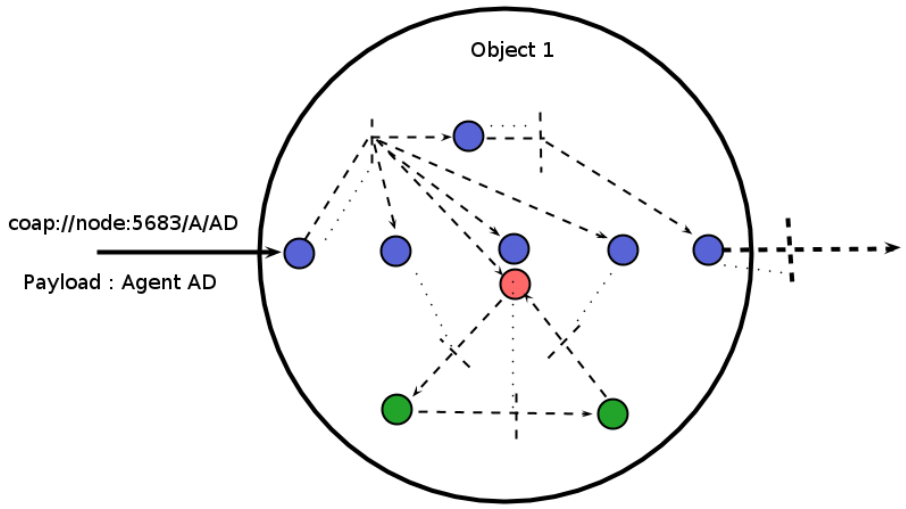


Figure 5.2 – Petri network of a DNA deployment container.

5.2.3.3 Self-Deployment Container

A *self-deployment container* is a *deployment container* that refers to itself in *PRE*, *POST* and/or *TARGET* fields; see Agent 4.3. Being sensitive to itself in the *PRE* field, it duplicates itself on other nodes on arrival. With the possibility to refer to itself in the *POST* field, it can update its definition during its copy process, which offers evolving possibilities. If this DNA deletes itself after copying itself in the *TARGET* field, however, it is a *mobile agent*.

At first deployment, all nodes have to receive their resources, while WSANs are not used by any service. Therefore, the use of a flooding technique to reach all nodes is possible. This DNA embeds all Service Choreographys (SCs) for deployment on the WSANs. When it arrives on a node, it sends itself to all the neighbors before deploying local resources. This initializer agent is built to be sensitive according to the */s/ns-uri* resource, which specifies the node type in order to only deploy the required resources for this node.

Moreover, this DNA can contain other DNAs, which can be used to deploy *self-repair agents* or other *deployment containers* responsible for a particular SC deployment. For example, the Agent 4.3, defined in Section 4.3.1.3, is a *self-deployment container*, as illustrated in Figure 5.3, that is automatically deployed on new neighbor nodes in order to install a set of Residual Network Agents (RNAs) with one local resource.

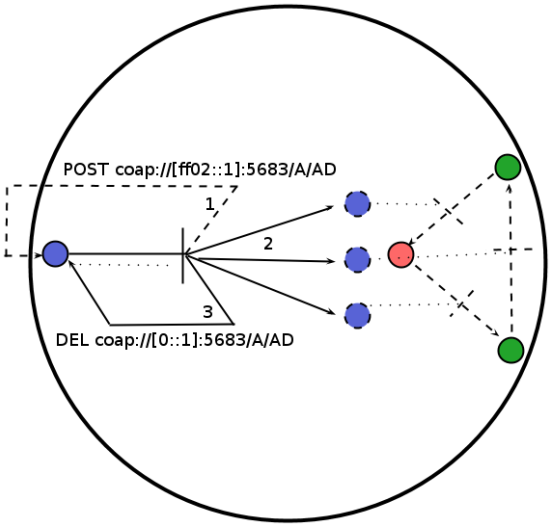


Figure 5.3 – Petri network of a DNA self-deployment container agent.

There are a lot of possible deployment strategies with such DNAs. Some of them can be used for initial deployment, others to install local SCs and, finally, others to deploy automatically minimal SCs, such as a discovery service and a system logger. Best practice is proposed at the end of this chapter.

5.3 Theoretical Background

The application mapping problem determines an efficient distribution of resources on the nodes to minimize communication load. Based on the functional constraints, the network topology and the hosting capacities of nodes, the instantiated graph must be mapped over the WSAN with its deployment process.

5.3.1 Model Definitions

5.3.1.1 Network

A WSAN is composed of a set of nodes $N = \{n_1, n_2, ...\}$ modeled by a distance matrix D, in which $d(n_1, n_2)$ represents the cost metric between n_1 and n_2. By default, the cost metric is determined by the routing algorithm according to the number of hops between two nodes. It can model any others parameters, such as the bandwidth, the link quality or an aggregation of them. If, and only if, the communications are bidirectional, $\forall n_1, n_2 \in N : d(n_1, n_2) = d(n_2, n_1)$. Due to node memory limitations, the routing tables of nodes are partial and a node n_1 can be unreachable from n_2, then $d(n_2, n_1) = -1$.

5.3.1.2 Resources

The resource r is an allocated space of memory on a node associated with a unique access path $URI(r)$, defined by */type/resource_name*. Each resource has a type $j \in T$ corresponding to its service ($T = \{\text{A, S, L, ...}\}$; see Section 4.2.3). $\forall n \in N$, $\forall j \in T$, R_n^j defines the set of resources of type j on node n. R_n then denotes the set of all resources on node n, $R_n = \bigcup_{j \in T} R_n^j$.

Definition 5.1.
$\text{card}_j(n)$ refers to the maximum number of resources of type j on node n, which is defined during the software compilation. Therefore, it is considered fixed for the mapping process. ■

5.3.1.3 Scopes

The scope s is a set of places P_s that must be mapped on the resource spaces of the same node. M_s refers to the multiplicity of scope s (see Section 5.2.1.1). The set of scopes $S = \{S_1, ..., S_m\}$ represents all the SCs to be deployed on the WSAN. Several scopes can be mapped onto the same node, as long as the node capacities are sufficient. Two scopes are considered linked if at least one place of the first scope interacts with one of the second. A communication a is modeled by a data exchange function $f_a(t)$ and a weight w_a, which represents the number of packets required to transmit payload. As the function $f_a(t)$ cannot be assessed a priori, we will instead consider an estimated frequency of f_a.

The set of all communications from scope s_1 to scope s_2 is denoted by A_{s_1,s_2}, as per the sum of all place communication costs defined in Eq. (5.1) between scopes s_1 and s_2. Communications between two scopes are generally asymmetric; hence, $c(s_1, s_2) \neq c(s_2, s_1)$.

$$c(s_1, s_2) = \sum_{a \in A_{s_1,s_2}} c_a = \sum_{a \in A_{s_1,s_2}} f_a \times w_a \tag{5.1}$$

5.3.1.4 Places

The place $p \in P_s$ is a requirement defined by an access path of $URI(p)$, such as */type/place_name* for the mapping of scope s on a node. There are two subsets, such as $P_s = \bigcup_{j \in T} \left\{ P_s^j \bigcup \dot{P}_s^j \right\}$.

- $\forall j \in T$, P_s^j defines the set of places that requires an available resource space of type j on the node. *Example:* The place $p \in P_s^L$ requires an empty resource space of type L on the node.
- $\forall j \in T$, $\dot{P}_s^j$ defines the set of places which requires a resource of type j on the node. *Example:* The place $\dot{p} \in \dot{P}_s^L$, specified with the $URI(p) = /L/temp$, requires a resource with the same URI on the node.

5.3.2 Problem Formulation

5.3.2.1 Knapsack Problems

The formulation of a SC mapping problem is composed of two variants of the knapsack problem:

- *Multiple Knapsack Problem (MKP)*: Each node is considered to be like a knapsack in which places should be mapped according to the node capacity.
- *Multiple Choice Knapsack Problem (MCKP)*: The nodes have a finite partition for each resource type. The number of elements by type is limited on each node because of node's hosting capacity.

Finally, the problem formulation can be summarized thus:

How can scopes, which require different numbers and types of places over a set of nodes and which do not have the same hosting capacities in terms of resource type and memory size, be mapped in order to minimize the network load?

5.3.2.2 Service Choreography Mapping

Definition 5.2.
The operator $size()$ is defined as follows:

- $size(r)$ refers to the memory space used by the resource r.
- $size(R)$, $R \in \{R_n, R_n^j\}$ refers to the memory space used by resources in R.
- $size(P_s^j)$ refers to the memory space used by all places in P_s^j.

- $size(n)$ refers to the total memory space of the node n.

Definition 5.3. ■
$\forall s \in S, N_s \subseteq N$ denotes the set of nodes on which the scope s is mappable. A scope s is mappable on a node $n \in N$ if, and only if, the following applies:

- Constraint (5.2) defines that the node n has a free resource for each place $p \in P_s^j$ in scope s of type j.

$$\forall j \in T, |P_s^j| + |R_n^j| \leq \text{card}_j(n) \tag{5.2}$$

- Constraint (5.3) defines that the node n has a resource of type j for each $\dot{p} \in \dot{P}_s^j$.

$$\forall \dot{p} \in \dot{P}_s^j, \exists r \in R_n^j \ / \ URI(\dot{p}) = URI(r) \tag{5.3}$$

- Constraint (5.4) defines that the node n has enough hardware memory to contain the places of scope s.

$$\sum_{j \in T} \text{size}(P_s^j) \leq \text{size}(n) - \text{size}(R_n) \tag{5.4}$$

Definition 5.4. ■
A Service Choreography (SC), defined by its set S of scopes, is not mappable if one of its scopes is not mappable, due to its multiplicity:

$$\exists s \in S, |N_s| < M_s \tag{5.5}$$

Definition 5.5. ■
The set of scopes that can be mapped on the node n is denoted $\forall n \in N, S_n \subseteq S$. ■

5.3.3 Pseudo-Boolean Optimization

The PBO minimizes a pseudo-Boolean function under a set of pseudo-Boolean constraints expressed by equations or inequations. The best SC mapping associates the places of the scopes with the available resource spaces on the nodes in order to minimize network load over the WSAN.

Definition 5.6.
The mapping of the scope $s \in S$ onto its possible hosting nodes in $n \in N_s$ is denoted by the Boolean vector x_s^n. The set X then determines the mappings of all scopes among their possible hosting nodes.

$$\forall s \in S, \forall n \in N_s : X = \{x_0^0, x_0^1 ..., x_1^0, x_1^1, ..., x_s^n\}$$

In turn, Eq. (5.6) resumes the total number of Boolean literals for the PBO, such as the number of possible mapping permutations on the nodes.

$$|X| = \sum_{s \in S} |N_s| \tag{5.6}$$

■

5.3.3.1 Communication Cost Function

The cost function $z(X)$ evaluates the impact of the mappings X on the network communication load. The pseudo-Boolean optimization (PBO) solver determines the best combinations of scopes and nodes among the set X of permutations in order to minimize the communication costs between the linked scopes. The communication cost of SC mapping is defined, for example, as the sum of its scope link costs $c(s_1, s_2)$, defined in Eq. (5.1), multiplied by the network distance $d(n, n')$ between their respecting hosting nodes. Then the pseudo-Boolean function of general WSAN communication costs, which must be minimized, is defined in Eq. (5.7).

$$z(X) = \sum_{s_1 \in S} \sum_{n \in N_{s_1}} \sum_{s_2 \in S} \sum_{n' \in N_{s_2}} c(s_1, s_2) d(n, n') x_{s_1}^{n} x_{s_2}^{n'} \tag{5.7}$$

5.3.3.2 Constraint Set

The minimization of the function $z(X)$ is constrained by the following set of (in)equations to define available mappings of the SC.

- Constraint (5.8) defines that each scope must be mapped several times according to its multiplicity parameter M_s.

$$\forall s \in S : \sum_{n \in N_s} x_s^n = M_s \tag{5.8}$$

- Constraint (5.9) forces the mapping of linked scopes to have a network route between their hosting nodes. If there are communications between s_1 and s_2 ($c(s_1, s_2) > 0$), they can be mapped respectively on n and n' if, and only if, $d(n, n') \geq 0$.

$$\forall (s_1, s_2) \in S^2, \forall n \in N_{s_1}, \forall n' \in N_{s_2}: \\ c(s_1, s_2) d(n, n') x_{s_1}^{n} x_{s_2}^{n'} \geq 0 \tag{5.9}$$

- Constraint (5.10) defines that the total available resource space required by all scopes mapped on a node n does not exceed its capacity.

$$\forall n \in N, \forall j \in T : \sum_{s \in S_n} |P_s^j| x_s^n \leq \text{card}_j(n) - |R_n^j| \tag{5.10}$$

- Constraint (5.11) limits the total memory usage by the mapped resources on a node to its available hardware memory.

$$\forall n \in N : \sum_{s \in S_n} \sum_{j \in T} \text{size}(P_s^j) x_s^n \leq \text{size}(n) - \text{size}(R_n) \tag{5.11}$$

5.4 Experimental Results

This experimental section presents results regarding mapping complexity and the execution time of deployment agents. Firstly, the number of constraints for the pseudo-Boolean optimization (PBO) problem is evaluated according to the size of the Service Choreography (SC) and the Wireless Sensor and Actor Network (WSAN). The second section evaluates the deployment agent execution for the service discovery mechanism over the Active Resource Middleware (ARM). Based on these results, deployment strategies are discussed in order to determine *best practice* concerning SC mapping and its deployment on an EMMA framework.

5.4.1 Dining Philosopher Mapping

The *dining philosopher* mapping process is a classical problem faced by distributed systems in terms of synchronization issues within concurrent resource access. In the case of a smart home under an energy contract with power delivery limitations, the different appliances must not simultaneously consume electricity. Therefore, they must be scheduled. One distributed approach involves each appliance negotiating permission with the others to be turned on. For example, the electric car and the hot water tank alternate their scheduling.

Figure 5.4 presents the functional design, while Figure 5.5 presents its mapping on the EMMA framework of a Petri network involving *dining philosophers*, as detailed in [HV+87]. Each appliance is represented by a philosopher and then a scope. They must exchange energy tokens in order that an appliance can consume energy, provided it has the tokens of the other appliances, which should not be turned on simultaneously.

The purpose of this experimentation is the evaluation of the resolution time of the mapping process according to the number of scopes and nodes. Figure 5.6 illustrates the number of generated constraints, which depend on the literals $x_s^n \in X$. They increase linearly and, in the main, according to the number of nodes, whereas the number of scopes has a lower impact. Figure 5.7 shows that the resolution time increases drastically according to the number of nodes, which is an expected result, such that the Knapsack problem is NP-complete.

Hence, the mapping process can be performed simultaneously for a lot of Service Choreographys (SCs), but only on a WSAN partition, which limits the number of nodes. This limitation is not particularly punitive because the deployment process is already partitioned by the composed agents.

Setup This experimentation has been realized with the PBO solver, based on an SAT4J framework on processor 4 cores at 2.2 GHz with multi-threading support and 8 GB of RAM memory. After several experimentations, available RAM memory for a Glucose solver seems to be the major factor on the resolution

time.

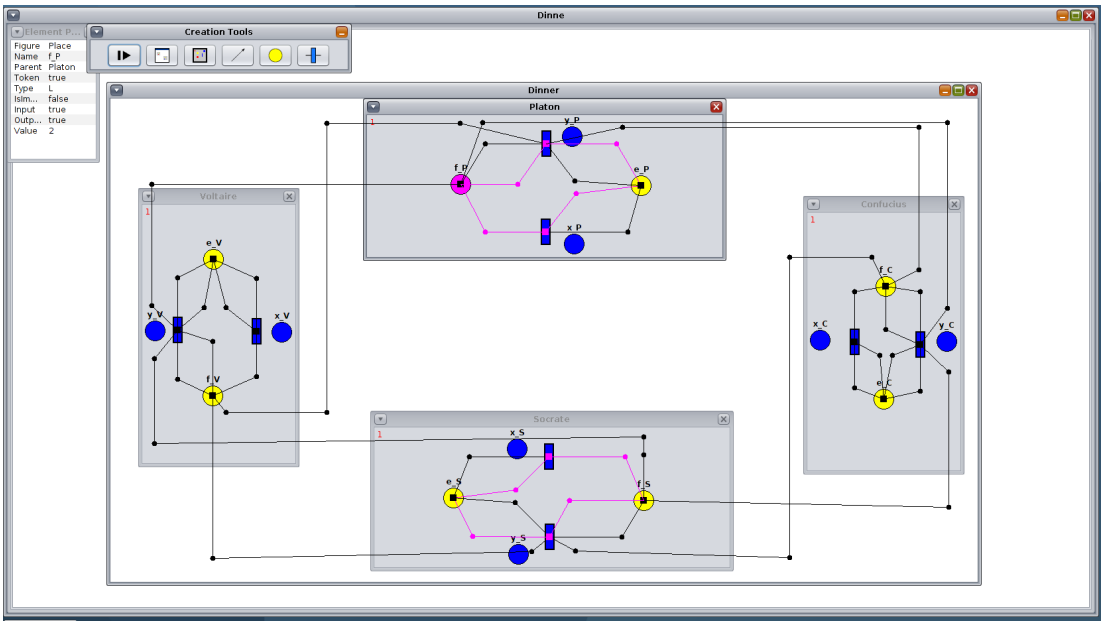


Figure 5.4 – Petri network of dining philosophers in an *emma-design* tool.

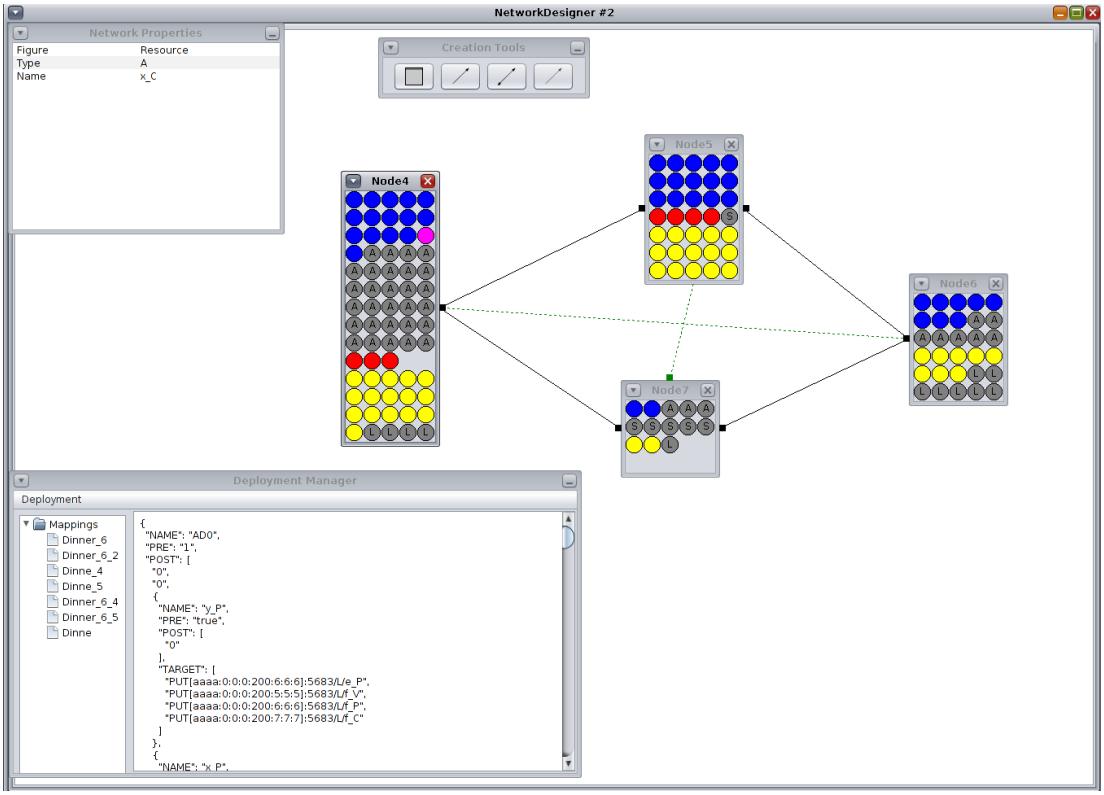


Figure 5.5 – Dining philosopher mapping in an *emma-design* tool.

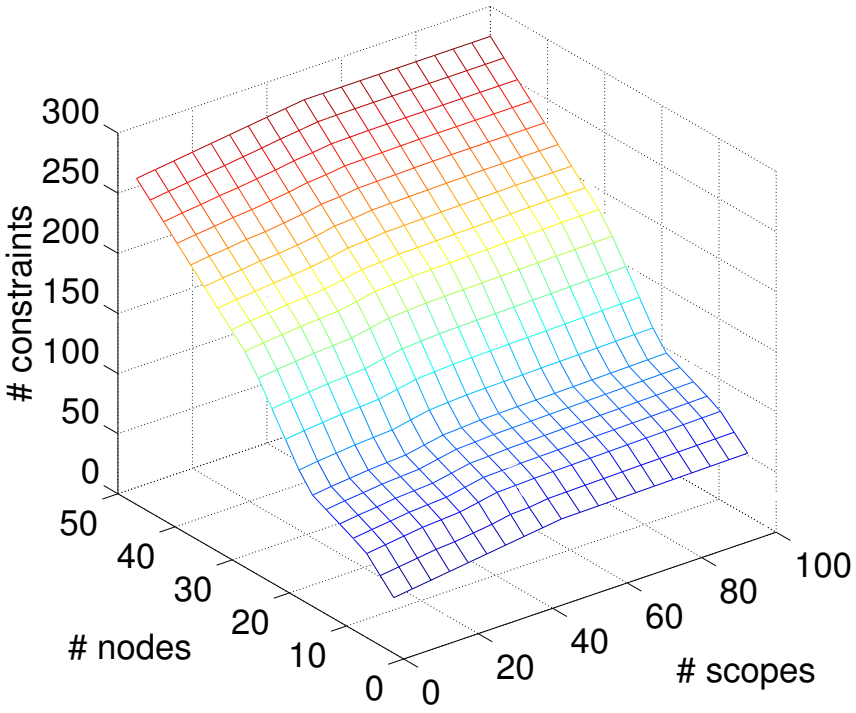


Figure 5.6 – Number of constraints according to the number of scopes and nodes.

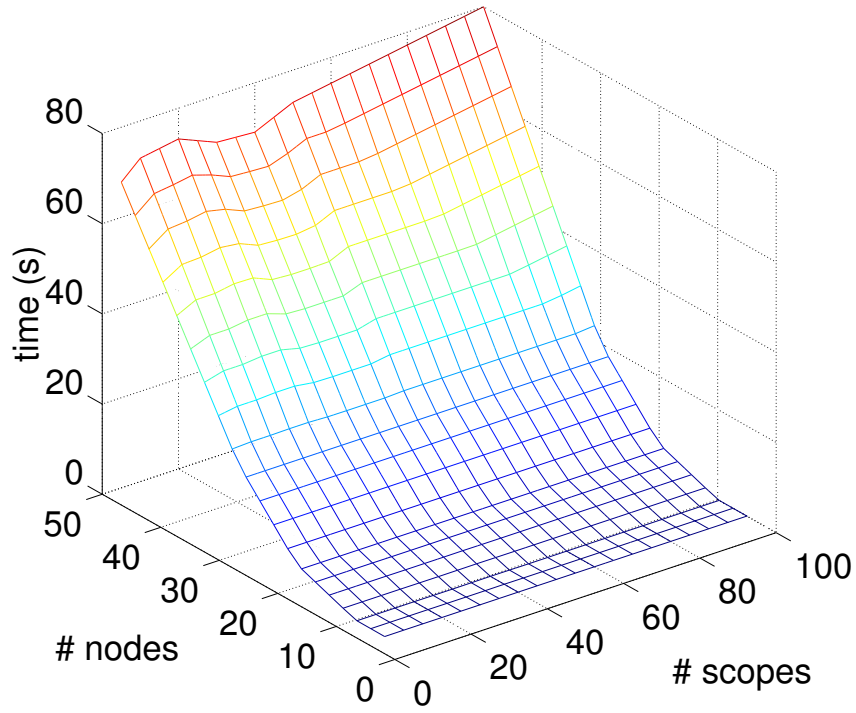


Figure 5.7 – Mapping resolution time according to the number of scopes and nodes.

5.4.2 Deployment Evaluation

EMMA agents allow the deployment process to be performed by three different approaches: *direct deployment*, *composed deployment* and *self-deployment*. Direct deployment is the common approach in most contributions for SC middleware. However, it produces significant network congestions around routers in a deep network because all deployments are transmitted from the supervisor. The below results compare the two proposed strategies by EMMA between a self-deployment Agent 4.3 (Figure 5.11) and a composed agent 4.1, as defined in Section 4.3.1.3 (Figure 5.10), deployed on a 14-hop network, as illustrated in Figure 5.9. The experimentation evaluates the deployment of the *service and network discovery* mechanism. Figure 5.10 prints the deployment time D as equal to the agent writing time W to store the agent contained in a payload of size P, transmitted in T ms to the node and executed in L ms on it. As the agent execution is processed en bloc, transmission time for the deployment agent i is equal to the total transmission time less the writing time on the next node, which is summarized in Eq. (5.12).

$$D(i) = W(i) + (T(i) - W(i+1)) + L(i) \tag{5.12}$$

This figure shows the impact of cumulated agent overhead along the deployment path. For each node, the deployment agent contains SC agents and the composed agents for the next node. This strategy is interesting in terms of delegating the deployment process to particular nodes over WSAN areas. Hence, several deployments can be performed simultaneously without any interference between them. The peer-to-peer communications between the supervisor and the nodes are resumed on the transmission of the deployment agents. In turn, all other communications for the deployment process are local and, therefore, without a long communications path. However, the important overhead produced because of the agent composition can be avoided in case of the redundancy of the SC to deploy. Figure 5.11 presents the deployment of a self-deployment agent, which is broadcast over the WSAN. It deploys the SC on the node on arrival before moving onto the next nodes. Its use is very interesting in relation to the deployment of SCs on several nodes, such that its constant overhead is low regarding contained SCs. However, the deployment of numerous different SCs implies that the payload is very large. Hence, *self-deployment* is efficient for the installation of common SCs on initialization in order to avoid redundant compositions, whereas *composed deployment* is used for resource deployment delegation to a local node in the WSAN area of interest.

Setup The different results have been experimented on using a COOJA simulator, with an Atmel AVR Raven board composed of an IEEE 802.15.4 radio transceiver and an ATmega1284PV 8-bit microcontroller at 8 MHz with 16 kB of RAM and 128 kB of flash memory.

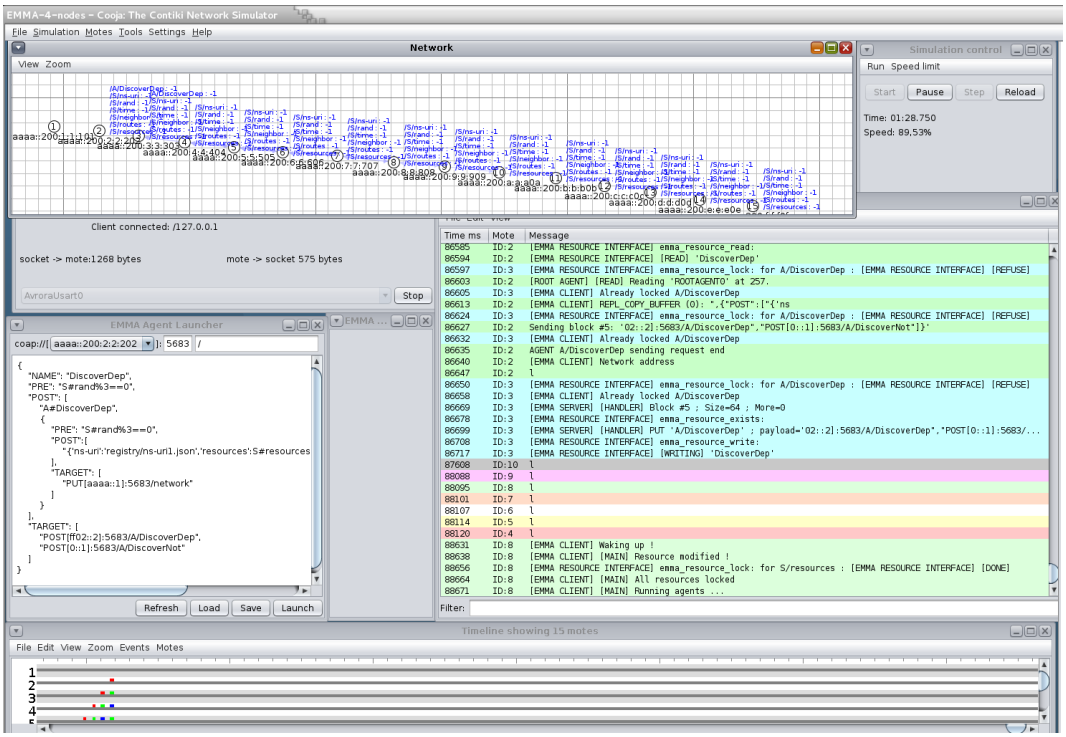


Figure 5.8 – Self-discovering agent deployment in the *emma-cooja* tool.

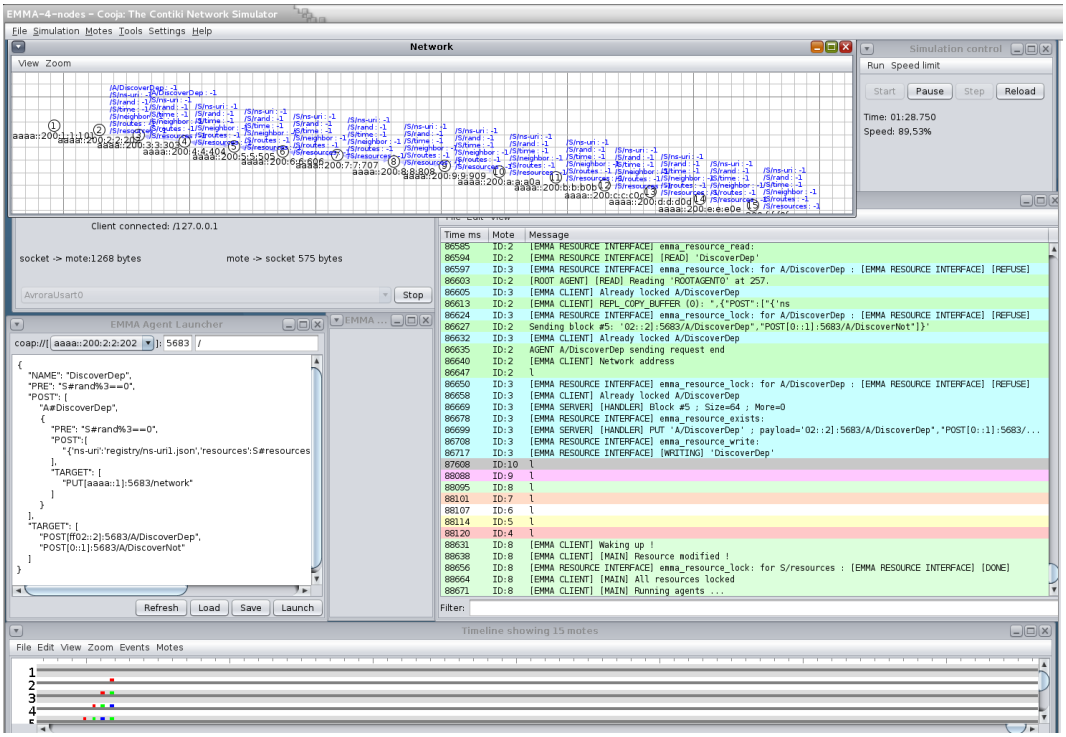


Figure 5.9 – Composed agent deployment in the *emma-cooja* tool.

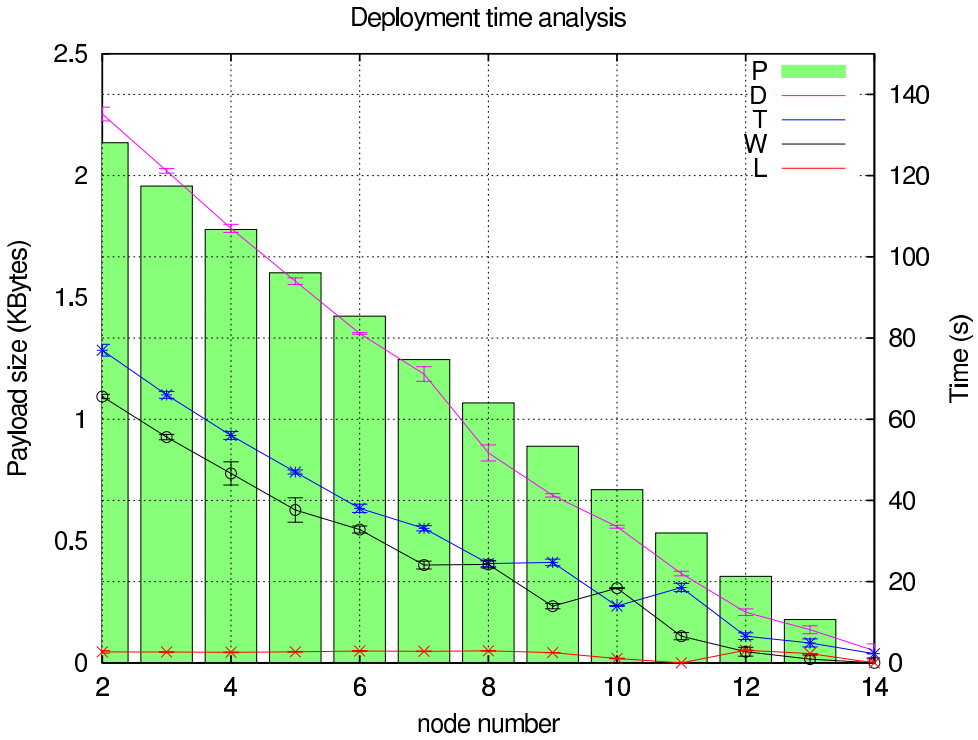


Figure 5.10 – Composed agent deployment time in a deep network.

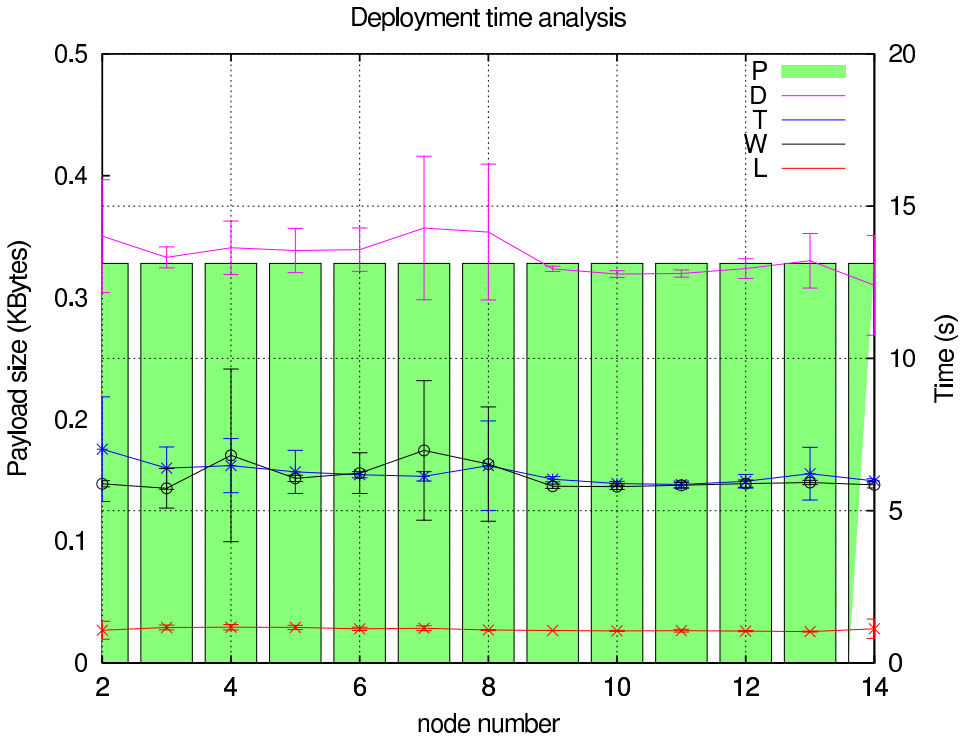


Figure 5.11 – Self-deployment agent deployment time in a deep network.

5.4.3 Deployment Strategy

Previous results have shown that the mapping problem complexity increases exponentially with WSAN size, while the composed agent overhead limits its capacity to deploy several Service Choreographys (SCs) simultaneously. Therefore, the deployment process must be performed for each WSAN partition and for each SC. The supervisor must schedule SC deployment to avoid network congestion and reduce mapping process complexity. All nodes require identical SCs, such as service discovery and system log collection, which can be deployed by a self-deployment agent. It does not require offline mapping;moreover it consumes free resources on the node, which reduces mapping this is combinatory for others' SC mapping.

1. *On WSAN initialization:* The network topology and node resources are unknown, while all bandwidth is available. The supervisor should deploy all SCs with a secure $NodeKeys$.

   (a) Self-deployment agents are sent by flooding in order to deploy independent SCs to the nodes, especially *service discovery* agents.

   (b) The mapping resolution of SC templates on available free resources according to the received resource lists from the *service discovery* agents.

   (c) Generation of Dynamic Network Agents (DNAs) to deploy Residual Network Agents (RNAs) with the $NSKeys$.

   (d) Scheduling of DNA transmission to manage the network load according to SC dependencies and their spatial localities.

2. *During WSAN running time:* SCs are already deployed and running on the ARM; new deployment should not disturb these SCs.

   (a) The mapping preprocessor adds new SCs to the current mapping.

   (b) RNA generations for fewer disturbances according to the simulator.

   (c) Scheduling DNA transmission to control the network flow.

3. *For local Corrections:* The target node is not accessible from the edge router and there is no free resource to host the delegation of the composed agent for deployment.

   (a) The supervisor (on a mobile device) moves spatially in the node area to obtain network access.

   (b) The supervisor collects information and applies deployments or corrections directly through COAP Web services with the $NodeKey$.

5.5 Summary

This chapter has presented, in detail, how to design and deploy services on an Environment Monitoring and Management Agent (EMMA) system. Based on *functional design* descriptions, these templates are instantiated and mapped according to Wireless Sensor and Actor Network (WSAN) specifications. The mathematical background has revealed that the Service Choreography (SC) graph can be mapped onto the WSAN topology by a pseudo-Boolean optimization (PBO) formulation. The EMMA system offers several ways to deploy, such a SC graph over a WSAN. Although the traditional approach by direct transmission to a node is possible, its use is, at least, inefficient and, at worst, impossible in large-scale networks. Two other approaches use the flexible language of an Active Resource Middleware (ARM) to design SCs to deploy SCs. The first one uses some temporary nodes to delegate the responsibilities of SC deployment by a composed agent. The other one uses mobile agents, which move across the WSAN to deploy SCs according to node requirements. The experimentation section concludes that all of these approaches have limitations and advantages. The proposed deployment procedure uses all of them according to the WSAN life cycle and constraints.

The impact of the EMMA system for service choreography over a WSAN is in relation to its interaction model. Firstly, it is a new paradigm in which services are completely abstracted from the WSAN. If common middleware provide hardware and network abstractions, an ARM is itself abstracted through a resource tuple. Secondly, this framework facilitates collaborative working between specialists through three levels of tools. Manufacturers use *emma-node* software to build their smart objects without considering future usages. Services served by the WSAN are designed through functional templates by software engineering experts. They use *emma-design-tools*, which do not require any knowledge about electronics or networks, allowing them to connect the WSAN to external internet services. Meanwhile, network administrators can deploy new services over a WSAN without considering SC implementation or node hardware. They can use the *emma-network-analyzer* to compare the network behavior prediction of *emma-design-tools* and real network monitoring through Simple Network Management Protocol (SNMP) interfaces. Finally, the totality of the EMMA system has been designed with privacy and security considerations at each level, from the Internet backbone to the resource level. All data stay contained inside the WSAN in which they are processed in order to address data privacy and security.

Part III

Toward Neural Intelligence

Chapter 6

Artificial Neural Controller[1]

Enseigner c'est apprendre deux fois.

To teach is to learn twice.

Joseph Joubert [Jou66]

Contents

[1] Published in Clement Duhart and Cyrille Bertelle. "Methodology for Artificial Neural controllers on wireless sensor network". In: *IEEE Conference on Wireless Sensors (ICWiSE)*. 2014, pages 67–72. DOI: 10.1109/ICWISE.2014.7042663

6.1 Introduction

Applications of Artificial Neural Network (ANN) [PSY88] on WSANs have recently emerged to define a new concept of Neural Wireless Sensor Networks (NWSNs). This has been formulated due to the similarity of these applications' graphic structures, which are jointly used to address complex issues [KFV11], such as the design and deployment of networks, localization, data aggregation and sensor fusion, energy-aware routing and clustering, scheduling, security and quality of services. Some implementations have demonstrated the technological suitability for deploying ANN on constrained hardware [Far+05], which allows real use cases to be validated, such as real-time forest fire detection [YWM05] or geolocalization in a grid [SZM07]. These applications use detection techniques relating to faults or unusual events [MS08; KD05] in complex distributed sensory systems addressed by a multilayered perceptron (supervised model) or an ART neural network (unsupervised model). Moreover, an ANN is used to optimize communication bandwidth in a WSAN [PA08; HZM09] by defining efficient cluster heads, data aggregation or parallel pattern recognition [KM04]. Indeed, fault tolerance is usually addressed by system redundancy, which increases infrastructural complexity. Noteworthy studies, however, have demonstrated that an ANN has the ability to correct incomplete or corrupted input patterns through an artificial recurrent neural network, such as the Hopfield model [OM06]. Based on data pattern recognition studies [Tei+14], new investigations have been conducted, which capture human behavior patterns in smart home system designs [BH06]. This chapter deals with the methodology to control actuators according to captured human behavior patterns. On the one hand, these patterns are learnt by recurrent neural networks, which can learn stochastic nonlinear systems [BB11]. On the other hand, the control system is ensured by using control theory studies of nonlinear dynamic systems [LN93; SSU94]. There are several motivations for using NWSNs in smart home information systems, such as load-balancing for bandwidth reduction and local data processing. Another very important motivation is the ability of an ANN to encode and compress the information in its weighted matrix in order to make it unreadable. This is helpful for security reasons, such as in password encryption [BB11] or in the case of EMMA, in terms of maintaining secret system functioning rules.

This chapter proposes agent-based architecture for extracting human behavior patterns in an object-centric approach. Each object learns how to drive its outputs according to its use by humans during the training phase. The proposed architecture for Artificial Neural Controller (ANC) [PSY88] uses neural classifiers for output control. After preliminary analysis, Section 6.2 presents an agent model based on ANC for managed environment control. Section 6.3 discusses the methodology behind neural behavior learning and symbolic rule inference for knowledge transfer. Finally, Section 6.4 looks at how ANC is implemented in relation to an EMMA service component.

6.2 Neural Control Architecture

In Internet of Things (IoT) applications, connected objects must adapt their behavior according to their running environment. Such managed environments can be unknown, open, dynamic and complex. Insertions of new objects or failures in the system can introduce uncertainty and incompleteness. Unfortunately, the behavior of humans in these environments cannot be easily modeled with concise mathematical expressions. More complex theories, however, are used to filter these inputs, such as fuzzy logic, possibility or probability models. The proposal investigates the potential for learning environment controls to use statistical neural classifiers. Operations performed by humans on actuators are learnt by Artificial Neural Network (ANN) according to the input state of sensors and actuators. Hence, there is no more requirement placed upon environment models. Such controllers are adaptive according to human behavior in a given environment. When system goals (energy saving, comfortable mode etc.) are evolving, object agents must adapt their behavior. Therefore, it must learn behavior for each system functioning mode, which is known as context.

6.2.1 Preliminary Analysis

Preliminary analysis evaluates the feasibility to learn all agent behaviors in a single Artificial Neural Network (ANN). Based upon experimentations with several sets of generated data, the size of learnt classifiers does not allow their execution by EMMA target devices. Hence, behaviors that split into several classifiers are investigated.

6.2.1.1 Artificial Neural Networks

An ANN is a learning technique to approximate control functions between output and input vectors. There are numerous models that can be used according to learning interests, such as delayed neural networks [LS07] for periodic functions or recurrent neural networks [PA08] for stochastic processes. The most used is an Multi-Layer Perceptron (MLP), which is a feedforward model in which all neurons are interconnected to join input and output layers by an acyclic directed graph. The *universal approximation theorem* [Csá01] proves that an MLP can approximate any continuous function, which associates input to output real number intervals for restricted classes of activation functions. Each of its intermediate neurons is equal to the weighted sum of the input neurons to compute the output through its decision function. The learning process determines these weight values in order to provide the right output (called a class) for a given input vector. Different algorithms are available, of which the most used is called the gradient descent algorithm by backpropagation. The algorithm 6.1 details the computation of output errors to update weight values by propagating the differential error on intermediate neurons layers.

In lighting applications, an Artificial Neural Network (ANN) must drive lights to maintain user brightness preferences. According to sensory inputs, it must decide whether to change luminosity. An ANN should learn transition functions in order to change the environment, as well as invariant functions when the system state is suitable. The set of actuator values is denoted $U = \{u_0, ..., u_M\}$ and the set of sensor values is denoted $S = \{s_0, ..., s_N\}$. At time t, the context state $C(t) = U(t) \cup S(t)$ evolves in line with unknown control function g_i, which can either be an invariant function or a transition function, learnt by the ANN to compute the next actuator vector U, as in Eq. (6.1).

$$u_i(t) = ANN_i(u_M(t-1), ..., u_i(t-1), ..., u_0(t-1), s_N(t-1), ..., s_0(t-1)) \tag{6.1}$$

MLP output prediction is a forward propagation of the information sample t across all layers in $L = \{1, \ldots, K\}$ that computes new values for each neuron $f = \{1, \ldots, M\}$ which composed them until the final output layer is reached. At the initial step, the first layer is equal to context state vector $f_0^t = u(t)$ for predicting output context state $u_i(t+1) = f_L^t$. To summarize, $f_{k,j}^t$ corresponds to neuron activation j in layer k for sample t, as defined in Eq. (6.2).

$$f_{k+1,j}^t = F(\sum_{j=1}^{M} w_{k,j} f_{k,j}^t + w_{k,0}) \tag{6.2}$$

Output feedback in an input layer allows an ANN to learn a stochastic process. The lack of a hidden context layer avoids stability issues for the gradient descent algorithm in relation to the recurrent neural network, even if it is more practical and tractable than in Markov chain process applications [Pea95].

Algorithm 6.1: Gradient descent learning algorithm

Data: T
Result: W

repeat
  T^i = RandomSelection(T)
  foreach *layer* k **do**
    foreach *neuron* j **do**
      $\delta_j = f_{k,j}(1 - f_{k,j}) \sum_{l \in dest(j)} \delta_k w_{j,l}$
      foreach $w_{l,j}$ *from neuron* l *to* j **do**
        $\Delta w_{l,j} = \varepsilon \delta_j f_{k,l}$
      end foreach
    end foreach
  end foreach
  foreach $w_{i,j}$ **do**
    $w_{i,j} \leftarrow w_{i,j} + \Delta w_{i,j}$
  end foreach
until $i < I_{max}$;

6.2.1.2 Classifier Learning Complexity

The number of layers and neurons in a classifier defines its complexity. It depends upon the difficulty in finding a efficient hyperplane in order to separate each possible output class. In a general way, the power of ANN discrimination is improved by increasing the size of neuron layers, as well as their numbers. However, an oversized ANN produces an overtraining effect, which prevents it from predicting the correct output if inputs are not perfectly accurate. Therefore, this study evaluates the minimal size needed by an ANN to learn a good classifier. These results are extended to address questions regarding multi-behavior learning.

According to the complexity of ANNs,
what is the best agreement between one or several classifiers
for the purpose of learning mixed data from different control functions?

Data sets are generated by a random state machine composed of 15 states with a maximal loop length of 3-hops. The training data set is composed of two million examples and the printed results are produced by a test data set of 500 examples. The value ranges of the states are bounded between continuous intervals $[-1; 1]$. Linear and quadratic transition functions are respectively evaluated in Figure 6.1 and Figure 6.2. Figure 6.3 presents a mixed data set of linear and quadratic functions. These figures present the error rate for the test data set according to the number of layers 1 to 5, the number of neurons by layer between 10 to 110 by steps of 20, and the learning rate ε as multi-curves (0.01, 0.001 and 0.0001).

Table 6.1 – ANN complexity synthesis

Data	Type	#Neurons	#Layers	ε	Error
Linear	A	10	1	0.01	0.06
Quadratic	B	70	2	0.01	0.09
Both	A+B	100	5	0.001	0.08

Table 6.1 summarizes the minimal size of an ANN for a given prediction error threshold in relation to test examples. The function of a linear transition requires a small number of neurons on a single layer, whereas quadratic data generation requires 14 times more neurons. The combination of these two kinds of transition function requires many more neurons (500 neurons) to predict output with the same error rate. Currently, there is no rule in the literature review that determines ANN complexity according to the input data structure. The general conclusion is not obvious; however, it is assumed that the parallel use of several classifiers consumes fewer neurons than a unique one. Given that EMMA nodes have memory and computation limitations, a multi-classifier model is preferred. For each different behavior, the minimal size of classifier is determined empirically by a *trial and error* approach. During the runtime, the classifier with the lower error rate, according to the input, is selected in order to drive output layers of actuators, as illustrated in Figure 6.4.

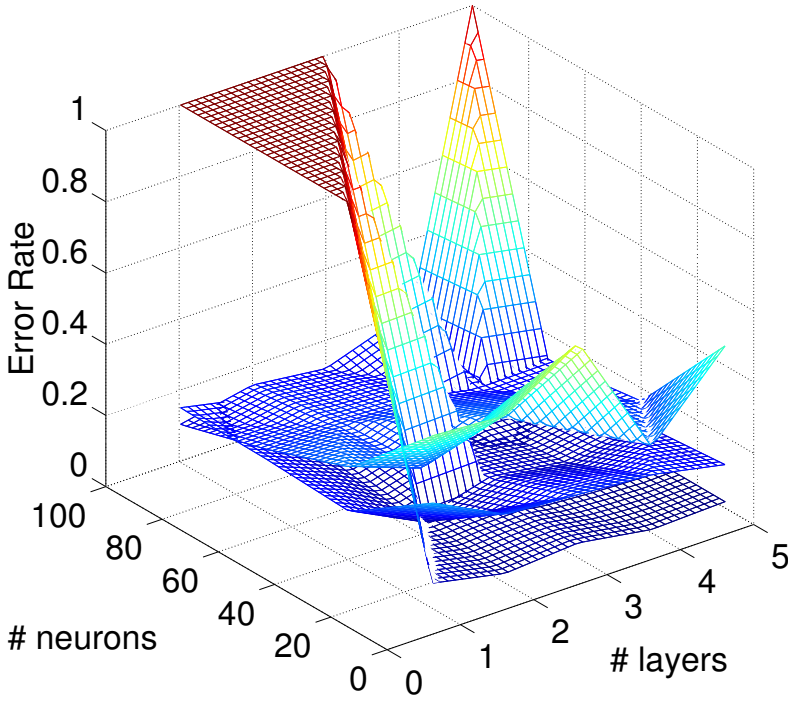


Figure 6.1 – ANN complexity for linear transition functions.

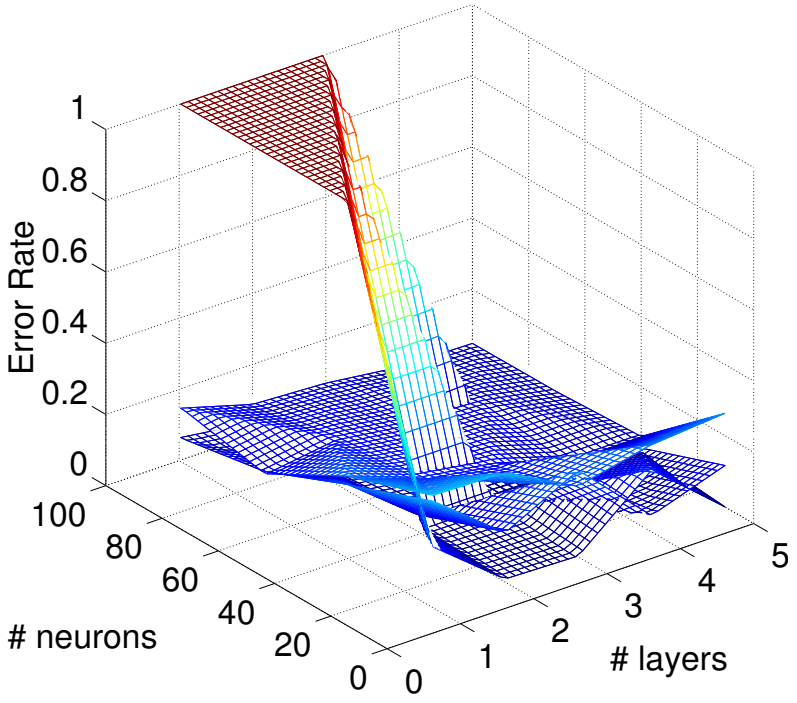


Figure 6.2 – ANN complexity for quadratic transition functions.

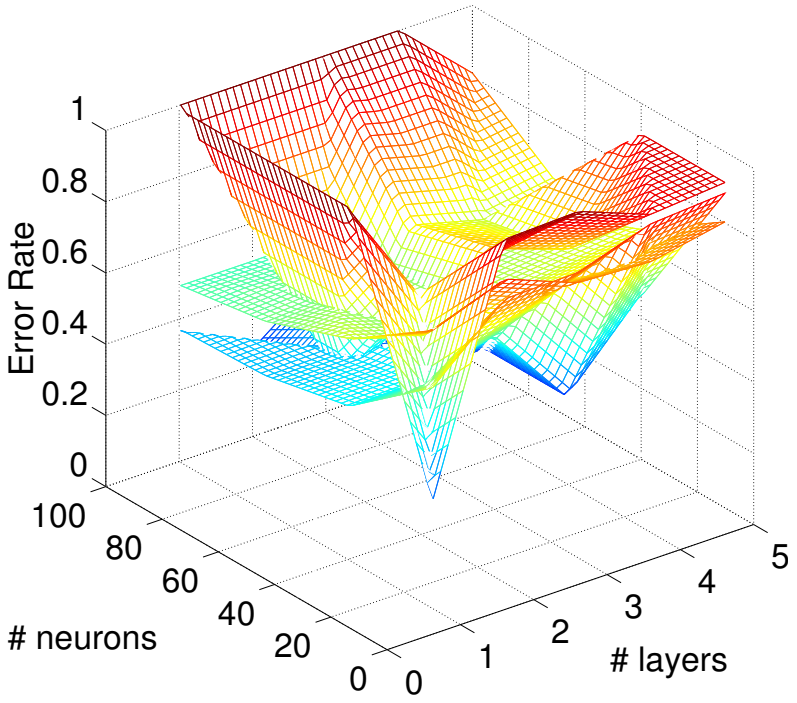


Figure 6.3 – ANN complexity for mixed transition functions.

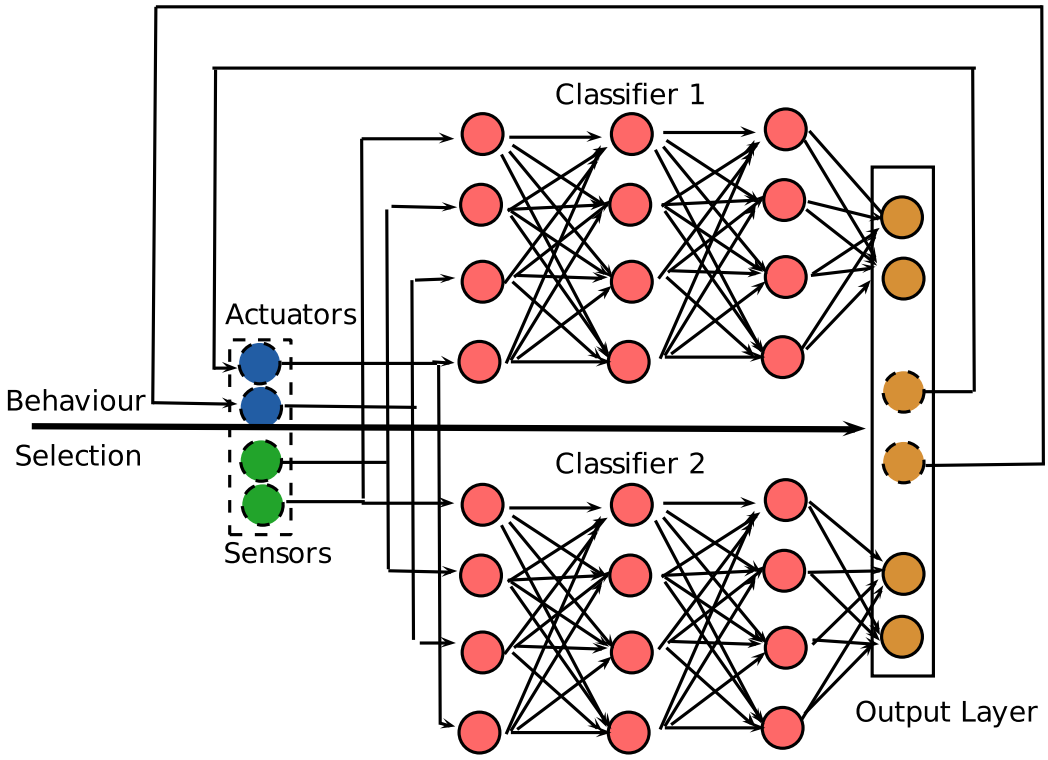


Figure 6.4 – Scheme of a multi-classifier ANN.

6.2.2 Agent Model

The previous experimentation has shown the suitability for correctly predicting the subsequent states using learning classifiers. As input vectors generated by multiprocessing are more difficult to learn, the use of a single ANN is not desirable.

Definition 6.1.
ANC agent A is a process for selecting the best ANN classifiers for the purpose of controlling an output vector according to an input vector. ■

The controller selects the best of the classifiers according to the successive success rate on a current input stream, as illustrated in Figure 6.5. It is assumed that, if the agent control is not suitable, humans will correct the actuator states. Therefore, human interaction produces a prediction error for the current agent classifier. There are two scenarios in this case: an agent does not know this environment behavior and has to learn it; or the current classifier must be corrected. Both scenarios are similarly managed. If the agent does not have the proper classifier (periodic errors), it starts a new learning process. According to error frequency, the agent decides between the creation of a new classifier and the update of the current one. In turn, the learning process is based on data production using manual control by humans in relation to environment actuators. Given the lack of captured data from real use examples, three kinds of transition function are used:

- *Counting* concerns an incremental state data series up until value 32 to demonstrate prediction suitability in the Markov chain process.
- *Fibo* is the famous Fibonacci data series up until value 32 to validate the examples of complex input-output relations.
- *Custom* data series involves a random state machine of 32 states.

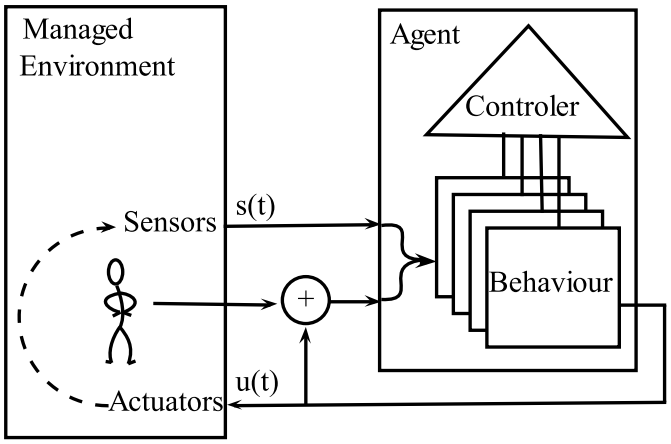


Figure 6.5 – Model overview of an ANC

6.2.2.1 Behavior Classifiers

Definition 6.2.
The behavior classifier $b \in B$ is a transition matrix of a finite N state machine of the input vector u, encoded in the weight matrix W for Artificial Neural Network (ANN), such as:

$$\forall t \in N, \mid b(u(t)) - u(t+1) \mid < \varepsilon \text{ with } \varepsilon << 1$$

■

A set of behavior classifiers represents agent knowledge to drive output according to the input data stream. As previously discussed, the learning process is a difficult challenge to determine minimal ANN complexity for a fixed prediction error rate. Therefore, a *trial and error* algorithm is used to explore possible combinations of neurons, layers and learning rates. Figure 6.6 presents the results of a classifier that learns the counting transition function that is encoded into a binary vector. Input samples of Figure 6.6 (a) are used to train the ANN. The algorithm progressively increases the number of neurons until that target of the error rate is reached, as illustrated in Figure 6.6 (c). Each step is evaluated with 100 input noise samples shown in Figure 6.6 (b), while the corresponding classifier response is shown in Figure 6.6 (d). These figures show that, with fewer than eight neurons, the behavior classifier do not succeed in counting up to 32.

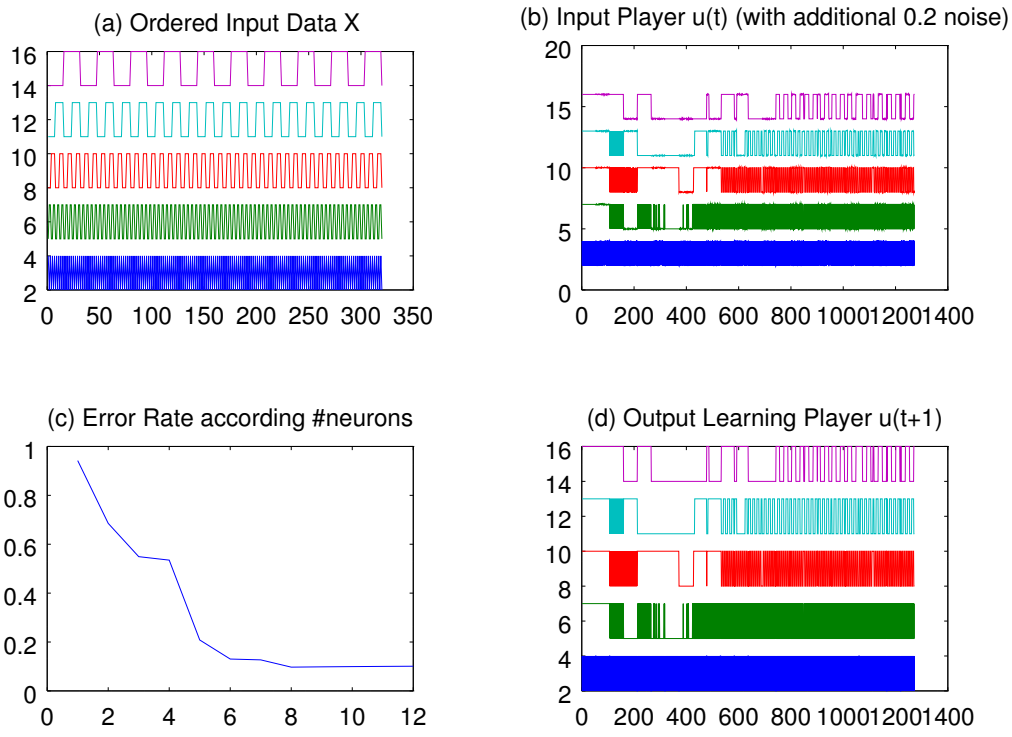


Figure 6.6 – Behavior classifier learning result.

6.2.2.2 Controller Scheduling

Controller scheduling evaluates which behavior classifier is the best for controlling outputs. Each classifier computes its outputs according to the input stream. Based on its predictions and the next occurring state, the preference index used for arbitration is built according to its success rate counter.

Definition 6.3.
The schedulability of behavior classifiers is determined by the agent ability to possess classifier $b \in B$ for input stream $u(t)$ of N states with a cumulative success rate that is greater than M, such that:

$$\forall t \in N, \exists b \in B \ / \ \forall j \in [t - M \ ; \ t], \mid b(u(j)) - u(j+1) \mid \ < \varepsilon \text{ with } \varepsilon << 1 \quad (6.3)$$

■

Figure 6.7 illustrates the behavior classifier scheduling according to the input of Figure 6.7 (a). Each of the three classifiers tries to predict the next state, while their error in relation to each sample is plotted in Figure 6.7 (b). According to their cumulative success rate for the definition in Eq. 6.3, a preference order of classifier is found in Figure 6.7 (c), with the purpose of selecting the best one for the current input data. It can be observed that several behavior classifiers can be matched; however, only those with the greater success cumulative rate can be scheduled, as illustrated in Figure 6.7 (d).

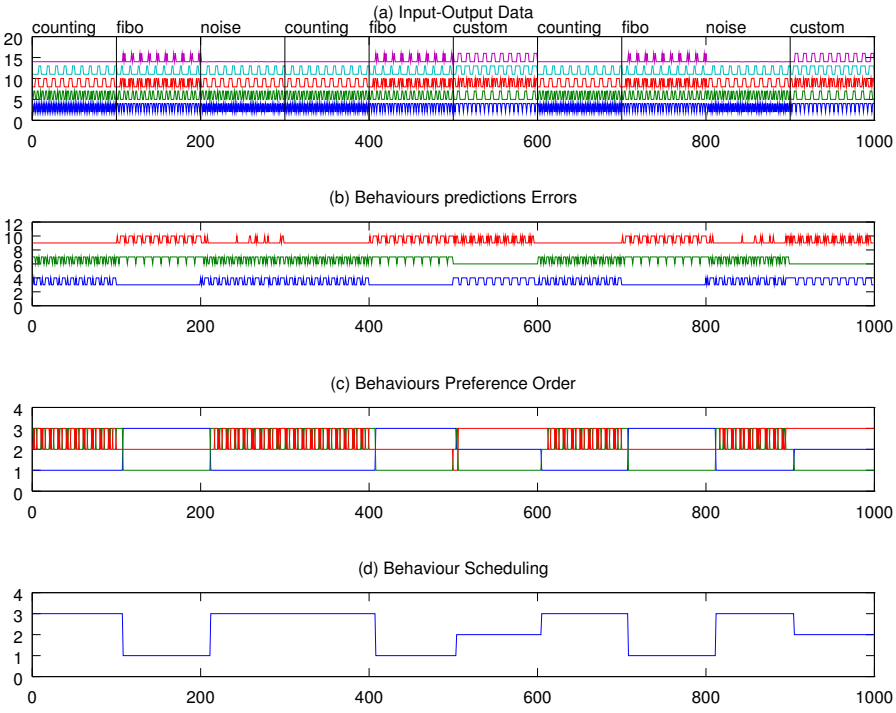


Figure 6.7 – ANC scheduling player example

6.2.2.3 Behavior Online Training

If Eq. 6.3 is not satisfied, the ANC is not able to properly drive the output vector. An unpredictable input stream means that there is no selected behavior classifier, while a new one must be trained. At ANC initialization, there is no behavior at all because it is a new environment with unknown human usages. Therefore, input stream recording starts until the learning buffer is full or the input stream becomes predictable again. In turn, the ANC creates a new behavior classifier, which is trained with recorded data.

Figure 6.8 shows an ANC containing only one behavior at the beginning (red line). As initial counting of the input signal is not predictable, ANC creates a new classifier, which is successfully used later. The same operation is observed for custom input data. In the end, each input stream is properly controlled by an ANC. Interesting results appear when the input stream changes during new classifier training. The ANC creates a general classifier for this mixed input stream, before creating specialized ones. Classifiers are refined until the definition Eq. 6.3 is satisfied. An analogy can be observed with human behavior, which tries to learn a complex process. It then tries to learn globally before it understands that it is composed of different subprocesses. Hence, the initial learning process is split into several iterations of learning that involve independent training.

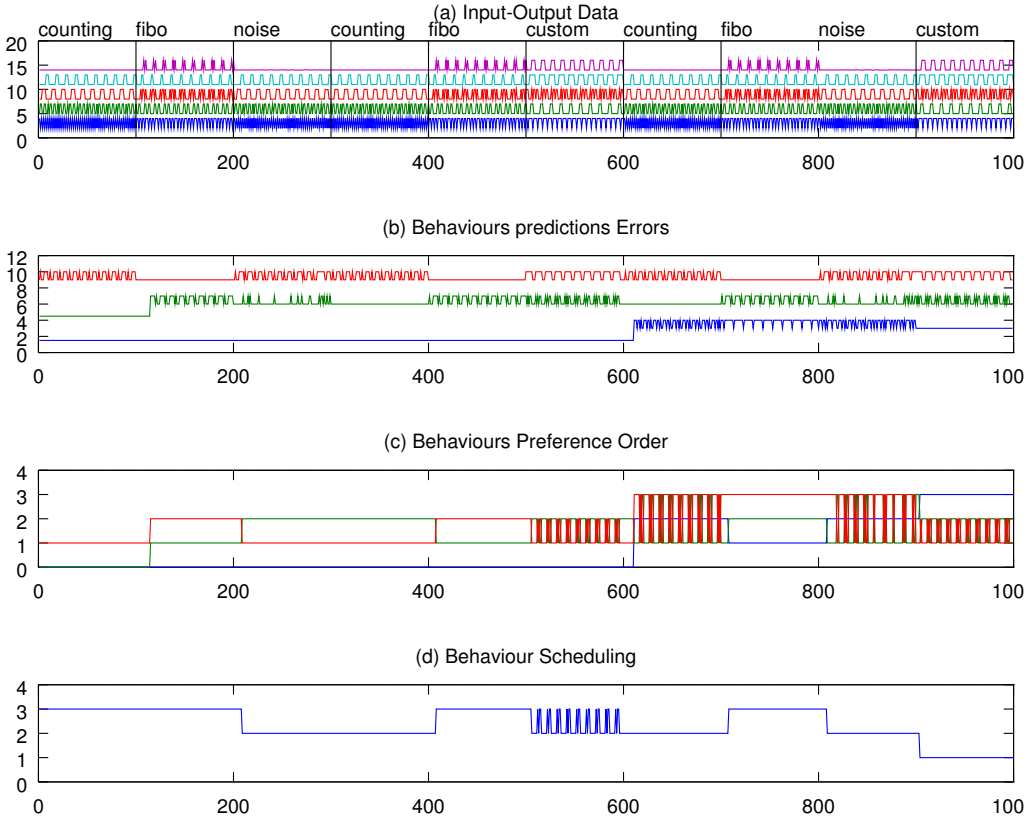


Figure 6.8 – ANC behavioral online training

6.3 Knowledge-Based Training

An Artificial Neural Controller (ANC) learns statistically desired output controls according to its input states and current environment context. From behavior learning to scheduling, the suitability of an ANC is validated for mixed output controlling. However, classifier training requires a lot of learning examples for all possible states. The collection of such data is often very difficult, while desired ANC behavior is fully known. Knowledge-based training is proposed to generate behavior classifiers based on logical descriptions, which are similar to the KBANN system [TS94]. Rules are used to generate training data, which are mixed with recorded ones from human behavior captures. Hence, uncaptured states during recording phases are also trained with a default behavior. In addition, the inverted process of associative rule extraction from behavior classifiers allows knowledge to be transferred between ANCs. All steps in an ANC training process are summarized in Figure 6.9. The first three concern behavior classifier training and logical rule inference for knowledge transfer. Classifiers are then deployed and executed on an Active Resource Middleware (ARM). The black arrow loop corresponds to previously presented behavior classifier learning processes, whereas dashed arrows concern training data generation and logical rules inference.

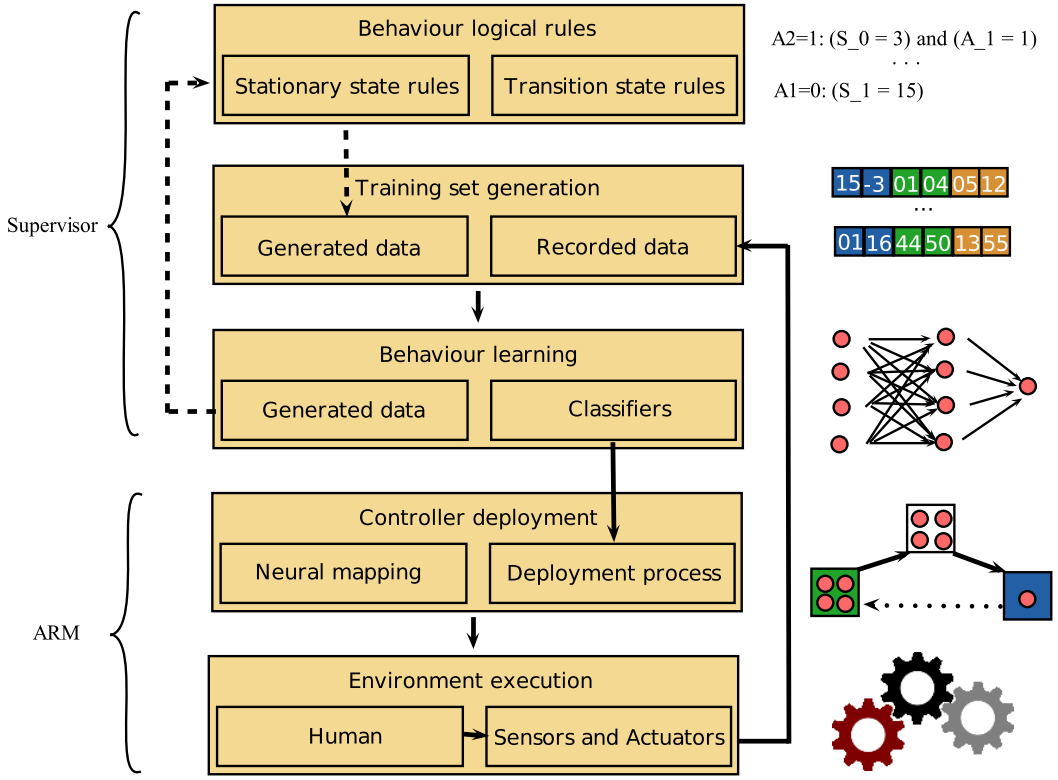


Figure 6.9 – Methodology process for ANC behavior training.

6.3.1 Training Data Generation

Statistical data generation for classifier training consists of building input vectors (composed of sensors and actuator states), which are associated with desired output vectors (future actuator states). Behavior classifiers are finite states machines encoded in an Artificial Neural Network (ANN) to manage uncertain transition state values. There are numerous forms for modeling such behaviors; examples include Markov chains and probabilistic finite state machines. However, decision trees are more practicable for agent-based approaches. An Artificial Neural Controller (ANC) is considered as an autonomous decision maker which must learn the best policy (a set of coupled state decisions). The state vector is composed of sensor and actuator values. They can be stationary or transitional. Indeed, the loop control between the ANC and the managed environment is continuously executed. The managed environment evolves in relation to the actuator effects in reaching a new state. If this state is suitable, the ANC must maintain the same output vector at which to stare. Therefore, on the one hand, behavior classifiers must be trained for transition conditions to change the environment state, while, on the other hand, it must learn not to change the control outputs when the managed environment is in a suitable context.

Finally, while any formalism can be used, it allows a couple of input-output vectors to be generated for each possible value of sensors and actuators, as illustrated in Figure 6.10. The generated data set must be blended and equally distributed to avoid classifier specialization in relation to particular states. Indeed, they would experience overtraining effects instead of an uniform error rate for each reachable state. Noise addition on input-output vectors during training increases fault tolerance for uncertain values produced by actuators or sensors. However, experimentations has shown that it can inversely increase learning complexity.

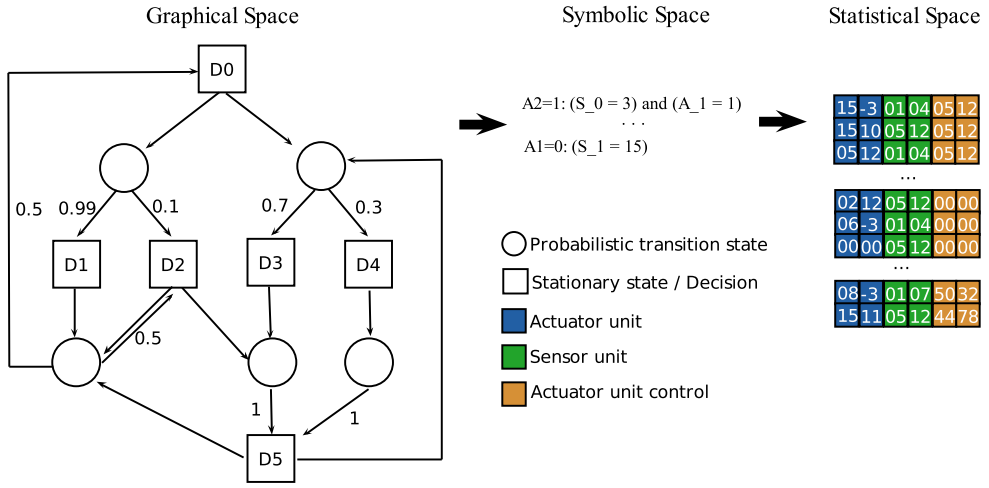


Figure 6.10 – Statistical data generation based on logical rules.

6.3.2 Inferred Knowledge Transfer

Training data generation helps experts to build behavior classifiers by providing associative rules between input and output vectors. In the previous section, some statistical considerations were discussed to ensure efficient learning for all states, which were represented by generated data. This allowed classifier behavior to be prepared offline with logical descriptions.

Knowledge transfer is discussed in terms of sharing human behavior captures between different ANCs. knowledge transfer studies for Artificial Neural Network (ANN) is an emphasized and very difficult research scope because of the neural encoding state representation. The common approach is the use of common hidden layers between ANN [DAG99]. Its application for multilingual structural transfer has been validated [Hua+13]. However, this kind of knowledge transfer implies similar neural topologies, which are not necessarily available in relation to the concerned ANC. Indeed, according to the input, ANC complexity is different, given the number of neurons and layers. Therefore, the literature review does not provide solutions for direct knowledge transfer at the neuron level. Others approaches consist of extracting knowledge to an external symbolic representation space, as illustrated in Figure 6.11. Based on generated data by the classifier, associative rules are inferred. Instead of transferring encoded knowledge between ANN, it is extracted by playing all possible input values in order to generate corresponding outputs. Therefore, associative rules inference algorithms are applied to extract data structural models, such as CHARADE [Gan87] or A-Priori [AS+94]. Several contributions exist in relation to the desired structural model, such as the finite state machine [Cas96] or associative rules [TS93]. However, the transition from a statistical to a symbolic space requires the literal definition for representing a continuous value by a discrete symbol. Implementation, in this thesis, is concerned with literals that are created for each value of each sensor. Therefore, the number of literals increases exponentially, whereas fuzzy logic is mostly used to reduce this combinatorial problem in corresponding data transcoding (it has not been investigated).

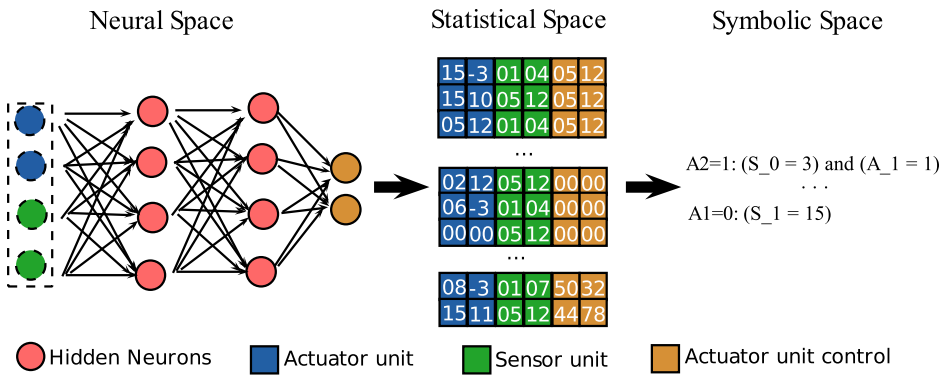


Figure 6.11 – Knowledge extraction for associative rules' inference.

6.4 EMMA System Integration

Artificial Neural Controller (ANC) integration in an EMMA framework can be performed by several approaches. One approach directly deploys an Artificial Neural Network (ANN) over Active Resource Middleware (ARM) resources. Each neuron is represented by a resource and connected by agents. A feedforward computation flow is performed by agent activations, which compute new target neuron values. This connectionist implementation has an important cost in relation to network communication, computation and memory storage. It can be used only for small sizes of a single-layered ANN (less than five neurons), such as a linear control of an actuator. For ANC, which requires a lot of neurons, as previously studied, there is another approach that consists of the use of ARM service components. ANC is encapsulated within them in order to execute all computations at a low level of execution instead of the agent interpreter service.

This section, which presents an ANC service component and its integration in an ARM. ANC framework, is separated into two parts. ANC runtime mechanisms are executed on nodes, whereas behavior classifiers are built remotely on the supervisor platform. Service choreography between nodes, in order to drive online actuator resources and behavior classifier deployment for the training phase, is performed by ARM agents.

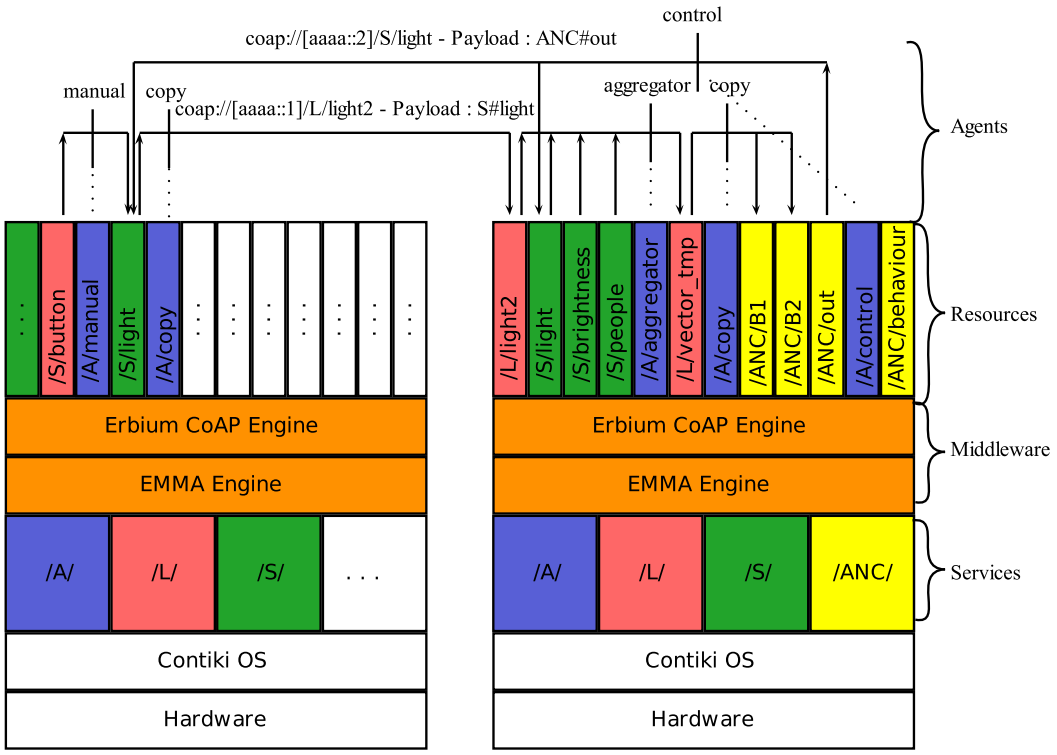


Figure 6.12 – ANC component implementation on an ARM

6.4.1 Controller Service

ANC is encapsulated in the service component to select the best behavior classifier according to its cumulative success rate of output prediction. Table 6.2 provides a set of resources produced by an ANC component. It is composed of:

- *Behavior register*, which contains the weighted matrix for all ANN that are remotely trained and deployed by the supervisor.
- *Out* resource, which provides the output vector for the ANC that is connected to the control actuator resources by the ARM agents.
- *Behavior* resource, which provides the current selected behavior ID and a manual interface to change the current behavior classifier.
- *Record* resource, which is an input data storage when stored behavior classifiers are not efficient to properly predict correct outputs.

The feedforward function of Eq. (6.2) is executed for each stored behavior classifier at each input event. According to the current input vector and its previous predictions, its cumulative success counter is incremented or decremented. According to Eq. 6.3, the ANC decides whether to update the resource with the prediction vector of best behavior classifiers or to store input data in the record resource for new classifier training. It should be noted that behavior classifiers can have different input vectors; however, the output vector has to be the same (and should be a part of all input vectors).

Resource URI	Methods	Description
/ANC/out	GET	Output vector
/ANC/behavior	GET/PUT	Current behavior used
/ANC/record	GET/PUT	Last input stream record
/ANC/Bx	GET/PUT/POST/DELETE	Behavior classifiers

Table 6.2 – Resource list of ANC services.

The ANC service component has a very lightweight implementation because it simply contains the feedforward function and selector. Storage functionalities are already provided by the Contiki OS. Hence, memory footprints in Table 6.3 are very small and efficient for the microcontroller target.

Modules	RAM	Program memory
emma-anc	561 B	11,635 B

Table 6.3 – Memory footprints of an ANC service on the Contiki OS.

6.4.2 Service Choreography

ANC Service Choreography (SC) is required to establish local control links between actuators, sensors and an ANC, as well as deploying behavior classifiers. Obviously, the training mechanisms are not executed directly on the nodes due to their computation limitations. Even if the training processes are delegated to the supervisor, the classifiers are installed and executed on the node, such as in relation to Service Choreography (SC) that are published over ARM resources.

6.4.2.1 Local Control

Local control links provide input vectors to behavior classifiers and transmit control outputs to actuator resources, as illustrated in Figure 6.12. For efficient implementation reasons, inputs and outputs of an ANC are vectors, which must be prepared by agents. Therefore, all input (and also output) resources are buffered locally on an ANC hosting node. Publish-subscribe agents copy the required remote resources to temporary local ones. Then, each of them is aggregated by another agent to be sent to each behavior classifier input. If they require different inputs, several *aggregator* agents are used. Conversely, output vectors are split into local temporary resources before they are sent to actuator resources in the proper format. These last transmission agents adapt Constrained Application Protocol (COAP) requests, while their payload is in relation to the actuator's Web service interface.

6.4.2.2 Remote Training

Behavior classifiers are trained on supervisor devices with recorded input data from nodes. In the current implementation, training algorithms are executed on Octave. When an ANC cannot control outputs due to the lack of behavior classifiers, it stores data until such time that the record buffer is full or input data become predictable again. At which point, these data are pushed onto record resources whose agents listen in order to transmit them to the supervisor of the Web service interface. Following new behavior classifier training, the supervisor creates new behavior resources on the ANC and pushes the learnt weighted matrix. Hence, it is automatically included in the ANC scheduler.

6.4.2.3 Initial Deployment

Agents are responsible for the connection establishment between resources of sensors, actuators and the ANC. They perform small processing for data aggregation, value range adaptation etc. They are also responsible for data collection for behavior classifier offline training and their deployment in relation to the ANC. All of these agents are deployable, as presented in Chapter 5, thanks to deployment agents. Moreover, classifier behavior can be carried out by such agents in order to be automatically installed on an ANC during the node initialization and the configuration step.

6.5 Summary

The proposed agent-based architecture is a first step towards an Artificial Neural Controller (ANC) for a smart home system design. Such approaches capture human behavior by learning about environment usages in order to automatically drive it. Instead of defining logical rules, which are limited to symbolic terms, correlations between current input states and the desired next states are extracted empirically. The proposal focuses on multi-behavior learning when the environment use is different according to the context, such as single user or multiple user, comfortable or saving-energy mode etc.). A single statistical classifier is not suitable for learning about such a situation, although it is possible theoretically. Therefore, an ANC behavior is learnt separately and selected automatically according to the best response in relation to the last input states. Meanwhile, topical discusses about knowledge transfer are concerned with generating approximative classifiers by extracting logical functioning rules relating to the management environment.

The integration on an ARM with an EMMA framework uses agents to delegate the training phase to a remote supervisor, whereas actuator control is performed locally, such as in the case of service choreography. Its implementation is suitable on microcontrollers, but real use cases have not yet been presented due to a lack of industrial partnership at the beginning of this project. However, these preliminary implementations show interesting results for the application of such kinds of technology on home automation. The main limitations concern an efficient sensory environment for building critical classifiers.

Interesting phenomena appear when there is several instances of context-switching during the training phase. An ANC firstly learns to be a bad classifier, which has some success in terms of predictions, but not enough to be eligible, as in the case of the selected behavior classifier. Hence, it will be replaced by several new ones, which are more specialized in terms of the different input streams of each context. In such situations, the ANC learns about its classifiers, just like the dichotomy approach towards roughly to specialized classifiers. An analogy with human behavior can be observed when humans learn several things simultaneously. They will try to address all tasks as much as they can until they discover how to split jobs into several modes of training.

Chapter 7

Neural Voting Procedure[1]

Si voter changeait quelque chose il y a longtemps que ça serait interdit.

If vote changed anything it would have been prohibited a long time ago.

Michel Colucci

Contents

[1]Published in Clement Duhart, Michel Cotsaftis, and Cyrille Bertelle. "Lightweight Distributed Adaptive Algorithm for Voting Procedures by Using Network Average Consensus". English. In: *PRIMA 2013: Principles and Practice of Multi-Agent Systems*. Volume 8291. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, pages 421–428. ISBN: 978-3-642-44926-0. DOI: 10.1007/978-3-642-44927-7_30

7.1 Introduction

Inside the Multi Agent System (MAS) community, the Consensus Seeking (CS) problem has been an attractive topic of research for a long time. Recent emphasis has been on using new approaches through the Network Average Consensus (NAC) framework. It has been studied initially in relation to Dynamic System (DS) and Control Theory (CT), particularly in nonholonomic systems, with topics including synchronization of coupled oscillators, flocking theory, fast consensus, rendezvous in space or distributed formation control around issues concerning continuous vs. discrete time consensus and undirected or directed graphs [OSFM07; RBA07] and references herein. Several contributions have demonstrated the insensitivity of NAC convergence according to given conditions for noised communication [KM07], time delay with switching topologies [OSM04; RB04; RB05] and network costs in real applications with communication failures that are localized in time [KM09; PBEA07]. After robust study, convergence time has been intensively investigated [HM05] in relation to network topology (regular lattice, random, small-world [OSFM07; OS05] and free-scale networks [WG08]) to define a performance indicator λ_2, known as algebraic connectivity. Finally, other update schemes are currently studied for finite time convergence [Cor06; WX10] and the definition of a framework for general function operators [Cor08]. This brief outline is not comprehensive regarding the amount of work on NAC, but illustrates its possibility for new kinds of distributed algorithms in MAS, especially for Voting Procedures (VP).

This chapter focuses on preference aggregation to allow a set of agents to make a common, consistent and unique decision in dynamic and fully distributed networks with constrained communication capacity. From its good properties in terms of robustness in relation to switching communication topologies and transmission time delay issues, distributed data fusion is performed by parallel NACs. As it has an asymptotic approximation of consensus equilibrium, finite time convergence is an issue. The proposed Voting Procedures (VP) solves this with a multi-scale adaptive algorithm to ensure likewise discrete result values. This kind of algorithm neither requires advanced semantics nor complex exchange protocols. Moreover, it requires simple mathematical operations, which are easily performable by microcontrollers. Therefore, its use over a WSAN is investigated in terms of allowing smart objects to select common system parameters or mode arbitration, such as comfort or energy-saving. The summary is organized as follows. Section 7.2 presents Voting Procedures (VP) architecture, as well as the theoretical background and implementation requirements. Section 7.3 analyzes its execution according to a number of nodes and choices, along with presenting a discussion around its alignment property. Its integration in an EMMA system with examples of possible uses are detailed in Section 7.4. A conclusion and possible next steps are provided in Section 7.5.

7.2 Voting Procedure Architecture

In terms of Voting Procedures (VP), a set of agents must select the best candidate, called a profile, by aggregating preferences. The orchestration of decisions over a Wireless Sensor and Actor Network (WSAN) is undesirable, as discussed in previous chapters. Therefore, aggregation and decision-making have to be distributed across the agents in order to guarantee consistency without involving external mechanisms. Network Average Consensus (NAC) is used as a distributed aggregation operator, while the proposed algorithm exploits its properties to define the distributed decision function.

7.2.1 Theoretical Background

The network is defined with topology G = (V,E) in which each agent only communicates with its neighboring agents $\Upsilon_i = \{j \in V : \{i,j\} \in E\}$, which are inside a range of radius R. It is assumed that communications are symmetrical and can be performed simultaneously by broadcast. Network agents are randomly distributed and constrained in their communication capacity, which prevents large payloads and a multi-exchange protocol.

Each node has its own preference order, which must be aggregated with those of the other nodes to represent the whole network preference order. The preferential model uses a utility function to numerically model the preference order for each node. Each of these utility functions are aggregated by executing multi-NAC on them to build up an unique aggregated utility function on which decision rules are applied. This work proposes a distributed algorithm that guarantees convergence to an unique preference order produced by the consensus. This allows nodes to take the same decision at the end of the algorithm without using extra mechanisms to ensure consistency and uniformity of node decisions.

7.2.1.1 Preference Model

The agent community is composed of N nodes, which must select a common profile from among a set $\rho_0, \rho_1 ... \rho_M$, of size M, without the centralization of all node preferences. The preference notation is defined, such that A is preferred to B by the node i, which is denoted $A \underset{i}{\succ} B$. It is assumed that each node can define a partial order of its profile preferences, which are modeled by a strictly monotonic discrete utility function, denoted $u^i(t)$, where $u^i_j(t)$ is the utility value of profile ρ_j for the node i at time t and $\dot{u}^i_j = \lim_{t \mapsto +\infty} u^i_j(t)$.

A profile preference partial order is representative of each node i, where it is established for all of them, as defined in Eq. (7.1).

$$\exists j / \ \forall k, \rho_j \succ \rho_k \Longleftrightarrow \exists j / \ \forall i,k, \rho_j \underset{i}{\succ} \rho_k \Longleftrightarrow \exists j / \forall i,k, \dot{u}^i_j > \dot{u}^i_k \tag{7.1}$$

7.2.1.2 Aggregation Process

The preference aggregation consists of computing the mean value of each profile utility assigned by each agent in Eq. (7.1). Based on the discrete utility function $u^i(0)$ of each node i, aggregation process builds utility function $\dot{u}$, which represents the preferences of the whole network. As this utility function is discrete, the aggregation process can be executed independently on each utility value. The global aggregation process is composed of a set of elementary aggregations by using the NAC algorithm [RBA05]. In Eq. (7.2), the M-uple $(\dot{u}_0, ..., \dot{u}_M)$ represents the result values of the consensus on each profile j, whose final value $\dot{u}_j$ is equal to the means of the utilities $u^i_j(0)$ weighted by their relevant node importance w^i. Therefore, as the NAC algorithm is a decentralized algorithm, the global aggregation process is also decentralized because it is composed of M NAC, which is executed simultaneously. The NAC algorithm has an asymptotic convergence with the initial value of a node's utility by using the gradient descent algorithm with the error approximation ε.

$$\dot{u} = (\dot{u}_0, ..., \dot{u}_M) = \frac{1}{N} \sum w^i u^i_j(0) \Longrightarrow \forall i, j | \dot{u}_j - \dot{u}^i_j | < \varepsilon \tag{7.2}$$

The analytic form of the iterative algorithm NAC is remembered in Eq. (7.3) under the necessary condition of convergence given in [OSFM07] : $\forall i, w^i < \frac{1}{\Delta}$ with $\Delta = max(\#\Upsilon_i)$ the graph degree of G and $\#\Upsilon_i$ the degree of vertex i .

$$u^i_j(t+1) = u^i_j(t) - w^i \sum_{k=0}^{N} (u^i_j(t) - u^k_j(t)) \tag{7.3}$$

By applying NAC to each profile, each node obtains the same profile utility function according to error interval $]-\varepsilon+\dot{u}; \dot{u}+\varepsilon[$ produced by the asymptotic convergence. Unfortunately, this error interval is unknown and cannot be computed analytically without full knowledge of the problem data. This theory limitation is bypassed by using an upper bound function as discussed in Section 7.2.2.1.

7.2.1.3 Distributed Decision Rules

After execution of the aggregation process, the nodes must choose the best profile. However, NAC has an asymptotic convergence in an infinite time process. Therefore, the aggregation process cannot converge with the ideal average values of the utility functions. Some agents may reach a lower value, whereas others may reach an upper one bounded by ε. According to the convergence error, some cases cannot be discriminated enough. Consequently, the theorem 7.1 stipulates conditions under which the interval between utility values is large enough to choose the elected profile without ambiguity. Otherwise, the definition 7.1 defines

the profile equivalence for ambiguous profile utility values. Therefore, arbitrary decisions can be made to ensure the unity of the decision for each node.

Theorem 7.1.
If, and only if, there exists a profile utility function $\dot{u_j^i}$ *greater than any other profile utility function* $\dot{u_k^i}$ *over* 4ε *for any decision makers* i *estimated by network average consensus, then this profile is preferred to any other profile by all agents.*

$$\forall i,k,\exists j/\dot{u_j^i} > \dot{u_k^i} \text{ and } |\dot{u_j^i} - \dot{u_k^i}| > 4\varepsilon \Longleftrightarrow \rho = \rho^i = j \tag{7.4}$$

■

Proof. Network average consensus converges with the average value $\dot{u}$ of node initial value $u(0)$ for any network topology if the convergence rate of the gradient descent algorithm is limited to $\frac{1}{\Delta}$ with $\Delta = \max(\#\Upsilon_i)$ [OSFM07]. The average value $\dot{u}$ is unique by definition, such that the set of the utility order $u_0^i(0) > ... > u_M^i(0)$ of each node i will evolve to an unique average utility order $\dot{u}_k > ... > \dot{u}_j,\ k,j \in [0,\ M]$ according to convergence error ε, which defines an error interval $]-\varepsilon+\dot{u};\dot{u}+\varepsilon[$. Assume that there exists a node, which prefers a profile j different from the other node preferences. This can happen if, and only if, an interval error of this preference utility is juxtaposed with another interval error of another near preference utility: $]-\varepsilon+\dot{u}_j,\dot{u}_j+\varepsilon[\ \bigcap\ ]-\varepsilon+\dot{u}_k,\dot{u}_k+\varepsilon[\neq\emptyset$ with $\dot{u}_j^i \in]-\varepsilon+\dot{u}_j,\dot{u}_j+\varepsilon[$ and $\dot{u}_k^i \in]-\varepsilon+\dot{u}_k,\dot{u}_k+\varepsilon[$. If $|\dot{u}_j^i - \dot{u}_k^i| > 4\varepsilon$, using triangle inequality that $]-\varepsilon+\dot{u}_j,\dot{u}_j+\varepsilon[\ \bigcap\ ]-\varepsilon+\dot{u}_k,\dot{u}_k+\varepsilon[=\emptyset$. Thus, if a node has a different preference to the other nodes, its preference utility cannot be spaced from its other preference utility by more than 4 ε. □

Definition 7.1.
Two profiles ρ_i and ρ_j are defined as equivalent if the distance between their estimated aggregated utility value is inferior to 4ε. ■

$$\rho_j \sim \rho_k \Longleftrightarrow |\dot{u_j^i} - \dot{u_k^i}| < 4\varepsilon \tag{7.5}$$

Based on the theorem 7.1 and the definition 7.1, a decision rule for the uniqueness of a profile is defined, as in Eq. (7.6). The theorem 7.1 guarantees that it is possible to build a unique aggregated order of preferences, which is representative of each node's profile preference where a sufficient discriminant interval exists. The definition 7.1 defines the equivalence state for partial order. Finally, space U^i is the set of the best equivalent utility on which an arbitrary rule is applied to select one unique, common, consistent and homogeneous profile ρ.

$$\rho = \begin{cases} U^i \in \mathbb{N}\ /\ j,k \in U^i \text{ if } \forall l,i,\ \rho_j \underset{i}{\sim} \rho_k \underset{i}{\succ} \rho_l \\ \rho_i = \min(U^i) \end{cases} \tag{7.6}$$

7.2.2 Implementation Arrangements

7.2.2.1 Finite Time Convergence

The aforementioned theoretical background presents a convergence process under infinite time. It can also be assumed that it is possible to determine a cone distance ε around ideal average value $\dot{u}_j$, where each estimated value $\dot{u_j^i}$ is inside it. Since each node cannot know the utility values of other nodes, however, it is impossible for them to know when consensus is reached. The Banach fixed-point theorem can give a bounded time for reaching convergence with the consensus because NAC uses the gradient descent algorithm. It has a Q-linear convergence and is a k-lipschitzienne function, as defined in Eq. (7.7).

$$|u_j^i(t+1) - \dot{u}_j| \leq k\ |u_j^i(t) - \dot{u}_j|\ ,\ k \in [0,1] \Longrightarrow \varepsilon \leq \frac{k^n}{(1-k)}[\max(\dot{u_j^i}) - \min(\dot{u_j^i})] \tag{7.7}$$

Unfortunately, and to the best of our knowledge, there is no analytical method to determine value k without any knowledge about the network topology (such as algebraic connectivity λ_2) and the initial values of each node. Indeed, based on the initial value, the Laplacian matrix graph and the algebraic value λ_2, it is possible to determine an upper bounded value of required iterations [Cor06] : $\frac{||Lu_j(0)||_2}{\lambda_2}$, when L is the Laplacian graph.

7.2.2.2 Multi-Scale Adaptive Accuracy

The implementation is limited by hardware accuracy and a finite time requirement. In this case, the proposed algorithm must refine its equivalence to the definition 7.1 between two utility values and its stop condition. The value ε determines the maximum error interval for a given utility value for all nodes. Then, after enough iterations, their equivalence in profile, according to ε and encoding base q, is defined by Eq. (7.8).

$$\forall i, \exists j,k\ /\ \lceil \frac{(\dot{u_j^i} - \dot{u_k^i})}{(q+1)\ \varepsilon} \rfloor = 0 \Longrightarrow \rho_j \sim \rho_k \tag{7.8}$$

Two profiles can be equivalent without equality of their utility values, according to ε, because of a lack of significant digits. In order to increase the accuracy of the estimated utility value, the refining process executes NAC with a decreasing error interval $\varepsilon = \frac{\varepsilon}{q}$. The refining process must continue until the utility values are discriminant enough to allow each node to extract the same U^i space, which is defined in Eq. (7.6). As the node decisions must be consistent when defining the common U space, the process of estimation-refining must stop if each node has its set U distant enough from the other utility values, according to the ε error interval and the encoding limitations, as defined in Eq. (7.9).

$$\forall i, \exists j,k\ /\ \dot{u_j^i} \in U^i, \dot{u_k^i} \notin U^i, |\dot{u_j^i} - \dot{u_k^i}| > q\varepsilon + \varepsilon \Longleftrightarrow \forall i, U = U^i \tag{7.9}$$

7.2.2.3 Voting Procedure Algorithm

The voting procedure algorithm is composed of the two previously presented steps: the aggregation of utility values and the selection of the best profile. During the aggregation step, the nodes communicate with each other by broadcasting their current values of utility $u^i(t)$ to their neighbors. The number of required iterations is fixed according to the criterion in Eq. (7.7). This step is iterated with a decreasing value ε until the value of aggregated utility is spaced enough according to the criterion of convergence defined in Eq. (7.8). Therefore, the decision function is applied by each node without the risk of an inconsistent decision. In algorithm 7.1, the decision is based on the *max* operator, although any Ordered Weighted Average (OWA) operator can be used.

Algorithm 7.1: Voting procedure algorithm executed on each node i.

Data: $u^i(0)$, w^i
Result: ρ

begin
  $\varepsilon \leftarrow 1$;
  repeat
    $\varepsilon \leftarrow \varepsilon * 0.1$;
    repeat
      foreach *j=1...M* **do**
        $u_j^i(t+1) \leftarrow u_j^i(t) - w^i \sum_{k=0}^{N} [u_j^i(t) - u_j^k(t)]$
      end foreach
    until $\frac{k^n}{(1-k)}[\max(\dot{u_j^i}) - \min(\dot{u_j^i})] < \frac{\varepsilon}{2}$;
    $[value, k] \leftarrow max(\dot{u^i})$
    DONE $\leftarrow$ true;
    foreach *j=1...M* **do**
      if $|value - \dot{u_j^i}| < (q+1)\ \varepsilon$ *and* $\left\lceil \frac{(value - \dot{u_j^i})}{(q+1)\ \varepsilon} \right\rceil \neq 0$ **then**
        DONE $\leftarrow$ false;
      end if
    end foreach
  until *! DONE*;
  [value, ρ] = $\max(\dot{u^i})$;
  foreach *j=1...M* **do**
    if $\lceil \frac{(value - \dot{u_j^i})}{(q+1)\ \varepsilon} \rfloor = 0$ **then**
      $\rho \leftarrow \min(\rho, j)$
    end if
  end foreach
end

7.3 Experimentations

This experimentation section analyzes the execution of the proposed Voting Procedures (VP) algorithm by studying a number of iterations required to reach consensus on the preference profile for all of the agents. This algorithm is adaptive to guarantee consistency in individual agent decisions. Therefore, the number of iterations depends on the number of aggregation loops, which depends on the network topology and the initial distribution of agent preferences. Firstly, a single experimentation is presented for the better understanding of its execution process. Convergence time is then studied according to the number of agents and profiles.

7.3.1 Execution Example

Experimentation is composed of 25 agents distributed in a random network graph. Each node can communicate with an average of five neighbors randomly assigned; there are no multi-hop communications. The algebraic connectivity of the network graph, illustrated in Figure 7.1, is $\lambda_2(L) = 0.93$.

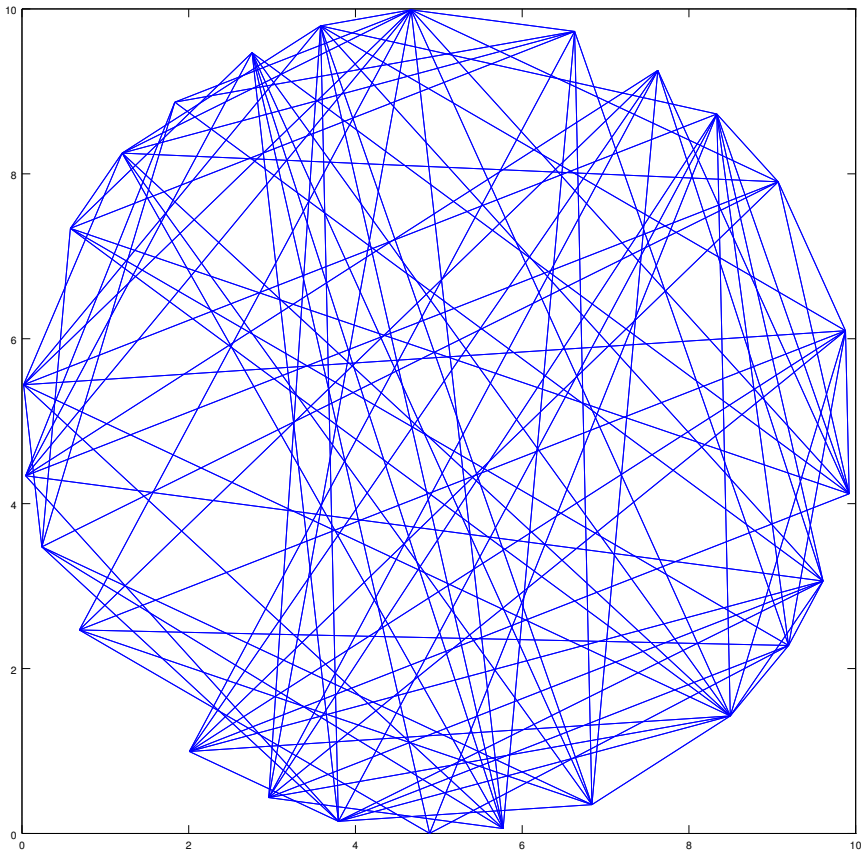


Figure 7.1 – Network graph of 25 nodes randomly connected to five neighbors.

Each agent defines its preference order for 10 profiles, such as for node 1 $\rho_3 \succ_1 \rho_2 \succ_1 ... \succ_1 \rho_6$. In this experimentation, the preference order is valued by a linear utility function, such as $\rho_0 \succ_i \rho_1 \Longleftrightarrow u_0^i(0) = u_1^i(0) + 1$. However, any kind of monotonic, discrete and bounded function can be used to convert logical order into utility functions. Figure 7.2 presents the utility functions $u^i(0)$ of each node i at the beginning of the VP algorithm in order to ensure the preferred profile is equal to 10, the second one to 9 etc. Each plot corresponds to the utility function of one node with $u_j^i(0) \in [0\ ;\ 10]$ with t=0. They represent the associated utility value in the y-axis of profile ρ_j in the x-axis. Initial preference order is generated randomly for each node. It is stated that numeric base is 10, given that it is an algorithm parameter for accuracy estimation (q=10).

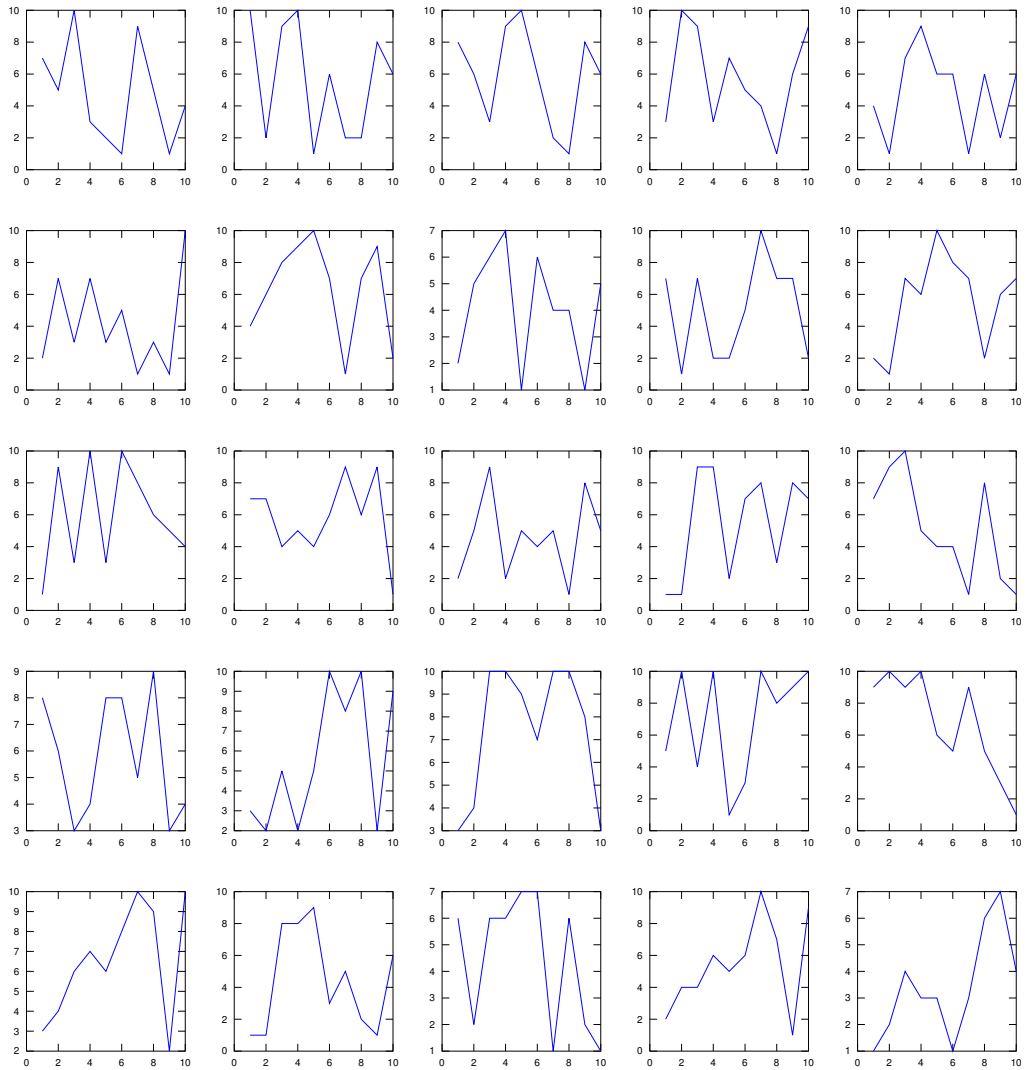


Figure 7.2 – Initial node's utility functions $u^i(0)$ of 25 nodes at the start of VP.

Figure 7.3 illustrates the evolution of a set of utility values for each profile j for each node i $u_j^i(t)$. This figure shows the evolution of 10 parallel Multi Network Average Consensus (MNAC) executions on each profile between all nodes, as defined in Eq. (7.3). Its Q-linear convergence reaches the final aggregated utility function $\dot{u}$ after 47 iterations. It can be observed that the aggregation process is repeated three times after iteration 19 and after iteration 35. Indeed, the number of NAC iterations computed by Eq. (7.7), according to the initial fixed accuracy ε, is not enough because the utility values are not significantly spaced to be discriminated by all nodes without ambiguity, in line with Eq. (7.4) and Eq. (7.9).

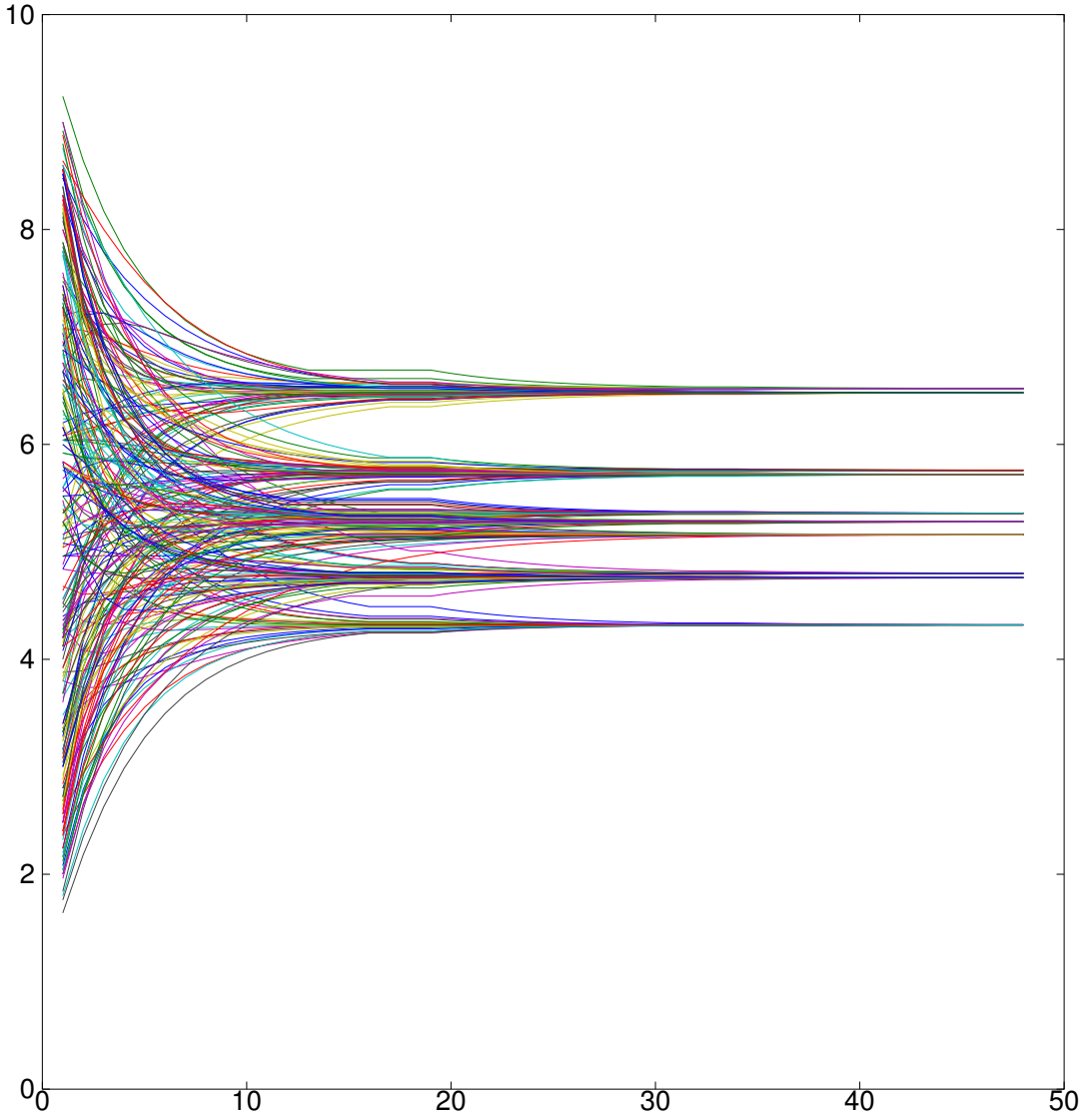


Figure 7.3 – Profile utility of each node convergence until reaching the final aggregated one $\dot{u}$.

Figure 7.4 presents extremum error intervals between utility values. It shows the error interval of utility values for each profile. The 10 error intervals of the Multi Network Average Consensus (MNAC) are plotted in addition to the fixed accuracy ε for the current aggregation loop. It can be observed that interval error decreases linearly because it is a Q-linear algorithm. When the error interval of all the NAC is lower than this maximal possible accuracy for the current aggregation loop and when the utility values are not sufficiently spaced, this accuracy decreases by a factor of 10 in order to start the aggregation loop again. Otherwise, the process stops when all the utility values are sufficiently spaced according to all the NAC error intervals.

This trial and error process of decreasing accuracy parameter minimizes the number of iterations needed to reach consensus. This approach provides interesting results in terms of reducing the number of network communications between agents.

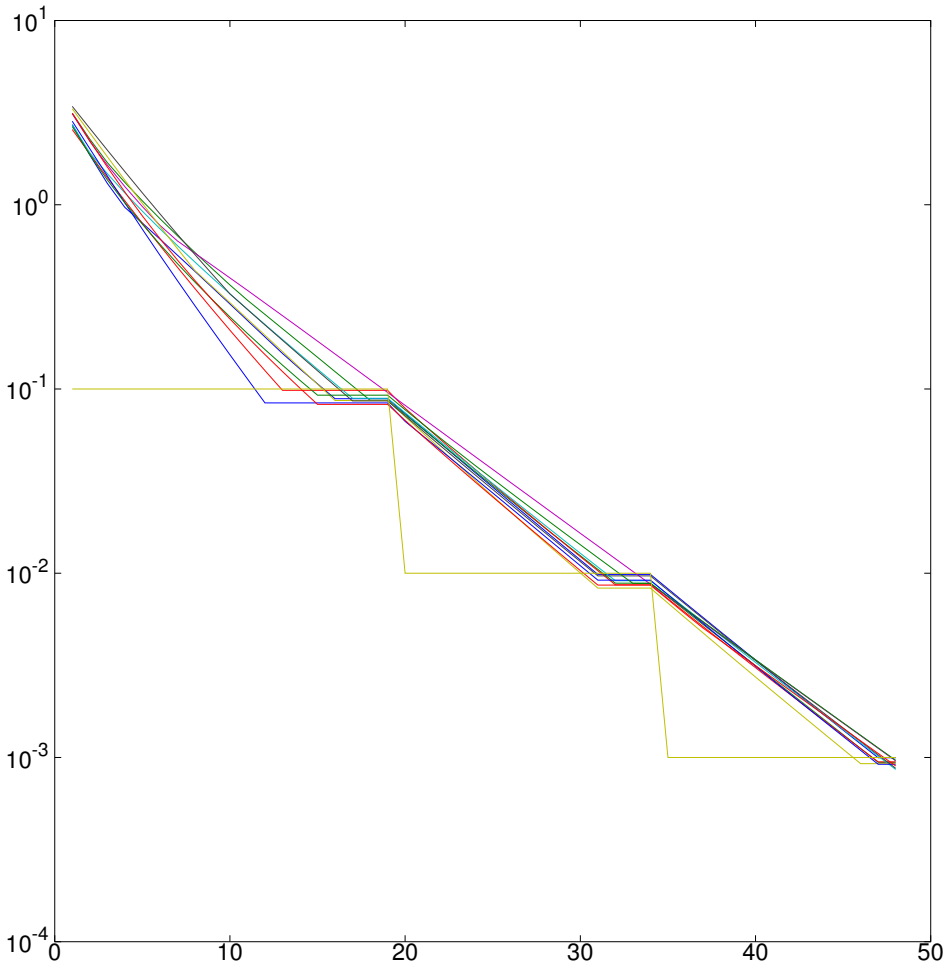


Figure 7.4 – Interval error ε between nodes for each profile according to NAC execution.

Finally, the aggregated utility function $\dot{u}$ is reached for each agent according to the last accuracy parameter value ε. In this experimentation, all nodes obtain the same utility function with a maximal error of 10^{-3} for each profile. Each agent can apply its Ordered Weighted Average (OWA) decision operator (in this simulation *max* operator) to select the elected profile. Their decisions are guaranteed to be consistent, given that the maximal error of aggregation is lower than the space between the utility values of a non-equivalent profile. To conclude, given that there is no method to determine the number of required iterations to reach consensus at the algorithm in the beginning, this proposal iteratively estimates an error interval according to the evolution of utility values. It solves the problem of inconsistent decisions due to the impossibility of determining the convergence error without full knowledge of the network topology and initial utility values [Cor06]. Moreover, this approach empirically reduces the number of iterations and, therefore, the agent communications.

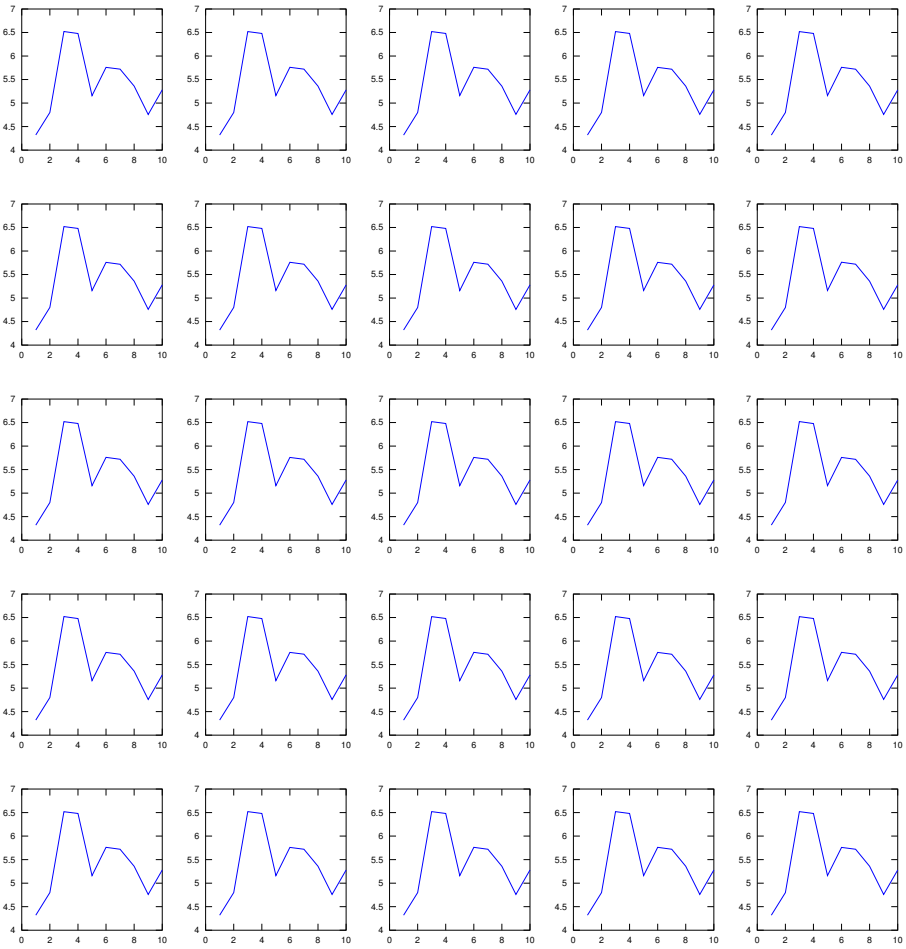


Figure 7.5 – Aggregated utility function $\dot{u}$ is reached for all agents according to the last ε value.

7.3.2 Time Convergence

The previous section presented, in detail, the aggregation step of the Voting Procedures (VP) algorithm. Figure 7.6 shows the number of iterations required to solve the VP problem according to the number of nodes and profiles. It can be observed that the node number increases linearly with the time of convergence, whereas the number of profiles has a limited impact. However, the convergence time is also impacted by the network topology, as demonstrated by NAC studies on random graph, small-world and scale-free topology [OSFM07; OS05; WG08]. Moreover, irregularities in Figure 7.6 are consequences of algorithm adaptation according to the initial distribution of utility values to ensure consistency in agent decisions.

To conclude, this kind of algorithm is very interesting with regard to solving the VP problem because it has fault tolerance properties concerning time delay, lost packets and switching topology [OSM04; RB04; RB05]. Its complexity increases linearly with the number of agents, while it only requires neighbor communications.

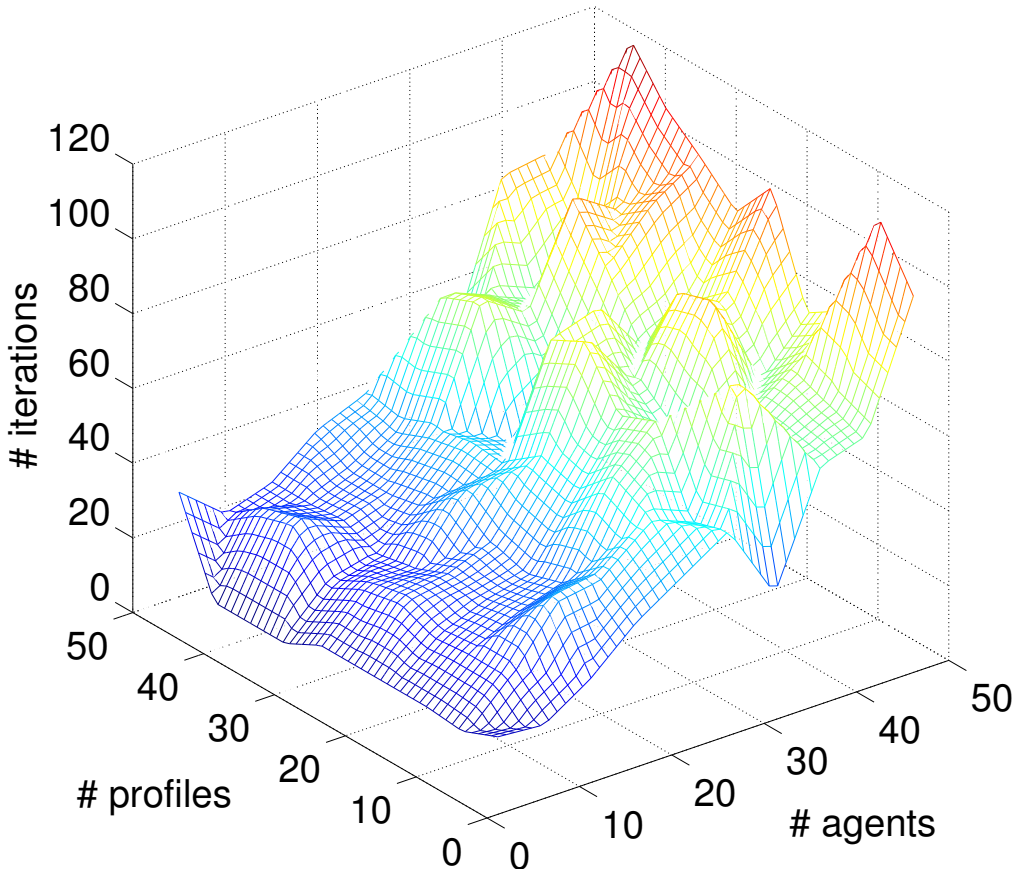


Figure 7.6 – Number of iterations according to the number of nodes and channels $\dot{u}$.

7.3.3 Alignment Property Discussions

The proposed Voting Procedures (VP) algorithm is composed of several parallel Network Average Consensuss (NACs). This last algorithm is based on the alignment property of the gradient descent type. It means that each agent asymptotically reaches an agreement according to its initial values [OSFM07]. Hence, the aggregation process for utility values inherits this property from an NAC. The main consequence is that all agents retain their utility values, which converge with a common type according to those of other agents. This is a direct consequence of the lack of centralization, which is the main purpose of this proposal. However, it means that, if an agent does not update its utility value according to NAC, all other agents converge in order to be in agreement with the former agent. While it can be an interesting feature of a veto policy, it can be also hazardous in the case of Byzantine agents.

7.3.3.1 Veto Policy

A veto policy forbids the selection of a particular profile by an agent. It is used when a profile cannot be accepted by an agent or when a human wishes to manually select the profile without bypassing the aggregation process. It simply has to ensure that its utility value is higher than all other profiles on one node. The use of this technique in such situations avoids the need to manually control all system components. The system runs autonomously, but its parameter and functioning modes are selected manually, rather than collectively.

Such operations are operated by bypassing the NAC update scheme of the interested profile, thereby keeping its value lower or higher than all the others.

7.3.3.2 Byzantine Threat

A Byzantine threat is produced by usurpation of a malicious agent. It is a common threat for system security, which is difficult to address. This Voting Procedures (VP) algorithm is extremely sensitive to such threats because of its alignment property. Moreover, the lack of information centralization avoids the use of analysis techniques in order to detect these situations. However, NAC has a Q-linear convergence, which is easily observable by comparing transmitted agent utility values. Therefore, a Byzantine agent that tries to alter how a VP is solved is easily detectable by observing a non-Q-linear convergence.

A veto policy provides this VP with the ability to select or exclude a particular profile. However, its use does not allow the system to detect a Byzantine threat because a veto policy and Byzantine agents have an identical response behavior regarding Q-linear convergence.

7.4 EMMA System Integration

7.4.1 Voting Procedure Choreography

The Voting Procedures (VP) can be implemented inside a service container, such as the ANC previously presented, or directly by a combination of agents, as illustrated in Figure 7.7.

The different mathematical operators used in the algorithm 7.1 are sufficiently simple in order to be performed by the agent over the middleware. For each profile x, there is an agent NAC_x that broadcasts and adds the preference value to those of its neighbors. The NAC_x agents of the different nodes are synchronized approximately at their first data reception. As the algorithm is of the fault tolerance type regarding lost packets and time delay, there is no need for it to be perfectly synchronized. This operation is repeated until the convergences are reached. The *aggregation* agent decides whether the aggregation phase must be repeated with a lower epsilon value. If it is not necessary, the *selection* agent applies the decision rules to select the elected profile.

The Service Choreography (SC) of the VP is then connected to other SCs or to node services through agents, which fill in the resources of preference values for each profile to start the algorithm.

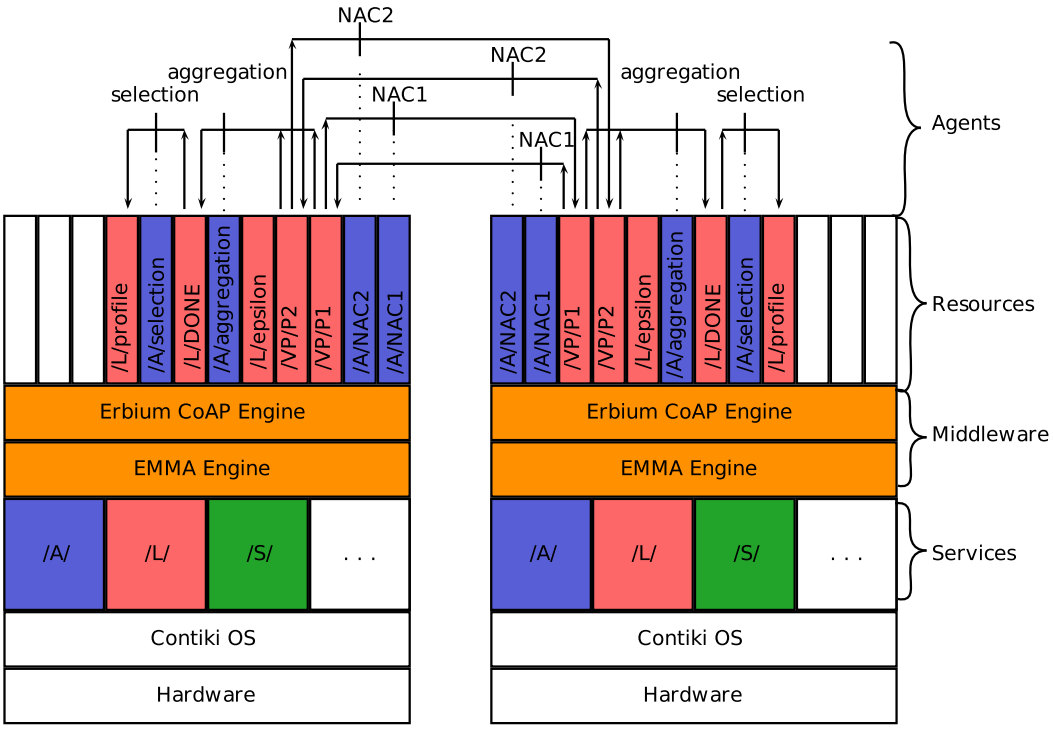


Figure 7.7 – Voting procedure algorithm implementation across ARM agents

7.4.2 Application Scenarios

Although the VP has several implications in Ambient Intelligence (AmI), it does not directly control the environment. It is used as an internal component to synchronize choices between different subsystems. Hence, there is a non-exhaustive list of possible scenarios for its use.

1. *Artificial Neural Controller (ANC) synchronization* is a scenario in which the different ANCs deployed to manage the actuators have a synchronized policy of behavior learning and scheduling. Instead of applying their own local decision about which classifier to use or the necessity to learn a new one, the ANC builds a preference order for their decision. They are aggregated and selected by the VP in order to apply the same collective decision. Hence, the AmI learns to manage the global environment contexts by the local and synchronized learning of each ANC. This scenario can be implemented by connecting the SC of a VP to those of an ANC through an agent between the resource $/ANC/behaviors/$ and the resources $/VP/P_x$.

2. *Functioning mode selection* can be used by other SCs that require common parameters. In a situation involving an Energy Management System (EMS), there are several profiles of energy consumption, such as a comfort mode without energy consumption restrictions and, conversely, an economic mode. Under a strict limitation of the total energy consumption in the smart home, the different appliances can vote for the common functioning mode in order to release energy when there is an energy shortage or surge. If the heaters do not have enough power to maintain the minimal temperature, they ask for more power by insisting that the other appliances change their functioning modes.

3. *Human Computer Interface (HCI)* is an example of an application to make several users come to an agreement. For example, in a traditional home, the user settings for brightness or temperature are changed by each user, with the negotiation is operated directly by them. The VP can be connected between the control thresholds of actuators and the HCI of each user. Hence, the system will automatically operate the preference aggregation to select the most popular settings or an intermediate value according to the VP configuration. In the future, smart environments will be more and more responsible in relation to multi-user preference management. In some situations, the lack of multi-user preference management results in a significant loss of time for users, such as in hospitals when several patients cannot agree on the right temperature.

7.5 Summary

This chapter presents a new distributed algorithm for a Voting Procedures (VP) in an Multi Agent System (MAS). Its implementation on an EMMA framework is performed in the same way as a Service Choreography (SC) by a combination of agents. Several scenarios of its use are proposed to illustrate the importance of this algorithm. It is composed of a distributed preferences aggregation step with adaptive refining in line with the requirements of a selection step. The latter is realized by each agent individually selecting the profile preferred by the agent community, based on a common decision function. As consistency of the result is ensured at the aggregation step, the selection step does not require any mechanism to check whether all the agents have made the same decision.

Alignment property of Network Average Consensus (NAC) allows the chosen preference model to be extended by adding a veto possibility. A veto is a node's behavior, which allows a profile to be excluded by the VP. If any node keeps the utility value of a profile unchanged and low, the community will converge asymptotically towards an agreement of exclusion. But, a Byzantine agent can hit back by using this property to change the profile that is preferred by the community. In all cases, nodes decisions stay consistent with or without Byzantine agents in the community. Moreover, the proposed Voting Procedures (VP) algorithm is interesting due to its good properties inherited from Network Average Consensus (NAC) research concerning mobile network constraints, such as robustness in switching topology and large-scale networks, as well as time delay. This algorithm is based on a utility function relating to preferences for each node and does not require any extra information about network topology other than a list of its neighbors. As this algorithm converges asymptotically and exponentially, as observed on Figure 7.3, it requires several iterations and exchanges over the network. But these exchanges remain localized in the local neighbors' area, which allows network load balancing. As these exchanges have a very small payload with a null overhead (it only contains the utility vector of the node), this algorithm is useful to Wireless Sensor and Actor Network (WSAN) applications, which are extremely limited by their communication capacity.

However, the aggregation operator studied in this chapter is the arithmetic mean for minimum or maximum operator decision functions. Future work will focus on studying this algorithm in relation to general functions for reaching consensus during the aggregation step. J.Cortes [Cor08] proposes several new NAC update schemes, which will be explored, such as the minimum and maximum operators, the harmonic mean, the geometric mean, the arithmetic mean and the root mean square. A study of an ordered weighted average operator as a generic decision function will also be extended to algorithm genericity.

Part IV

MIT Medialab Experience

Chapter 8

Ambient Sound Recognition[1]

Vous ne comprenez rien tant que vous ne l'avez pas appris par plus d'une approche.

You don't understand anything until you learn it more than one way.

Marvin Minsky

Contents

[1]Submission in Brian Mayton, Gershon Dublon, Spencer Russell, Evan F. Lynch, Vasant Ramasubramanian, Donald Derek Haddad, Clement Duhart, Qiansheng Li, Glorianna Davenport, and Joseph A. Paradiso. "Deploying the Living Observatory: From Environmental Sensor Network to Networked Sensory Landscape". In: Submission in ACM. 2016

8.1 Tidmarsh Living Observatory

Tidmarsh is a 600-acre old cranberry farm located in the south of Massachusetts. The Living Observatory project is a restoration program on this site closely situated to marshland. Industrial farms and artificial structures have profound impacts at an ecological and biodiversity level. The establishment of studies and techniques to restore the natural environment is an important challenge facing the future if the industrial development of the last century is to be erased. This study focuses on the monitoring and analysis, in real time, of those natural processes whose aim is to help nature to return, as well as providing feedback on the theoretical models of restoration processes. The wireless sensor network (WSN) deployed by the Responsive Environment Group (REG) in the MIT Media Lab provides a large set of data for analyzing the evolution of micro and macro climates, as well as their impact on life cycles and the re-emergence of bacteria, flora and animals. The intention has been to design a sensing engine that is able to geolocalize animals based on their calls, which are recorded in real time using microphones distributed on the site.

8.1.1 Environment Sensing and Network

The sensing network is composed of sensor nodes distributed over a tight grid under hostile conditions, in which the hilly land is continuously evolving with high humidity conditions. Therefore, the whole system must be waterproof and powered with enough 3-AAA batteries for one year, given that the sensor nodes are only accessible during sporadic periods of the year. The base station, powered by solar panels, collects the different data, which are forwarded to the Internet over long-range Wi-Fi to a fiber connection located at the entry of the site. A set of secondary stations distributed over a large grid is powered by solar panels in order to provide real-time pictures and audio streams of the site.

8.1.1: Base station

8.1.2: Station-2

8.1.3: Sensor node

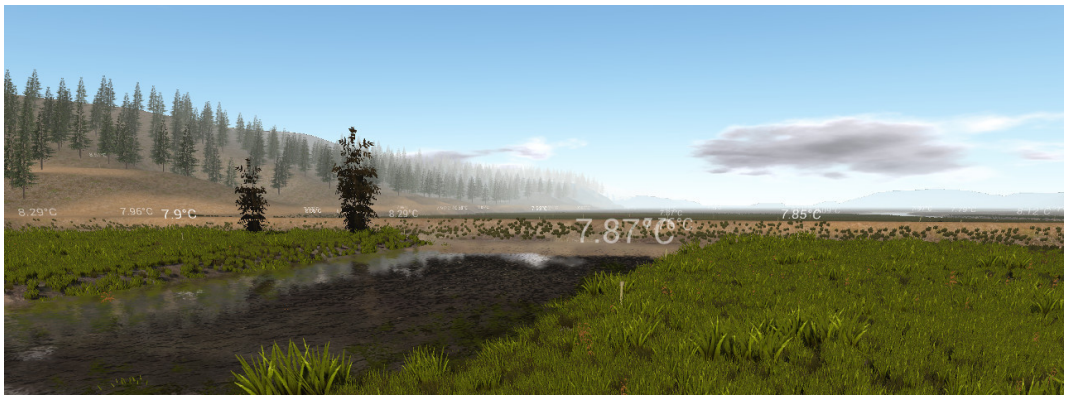
8.1.4: Microphone

Figure 8.1 – WSN on Tidmarsh.

8.1.2 Data Visualization: Cross-Reality and Sonification

Data visualization is an important challenge regarding the huge amount of data produced by the Living Observatory project. Traditionally, data reduction is applied to extract interesting information; however, the REG is more interested in looking for alternative data representation instead of their extraction. On the one hand, all data must be conserved, while, on the other hand, a new kind of user experience is investigated thanks to cross-reality and sonification.

Cross-reality is a bidirectional meeting between virtual reality and reality itself. Data representation is replaced by perception in which augmented reality is used in the environment *in situ*, while virtual reality provides environment ubiquity. Hence, in Figure 8.2.1, the gaming platform provides a real-time experience of the Tidmarsh site, whereas the use of Google Glass notices information about current and previous states of an observed area on the site. An additional component named sonification is interested in data representation, due to the auditory sense. Instead of printing all the information about visual supports, the auditory sense is used to augment human perception field. Musical contexts are generated according to the data received from Tidmarsh to indicate any anomaly or environment context in real time.



8.2.1: Unity virtual environment

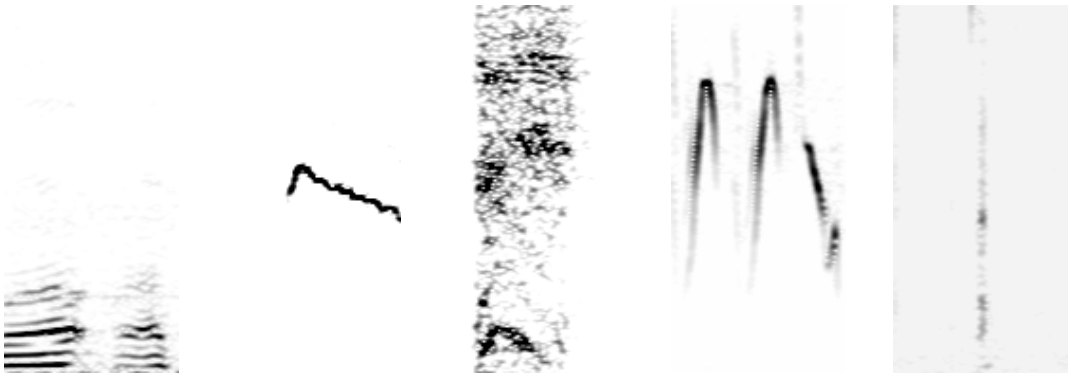
8.2.2: Real environment in Tidmarsh

Figure 8.2 – Cross-reality to visualize in situ marsh environment evolution.

8.1.3 Towards Wildlife Geolocalization

Wildlife geolocalization is another kind of sensing, although it cannot be operated by specialized sensors. Based on the audio streams, however, the system must be able to identify animal calls in order to localize them in the microphone range areas and represent them in the virtual environment. While it may be easy to discriminate a frog sound from a human voice, it is a much more difficult challenge to differentiate between bird calls.

Figure 8.3 presents the different categories of animal call spectrograms. In Brand et al. [Bra08], the authors present different approaches, such as Bayesian classifiers, neural networks and Gaussian mixture models, that work well to discriminate between bird calls according to their group. They concluded that, in 2008, there was no available method that works for any of them, although deep learning had not been evaluated at that stage. If the human voice seems complicated, with a range between 70 Hz and 4.4 Khz, it is composed of a fundamental frequency around 70 Hz for men and 100 Hz for women within a set of formants. In Bevis et al. [Bev10], the complexity of bird calls is significantly revealed with some birds able to change their voice. Therefore, a single bird must be considered in relation to several voices. The term voice or song is preferred, while their vocal structure includes different kinds of phonemes in a large frequency range, complex rhythmics, syllables and even sentences from some birds. Recent studies have demonstrated major improvements in bird call classifications by self-extraction of such features in an unsupervised fashion [SP14].



8.3.1: Human 8.3.2: Blue jay 8.3.3: Crow 8.3.4: Sparrow 8.3.5: Frog

Figure 8.3 – Example of the main different categories of spectrograms (frequency between 50 Hz and 7 kHz vs. time of 500 ms) at Tidmarsh.

TidZam project has focused on vocal snapshots without considering time-dependent identification regarding such complexity. The result is that the system can make mistakes given that some birds can change their voice. The system will provide several proposals without the capacity to exclude some of them.

8.2 TidZam Contribution

The contribution entitled TidZam is a Web-based platform connected to Tidmarsh's audio stream in order to analyze, in real time, the different microphone channels. The different detected events, such as bird songs, mechanical noise and human voice, are pushed and stored in Tidmarsh's information system 2014hypermedia. TidZam is composed of a classifier player, a Web-based HCI and a deep learning stack.

8.2.1 Architecture Overview

Figure 8.4 presents a schematic overview of the system. The input multi-channel stream is provided by an Icecast platform and transformed into spectrogram samples (50 Hz to 15 kHz over 500 ms) overlapped by factor of 0.5 to avoid missing cut samples. They are presented to a pull of binary classifiers, which are triggered if their learnt signal is matched (animal call, mechanical noise etc.). Each input sample crosses a band-pass filter, which depends on each classifier, in order to determine a region of interest to analyze. This step reduces the complexity of the classifier task because of a focus on the frequency range of the signal. Finally, a decision function determines whether the classifier outputs are consistent over the time window of 1.25 s (four samples). If none of the classifiers is triggered, the signal is considered unknown and stored in the record database until it is identified by experts.

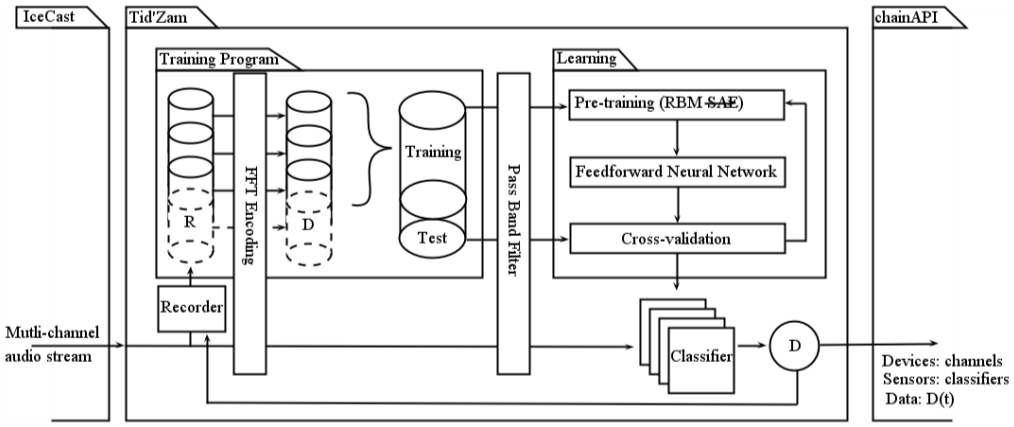


Figure 8.4 – TidZam architecture overview.

Based on the Web interface, experts listen to unknown signals in order to associate them with a category or create a new one. For example, if a new kind of sound is registered, such as a airplane noise, the expert can create a new data set to create a new classifier. In the same way, a classifier could be improved by adding new samples. The learning process is composed of two main steps. Firstly, the system builds a training program in order to generate effective training and evaluation data sets, which are transmitted to the deep learning stack to create

or improve a classifier. The classifiers are autonomous and, therefore, can be loaded and unloaded online.

8.2.2 Signal Footprint Background

Classical approaches focus on the discrimination between different classes, which must be known during the learning process. However, the nature of TidZam project does not allow all kinds of audio signal present on the site to be known, such as the return of a new animal, in this the study. Therefore, the deep learning stack is not interested in the discrimination of the signals but in their identification.

Each classifier is trained to learn a signal footprint, which is evaluated according to its ability to generate samples that are able to imitate the initial signal. Firstly, the system determines the features that are useful for imitating the signal beforehand, which will refined and transformed into a classifier. In this approach, the deep learning stack looks for efficient feature spaces to represent the signal by unsupervised learning. When the feature space is smaller than the initial space, the signal is compressed thanks to a dimension reduction. The size of feature space depends of the complexity of the input signal in terms of information redundancy, which is similar to that in Principal Component Analysis (PCA), but also in terms of feature diversity, such as the same bird having different voices. Therefore, the feature space is initially fixed and refined according to the difficulty of the system in finding a good feature space to reconstruct the signal. This operation can be stacked by applying an additional layer, in which the input space is the previously learnt feature space for extracting a higher level of feature abstraction. For the complexity of audio signals at Tidmarsh, but also for speaker recognition, a single layer of feature space is enough. In turn, these feature spaces are used to initialize a neural network that is trained to build a binary classifier. This step is very light given that the initial configuration of the neural weights is already able to match the input signal. The weights are updated in order to fix output neurons when the input does not contain the signal. Instead of recognizing the signal, the neural network must learn what the signal is not, which is a simpler task regarding its initial configuration. This step can be compared to a kind of crystallization or shear-off of the neural network. Hence, the learning process is simple, controllable by observing feature space and fast, as is presented in the experimental section 8.4.

A training program is composed of three different mixed data sets. The first one is only composed of samples containing the signal, which should be learnt in order to determine the feature space. The second one mixes samples from the signal and a random distribution of other samples for the crystallization process. Finally, the evaluation data set is composed of unused samples of the signal and all available samples from other signals.

8.3 Deep Learning Stack

Deep learning refers to neural networks composed of a large number of neural layers and the set of techniques used to train them. G. Hinton et al. [HS06] present a framework for reducing the dimensionality of data by using a neural network with a fully unsupervised approach. It is composed of three main steps. Firstly, the authors use a restricted Boltzmann machine in order to self-extract relevant features of the model instead of defining them manually. As such, these learnt features are used to initialize a neural network trained to reconstruct the input layer in its output layer. This model, which is a stacked autoencoder, produces a good neural configuration to model the input data to the extent that it is able to imitate it. Finally, lightweight supervised learning, which uses a gradient descent algorithm, is applied to fix class labels in the output layer.

8.3.1 Restricted Boltzmann Machine

An Restricted Boltzman Machine (RBM) is a two-layer network in which binary stochastic visible inputs are connected to binary stochastic features in an hidden layer,as illustrated in Figure 8.5. The two layers are connected thanks to symmetric weights $w_{i,j}$, which form a joint distribution $E(v,h)$, which is defined as an energy function, as in Eq. 8.1, to model probability of correlations between visible and hidden layers. This model seeks to compute successively visible and hidden layers until reaching a thermal equilibrium in order to apply the update function 8.2, known as constructive divergence (CD). When the energy function is stabilized for a sample, the hidden layer or feature space reaches its capacity to model the visible layer with its current configuration. Then the weights can be updated according to the learning rate ε in order to improve the reconstruction for the current sample.

$$E(v,h) = -\sum_{i \in input} b_i v_i - \sum_{f \in features} b_j h_j - \sum_{i,j} v_i h_j w_{ij} \tag{8.1}$$

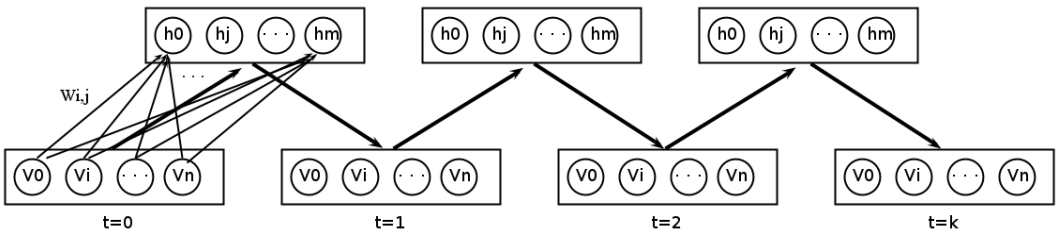
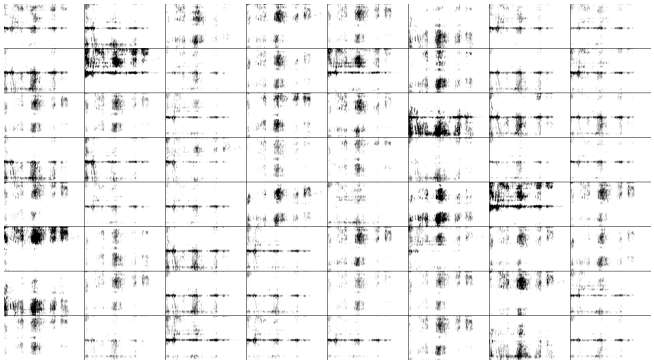


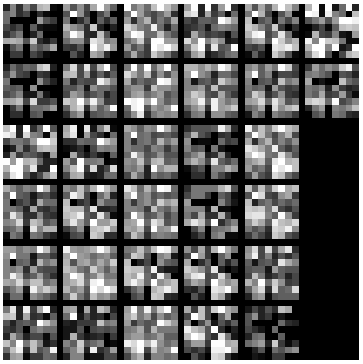
Figure 8.5 – Restricted Boltzmann machine execution until thermal equilibrium.

$$\begin{aligned} \Delta w_{ij} &= \varepsilon(< v_i h_j >^0_{input} - < v_i h_j >^\infty_{recon}) \\ \text{where } < v_i h_j >^t &= \frac{1}{1 + e^{\sum w_{ij} v_j + c_i}} \end{aligned} \tag{8.2}$$

Each neuron in a hidden layer represents a particular feature, which can be visualized like a picture. Figure 8.6 represents two stacked layers of an RBM in which each small square represents the values of weights connected to one feature neuron. In the first layer, it can be observed that these low features look like the input, whereas the second layer is an higher level of abstraction. The fraction dropout technique has been used to avoid coadaptation of feature detectors (several neurons learn the same feature as in the second layer) by randomly forgetting the feature neurons in the update scheme [Hin+12].



8.6.1: L1



8.6.2: L2

Figure 8.6 – Two-layer feature space on a human voice signal.

The $CD-t$ algorithm is based on the computation of the t Markov chain transitions to estimate sample reconstruction. According to t, it may be time-consuming because it ought to be operated for each sample and could be useless if the parameter configuration (v,h) is not good. Hence, $CD-1$ is preferred most of the time because it provides a good approximation of the gradient of a likelihood objective function. However, $CD-10$ considerably improves the feature detectors, which is required when the signals of the different classes have similar features. Therefore, $CD-1$ is fast and effective in most situations, although $CD-10$ is preferred for difficult tasks, such as bird song classification. In Tieleman et al. [Tie08], a trick is proposed, called persistent constructive divergence, to reduce time computation in $CD-t$ with $t>1$ using mini-batches to successively compute hidden and visible layers. The authors demonstrate that it works well and is a very good trade-off between the accuracy of feature detectors and computation time, which has subsequently been confirmed by the TidZam project.

8.3.2 Stacked Autoencoder

An Stacked Auto-Encoder (SAE) is deep network architecture composed of stacked feature spaces to reduce data dimensions for supervised learning. The feature space learnt by an RBM is unrolled into the stacked architecture of Figure 8.7. Then the backpropagation of error derivatives is applied across encoding and

decoding stacks to link the feature spaces in order to reconstruct the input signal. As the feature spaces are already close to the signal model, this learning is very light. Finally, the encoding stack of the SAE is extracted in order to build the classifier, into which is added a last layer for output class labels. This last training step is very fast, such that the Multi-Layer Perceptron (MLP) only needs to learn the labeled neurons based on the catalogue of features in the last encoding layer.

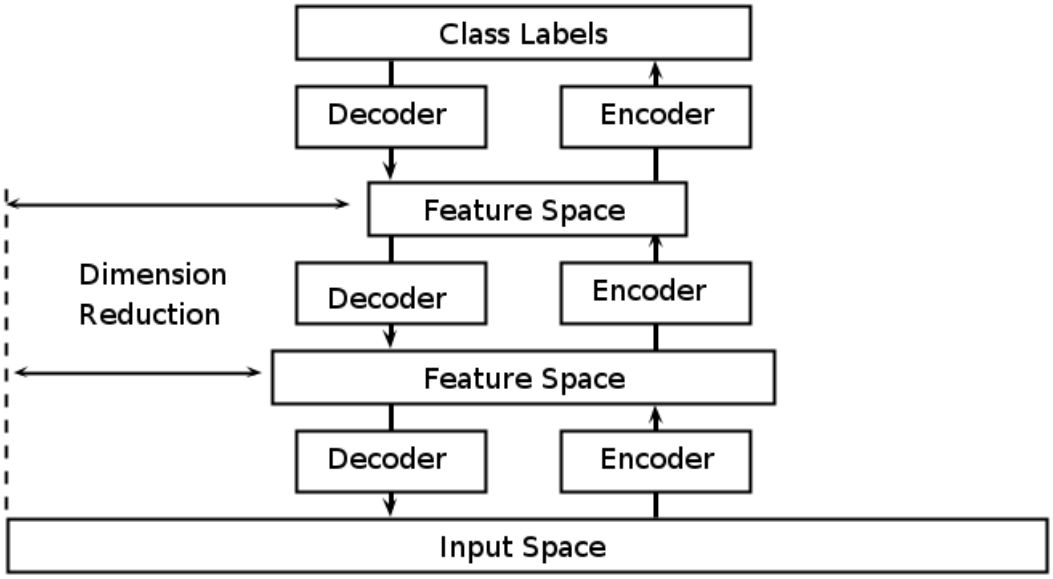


Figure 8.7 – Stacked autoencoder architecture.

8.3.3 Classifier Decision Function

The different microphone streams are filtered and presented in parallel to the pull of classifiers trained to evaluate the probability $P(A)$ in the presence of A, as well as $P(\neg A)$ in its absence, in the input signals. Hence, $C(A) = P(A) - \alpha P(\neg A)$ reveals the confidence degree of the classifier. If no classifier is triggered according to $C(A)$, the signal is considered unknown and extracted dynamically to the database in order to be associated with a label by an expert.

Finally, the decision function of TidZam is a sigmoid function of the weighted average of the classifier response $C(A)$ over T samples for each classifier A, as defined in Eq. 8.3. Indeed, some signals with a low feature complexity, such as frog calls, can easily trigger other classifier features. This stochastic function limits the number of false detections by checking whether this signal continues to trigger the classifier after a number of time translations due to signal overlapping.

$$D(A) = \frac{1}{1 + e^{\frac{1}{2} - \sum_i^T \frac{P(A) - \alpha P(\neg A)}{T}}} - \frac{1}{2} \tag{8.3}$$

8.4 Experimentations

TidZam, which has been evaluated in relation to two different applications, has been crucial to the improvement of its deep learning stacks by a cross-experimentation. The complexity of bird and animal calls has required the study of feature space dimension, given that birds have very different characteristics of phonemes and rhythmics in different frequency ranges. Meanwhile, speaker recognition needs to optimize learning parameters in the SAE in order to distinguish human voice footprints, given that they share a common feature space.

The following results have been produced on an independent evaluation data set composed of samples that are not used during the learning process. They are, at least, composed of 100 positive samples according to their availability and all other samples, including negative ones. Therefore, the evaluation of signal overestimation is always more accurate than its underestimation.

8.4.1 Wildlife Recognition

Wildlife recognition has been applied to the *53 All-American Bird Songs and Calls* and *Peterson Field Guides* recording databases, in which samples have been extracted lasting over four minutes across 97 tracks. Table 8.1 is a subset of the presented training results of Table 9.1 in the Appendix. It details the percentage of *underestimation* and *overestimation*, the layer structure of the neural classifiers, the *region of interest* on which the classifier is trained, and the *training time*. The presented results seem very good with an average training time of around two minutes and an error rate in the evaluation data set lower than 2%. However, their deployment in the real Tidmarsh environment has revealed some limitations regarding surrounding noises produced by rain, wind, microphone sensitivity etc. Hence, the results in real conditions are still under experimentation for evaluating these issues, as well as improving the signal preprocessing and the deep learning stack configuration in future works.

Classifiers	UE	OE	Structure	ROI	TT
Nothing	0.94	0.01	[16928 24 2]	50-14191 Hz	108 s
Cricket	0	0	[54188 16 2]	1003-14786 Hz	144 s
Crow	0	0.21	[25208 16 2]	503-6786 Hz	88 s
Egret	0	1.37	[25208 16 2]	503-6786 Hz	89 s
Frog	1.25	3.64	[54188 24 2]	1003-14786 Hz	150 s
Red woodpecker	0	0.94	[25208 16 2]	503-6786 Hz	101 s
White sparrow	0	0	[25208 16 2]	503-6786 Hz	66 s
...	...	...	...	...	...

Table 8.1 – TidZam results (%) for the wildlife recognition application.

8.4.2 Human Computer Interface

The interactive TidZam HCI provides training feedback mechanisms between users/experts and the neural system in order to improve the knowledge for the system and the users. On the one hand, the system can be used by users in order to learn different animal calls, whereas experts can create new classifiers or improve some of them when mistakes appear. Figure 8.8 presents an overview of the interface. On the right side, each input channel has an independent plot in which the classifier outputs $D(A)$ are printed in real time, as defined in Eq. 8.3: nothing, cricket, frog, crow etc. The different buttons on the bottom of the *classifier management* window can be used to create a new database of records, add currently played samples to an existing database, mark mistakes on a classifier or obtain information on loaded classifiers. The left side provides the event detection distribution in light of the playback of input sources. It can be observed that, in 75% of the time, the input signal is undetermined, which suggests the need to build new classifiers. In Figure 8.9, these recognition events are used to render wildlife in the virtual environment.

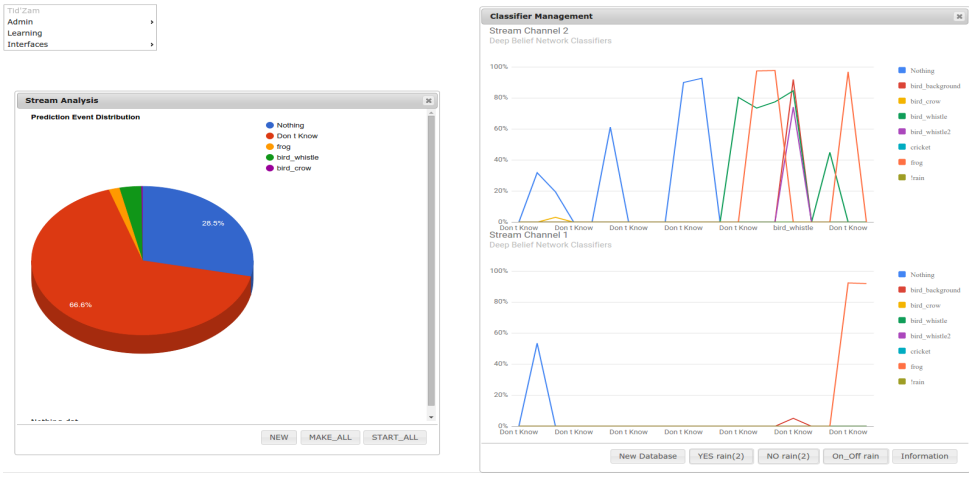


Figure 8.8 – Screenshot of TidZam Web administration interface.

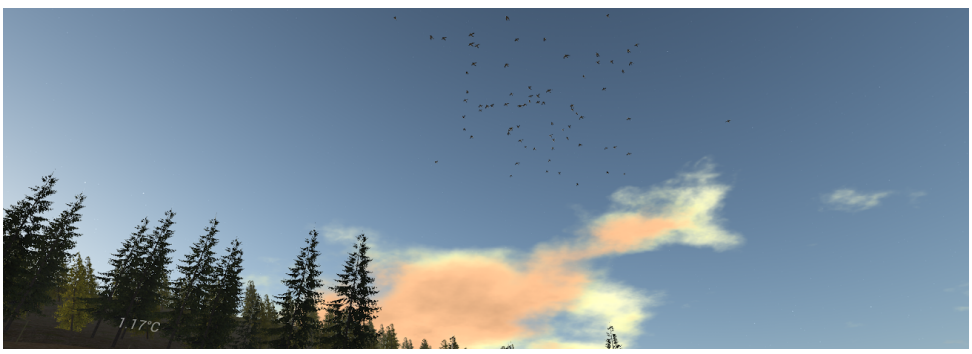


Figure 8.9 – Bird rendering in the (VR) based on TidZam detection.

8.4.3 Speaker Recognition

Speaker recognition has been prompted by discussions with Rebecca Kleinberger, a Ph.D. candidate at the MIT Media Lab and a specialist in voice and emotional analysis. During a meeting, different types of people interact in order to propose ideas and solve problems. Understanding people's mood should be a powerful tool for improving meeting efficiency. For the current study, experimentations were performed by allocating microphones to each speaker in order to determine who is speaking. TidZam requires only one microphone stream, which is analyzed by the classifiers to provide feedback in real time for the experimenters or for the members of the meeting.

The experimentation on speaker recognition was conducted in order to build statistical distribution over time in relation to the speaking time of each person. While people presented themselves around the table, TidZam recorded their voices in order to build their vocal footprints. This recording was possible due to a prebuilt classifier, which can recognizes *nothing*; in other words, when nobody is speaking. Therefore, the system is able to dynamically extract the samples. This classifier is a main key, given that it should not be sensitive to room acoustics and could be refined before the meeting.

Table 8.2 presents the results of TidZam in relation to the task of speaker recognition during a meeting of three men and three women. Their vocal footprints were created using the first two minutes of recording. From these records, 50 samples were extracted, of which 40 were allocated to training and 10 to its evaluation of (*training underestimation* and *training overestimation*. *Real underestimation* and *real overestimation* are evaluated on a meeting simulation of 30 minutes without rules (people can speak simultaneously). This preliminary experimentation is a *proof of concept*, which cannot provide any more information that TidZam can use on an additional application. Indeed, these results can be very different according to the proximity between speaker voices. However, the research community interested in speaker recognition has recently published results that indicate that the TidZam deep learning stack could see interesting developments [RRD15; Sai+13].

Classifiers	TUE	TOE	RUE	ROE	Structure	TT
#1 Man	0	4	2.7	9.4	[16560 24 2]	32 s
#2 Man	0	2.5	3.2	6.7	[16560 24 2]	41 s
#3 Man	0	0.5	5.1	7.8	[16560 24 2]	28 s
#4 Women	0	8	4.9	10.1	[16560 24 2]	66 s
#5 Women	10	12.5	15.3	12.8	[16560 24 2]	57 s
#6 Women	7.5	7	9.5	8.2	[16560 24 2]	48 s

Table 8.2 – TidZam results (%) for the speaker recognition experimentation.

8.5 Summary

In this chapter, deep learning technology has been investigated in order to build virtual sensors for wildlife detection in the outdoor environments linked to the Tidmarsh project. Based a set of distributed microphones, TidZam detects animal calls in order to timestamp, in real time, their localization in the microphone range. It has presented interesting results in terms of fast classifier training, while maintaining a relative low error rate, according to its task of wildlife detection. Given that the neural classifiers have a small size, they can be executed in parallel and in real time on different microphone channels. Their encapsulation into a Web-based framework facilitates the system integration into the network architecture of Tidmarsh. In addition, it has offered the possibility for designing interactive HCI; this offers to the user the ability to learn more about wildlife in Tidmarsh, as well the possibility of improving the TidZam sensors by upgrading or creating new classifiers.

This work has been conducted over a three-month period among the Responsive Environment Group (REG) at the MIT Medialab. It follows on from previous Ph.D. studies on Wireless Sensor and Actor Networks (WSANs) and Artificial Neural Controller (ANC), even when ANC runs on a dedicated server instead of nodes. The deployment of such technologies in outdoor environments, under real conditions, requires several iterations in order to validate the concept, evaluate its reliability and start the design for an integrated solution. The experience during this stay at Tidmarsh has revealed the complexity of outdoor environments that are open and intractable. In future work, the design of an embedded chipset, which is able to execute these tiny classifiers, will be investigated, given that it could offer the ability to analyze audio streams in situ and, in turn, emit event information over a WSAN instead of the audio streams. The trade-off between the energy consumption of these chipsets versus the energy consumption of the WSAN in streaming these audio channels would be the main problematic in this context.

Part V

Conclusion and perspectives

Chapter 9

Conclusion

Nous ne pouvons voir uniquement en avant qu'à courte distance mais nous pouvons y voir des choses à faire.

We can only see a short distance ahead but we can see plenty there that needs to be done.

Alan Turing [Tur50]

Contents

Since the last century, Artificial Intelligence (AI) has been a dream of human beings. From the first simulations of brain capacities, this research area has been motivated by its incredible potential in terms of future applications. It can be observed that, according to technological evolution, AI techniques increase in sophistication as they take less and less inspiration from biological systems. Hence, expert systems have been inspired by deductive language, Artificial Neural Network (ANN) has been inspired by the brain and Multi Agent System (MAS) has been inspired by society, amongst others. These inspirations have been possible thanks to the technological evolution, which has provided new abstraction ability. Nowadays, natural emergence and self-organized processes are copiously studied in Complex System (CS). The hope is that this will result in the application of evolving features to AI systems. In turn, new kinds of architecture can be discussed with a view to controlling such systems with self-x properties.

In this thesis, the Environment Monitoring and Management Agent (EMMA) framework has been proposed for the continuity of the AI impulse. It provides an abstraction layer over the Internet of Things (IoT) infrastructure for the purpose of designing distributed AI with self-x properties at the scale of Organic Computing (OC). Two algorithms are proposed in order to illustrate the design of such algorithms: an Artificial Neural Controller (ANC) and a neural Voting Procedures (VP). Both of them have been designed as first elements of a toolbox for the design of a future cybernetic brain distributed over a Wireless Sensor and Actor Network (WSAN).

9.1 An Organic Internet of Things Framework

The Internet of Things (IoT) is still an emerging concept promising the next technological revolution. Even if it is still difficult to predict when, how and and in what form its future applications will happen, the scientific community is already exploring new technologies in this vein. One major dimension is its inherent multidisciplinarity of embedded systems, wireless and lossy networks, and ubiquitous services that require independently advanced expertise. These different research areas are evolving very quickly and raise several challenges regarding end-to-end connectivity, large-scale networks, data heterogeneity, communication and security, along with their derivative constraints. EMMA proposal is a framework that considers these aspects in order to design multi-scale and hybrid IoT architecture, in which services are located indifferently in the cloud, at the gateway, and on mobile devices and appliances. It is assumed that a single entity would not be able to manage appliances, their services and the network.

Part II is composed of three chapters in order to present different abstractions for the different actors identified of the IoT: the manufacturers, the service providers and the infrastructure administrators. Chapter 3 presents architecture based on an Internet Protocol (IP) in order to abstract the network and locality of services for communications. Hence, a service can communicate indifferently with appliances and other services, which can be local, for example, in the same Personal Area Network (PAN) or remotely, such as over the Internet. Chapter 4 details with the EMMA middleware used for the Service Choreography (SC), along with its implementation by the manufacturers and the SC design model, which is based on a Petri network, for the service providers. Finally, Chapter 5 presents the methodology, the theoretical background and the implementation of the SC deployment over the EMMA appliances in order to preserve the network.

This abstraction stack opens up the research area for distributed Ambient Intelligence (AmI). Traditionally, this research area has not been correlated with the technological aspects of its hardware and network supports. Hence, the AI is located on a central computer, which manages the system as if it were external. This work has been motivated by the necessity to prevent such failure points caused by the distribution of intelligence over the system. In addition, any solution must be able to relocate its functions in case of failures in their execution supports, similar to brain plasticity. Hence, the different abstractions in the EMMA framework have been designed for OC in order to investigate a new bio-inspired scale for AI technologies. This proposal for an organic framework presents general architecture with one main self-x property: self-deployment. This elementary property can be used to allow the system to adapt itself to new usages, hardware and network failures, as well as colonize new appliances and evolve by rewriting rules. Although significant effort must be made to address the different properties required by a complete OC system, this proposal offers primary materials with which to derive other self-x properties.

9.2 Towards Neural Ambient Intelligence

The biological organisms have the features needed to adapt and evolve according to their environment, which makes them more powerful than any artificial system. The major difference resides in their design, while their high level functions are produced as a result of complex low-level interaction mechanisms. The significant prevalence of their possible organizations allows them to model their high level functions differently according to their environment. For example, brain plasticity allows a cognitive function to be moved when there are lesions in the brain's initial locality. Even if a much work is still required before simulating high-level cognitive functions, due to low-level bio-inspired execution supports, this abstraction has been investigated in relation to the EMMA framework. The concern is with the design of high-level functionality, which is based on the interaction rules of the EMMA framework for the purpose of exploiting its self-x properties at the execution level.

Part III is composed of two chapters, in order to propose two elementary functions for the design of a distributed cybernetic brain using EMMA. Chapter 6 presents an Artificial Neural Controller (ANC) for the control of outputs according to an input context, including the aforementioned outputs. The purpose, here, is that the system learns empirically how it should work through its usage, as operated by a human. A major conclusion is that it is preferable to use several learning classifiers, which are switched according to their usage context, instead of a big one. This requirement is also observed in the biological brain, in which there are several specialized brain areas that manage an identical set of organs or muscles. Their activations are switched according to the brain's requirements in order to prevent deadlocks between some brain functions, which can be observed in some diseases, such as epilepsy. Hence, Chapter 7 presents a Voting Procedures (VP) for the purpose of managing the synchronization of decisions among different appliances. This connectionist algorithm is fully distributed over the WSAN, which guarantees consistency in the appliance decisions. In addition, it has good properties in terms of fault tolerance regarding time delay, lossy communications and switching topology, while being strongly dependent on Byzantine threats.

These two algorithms are functionalities identified in the biological brain by the *motor cortex* and the *anterior cingulate cortex*. They are crucial components, respectively, in the learning process and the behavioral consistency of different brain regions. Even if these two elements have been successfully implemented, in addition to the plasticity feature, thanks to EMMA model, there is still a lot of work required to connect them. Indeed, the design of a complete cybernetic brain, based on a distributed neural network, in which the appliances learn different local behavior that are switched according to global voting behavior by the aforementioned appliances, should be lead to interesting and intensive work in the future.

9.3 Perspectives

A lot of general discussions and framework improvements can be derived from this thesis. A non-exhaustive list, given below, presents a selection of them:

- General Discussions
  - *Internet of Things (IoT)*: The scope of IoT technology is very wide and evolves at a pace. Hence, a framework cannot survive without collaborations if it is to be integrated in the landscape of the IoT. Even if much effort is made in relation to the use of standards, the EMMA framework requires support from a developer community if technological progress is to be pursued.
  - *Organic Computing (OC)*: The control of emergent processes is a general issue in OC. The use of a self-rewriting feature in the EMMA agent could be used to design evolutionary agents. This aspect has not been investigated in the thesis, but its study through the proposed Dynamic Network Agent (DNA)-Residual Network Agent (RNA) process ought to lead to a new perspective for AmI based on emergent processes.
  - *Neural ambient intelligence*: The proposed neural-based approach is bio-inspired by brain organization, thanks to the observation that its properties, like plasticity, is inherited from a neural interaction layer, which produces high-level cognitive functions. Hence, this work has tried to design an interaction model with neural properties of plasticity and high-level functions based on elementary interaction rules. In this context, new neural components should be investigated with the aim of building a general toolbox to design, deploy and execute distributed neural-based AI with self-x properties.
- Framework Improvements
  - *Optimization*: The agent implementation by JavaScript Object Notation (JSON) is very heavy regarding computation and memory capacities for target platforms. Indeed, the design of advanced SC, which is composed of numerous agents, has not been undertaken because of this issue. This major limitation should be bypassed through the use of a compressed binary encoding for the agents, in the same way as the data format Open Building Information Xchange (oBIX).
  - *Agent logic*: The agents are currently limited to basic logic and arithmetic operations. The addition of other operators should permit the use of other logical models, such as fuzzy, modal and symbolic logics. Their implementation should be performed easily on the Active Resource Middleware (ARM), due to the agent service, and on the Petri network simulator.

- *Model checking*: The adaptation of a Petri network in the design of SC cannot be used in the present state with model checking found in the literature. Therefore, a theoretical study should be performed to evaluate the implication of the EMMA model on a traditional numerical Petri network and its framework of analysis.
- *Web semantics*: The semantic schemes of nodes and SC are currently used to identify their services and resources. However, they could be used to infer compatibility schemes between different resources. For example, the design of agents between EMMA and other Constrained Application Protocol (COAP) nodes is currently performed manually, whereas the use of semantic schemes can automate this process. Therefore, description schemes, based on ontology technology, should be used to automatically adapt a SC to derive resources from the intended initial ones.

Future collaborations are discussed in order to extend this thesis work. *Thales Security* is a society specializing in the development of military projects. The team headed by Jean-Marc Lacroix has expressed a strong interest in EMMA technology from the perspective of their projects on the IoT. The developed administration tools and the secured and flexible ARM for SC have led to the start of a collaboration on experimental test beds. In addition, the experience of the *Massachusetts Institute of Technology (MIT) Media Lab* has been rich in terms of content and collaborations. A possible post-doc is discussed.

Appendix

During the stay at the MIT Media Lab, an electronic platform was designed for the purposes of building an electronic devil stick for running the EMMA middleware. The hardware was composed of an AVR 8-bit Atmega128rfa1 with an IEEE 802.15.4 transceiver, an MPU6050 accelerometer and an FTDI chipset for serial line interface. The different inputs and outputs, including the accelerometer, were connected to the EMMA resources. Therefore, all IOs were accessible over the Internet thanks to an AVR USB stick, which ensured 6LoWPAN <-> IPv6 translation. The administration software was implemented in the JavaScript language in order to configure the EMMA resources. The serial line was connected to internal shell interface to administrate and debug the Contiki OS or the EMMA application. More details on this open-source project are available at https://github.com/slash6475/DVS.

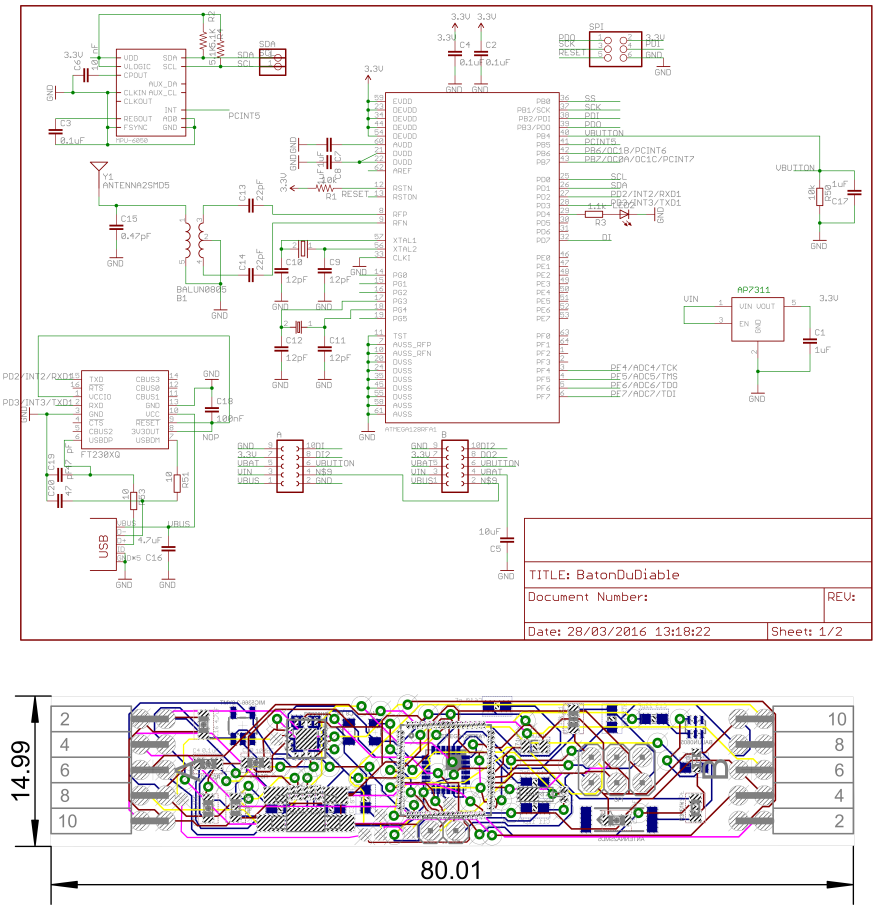


Figure 9.1 – DVS scheme and PCB for EMMA middleware (MIT Media Lab).

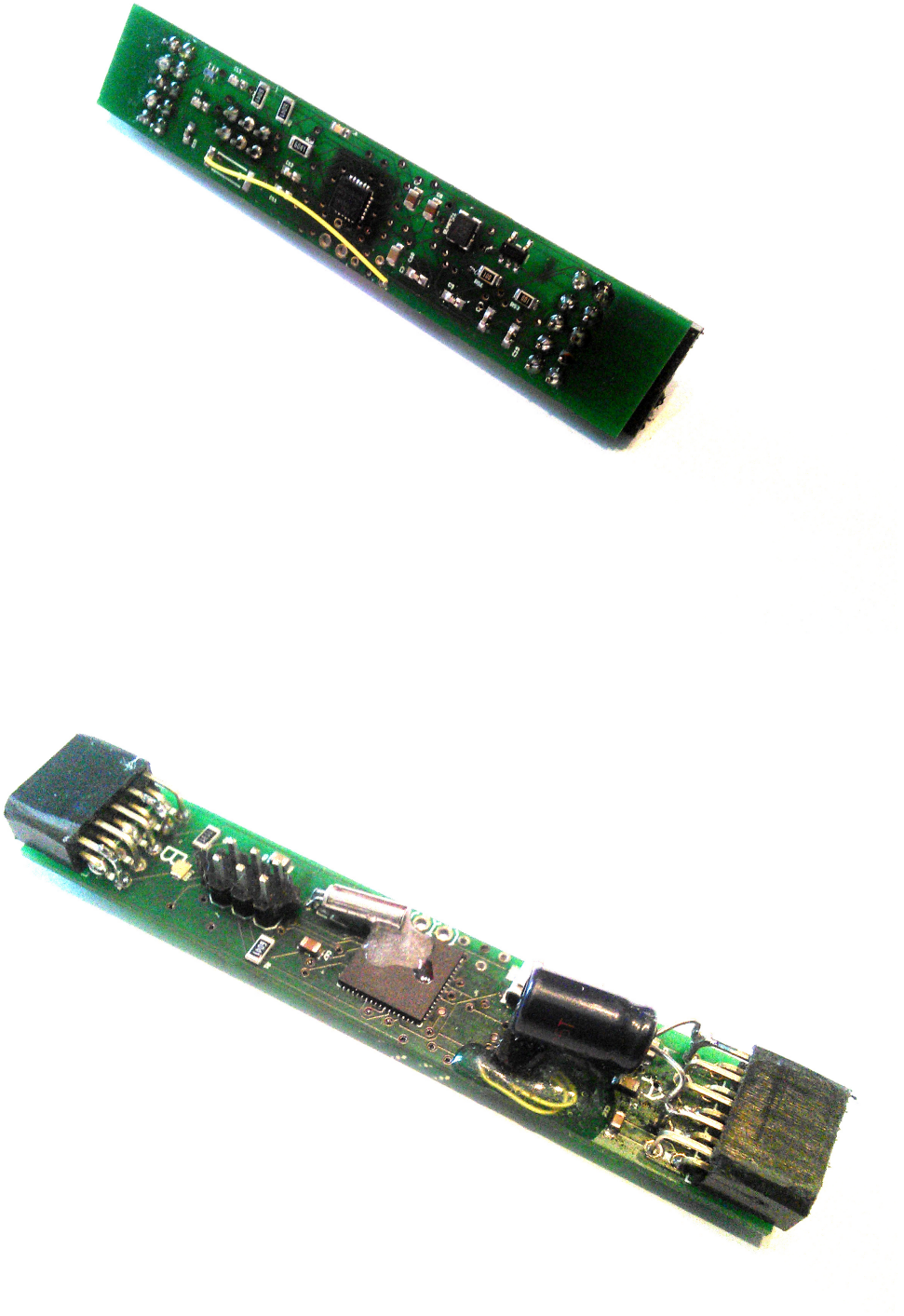


Figure 9.2 – DVS platform built for EMMA middleware (MIT Medialab).

Classifiers	UE	OE	Structure	RoI	TT
Nothing	0.94	0.01	[16928 24 2]	50-14191 Hz	108 s
Bachman sparrow	0	0.21	[25208 24 2]	503-6786 Hz	106 s
Black grosbeak	2.05	0.88	[25208 24 2]	503-6786 Hz	125 s
Bluebird	0	1.50	[25208 24 2]	503-6786 Hz	89 s
Blue jay	0	0.31	[25208 24 2]	503-6786 Hz	144 s
Bobolink	0.03	0.64	[25208 24 2]	503-6786 Hz	188 s
Canyon wren	0.76	1.91	[25208 24 2]	503-6786 Hz	114 s
Cardinal	2.12	1.34	[25208 24 2]	503-6786 Hz	126 s
Carolina wren	0	0.10	[25208 16 2]	503-6786 Hz	154 s
Catbird	1.55	0.42	[25208 16 2]	503-6786 Hz	156 s
Chachalaca	0.39	0.74	[25208 16 2]	503-6786 Hz	108 s
Coue flycatcher	0	0.52	[25208 16 2]	503-6786 Hz	190 s
Cricket	0	0	[54188 16 2]	1003-14786 Hz	144 s
Crow	0	0.21	[25208 16 2]	503-6786 Hz	88 s
Dowitcher	0	1.18	[25208 16 2]	503-6786 Hz	97 s
Eastern wood pewee	0	0.52	[25208 16 2]	503-6786 Hz	102 s
Egret	0	1.37	[25208 16 2]	503-6786 Hz	89 s
Fox sparrow	0	1.38	[25208 16 2]	503-6786 Hz	88 s
Frog	1.25	3.64	[54188 24 2]	1003-14786 Hz	150 s
Gallinule	0	1.40	[25208 16 2]	503-6786 Hz	177 s
Green heron	0	0.63	[25208 16 2]	503-6786 Hz	144 s
King bird	0	1.59	[25208 16 2]	503-6786 Hz	195 s
Lark bunting	0	2.09	[25208 16 2]	503-6786 Hz	102 s
Marsh wren	0	1.56	[25208 24 2]	503-6786 Hz	138 s
Mockingbird	2.56	0	[25208 16 2]	503-6786 Hz	122 s
Nuthatch	1.16	0.10	[25208 16 2]	503-6786 Hz	107 s
Painted bunting	2.42	1.05	[25208 16 2]	503-6786 Hz	103 s
Pauraque	0	1.90	[25208 16 2]	503-6786 Hz	125 s
Purple finch	6.02	0.6	[25208 16 2]	503-6786 Hz	133 s
Red-tailed hawk	1.45	0	[25208 24 2]	503-6786 Hz	174 s
Red woodpecker	0	0.94	[25208 16 2]	503-6786 Hz	101 s
Robin	0	1.05	[25208 16 2]	503-6786 Hz	126 s
Rose grosbeak	0	2.19	[25208 16 2]	503-6786 Hz	191 s
Rufous towhee	0	0	[57868 24 2]	50-14786 Hz	173 s
Sanderling	2.35	2.85	[25208 16 2]	503-6786 Hz	151 s
Scott oriole	0	0.83	[25208 16 2]	503-6786 Hz	94 s
Scrub jay	3.14	0.95	[25208 16 2]	503-6786 Hz	126 s
Stellar jay	1.12	1.60	[25208 16 2]	503-6786 Hz	142 s
Summer tanager	0	0.62	[25208 16 2]	503-6786 Hz	103 s
Tailed towhee	0	1.28	[25208 16 2]	503-6786 Hz	109 s
Warbler	5.85	0.10	[25208 16 2]	503-6786 Hz	143 s
Western tanager	3.07	1.80	[25208 24 2]	503-6786 Hz	156 s
White sparrow	0	0	[25208 16 2]	503-6786 Hz	66 s
White vireo	0	0.10	[25208 16 2]	503-6786 Hz	83 s
Yellowlegs	0	0.84	[25208 16 2]	503-6786 Hz	95 s

Table 9.1 – TidZam classifier details on wildlife recognition application.

Glossaries

Acronyms

6LoWPAN

IPv6 LoW Power Wireless Area Networks. 9, 14–16, 18, 20, 38, 45–48, 50, 53–56, 61, 75, 192

ACL

Agent Communications Language. 27

ACL

Access Control List.
— Glossary: ACL

AES

Advanced Encryption Standard. 16,
— Glossary: AES

AI

Artificial Intelligence. 3, 7, 37, 147, 148, 150, 192

AmI

Ambient Intelligence. 7–10, 22, 23, 25–27, 33, 37, 38, 126, 148, 150, 192

ANC

Artificial Neural Controller. 8, 25, 38, 39, 94, 100, 103–110, 125, 126, 143, 147, 149, 192

ANN

Artificial Neural Network. 7, 8, 10, 12, 37, 39, 94–101, 105–108, 147,
— Glossary: ANN

AO

Artificial Organism. 7, 37

API

Application Programming Interface. 20

ARM

Active Resource Middleware. 58, 65, 70, 72, 75, 76, 83, 89, 90, 104, 107–110, 150, 151

BER

Border Edge Router. 45–47, 49

CMOS

Complementary Metal Oxide Semiconductor. 31,
— Glossary: CMOS

COAP

Constrained Application Protocol. 16, 18, 55, 58–63, 68, 70, 89, 109, 151,
— Glossary: COAP

CPU

Central Processing Unit. 13

CS

Consensus Seeking. 112

CS

Complex System. 147,
— Glossary: CS

CT

Control Theory. 112

DAG

Directed Acyclic Graph. 52

DNA

Dynamic Network Agent. 8, 76–78, 89, 150

DNS

Dynamic Name Server.
— Glossary: DNS

DoS

Denial of Service. 16, 49,
— Glossary: DoS

DPWS

Devices Profile for Web Service. 18,
— Glossary: DPWS

DS

Dynamic System. 112

ECA

Event-Condition-Action. 24

EMMA

Environment Monitoring and Management Agent. 7–9, 37–39, 54, 55, 59, 61–63, 66, 68–70, 72, 77, 83, 86, 90, 94, 95, 97, 107, 110, 112, 127, 147–151, 153, 192

EMS

Energy Management System. 126

ES

Embedded System. 7,
— Glossary: ES

FIPA

Foundation for Intelligent Physical Agents. 27,
— Glossary: FIPA

FPGA

Field-Programmable Gate Array. 13, 31,
— Glossary: FPGA

FS

File System. 59

HCI

Human Computer Interface. 126, 135, 141, 143, 192

HIS

Home Information System. 9, 47–50, 56

HTTP

Hyper Text Transfer Protocol. 18, 51, 55

ICMP

Internet Control Message Protocol. 52, 53,
— Glossary: ICMP

IEEE

Institute of Electrical and Electronics Engineers. 7

IETF

Internet Engineering Task Force. 7, 16, 18, 45

IoT

Internet of Things. 3–11, 14, 20, 33, 44, 45, 49–51, 56, 68, 69, 95, 147, 148, 150, 151

IP

Internet Protocol. 4, 14, 44–47, 49, 50, 148

IPv4

Internet Protocol version 4. 9, 49, 50

IPv6

Internet Protocol version 6. 14, 39, 44, 46–50, 53, 56

ISP

Internet Service Provider. 48

IT

Information Technology. 49

JSON

JavaScript Object Notation. 18, 63, 64, 150

MAC

Media Access Control. 14, 61,
— Glossary: MAC

MAS

Multi Agent System. 7, 27–30, 112, 127, 147,
— Glossary: MAS

MCKP

Multiple Choice Knapsack Problem. 80

MIT

Massachusetts Institute of Technology. 151

MKP

Multiple Knapsack Problem. 80

MLP

Multi-Layer Perceptron. 95, 96, 139,
— Glossary: MLP

MNAC

Multi Network Average Consensus. 120, 121

MOSFET

Metal Oxide Semiconductor Field Effect Transistor. 31,
— Glossary: MOSFET

NAC

Network Average Consensus. 112–114, 116, 120, 121, 123, 124, 127

NAS

Network Attached Storage. 47–49

NPN

Numerical Petri Network. 39

NS

Name Space. 69

NWSN

Neural Wireless Sensor Network. 94

oBIX

Open Building Information Xchange. 150,
— Glossary: oBIX

OC

Organic Computing. 8, 10, 28–30, 32, 33, 37, 147, 148, 150

OPC

Organic Processing Cells. 31

OS

Operating System. 7, 48

OSGi

Open Services Gateway initiative. 27,
— Glossary: OSGi

OWA

Ordered Weighted Average. 117, 122

PAN

Personal Area Network. 148

PBO

pseudo-Boolean optimization. 39, 72, 75, 81–83, 90

PCA

Principal Component Analysis. 136

QoS

Quality of Service. 15, 20,
— Glossary: QoS

RBM

Restricted Boltzman Machine. 137, 138

RDF

Resource Description Framework. 18,
— Glossary: RDF

RE

Responsive Environments. 3, 6, 56, 67, 192

REG

Responsive Environment Group. 132, 133, 143

REST

REpresentational State Transfer. 51, 61

RESTFUL

REpresentational State Transfer. 18,
— Glossary: RESTFUL

RFID

Radio Frequency IDentification. 12, 69,
— Glossary: RFID

RNA

Residual Network Agent. 8, 76, 78, 89, 150

RNDIS

Remote Network Driver Interface Specification. 48, 52,
— Glossary: RNDIS

ROA

Resource Oriented Architecture. 8, 38, 39, 51, 54, 58, 192,
— Glossary: ROA

RPL

Routing Protocol for Low power and Lossy Networks. 16, 46, 47, 52

SAE

Stacked Auto-Encoder. 138–140

SC

Service Choreography. 8, 38, 39, 62, 65, 66, 72–76, 78–83, 86, 89, 90, 109, 125–127, 148, 150, 151

SC

Service Choreography. 9, 12, 51–56, 58, 62, 65–70, 75, 77, 109

SG

Smart Grid.
— Glossary: SG

sMAP

Simple Measurement and Actuation Profile. 18

SNMP

Simple Network Management Protocol. 55, 90,
— Glossary: SNMP

SO

Service Orchestration. 9, 51, 52, 56

SOA

Service Oriented Architecture. 44, 51, 52, 56,
— Glossary: SOA

SOAP

Simple Object Access Protocol. 18,
— Glossary: SOAP

SoC

System on Chip. 31, 38,
— Glossary: SoC

SSL

Secure Socket Layer.
— Glossary: SSL

SuOC

System under Observation and Control. 29, 30, 38

TDMA

Time Division Multiple Access. 61,
— Glossary: TDMA

TunSLIP

Tunnel Serial Line Internet Protocol. 48

UC

Ubiquitous Computing. 6,
— Glossary: UC

UML

Unified Modelling Language. 29

UPnP

Universal Plug and Play. 47,
— Glossary: UPnP

URI

Unified Resource Identifier. 54, 60, 61, 63, 68, 69

VP

Voting Procedures. 8, 10, 112, 113, 118, 119, 123–127, 147, 149, 192

VPN

Virtual Personal Network. 50,
—
Glossary: VPN

W3C

World Wide Web Consortium.
—
Glossary: W3C

WiFi

Wireless Fidelity. 49

WOA

Web Oriented Architecture.
—
Glossary: WOA

WSAN

Wireless Sensor and Actor Network. 8–21, 37–39, 45–56, 58, 62, 65, 70, 72, 74–79, 81–83, 86, 89, 90, 94, 112, 113, 127, 143, 147, 149, 192

XML

Extensible Markup Language. 18, 68

Glossary

ACL

Wikipedia. “An access control list (ACL), with respect to a computer file system, is a list of permissions attached to an object. An ACL specifies which users or system processes are granted access to objects, as well as what operations are allowed on given objects [...]”. 157

AES

Wikipedia. “The Advanced Encryption Standard (AES), also known as Rijndael (its original name), is a specification for the encryption of electronic data established by the U.S. National Institute of Standards and Technology (NIST) in 2001 [...]”. 157

ANN

Wikipedia. “In machine learning and cognitive science, artificial neural networks (ANNs) are a family of statistical learning models inspired by biological neural networks (the central nervous systems of animals, in particular the brain) and are used to estimate or approximate functions that can depend on a large number of inputs and are generally unknown. Artificial neural networks are generally presented as systems of interconnected "neurons" which send messages to each other. The connections have numeric weights that can be tuned based on experience, making neural nets adaptive to inputs and capable of learning [...]”. 157

CMOS

Wikipedia. “Complementary metal–oxide–semiconductor (CMOS) is a technology for constructing integrated circuits. CMOS technology is used in microprocessors, microcontrollers, static RAM, and other digital logic circuits [...]”. 158

COAP

Wikipedia. “Constrained Application Protocol (CoAP) is a software protocol intended to be used in very simple electronics devices that allows them to communicate interactively over the Internet. It is particularly targeted for small low power sensors, switches, valves and similar components that need to be controlled or supervised remotely, through standard Internet networks. CoAP is an application layer protocol that is intended for use in resource-constrained internet devices, such as WSN nodes. CoAP is designed to easily translate to HTTP for simplified integration with the web, while also meeting specialized requirements such as multicast support, very low overhead, and simplicity [...]”. 158

CS

Wikipedia. “Complex systems present problems both in mathematical modelling and philosophical foundations. The study of complex systems represents a new approach to science that investigates how relationships between parts give rise to the collective behaviors of a system and how the system interacts and forms relationships with its environment [...]”. 158

DNS

Wikipedia. “The Domain Name System (DNS) is a hierarchical distributed naming system for computers, services, or any resource connected to the Internet or a private network. It associates various information with domain names assigned to each of the participating entities. Most prominently, it translates domain names, which can be easily memorized by humans, to the numerical IP addresses needed for the purpose of computer services and devices worldwide [...]”. 158

DoS

Wikipedia. “In computing, a denial-of-service (DoS) attack is an attempt to make a machine or network resource unavailable to its intended users [...]”. 158

DPWS

Wikipedia. “The Devices Profile for Web Services (DPWS) defines a minimal set of implementation constraints to enable secure Web Service messaging, discovery, description, and eventing on resource-constrained devices. Its objectives are similar to those of Universal Plug and Play (UPnP) but, in addition, DPWS is fully aligned with Web Services technology and includes numerous extension points allowing for seamless integration of device-provided services in enterprise-wide application scenarios [...]”. 159

ES

Wikipedia. “An embedded system is a computer system with a dedicated function within a larger mechanical or electrical system, often with real-time computing constraints. It is embedded as part of a complete device often including hardware and mechanical parts. Embedded systems control many devices in common use today [...]”. 159

FIPA

Wikipedia. “The Foundation for Intelligent Physical Agents (FIPA) is a body for developing and setting computer software standards for heterogeneous and interacting agents and agent-based systems [...]”. 159

FPGA

Wikipedia. “A field-programmable gate array (FPGA) is an integrated circuit designed to be configured by a customer or a designer after manufacturing – hence "field-programmable". The FPGA configuration is generally specified using a hardware description language (HDL), similar to that used for an application-specific integrated circuit (ASIC) [...]”. 159

ICMP

Wikipedia. “The Internet Control Message Protocol (ICMP) is one of the main protocols of the Internet Protocol Suite. It is used by network devices, like routers, to send error messages indicating, for example, that a requested service is not available or that a host or router could not be reached. ICMP can also be used to relay query messages [...]”. 160

MAC

Wikipedia. “In the seven-layer OSI model of computer networking, media access control (MAC) data communication protocol is a sublayer of the data link layer (layer 2). The MAC sublayer provides addressing and channel access control mechanisms that make it possible for several terminals or network nodes to communicate within a multiple access network that incorporates a shared medium, e.g. an Ethernet network. The hardware that implements the MAC is referred to as a media access controller [...]”. 160

MAS

Wikipedia. “A multi-agent system (M.A.S.) is a computerized system composed of multiple interacting intelligent agents within an environment. Multi-agent systems can be used to solve problems that are difficult or impossible for an individual agent or a monolithic system to solve [...]”. 160

MLP

Wikipedia. “A multilayer perceptron (MLP) is a feedforward artificial neural network model that maps sets of input data onto a set of appropriate outputs. A MLP consists of multiple layers of nodes in a directed graph, with each layer fully connected to the next one. Except for the input nodes, each node is a neuron (or processing element) with a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training the network [...]”. 161

MOSFET

Wikipedia. “The metal–oxide–semiconductor field-effect transistor (MOSFET, MOS-FET, or MOS FET) is a type of transistor used for amplifying

or switching electronic signals [...]". 161

oBIX

Wikipedia. "oBIX (for Open Building Information Exchange) is a standard for RESTful Web Services-based interfaces to building control systems. oBIX is about reading and writing data over a network of devices using XML and URIs, within a framework specifically designed for building automation [...]". 161

OSGi

Wikipedia. "The OSGi specification describes a modular system and a service platform for the Java programming language that implements a complete and dynamic component model, something that does not exist in standalone Java/VM environments. Applications or components, coming in the form of bundles for deployment, can be remotely installed, started, stopped, updated, and uninstalled without requiring a reboot; management of Java packages/classes is specified in great detail. Application life cycle management is implemented via APIs that allow for remote downloading of management policies. The service registry allows bundles to detect the addition of new services, or the removal of services, and adapt accordingly [...]". 162

QoS

Wikipedia. "Quality of service (QoS) is the overall performance of a telephony or computer network, particularly the performance seen by the users of the network [...]". 162

RDF

Wikipedia. "The Resource Description Framework is a family of World Wide Web Consortium (W3C) specifications[1] originally designed as a metadata data model. It has come to be used as a general method for conceptual description or modeling of information that is implemented in web resources, using a variety of syntax notations and data serialization formats. It is also used in knowledge management applications [...]". 162

RESTFUL

Wikipedia. "Representational State Transfer is a software architecture style consisting of guidelines and best practices for creating scalable web services. REST is a coordinated set of constraints applied to the design of components in a distributed hypermedia system that can lead to a more performance and maintainable architecture [...]". 162

RFID

Wikipedia. “Radio-frequency identification (RFID) is the wireless use of electromagnetic fields to transfer data, for the purposes of automatically identifying and tracking tags attached to objects. The tags contain electronically stored information [...]”. 163

RNDIS

Wikipedia. “The Remote Network Driver Interface Specification (RNDIS) is a Microsoft proprietary protocol used mostly on top of USB. It provides a virtual Ethernet link to most versions of the Windows and Linux operating systems. A partial RNDIS specification is available from Microsoft, but Windows implementations have been observed to issue requests not included in that specification, and to have undocumented constraints [...]”. 163

ROA

Wikipedia. “In software engineering, a resource-oriented architecture (ROA) is a style of software architecture and programming paradigm for designing and developing software in the form of resources with REpresentational State Transfer (RESTFUL) interfaces. These resources are software components (discrete pieces of code and/or data structures) which can be reused for different purposes. ROA design principles and guidelines are used during the phases of software development and system integration [...]”. 163

SG

Wikipedia. “Complex systems present problems both in mathematical modelling and philosophical foundations. The study of complex systems represents a new approach to science that investigates how relationships between parts give rise to the collective behaviors of a system and how the system interacts and forms relationships with its environment [...]”. 163

SNMP

Wikipedia. “Simple Network Management Protocol (SNMP) is an "Internet-standard protocol for managing devices on IP networks". Devices that typically support SNMP include routers, switches, servers, workstations, printers, modem racks and more.[1] SNMP is widely used in network management systems to monitor network-attached devices for conditions that warrant administrative attention [...]”. 163

SOA

Wikipedia. “A service-oriented architecture (SOA) is an architectural pattern in computer software design in which application components provide services to other components via a communications protocol, typically over

a network. The principles of service-orientation are independent of any vendor, product or technology [...]". 164

SOAP

Wikipedia. "SOAP, originally an acronym for Simple Object Access protocol, is a protocol specification for exchanging structured information in the implementation of web services in computer networks. It uses XML Information Set for its message format, and relies on other application layer protocols, most notably Hypertext Transfer Protocol (HTTP) or Simple Mail Transfer Protocol (SMTP), for message negotiation and transmission [...]". 164

SoC

Wikipedia. "A system on a chip or system on chip (SoC or SOC) is an integrated circuit (IC) that integrates all components of a computer or other electronic system into a single chip. It may contain digital, analog, mixed-signal, and often radio-frequency functions—all on a single chip substrate. SoCs are very common in the mobile electronics market because of their low power consumption. A typical application is in the area of embedded systems [...]". 164

SSL

Wikipedia. "Transport Layer Security (TLS) and its predecessor, Secure Sockets Layer (SSL), are cryptographic protocols designed to provide communications security over a computer network.[1] They use X.509 certificates and hence asymmetric cryptography to authenticate the counterparty with whom they are communicating,[2] and to negotiate a symmetric session key. This session key is then used to encrypt data flowing between the parties. This allows for data/message confidentiality, and message authentication codes for message integrity and as a by-product, message authentication [...]". 164

TDMA

Wikipedia. "Time division multiple access (TDMA) is a channel access method for shared medium networks. It allows several users to share the same frequency channel by dividing the signal into different time slots. The users transmit in rapid succession, one after the other, each using its own time slot. This allows multiple stations to share the same transmission medium (e.g. radio frequency channel) while using only a part of its channel capacity [...]". 164

UC

Wikipedia. "Ubiquitous computing (ubicomp) is a concept in software engineering and computer science where computing is made to appear everywhere and anywhere. In contrast to desktop computing, ubiquitous computing can occur using any device, in any location, and in any format. A user interacts with the computer, which can exist in many different forms, including laptop computers, tablets and terminals in everyday objects such as a fridge or a pair of glasses. The underlying technologies to support ubiquitous computing include Internet, advanced middleware, operating system, mobile code, sensors, microprocessors, new I/O and user interfaces, networks, mobile protocols, location and positioning and new materials [...]". 164

UPnP

Wikipedia. "Universal Plug and Play (UPnP) is a set of networking protocols that permits networked devices, such as personal computers, printers, Internet gateways, Wi-Fi access points and mobile devices to seamlessly discover each other's presence on the network and establish functional network services for data sharing, communications, and entertainment. UPnP is intended primarily for residential networks without enterprise-class devices [...]". 164

VPN

Wikipedia. "A virtual private network (VPN) extends a private network across a public network, such as the Internet. It enables a computer or network-enabled device to send and receive data across shared or public networks as if it were directly connected to the private network, while benefiting from the functionality, security and management policies of the private network [...]". 165

W3C

Wikipedia. "The World Wide Web Consortium (W3C) is the main international standards organization for the World Wide Web (abbreviated WWW or W3) [...]". 165

WOA

Wikipedia. "Web-oriented architecture (WOA) was coined in 2006 by Nick Gall of the Gartner's group. It is a software architecture style that extends service-oriented architecture (SOA) to web-based applications. WOA was originally created by many web applications and sites, such as social websites and personal websites [...]". 165

Bibliography

[AAS13] Charu C Aggarwal, Naveen Ashish, and Amit Sheth. "The internet of things: A survey from the data-centric perspective". In: *Managing and mining sensor data*. Springer, 2013, pages 383–428. (Cited on page 4).

[AS+94] Rakesh Agrawal, Ramakrishnan Srikant, et al. "Fast algorithms for mining association rules". In: *Proc. 20th int. conf. very large data bases, VLDB*. Volume 1215. 1994, pages 487–499. (Cited on page 106).

[AK04] Ian F Akyildiz and Ismail H Kasimoglu. "Wireless sensor and actor networks: research challenges". In: *Ad hoc networks* 2.4 (2004), pages 351–367. (Cited on page 11).

[AE10] Hande Alemdar and Cem Ersoy. "Wireless sensor networks for healthcare: A survey". In: *Computer Networks* 54.15 (2010), pages 2688–2710. (Cited on page 23).

[ATK11] Aliaksei Andrushevich, Stephan Tomek, and Alexander Klapproth. "The autonomic computing paradigm in adaptive building/ambient intelligence systems". In: *Ambient Intelligence*. Springer, 2011, pages 98–104. (Cited on page 26).

[AIM10] Luigi Atzori, Antonio Iera, and Giacomo Morabito. "The internet of things: A survey". In: *Computer networks* 54.15 (2010), pages 2787–2805. (Cited on page 11).

[Aug09] Juan Carlos Augusto. "Ambient intelligence: Opportunities and consequences of its use in smart classrooms". In: *Innovation in Teaching and Learning in Information and Computer Sciences* 8.2 (2009), pages 53–63. (Cited on page 22).

[ANA10] Juan Carlos Augusto, Hideyuki Nakashima, and Hamid Aghajan. "Ambient intelligence and smart environments: A state of the art". In: *Handbook of ambient intelligence and smart environments*. Springer, 2010, pages 3–31.
(Cited on page 22).

[AN04] Juan Carlos Augusto and Chris D Nugent. "The use of temporal reasoning and management of complex events in smart homes". In: *European Conference on Artificial Intelligence (ECAI)*. Volume 16. 2004, page 778.
(Cited on page 24).

[Aug+08] Juan Carlos Augusto, Jun Liu, Paul McCullagh, Hui Wang, and Jian-Bo Yang. "Management of uncertainty and spatio-temporal aspects for monitoring and diagnosis in a smart home". In: *International Journal of Computational Intelligence Systems* 1.4 (2008), pages 361–378.
(Cited on page 24).

[AIA10] Asier Aztiria, Alberto Izaguirre, and Juan Carlos Augusto. "Learning patterns in ambient intelligence environments: a survey". In: *Artificial Intelligence Review* 34.1 (2010), pages 35–51.
(Cited on page 25).

[BB11] Amit Badlani and Surekha Bhanot. "Smart Home System Design based on Artificial Neural Networks". In: *Proceedings of the World Congress on Engineering and Computer Science*. Volume 1. 2011, pages 146–164.
(Cited on page 94).

[Ban+11] Soma Bandyopadhyay, Munmun Sengupta, Souvik Maiti, and Subhajit Dutta. "Role of middleware for internet of things: A study". In: *International Journal of Computer Science & Engineering Survey (IJCSES)* 2.3 (2011), pages 94–105.
(Cited on page 20).

[Bec+10] Birger Becker et al. "Decentralized energy-management to control smart-home architectures". In: *Architecture of Computing Systems-ARCS 2010*. Springer, 2010, pages 150–161.
(Cited on page 32).

[BH06] Rezaul Begg and Rafiul Hassan. "Artificial Neural Networks in Smart Homes". In: *Designing Smart Homes*. Edited by JuanCarlos Augusto and ChrisD. Nugent. Volume 4008. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006, pages 146–164. isbn: 978-3-540-35994-4. doi: 10.1007/11788485_9. url: http://dx.doi.org/10.1007/11788485_9.
(Cited on page 94).

[Bel+05] Stephen J Bellis et al. "Development of field programmable modular wireless sensor network nodes for ambient systems". In: *Computer Communications* 28.13 (2005), pages 1531–1544.
(Cited on page 13).

[Bev10] John Bevis. *aaaaw to zzzzzd The Words of Birds*. 2010.
(Cited on page 134).

[BWWH91] Jonathan Billington, Geoffrey R. Wheeler, and Michael C. Wilbur-Ham. "PROTEAN: a high-level Petri net tool for the specification and verification of communication protocols". In: *High-level Petri Nets*. Springer, 1991, pages 560–575.
(Cited on page 66).

[BBB08] Zoran S Bojkovic, Bojan M Bakmaz, and Miodrag R Bakmaz. "Security issues in wireless sensor networks". In: *International Journal of Communications* 2.1 (2008), pages 106–115.
(Cited on page 16).

[Bou+06] Abdelmajid Bouajila et al. "Organic computing at the system on chip level". In: *Very Large Scale Integration, 2006 IFIP International Conference on*. IEEE. 2006, pages 338–341.
(Cited on page 31).

[BN08] David Boyle and Thomas Newe. "Securing wireless sensor networks: security architectures". In: *Journal of Networks* 3.1 (2008), pages 65–77.
(Cited on page 16).

[Bra08] T Scott Brandes. "Automated sound recording and analysis techniques for bird surveys and conservation". In: *Bird Conservation International* 18.S1 (2008), S163–S173.
(Cited on page 134).

[Bra+06] Jürgen Branke, Moez Mnif, Christian Muller-Schloer, and Holger Prothmann. "Organic Computing–Addressing complexity by controlled self-organization". In: *Leveraging Applications of Formal Methods, Verification and Validation, 2006. ISoLA 2006. Second International Symposium on*. IEEE. 2006, pages 185–191. (Cited on page 30).

[BKK08] Rainer Buchty, David Kramer, and Wolfgang Karl. "An Organic Computing Approach to Sustained Real-time Monitoring". In: *Biologically-Inspired Collaborative Computing*. Springer, 2008, pages 151–162. (Cited on page 31).

[CY05] Seyit A Camtepe and Bülent Yener. "Key distribution mechanisms for wireless sensor networks: a survey". In: *Rensselaer Polytechnic Institute, Troy, New York, Technical Report* (2005), pages 05–07. (Cited on page 16).

[Cas96] Mike Casey. "The dynamics of discrete-time computation, with application to recurrent neural networks and finite state machine extraction". In: *Neural computation* 8.6 (1996), pages 1135–1178. (Cited on page 106).

[Cer+09] Matteo Ceriotti et al. "Monitoring heritage buildings with wireless sensor networks: The Torre Aquila deployment". In: *Proceedings of the 2009 International Conference on Information Processing in Sensor Networks*. IEEE Computer Society. 2009, pages 277–288. (Cited on page 5).

[Che+11] Sylvain Cherrier, Yacine M Ghamri-Doudane, Stéphane Lohier, and Gilles Roussel. "D-lite: Distributed logic for internet of things services". In: *Internet of Things (iThings/CPSCom), 2011 International Conference on and 4th International Conference on Cyber, Physical and Social Computing*. IEEE. 2011, pages 16–24. (Cited on page 19).

[Che+13] Sylvain Cherrier, Yacine M Ghamri-Doudane, Stephane Lohier, and Gilles Roussel. "SALT: a simple application logic description using transducers for internet of things". In: *Communications (ICC), 2013 IEEE International Conference on*. IEEE. 2013, pages 3006–3011.
(Cited on page 19).

[Cio95] Emil Michel Cioran. "Aveux et anathèmes in Œuvres". In: *Paris: Gallimard* (1995).
(Cited on page 9).

[CB15] Daniel Cohn Bendit. "L'utopie est le rêve nécessaire et la réalité le défi permanent." In: *Le Monde* (2015).
(Cited on page 57).

[Com+06] Autonomic Computing et al. "An architectural blueprint for autonomic computing". In: (2006).
(Cited on page 26).

[Cor08] Jorge Cortés. "Distributed algorithms for reaching consensus on general functions". In: *Automatica* 44.3 (2008), pages 726–737.
(Cited on pages 112, 127).

[Cor06] Jorge Cortés. "Finite-time convergent gradient flows with applications to network consensus". In: *Automatica* 42.11 (2006), 1993–2000. URL: http://www.sciencedirect.com/science/article/pii/S000510980600269X (visited on 06/27/2013).
(Cited on pages 112, 116, 122).

[Cos+07] Paolo Costa, Luca Mottola, Amy L Murphy, and Gian Pietro Picco. "Programming wireless sensor networks with the TeenyLime middleware". In: *Middleware 2007*. Springer, 2007, pages 429–449.
(Cited on page 21).

[Csá01] Balázs Csanád Csáji. "Approximation with artificial neural networks". In: *Faculty of Sciences, Etvs Lornd University, Hungary* 24 (2001).
(Cited on page 95).

[DH+10] Stephen Dawson-Haggerty, Xiaofan Jiang, Gilman Tolle, Jorge Ortiz, and David Culler. "sMAP: a simple measurement and actuation profile for physical information". In: *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*. ACM. 2010, pages 197–210.
(Cited on page 18).

[Del+06] Flávia C Delicato, Luci Pirmez, Paulo F Pires, and José Ferreira De Rezende. "Exploiting web technologies to build autonomic wireless sensor networks". In: *Mobile and Wireless Communication Networks*. Springer, 2006, pages 99–114.
(Cited on page 17).

[DAG99] Zoltán Dienes, Gerry Altmann, and Shi-Ji Gao. "Mapping across domains without feedback: A neural network model of transfer of implicit knowledge". In: *Cognitive Science* 23.1 (1999), pages 53–82.
(Cited on page 106).

[DHC05] Faiyaz Doctor, Hani Hagras, and Victor Callaghan. "A fuzzy embedded agent-based approach for realizing ambient intelligence in intelligent inhabited environments". In: *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans* 35.1 (2005), pages 55–65.
(Cited on page 25).

[DB14] Clement Duhart and Cyrille Bertelle. "Methodology for Artificial Neural controllers on wireless sensor network". In: *IEEE Conference on Wireless Sensors (ICWiSE)*. 2014, pages 67–72. doi: 10.1109/ICWISE.2014.7042663.
(Cited on pages 1, 93).

[DB15] Clement Duhart and Cyrille Bertelle. "Toward Organic Computing Approach for Cybernetic Responsive Environment". In: *International Journal of Ambient Systems and Applications (IJASA)* 3.4 (2015). doi: DOI:10.5121/ijasa.2015.3401.
(Cited on pages 1, 35).

[DCB13] Clement Duhart, Michel Cotsaftis, and Cyrille Bertelle. "Lightweight Distributed Adaptive Algorithm for Voting Procedures by Using Network Average Consensus". English. In: *PRIMA 2013: Principles and Practice of Multi-Agent Systems*. Volume 8291. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, pages 421–428. ISBN: 978-3-642-44926-0. DOI: 10.1007/978-3-642-44927-7_30.

(Cited on pages 1, 111).

[DCB14] Clement Duhart, Michel Cotsaftis, and Cyrille Bertelle. "Wireless Sensor Network Cloud Services: Towards a Partial Delegation". In: *Proceedings of 5th International Conference on Smart Communications in Network Technologies 2014 (IEEE SaCoNeT 2014)*. Vilanova i la Geltru, Spain, June 2014.

(Cited on pages 1, 57).

[DSB] Clement Duhart, Pierre Sauvage, and Cyrille Bertelle. "A Resource Oriented Framework for Service Choreography over Wireless Sensor and Actor Networks". In: *Submission in International Journal of Wireless Information Networks (IJWI)* ().

(Cited on pages 1, 71).

[Dun03] Adam Dunkels. "uIP-dunkels-2003". In: *Full TCP/IP for 8-bit architectures*. Edited by ACM. San Francisco, CA, USA: Proceedings of the 1st international conference on Mobile systems, applications and services, 2003, pages 85–98.

(Cited on pages 15, 20, 54).

[Dun+06] Adam Dunkels, Niclas Finne, Joakim Eriksson, and Thiemo Voigt. "Run-time dynamic linking for reprogramming wireless sensor networks". In: *Proceedings of the 4th international conference on Embedded networked sensor systems*. ACM. 2006, pages 15–28.

(Cited on page 20).

[Ein71] Albert Einstein. "Our entire much-praised technological progress, and civilization generally, could be compared to an axe in the hand of a pathological criminal." In: *Letter of 6 December to Heinrich Zaggler* (1971).

(Cited on page 43).

[Far+05] S. Farshchi, P.H. Nuyujukian, A. Pesterev, I. Mody, and J.W. Judy. "A TinyOS-Based Wireless Neural Sensing, Archiving, and Hosting System". In: *Proceedings of 2nd International IEEE EMBS Conference on Neural Engineering*. 2005, pages 671–674. doi: 10.1109/CNE.2005.1419714.
(Cited on page 94).

[FS05] Dietmar Fey and Daniel Schmidt. "Marching-pixels: a new organic computing paradigm for smart sensor processor arrays". In: *Proceedings of the 2nd conference on Computing frontiers*. ACM. 2005, pages 1–9.
(Cited on page 31).

[Gan87] Jean-Gabriel Ganascia. "CHARADE: A Rule System Learning System." In: *IJCAI*. Volume 87. 1987, pages 234–239.
(Cited on page 106).

[GS07] Alessandro Giua and Carla Seatzu. "A systems theory view of Petri nets". In: *Advances in Control Theory and Applications*. Springer, 2007, pages 99–127.
(Cited on page 66).

[Gub+13] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. "Internet of Things (IoT): A vision, architectural elements, and future directions". In: *Future Generation Computer Systems* 29.7 (2013), pages 1645–1660.
(Cited on page 6).

[GTW10] Dominique Guinard, Vlad Trifa, and Erik Wilde. "A resource oriented architecture for the web of things". In: *Internet of Things (IOT), 2010*. IEEE. 2010, pages 1–8.
(Cited on page 58).

[Gun+07] Vehbi C Gungor, Chellury Sastry, Zhen Song, and Ryan Integlia. "Resource-aware and link quality based routing metric for wireless sensor and actor networks". In: *Communications, 2007. ICC'07. IEEE International Conference on*. IEEE. 2007, pages 3364–3369.
(Cited on page 15).

[HM06] Salem Hadim and Nader Mohamed. "Middleware: Middleware challenges and approaches for wireless sensor networks". In: *IEEE distributed systems online* 3 (2006), page 1.
(Cited on page 20).

[HM05] Yuko Hatano and Mehran Mesbahi. "Agreement over random networks". In: *Automatic Control, IEEE Transactions on* 50.11 (2005), pages 1867–1872.
(Cited on page 112).

[HZM09] Hongmei He, Zhenhuan Zhu, and E. Makinen. "A Neural Network Model to Minimize the Connected Dominating Set for Self-Configuration of Wireless Sensor Networks". In: *Neural Networks, IEEE Transactions on* 20.6 (2009), pages 973–982. issn: 1045-9227. doi: 10.1109/TNN.2009.2015088.
(Cited on page 94).

[HS06] Geoffrey E Hinton and Ruslan R Salakhutdinov. "Reducing the dimensionality of data with neural networks". In: *Science* 313.5786 (2006), pages 504–507.
(Cited on page 137).

[Hin+12] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. "Improving neural networks by preventing co-adaptation of feature detectors". In: *arXiv preprint arXiv:1207.0580* (2012).
(Cited on page 138).

[Hog+12] Joel Hoglund, Dejan Ilic, Stamatis Karnouskos, Robert Sauter, and P Goncalves Da Silva. "Using a 6LoWPAN smart meter mesh network for event-driven monitoring of power quality". In: *IEEE Third International Conference on Smart Grid Communications (SmartGridComm)*. IEEE. 2012, pages 448–453.
(Cited on page 14).

[HV+87] Mark Holliday, Mary K Vernon, et al. "A generalized timed Petri net model for performance analysis". In: *IEEE Transactions on Software Engineering* 12 (1987), pages 1297–1310.
(Cited on page 83).

[Hua+13] Jui-Ting Huang, Jinyu Li, Dong Yu, Li Deng, and Yifan Gong. "Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers". In: *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE. 2013, pages 7304–7308.
(Cited on page 106).

[HJ08] Vincent Huang and Muhammad Kashif Javed. "Semantic sensor information description and processing". In: *Sensor Technologies and Applications, 2008. SENSORCOMM'08. Second International Conference on*. IEEE. 2008, pages 456–461.
(Cited on page 18).

[JMS05] François Jammes, Antoine Mensch, and Harm Smit. "Service-oriented device communications using the devices profile for web services". In: *Proceedings of the 3rd international workshop on Middleware for pervasive and ad-hoc computing*. ACM. 2005, pages 1–8.
(Cited on page 18).

[JZS12] Antonio J Jara, Miguel A Zamora, and Antonio Skarmeta. "Glowbal IP: An adaptive and transparent IPv6 integration in the Internet of Things". In: *Mobile Information Systems* 8.3 (2012), pages 177–197.
(Cited on page 15).

[JL93] A Jirachiefpattana and R Lai. "Verifying Estelle specifications: numerical Petri nets approach". In: *Network Protocols, 1993. Proceedings., 1993 International Conference on*. IEEE. 1993, pages 334–341.
(Cited on page 66).

[Jou66] Joseph Joubert. "To teach is to learn twice." In: *De l'éducation, LXVIII* (1866).
(Cited on page 93).

[Kaa+12] Mohamed Kaaniche, Paolo Lollini, Andrea Bondavalli, and Karama Kanoun. "Modeling the resilience of large and evolving systems". In: *arXiv preprint arXiv:1211.5738* (2012).
(Cited on page 67).

[KM09] S. Kar and J.M.F. Moura. "Distributed Consensus Algorithms in Sensor Networks With Imperfect Communication: Link Failures and Channel Noise". In: *IEEE Transactions on Signal Processing* 57.1 (2009), pages 355–369. ISSN: 1053-587X, 1941-0476. DOI: 10.1109/TSP.2008.2007111. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4663899 (visited on 07/01/2013).
(Cited on page 112).

[KM07] Soummya Kar and José MF Moura. "Distributed average consensus in sensor networks with random link failures". In: *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*. Volume 2. Honolulu, Hawaii, USA, 2007, II–1013. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4217583 (visited on 07/01/2013).
(Cited on page 112).

[KB05] Holger Kasinger and Bernhard Bauer. "Towards a Model-Driven Software Engineering Methodology for Organic Computing Systems." In: *Computational Intelligence*. 2005, pages 141–146.
(Cited on page 29).

[KM04] A.I. Khan and P. Mihailescu. "Parallel pattern recognition computations within a wireless sensor network". In: *Proceedings of the 17th International Conference on Pattern Recognition*. Volume 1. 2004, 777–780 Vol.1. DOI: 10.1109/ICPR.2004.1334332.
(Cited on page 94).

[Kim+06] Sukun Kim et al. "Wireless sensor networks for structural health monitoring". In: *Proceedings of the 4th international conference on Embedded networked sensor systems*. ACM. 2006, pages 427–428.
(Cited on page 4).

[Ko+10] JeongGil Ko et al. "Wireless sensor networks for healthcare". In: *Proceedings of the IEEE* 98.11 (2010), pages 1947–1960.
(Cited on page 5).

[Kof77] Kurt Koffka. "The whole is different than the sum of its parts." In: *Heider* (1977), page 383.
(Cited on page 71).

[KDD11] Matthias Kovatsch, Simon Duquennoy, and Adam Dunkels. "A Low-Power CoAP for Contiki". In: *2011 IEEE Eighth International Conference on Mobile Ad-Hoc and Sensor Systems* (Oct. 2011), pages 855–860. DOI: 10.1109/MASS.2011.100.
(Cited on pages 18, 61).

[KLD12] Matthias Kovatsch, Martin Lanter, and Simon Duquennoy. "Actinium: A restful runtime container for scriptable internet of things applications". In: *Internet of Things (IOT), 2012 3rd International Conference on the*. IEEE. 2012, pages 135–142.
(Cited on page 19).

[KRM13] Konrad-Felix Krentz, Hosnieh Rafiee, and Christoph Meinel. "6LoWPAN security: adding compromise resilience to the 802.15. 4 security sublayer". In: *Proceedings of the International Workshop on Adaptive Security*. ACM. 2013, page 1.
(Cited on page 16).

[KEW02] Bhaskar Krishnamachari, Deborah Estrin, and Stephen Wicker. "The impact of data aggregation in wireless sensor networks". In: *Distributed Computing Systems Workshops, 2002. Proceedings. 22nd International Conference on*. IEEE. 2002, pages 575–578.
(Cited on page 12).

[KD05] A. Kulakov and D. Davcev. "Tracking of unusual events in wireless sensor networks based on artificial neural-networks algorithms". In: *Proceedings of International Conference on Information Technology: Coding and Computing*. Volume 2. 2005, 534–539 Vol. 2. DOI: 10.1109/ITCC.2005.281.
(Cited on page 94).

[KFV11] R.V. Kulkarni, A. Forster, and G.K. Venayagamoorthy. "Computational Intelligence in Wireless Sensor Networks: A Survey". In: *Communications Surveys Tutorials, IEEE* 13.1 (2011), pages 68–96. ISSN: 1553-877X. DOI: 10.1109/SURV.2011.040310.00002.
(Cited on page 94).

[Kus+07] Manish Kushwaha, Isaac Amundson, Xenofon Koutsoukos, Sandeep Neema, and Janos Sztipanovits. "OASiS: A Programming Framework for Service-Oriented Sensor Networks". In: *2007 2nd International Conference on Communication Systems Software and Middleware* (Jan. 2007), pages 1–8. DOI: 10.1109/COMSWA.2007.382431.

(Cited on page 21).

[Kwo+06] YoungMin Kwon, Sameer Sundresh, Kirill Mechitov, and Gul Agha. "ActorNet: An actor platform for wireless sensor networks". In: *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*. ACM. 2006, pages 1297–1300.

(Cited on page 21).

[Lay13] MHz Physical Layer. "Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)". In: (2013).

(Cited on page 14).

[LN93] A.U. Levin and K.S. Narendra. "Control of nonlinear dynamical systems using neural networks: controllability and stabilization". In: *Neural Networks, IEEE Transactions on* 4.2 (1993), pages 192–206. ISSN: 1045-9227. DOI: 10.1109/72.207608.

(Cited on page 94).

[LS07] Kuang-Hui Lin and Chih-Wen Shih. "Multiple almost periodic solutions in nonautonomous delayed neural networks". In: *Neural computation* 19.12 (2007), pages 3392–3420.

(Cited on page 95).

[Mad+05] Samuel R Madden, Michael J Franklin, Joseph M Hellerstein, and Wei Hong. "TinyDB: an acquisitional query processing system for sensor networks". In: *ACM Transactions on database systems (TODS)* 30.1 (2005), pages 122–173.

(Cited on page 21).

[Mag+06] Carsten Magerkurth et al. "An intelligent user service architecture for networked home environments". In: *2nd IET International Conference on Intelligent Environments*. Volume 1. IET. 2006, pages 361–370.

(Cited on page 25).

[MKD11] Naoki Matsumaru, Peter Kreyssig, and Peter Dittrich. "Organisation-Oriented Chemical Programming". In: *Organic Computing — A Paradigm Shift for Complex Systems*. Springer, 2011, pages 207–220.
(Cited on page 30).

[May+16] Brian Mayton et al. "Deploying the Living Observatory: From Environmental Sensor Network to Networked Sensory Landscape". In: Submission in ACM. 2016.
(Cited on pages 1, 131).

[Mio+12] Daniele Miorandi, Sabrina Sicari, Francesco De Pellegrini, and Imrich Chlamtac. "Internet of things: Vision, applications and research challenges". In: *Ad Hoc Networks* 10.7 (2012), pages 1497–1516.
(Cited on pages 11, 12).

[MA06] M.M. Molla and S.I. Ahamed. "A Survey of Middleware for Sensor Network and Challenges". In: *2006 International Conference on Parallel Processing Workshops (ICPPW'06)* (2006), pages 223–228. doi: 10.1109/ICPPW.2006.18.
(Cited on page 20).

[MS08] A.I. Moustapha and R.R. Selmic. "Wireless Sensor Network Modeling Using Modified Recurrent Neural Networks: Application to Fault Detection". In: *Instrumentation and Measurement, IEEE Transactions on* 57.5 (2008), pages 981–988. issn: 0018-9456. doi: 10.1109/TIM.2007.913803.
(Cited on page 94).

[MSSU11] Christian Müller-Schloer, Hartmut Schmeck, and Theo Ungerer. *Organic computing—A paradigm shift for complex systems*. Volume 1. Springer Science & Business Media, 2011.
(Cited on page 28).

[Naf+10] Florian Nafz, Hella Seebach, Jan-Philipp Steghöfer, Simon Bäumler, and Wolfgang Reif. "A formal framework for compositional verification of organic computing systems". In: *Autonomic and Trusted Computing*. Springer, 2010, pages 17–31.
(Cited on page 28).

[Nie17] Friedrich Nietzsche. *Ainsi parlait Zarathoustra*. Hayes Barton Press, 1917.
(Cited on page 3).

[OM06] F. Oldewurtel and P. Mahonen. "Neural Wireless Sensor Networks". In: *Proceedings of International Conference on Systems and Networks Communications*. 2006, pages 28–28. doi: 10.1109/ICSNC.2006.56.
(Cited on page 94).

[OS05] Reza Olfati-Saber. "Ultrafast consensus in small-world networks". In: *American Control Conference, 2005. Proceedings of the 2005*. IEEE. Portland Oregon, USA, 2005, pages 2371–2378.
(Cited on pages 112, 123).

[OSFM07] Reza Olfati-Saber, J. Alex Fax, and Richard M. Murray. "Consensus and Cooperation in Networked Multi-Agent Systems". In: *Proceedings of the IEEE* 95.1 (Jan. 2007), 215–233. issn: 0018-9219. doi: 10.1109/JPROC.2006.887293. url: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4118472 (visited on 05/14/2013).
(Cited on pages 112, 114, 115, 123, 124).

[OSM04] Reza Olfati-Saber and Richard M Murray. "Consensus problems in networks of agents with switching topology and time-delays". In: *Automatic Control, IEEE Transactions on* 49.9 (2004), pages 1520–1533.
(Cited on pages 112, 123).

[Ost+06] Fredrik Osterlind, Adam Dunkels, Joakim Eriksson, Niclas Finne, and Thiemo Voigt. "Cross-level sensor network simulation with cooja". In: *Local Computer Networks, Proceedings 2006 31st IEEE Conference on*. IEEE. 2006, pages 641–648.
(Cited on page 54).

[PBG07] Federica Paganelli, Gabriele Bianchi, and Dino Giuli. "A context model for context-aware system design towards the ambient intelligence vision: experiences in the eTourism domain". In: *Universal access in ambient intelligence environments*. Springer, 2007, pages 173–191.
(Cited on page 22).

[PLH06] ASK Pathan, Hyung-Woo Lee, and Choong Seon Hong. "Security in wireless sensor networks: issues and challenges". In: *Advanced Communication Technology, 2006. ICACT 2006. The 8th International Conference*. Volume 2. IEEE. 2006, 6–pp.
(Cited on page 16).

[PA08] Rajendra M. Patrikar and Sudhir G. Akojwar. "Neural Network Based Classification Techniques for Wireless Sensor Network with Cooperative Routing". In: *Proceedings of the 12th International Conference on Communications*. ICCOM'08. Heraklion, Greece: World Scientific, Engineering Academy, and Society (WSEAS), 2008, pages 433–438. ISBN: 978-960-6766-84-8. URL: http://dl.acm.org/citation.cfm?id=1580987.1581063.
(Cited on pages 94, 95).

[PBEA07] Stacy Patterson, Bassam Bamieh, and Amr El Abbadi. "Distributed average consensus with stochastic communication failures". In: *Decision and Control, 2007 46th IEEE Conference on*. New Orleans, Louisiana, USA, 2007, 4215–4220. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4434917 (visited on 07/01/2013).
(Cited on page 112).

[Pea95] Barak A Pearlmutter. "Gradient calculations for dynamic recurrent neural networks: A survey". In: *Neural Networks, IEEE Transactions on* 6.5 (1995), pages 1212–1228.
(Cited on page 96).

[Pro+08] Holger Prothmann et al. "Organic control of traffic lights". In: *Autonomic and Trusted Computing*. Springer, 2008, pages 219–233.
(Cited on page 32).

[Pro+11] Holger Prothmann et al. *Organic traffic control*. Springer, 2011.
(Cited on page 32).

[PSY88] Demetri Psaltis, Athanasios Sideris, and Alan Yamamura. "A multilayered neural network controller". In: *IEEE control systems magazine* 8.2 (1988), pages 17–21.
(Cited on page 94).

[Rah06] MA Rahman. "Middleware for wireless sensor networks: Challenges and Approaches". In: *IEEE Distributed System Online* 7.3 (2006), pages 2–6.
(Cited on page 20).

[RM13] Parisa Rashidi and Alex Mihailidis. "A survey on ambient-assisted living tools for older adults". In: *IEEE journal of biomedical and health informatics* 17.3 (2013), pages 579–590.
(Cited on page 23).

[RB05] Wei Ren and R.W. Beard. "Consensus seeking in multiagent systems under dynamically changing interaction topologies". In: *IEEE Transactions on Automatic Control* 50.5 (May 2005), pages 655–661. issn: 0018-9286. doi: 10.1109/TAC.2005.846556. url: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1431045 (visited on 06/27/2013).
(Cited on pages 112, 123).

[RB04] Wei Ren and Randal W Beard. "Consensus of information under dynamically changing interaction topologies". In: *American Control Conference, 2004. Proceedings of the 2004*. Volume 6. IEEE. Boston, USA, 2004, pages 4939–4944.
(Cited on pages 112, 123).

[RBA05] Wei Ren, Randal W. Beard, and Ella M. Atkins. "A survey of consensus problems in multi-agent coordination". In: *American Control Conference, 2005. Proceedings of the 2005*. Portland Oregon, USA, 2005, 1859–1864. url: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1470239 (visited on 04/30/2013).
(Cited on page 114).

[RBA07] Wei Ren, Randal W. Beard, and Ella M. Atkins. "Information consensus in multivehicle cooperative control". In: *Control Systems, IEEE* 27.2 (2007), 71–82. url: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4140748 (visited on 07/01/2013).
(Cited on page 112).

[RRD15] Fred Richardson, Do ʌs Reynolds, and Najim Dehak. "Deep neural network apı ıes to speaker and language recognition". In: *Signal Processin ters, IEEE* 22.10 (2015), pages 1671–1675.
(Cited on page 142).

[Ric+06] Urban Richter, Moez Mnif, Jürgen Branke, Christian Müller-Schloer, and Hartmut Schmeck. "Towards a generic observer/controller architecture for Organic Computing." In: *Gesellschaft für Informatik Jahrestagung (1)* 93 (2006), pages 112–119.
(Cited on page 29).

[RN10] Joel JPC Rodrigues and Paulo ACS Neves. "A survey on IP-Based wireless sensor network solutions". In: *International Journal of Communication Systems* 23.8 (2010), pages 963–981.
(Cited on pages 14, 15).

[RDT07] B. Rubio, M. Diaz, and J.M. Troya. "Programming Approaches and Challenges for Wireless Sensor Networks". In: *Second International Conference on Systems and Networks Communications, ICSNC 2007*. 2007, pages 36–36. doi: 10.1109/ICSNC.2007.63.
(Cited on pages 13, 17, 20).

[Rus+06] Nick Russell, Arthur, Wil M. P. van der Aalst, and Natalya Mulyar. "Workflow Control-Flow Patterns: A Revised View". In: *BPM Center Report BPM-06-22* (2006).
(Cited on page 66).

[RP14] Spencer Russell and Joseph A Paradiso. "Hypermedia APIs for Sensor Data: A pragmatic approach to the Web of Things". In: *Proceedings of the 11th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering). 2014, pages 30–39.

[Sad11] Fariba Sadri. "Ambient intelligence: A survey". In: *ACM Computing Surveys (CSUR)* 43.4 (2011), page 36.
(Cited on page 23).

[Sai+13] Tara N Sainath, Brian Kingsbury, Abdel-rahman Mohamed, and Bhuvana Ramabhadran. "Learning filter banks within a deep neural network framework". In: *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE. 2013, pages 297–302.
(Cited on page 142).

[SSU94] P. S. Sastry, G. Santharam, and K. P. Unnikrishnan. "Memory neuron networks for identification and control of dynamical systems". In: *Neural Networks, IEEE Transactions on* 5.2 (1994), pages 306–319. ISSN: 1045-9227. DOI: 10.1109/72.279193.
(Cited on page 94).

[SHB09] Christian Schuck, Bastian Haetzer, and Jürgen Becker. "An interface for a decentralized 2d reconfiguration on xilinx virtex-fpgas for organic computing". In: *International Journal of Reconfigurable Computing* 2009 (2009), page 7.
(Cited on page 31).

[SZM07] Ali Shareef, Yifeng Zhu, and Mohamad Musavi. "Localization Using Neural Networks in Wireless Sensor Networks". In: *Proceedings of the 1st International Conference on MOBILe Wireless MiddleWARE, Operating Systems, and Applications*. MOBILWARE '08. Innsbruck, Austria: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2007, 4:1–4:7. ISBN: 978-1-59593-984-5. URL: http://dl.acm.org/citation.cfm?id=1361492.1361497.
(Cited on page 94).

[SB11] Zach Shelby and Carsten Bormann. *6LoWPAN: the wireless embedded internet*. Volume 43. John Wiley & Sons, 2011.
(Cited on page 14).

[SV05] Flávio Soares Corrêa da Silva and Wamberto W Vasconcelos. "Agent-based management of responsive environments". In: *Advances in Artificial Intelligence*. Springer, 2005, pages 224–236.
(Cited on page 27).

[SR02] Manuel Silva and Lau .ecalde. "Petri nets and integrality relaxations: A view of co .ous Petri net models". In: *Systems, Man, and Cybernetics, F :: Applications and Reviews, IEEE Transactions on* 32.4 (2002), pages 314–327.
(Cited on page 66).

[Sim+06] Richard Simpson, Debra Schreckenghost, Edmund F Lo-Presti, and Ned Kirsch. "Plans and planning in smart homes". In: *Designing Smart Homes*. Springer, 2006, pages 71–84.
(Cited on page 24).

[SSS+10] Shio Kumar Singh, MP Singh, DK Singh, et al. "Routing protocols in wireless sensor networks–A survey". In: *International Journal of Computer Science & Engineering Survey (IJCSES) Vol* 1 (2010), pages 63–83.
(Cited on page 15).

[SM11] Ridha Soua and Pascale Minet. "A survey on energy efficient techniques in wireless sensor networks". In: *Wireless and Mobile Networking Conference (WMNC), 2011 4th Joint IFIP*. IEEE. 2011, pages 1–9.
(Cited on page 12).

[Sou+04] Eduardo Souto et al. "A message-oriented middleware for sensor networks". In: *Proceedings of the 2nd workshop on Middleware for pervasive and ad-hoc computing* - (2004), pages 127–134. doi: 10.1145/1028509.1028514.
(Cited on page 20).

[SM06] Nikolaos I Spanoudakis and Pavlos Moraitis. "Agent-based Architecture in An Ambient Intelligence Context." In: *EUMAS*. 2006.
(Cited on page 27).

[ST06] Vlado Stankovski and Jernej Trnkoczy. *Application of Decision Trees to Smart Homes*. English. Edited by JuanCarlos Augusto and ChrisD. Nugent. Volume 4008. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006, pages 132–145. isbn: 978-3-540-35994-4. doi: 10.1007/11788485_8. url: http://dx.doi.org/10.1007/11788485_8.
(Cited on page 24).

[Ste+10] Jan-Philipp Steghöfer et al. "Trustworthy organic computing systems: Challenges and perspectives". In: *Autonomic and Trusted Computing*. Springer, 2010, pages 62–76.
(Cited on page 28).

[SP14] Dan Stowell and Mark D Plumbley. "Automatic large-scale classification of bird sounds is strongly improved by unsupervised feature learning". In: *PeerJ* 2 (2014), e488.
(Cited on page 134).

[Tei+14] Tobias Teich, Falko Roessler, Daniel Kretz, and Susan Franke. "Design of a Prototype Neural Network for Smart Homes and Energy Efficiency". In: *Procedia Engineering* 69.0 (2014). 24th {DAAAM} International Symposium on Intelligent Manufacturing and Automation, 2013, pages 603 –608. ISSN: 1877-7058. DOI: http://dx.doi.org/10.1016/j.proeng.2014.03.032. URL: http://www.sciencedirect.com/science/article/pii/S1877705814002781.
(Cited on page 94).

[Tie08] Tijmen Tieleman. "Training restricted Boltzmann machines using approximations to the likelihood gradient". In: *Proceedings of the 25th international conference on Machine learning*. ACM. 2008, pages 1064–1071.
(Cited on page 138).

[TS94] Geoffrey G Towell and Jude W Shavlik. "Knowledge-based artificial neural networks". In: *Artificial intelligence* 70.1 (1994), pages 119–165.
(Cited on page 104).

[TS93] Geoffrey G Towell and Jude W Shavlik. "The extraction of refined rules from knowledge-based neural networks". In: *Machine learning*. Citeseer. 1993.
(Cited on page 106).

[TED10] Nicolas Tsiftes, Joakim Eriksson, and Adam Dunkels. "Low-power wireless IPv6 routing with ContikiRPL". In: *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*. ACM. 2010, pages 406–407.
(Cited on page 16).

[Tur50] A. M. Turing. *Compu* *Machinery and Intelligence*. One of the most influential pape the history of the cognitive sciences: http://cogsci.umn.edu/m nium/final.html. 1950. URL: http://cogprints.org/499/.

(Cited on page 147).

[Uki+13] Arijit Ukil, Soma Bandyopadhyay, Abhijan Bhattacharyya, and Arpan Pal. "Lightweight security scheme for vehicle tracking system using CoAP". In: *Proceedings of the International Workshop on Adaptive Security*. ACM. 2013, page 3.

(Cited on page 16).

[VD10] Jean-Philippe Vasseur and Adam Dunkels. *Interconnecting smart objects with ip: The next internet*. Morgan Kaufmann, 2010.

(Cited on page 14).

[WG08] Hua Wang and Yi Guo. "Consensus on scale-free network". In: *American Control Conference, 2008*. IEEE. Seattle, Washington, USA, 2008, pages 748–752.

(Cited on pages 112, 123).

[WX10] Long Wang and Feng Xiao. "Finite-time consensus problems for networks of dynamic agents". In: *Automatic Control, IEEE Transactions on* 55.4 (2010), pages 950–955.

(Cited on page 112).

[Wan+08] MM Wang, JN Cao, J Li, and SK Dasi. "Middleware for wireless sensor networks: A survey". In: *Journal of computer science and ...* January (2008).

(Cited on page 20).

[YLD07] Yi Yang, Frank Lambert, and Deepak Divan. "A survey on technologies for implementing sensor networks for power delivery systems". In: *Power Engineering Society General Meeting, 2007. IEEE*. IEEE. 2007, pages 1–8.

(Cited on page 5).

[YF04] Ossama Younis and Sonia Fahmy. "HEED: a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks". In: *Mobile Computing, IEEE Transactions on* 3.4 (2004), pages 366–379.

(Cited on page 15).

[YWM05] Liyang Yu, Neng Wang, and Xiaoqiao Meng. "Real-time forest fire detection with wireless sensor networks". In: *Proceedings of International Conference on Wireless Communications, Networking and Mobile Computing*. Volume 2. 2005, pages 1214–1217. doi: 10.1109/WCNM.2005.1544272.

(Cited on page 94).

Toward Organic Ambient Intelligences ? EMMA

Abstract

The Ambient Intelligence (AmI) is a research area investigating AI techniques to create Responsive Environments (RE). Wireless Sensor and Actor Network (WSAN) are the supports for communications between the appliances, the deployed services and Human Computer Interface (HCI).

This thesis focuses on the design of RE with autonomic properties i.e. system that have the ability to manage themselves. Such environments are open, large scale, dynamic and heterogeneous which induce some difficulties in their management by monolithic system. The bio-inspired proposal considers all devices like independent cells forming an intelligent distributed organism. Each cell is programmed by a DNA-RNA process composed of reactive rules describing its internal and external behaviour. These rules are modelled by reactive agents with self-rewriting features offering dynamic reprogramming abilities.

The EMMA framework is composed of a modular *Resource Oriented Architecture (ROA) Middleware* based on IPv6 LoW Power Wireless Area Networks (6LoWPAN) technology and a MAPE-K architecture to design multi-scale AmI. The different relations between technical issues and theoretical requirements are discussed through the platforms, the network, the middleware, the mobile agents, the application deployment to the intelligent system. Two algorithms for AmI are proposed: an Artificial Neural Controller (ANC) model for automatic control of appliances with learning processes and a distributed Voting Procedures (VP) to synchronize the decisions of several system components over the WSAN.

Keywords: Ambient Intelligence, Organic Computing, Autonomic System, Multi-Agent System, Wireless Sensor and Actor Networks, Resource Oriented Architecture, Artificial Neural Network and Voting Procedure

---

Vers des Intelligences Ambiantes Organiques ? EMMA

Résumé

L'Intelligence Ambiamte (AmI) est un domaine de recherche investigant les techniques d'intelligence artificielle pour créer des environnements réactifs. Les réseaux de capteurs et effecteurs sans-fils sont les supports de communication entre les appareils ménagers, les services installés et les interfaces homme-machine.

Cette thèse s'intéresse à la conception d'Environements Réactifs avec des propriétés autonomiques i.e. des systèmes qui ont la capacité de se gérer eux-même. De tels environements sont ouverts, à grande échelle, dynamique et hétérogène, ce qui induit certains problèmes pour leur gestion par des systèmes monolithiques. L'approche proposée est bio-inspirée en considérant chacune des plate-formes comme une cellule indépendente formant un organisme intelligent distribué. Chaque cellule est programmée par un processus ADN-RNA décrit par des règles réactives décrivant leur comportement interne et externe. Ces règles sont modelées par des agents mobiles ayant des capacités d'auto-réécriture et offrant ainsi des possibilités de reprogrammation dynamique.

Le *framework* EMMA est composé d'un *middleware* modulaire avec une architecture orientée ressource basée sur la technologie 6LoWPAN et d'une architecture MAPE-K pour concevoir des AmI à plusieurs échelles. Les différentes relations entre les problèmes techniques et les besoins théoriques sont discutées dans cette thèse depuis les plate-formes, le réseau, le middleware, les agents mobiles, le déploiement des applications jusqu'au système intelligent. Deux algorithmes pour AmI sont proposés : un modèle de contrôleur neuronal artificiel pour le contrôle automatique des appareils ménagers avec des processus d'apprentissage ainsi qu'une procédure de vote distribuée pour synchroniser les décisions de plusieurs composants systèmes.

Mots-clés : Intelligence Ambiante, Informatique Organique, Système Autonomique, Système Multi-Agent, Réseau de Capteurs et d'Actuateurs sans-Fils, Architecture Orientée Ressource, Réseau de Neurones Artificiels et Procédure de Vote