UNIVERSITÀ DELLA CALABRIA

UNIVERSITA' DELLA CALABRIA

Dipartimento di Ingegneria Informatica, Modellistica, Elettronica e Sistemistica

Scuola di Dottorato

In Ingegneria dei Sistemi ed Informatica

CICLO

XXVIII

A CLOUD-ASSISTED, AGENT-BASED FRAMEWORK FOR CYBER-PHYSICAL SYSTEMS

Settore Scientifico Disciplinare ING-INF/05

Direttore:	Ch.mo Prof. Felice Crupi
	Firma [illegible]
Supervisore:	Ch.mo Prof. Giandomenico Spezzano
	Firma [illegible]

Dottorando: Dott. Andrea Vinci

Firma Andrea Vinci

Andrea Vinci

A Cloud-Assisted Agent-Based Framework for Cyber-Physical Systems

and its applications

Ph.D. Thesis

November 28, 2015

to my beloved girls

Aknowledgements

Firstly, I would like to express my sincere gratitude to my advisor, professor Giandomenico Spezzano, for his expert guidance and for allowing me to grow as a researcher.

I would like to thank my colleagues, Andrea Giordano, for all the work done together, for his advices and for the exhausting but stimulating discussions, and Alessia Amelio, for her support during my PhD studies and for her advices in both research and life.

I thank all my colleagues and friends at ICAR and Unical, for the stimulating discussions and for all the fun we have had in the last three years.

I would like to thank my parents and my brother for supporting me spiritually throughout writing this thesis and my life in general.

Finally, I would like to thank Anna, for her support, presence, patience and love, and Lara, whose smiles light my way even during the hardest times.

Preface

Since the early 2000s, we have experienced a disruptive trend in the integration of information and communication technologies with physical objects and systems which can potentially change, and sometimes already has, the way people live and interact with the physical world. The same trend affects engineered systems, which can exploit embedded networking and computation devices to improve their efficiency and capabilities, as well as physical infrastructures and environments, which can be augmented with sensing, actuation, computational and networking capabilities, so as to became "smarte" systems, acquiring an autonomous behaviour or providing new services and functionalities to their users, either human or other systems.

When a system encompasses a large number of physical devices, each with computational and networking capabilities, being seamlessly integrated with a physical environment and capable of sensing and/or changing its status, and when all these devices cooperate with the others so as to provide a certain physical behaviour or new service and functionalities, that system can be called a *cyber-physical system* (CPS) [7].

According to NSF[93], research advances in cyber-physical systems promise to transform our world with systems that respond more quickly (e.g., autonomous collision avoidance), are more precisely (e.g., robotic surgery and nano-tolerance manufacturing), work in dangerous or inaccessible environments (e.g., autonomous systems for search and rescue, firefighting, and exploration), provide large-scale, distributed coordination (e.g., automated traffic control), are highly efficient (e.g., zero-net energy buildings), augment human capabilities, and enhance societal well-being (e.g., assistive technologies and ubiquitous healthcare monitoring and delivery).

This vision lays down a challenge for new frameworks, models and algorithms to be proposed, so as to support the design of these kinds of complex system, taking into account their requirements and related issues such as adaptivity, devices heterogeneity, large-scale, reconfiguration, responsiveness, decentralization and reliability.

The main contributions of this thesis concern:

1. the definition of a cloud-assisted, agent-based framework for designing large-scale Cyber Physical Systems, tackling issues like heterogeneity, scalability and reconfigurability;
2. the proposal of models and patterns for the design and implementation of CPS, through the use of decentralized agent-based algorithms in various application scenarios;
3. the proposal of a decentralized control system for urban drainage networks, which can improve the performance of an existing network reducing its environmental impact.

The rest of this thesis is structured in four chapters, as follows:

Chapter 1 presents the fundamental concepts of Cyber-Physical Systems, characterizing them, highlighting open issues and presenting possible application scenarios.

Chapter 2 is devoted to present Rainbow, a cloud-assisted, agent-based platform for Cyber-Physical Systems, detailing its main components and features.

Chapter 3 shows how the Rainbow platform can be exploited in multiple application scenario, using proper decentralized algorithms and cooperation mechanism.

Chapter 4 presents a methodology for managing an urban drainage network, seen as a Cyber Physical System, so as to provide an optimized behaviour to the system, which is capable of reducing effectively both urban flooding and combined sewer overflow phenomena, as well detailed in the experimental section.

Rende, November 2015, *Andrea Vinci*

Contents

List of Figures

List of Tables

1

An introduction to Cyber-physical Systems

1.1 Introduction

Over the last two decades, the advancements in computing and communication technologies have been so significant that have given rise to an information technology (IT) revolution. The Internet has changed the way we interact and communicate each other and also how we create, distribute and consume information. Furthermore, the advent of ubiquitous embedded computing, sensing, and wireless networking technologies has also changed how we interact, control, and build physical engineered systems such as automobiles, aircrafts, power grids, manufacturing plants, medical systems and building systems, on which our modern society and economy are becoming highly dependent.

The potential benefits of the convergence between the cyber-world, i.e. computation and networking, and the physical world for developing new engineered systems are transformative and wide ranging. Through embedded systems for distributed sensing, computation and actuation over wired or wireless communication networks, multi-objective optimization and high level decision-making algorithms, engineered systems in many application domains, such as transportation, energy, domotic and medical systems, can be designed and developed to be much smarter, more reliable, secure, efficient and robust.

Recently, we have experienced an ongoing seamless integration between the cyber and the physical worlds, at every scale. Everyday objects can be enhanced with computation and networking capabilities, so as to provide additional features, like facility to notify a change of status to a remote server and to be remotely controlled by smartphones. These so-called *smart objects* can even execute simple tasks, such as keeping a given temperature in a room or dispensing the correct amount of water in the soil to keep the right humidity for the health of a vegetable. *Smart Objects* are the building blocks of the *Internet of Things* (IoT) vision [44].

The Internet of Things[52] is the network of physical objects, or “things” , embedded with electronics, software, sensors, and network connectivity, which enables these objects to collect and exchange data. IoT can be seen as a realization of the concept of ubiquitous computing, proposed by Mark Weiser in the early '90s [90].

When a set of these networked *things* are placed in the same environment and cooperate to provide new functionalities or to achieve a common goal, it is possible to extend the abstraction of smart objects so as to define *smart environments*. In the literature, there are many different definition of smart environment, among them: (i) "*a small world, where all kinds of smart devices are continuosly working to make inhabitants' lives more confortable*"[20] and (ii) "*an environment that is able to acquire and apply knowledge about the environment and its inhabitants, in order to improve their experience in that environment*"[94].

Besides and in parallel with the IoT vision, the engineering research field of cyber-physical systems (CPS) has attracted a great deal of attention from academia, industry and governments. As a whole, CPSs can be described as "*smart systems that encompass computational (i.e., hardware and software) and physical components, seamlessly integrated and closely interacting to sense the changing state of the real world*"[93], or even as the "*next generation of engineered systems that require tight integration of computing, communication, and control technologies to achieve stability, performance, reliability, robustness and efficiency in dealing with physical systems in many application domains*" [66].

CPSs are different from desktop computing, traditional embedded/real-time systems, and Wireless Sensor Networks; however, they have some defining characteristics as follows[84]:

- Cyber capability in every physical component and resource constraint. The software is embedded in every embedded system or physical component, and the system resources (e.g., computing and network bandwidth) are usually limited.
- Closely integrated. CPSs deeply integrate computation with physical processes.
- Networked at multiple and extreme scales. CPSs, the networks of which include wired/wireless network, Wi-Fi, Bluetooth, and GSM, among others, are distributed systems. Moreover, the system scales and device categories appear to be highly varied.
- Complex multiple temporal and spatial scales. In CPSs, the different components are likely to have unequal granularity of time and spatiality. CPSs are strictly constrained by spatiality and real-time capacity.
- Dynamically reorganizing/reconfiguring. CPSs, as very complicated and large-scale systems, must have adaptive capabilities.
- Closed-loop control and high degrees of automation. CPSs favor convenient man-machine interaction, and advanced feedback control technologies are widely applied to these systems.

- Operation must be dependable and certified in some cases. Reliability and security are necessary for CPSs because of their extreme scales and complexities.

IoT and CPS research topics share similar goals and technologies but have different research communities and focus on different approaches and problems. Both of them concerns large-scale distributed computing systems of systems, where computation and "intelligence" is not decoupled from the environment. Conversely, while the IoT focuses on openness, interoperability and cooperation of objects in dynamic environments, whose elements continuously change over time, CPSs focus on more "static environments", where interactions take place between well-known and controlled participants. A typical CPS is often an engineered closed-loop controlled system, even though networked and with bigger scale.

For a better understanding of the different visions, we can look at Figure 1.1. While the Internet of Things is about extending the existing Internet so as to connect the objects in the physical world, a CPS is an application-specific system that integrates cyber and physical components to achieve a certain goal.

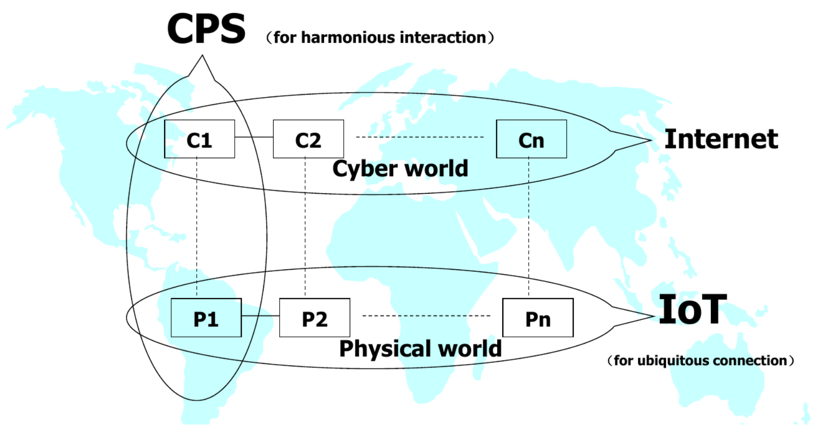


Fig. 1.1. Internet of Things versus Cyber-Physical Systems.

1.2 Enabling Technologies

The integration between the cyber and the physical world is made possible due to recent advancements in microelectronics, telecommunication systems, embedded systems, communication protocols and computer science, as detailed in the following.

1.2.1 Hardware miniaturization and energy efficiency

Traditionally, computing devices were stationary and mains powered, which meant processor units being designed with computing performance as the main goal. Since the early 2000s, with the end of Dennard Scaling[25, 35] and the growth of the demand for mobile devices, manufacturers have switched their focus from computational power to energy efficiency[22]. Nowadays all the main manufacturers produce devices and systems-on-a-chip that can be suitably embedded in objects and things (as well as in smartphones), thanks to their small form factor and low energy consumption. Obviously, the computational power of these kinds of device is not comparable with a desktop or server computer, but is enough to provide general purpose computational capabilities to an embedded object.

1.2.2 Networking technologies

Cyber-physical systems are built using networked components, which can be spread over small areas, like a room or an house, or wider areas, as a sewer network or a power grid. The telecommunications industry and research community developed new technologies that enable reliable wired and wireless connectivity both in wide area domains and in small area ones. Wireless internet connectivity is granted world-wide by digital cellular networks, whose technologies and standard evolved from the first digital mobile system (2G), providing more and more bandwidth [24]. The current standard, LTE-Advanced, is expected to offer peak rates up to 1 Gbit/s fixed speeds and 100 Mb/s to mobile users[76]. In a local area network, wireless connectivity is reached using the IEEE 802.11 standard. The current iteration is IEEE 802.11ac, which is an advancement in terms of both maximum transmission data rate and energy consumption[58]. Furthermore, the IEEE 802.11ah amendment is currently in a draft state, aiming at achieving a transmission range of up to 1 km at the minimum data rate of 100 kbps and low power consumption, so as to be used in wireless sensor networks and IoT on even an urban scale. In personal area networks, the reference is currently represented by ZigBee and Bluetooth specifications. The former is a low-cost, low-power, wireless mesh network standard targeted at wide development of long battery life devices in wireless control and monitoring applications[11] and is widely used in home automation applications. The latter provides a bigger data rate compared to ZigBee, at the cost of lesser transmission range and greater energy consumption[37]. Bluetooth is mainly used to establish single-hop connection between two devices for exchanging multimedia streams/data.

Given the large number of devices that can be involved in a CPS, other networking issues that need to be dealth with are the unique identification and the addressability of the devices. Back in 1999, the term ”Internet of things” was firstly introduced by Ashton [8] in a presentation about Radio Frequency Identification (RFID)[89] technology. He show how easily a computer can recognize objects that are attached with proper RFID tags, using a proper reader. The Electronic Product Code Tag Data Standard (EPC TDS)[77] specifies what information is encoded on an RFID tag to uniquely identify a certain object, together with some additional product information. Extensions to this standard are proposed, for example in [75], to better full-fill the needs for openness and interoperability of the Internet of Things.

The addressability issue is dealt with using the Internet Protocol Version 6 (IPv6), which has a larger address space than IPv4. The length of an IPv6 address is 128 bits, therefore, the address space has 2^{128} or approximately $3.4 * 10^{38}$ addresses, thus being more than sufficient for meeting the need of unique addresses for the next decades. Furthermore, the previously mentioned ZigBee and Bluetooth specifications provide extensions to integrate their stack with the IPv6 protocol[88, 71].

1.2.3 Computer science

Enabling technologies from a computer science perspective cover a wide range of research area. Back in 1946, the scientists behind the first computer developed it to perform ballistic computations. The first general purpose computer, ENIAC (Electronic Numerical Integrator and Computer) was capable of performing all the four basic mathematical operations plus the square root, using as working memory twenty accumulators, each of which were capable of storing a ten-digit decimal number. Actually, the first computers were targeted to do mathematical computation, and their only requirement was the correctness of the results. When they began to be used to close control loops around physical systems, only correctness was not enough, since time related issues have to be taken into account. This motivated the development of real-time computation[82, 50], which involved the problems of how to schedule computational tasks so that every job in every task is completed before its deadline. Many cyber-physical systems applications involve automation and actuation to a distributed physical environment with soft or hard real time constraints[43].

Cyber-physical systems can be seen also as a natural evolution of Wireless Sensor Networks (WSN). Under the flag of the Smart Dust[69] project, it was given the concept of mote, which is a tiny device capable of sensing, computation and communication. These motes allowed the attachment of sensors to the nodes, bringing information about the physical environment into the interconnected wireless sensor network [92] of computational nodes. When the nodes in a communication network are connected both to sensors and actuators, one obtains a Wireless Sensor and Actuator Network (WSAN)[87], which can be seen as a networked control system very similar to the concept of CPS.

Each device involved in a CPS can produce data about its internal state and measures about the environment in which it is deployed at a very high rate, thus techniques are needed to manage the huge amount of data coming from all the devices involved in a large-scale system. Traditionally, computing on data requires (i) the data to be completely stored somewhere and available as a whole and (ii) programs that are able to compute on a large piece of input data. When the data size is overwhelming, as in CPSs, smarter techniques need to be used, and the precision of results can be sacrificed in order to meet time and space requirements. Models and algorithms for dealing with these so-called data streams are surveyed, for example, in [55] and [4], where data streams are formally defined and algorithms are presented for statistical aggregation and for data mining tasks.

A data stream processing algorithm is an on-line algorithm that produces/updates the result as soon as a new element arrives from a stream. This requires that the algorithm can process each element only once (or a fixed number of times), then the processed element must be dropped. On the contrary, in some applications there is the need to store the data for further analysis and future uses. The "Big Data"[18] research topic refers to the collection of techniques that tackles the issues of storing, computing and mining on this kind of huge data-set, and there is a growing attention in the CPSs research community to this subject [47].

1.3 Applications

The effect of CPS research can be significant in many application domains, where large scale systems can exploit a closer integration between computation and physical processes so as to obtain new control functionalities and capabilities, or improvements in efficiency, reliability and adaptivity.

1.3.1 Energy Systems

Energy generation, transmission, and distribution for a clean and sustainable society are high-priority issues that are currently being investigated by both the industry and the scientific community.

Power electronics, power grid and embedded control form a CPS, whose design is heavily influenced by fault tolerance, security, decentralized control and economic, ethical and social aspects. The smart grid[6] is a next-generation infrastructure for electric power systems that can help to produce, distribute and use electricity in a cleaner, more cost effective and efficient manner through the integration of computing, communication and control technologies. The production and distribution of electric energy can be made more responsive and reliable through real-time distributed sensing, measurement and analysis. Furthermore, communication and information technology can contribute to improving the efficiency of overall electric consumption by encouraging consumers to avoid consumption peak times through a dynamic pricing mechanism and by providing useful real-time price information to consumers. Thanks to the infrastructure and mechanism for bidirectional exchange of information and electricity, smart grids allow traditional electric energy consumers to become providers. Electric energy that is stored or generated at residential and industrial facilities from renewable energy sources such as wind and solar power can be sold to other consumers in the neighbourhood or electric power providers. Recent advancement in CPSs for smart power grid are presented, for example, in [42].

Computing, communication, and control technologies can play an important role in improving efficiency in the home and in office building energy consumption. Appliances, lighting and HVAC (heating, ventilating and air conditioning) can be more efficiently managed through distributed sensing, by monitoring inhabitants behaviour and dynamically reacting to human activities and weather conditions[83, 95].

1.3.2 Transportation systems

The development of vehicles, mass transit, and traffic systems to address sustainability, efficiency, congestion and safety is an important research issue for the benefit of our environment, economy and safety. Next generation transportation systems can potentially integrate intelligent vehicles and intelligent infrastructures. Intelligent vehicles can be equipped with seamlessly integrated embedded computing systems and in-vehicle networking systems. Vehicles can exchange information through wireless communication between vehicle to vehicle and vehicle-to-infrastructure. Intelligent mass traffic systems can be more adaptive to the needs of users. Through these capabilities, vehicles can assist drivers or even drive autonomously by monitoring and estimating traffic conditions, planning ahead their behaviour, and implementing the plan through drive-by-wire functionalities such as stability control, speed control, braking and steering. Intelligent traffic infrastructures can be operated to manage the throughput of entire traffic systems. Intelligent mass transit systems can be better adaptive to the need of users.

1.3.3 Healtcare and medical Systems

It is an important challenge to design and develop medical devices and systems with better efficiency, reliability, intelligence, and interoperability. Medical devices need to be highly reliable; moreover, they should be operated in a patient-specific manner since patients have different physiological characteristics. Formal models of patient physiological dynamics, and the hardware and software systems of medical devices, and their interactions, can play an important role in designing and verifying the safety properties of devices. The integration of wireless networking and distributed sensing and computing infrastructure for interconnectivity and interoperability with medical devices enables the development of medical systems by which patient physiological conditions can be diagnosed and treated in a more integrated and intelligent manner.

1.3.4 Hydraulic infrastructures

Urban hydraulic infrastructures such as water supply or drainage systems, can benefit from CPSs so as to make them capable of autonomously adapting to exceptional events. Water supply systems, when properly equipped with actuation components, such as pump and gates, and sensors monitoring the state of the pipelines, can react to pipeline breakages or pressure drops, so as to favour high-priority buildings such as hospitals or schools. Drainage systems can use their storage capacity more efficiently, for reducing or avoiding urban flooding even when heavy rainfall events occur.

1.3.5 Home automation and ambient assisted living

Improving the quality of life for the disabled and the increasing proportion of elderly people is becoming a more and more essential task for today's societies. One way to achieve this goal is by making the home environment a more comfortable place to live in, equipping it with sensors and actuators devices that can be used for monitor and automation tasks. The elderly or physically disabled people can be continuously monitored, and their daily activities can be properly recognized using relevant sensors data and classification techniques. Predictive algorithms can be used to anticipate inhabitants' behaviour so as to forecast their needs and properly control light and appliances. Exceptional or risky situations, such as a fall or diseases, can be recognized too, so as to immediately send alarms to relatives or doctors, and switch off potentially dangerous appliances.

1.4 Challanges

Owing to their inherent complexity, the design of CPSs is faced with several challenges and issues, as described in [66][74][16][54].

1.4.1 Heterogeneity

Owing to the nature of many of the possible application contexts, there is the need to coordinate a large number of heterogeneous devices. If a CPS is designed for enhancing an existing system, most of the hardware devices previously involved can still be used in the new system, and only small changes are allowed. Devices can be heterogeneous in different ways. There are devices with no computational power on board, that are only simple electronically controllable sensors or actuators, and more sophisticated, "smart" devices, with little or even high computational power on board. Among the latter, devices can run different operating system, from embedded OS, as TinyOS, to linux-based or android-based systems. Furthermore, devices can be capable of communicating each with a different protocol, which can be even a proprietary one. In order to cope with this heterogeneity, integration techniques are needed, so as to allow all the devices involved to be capable of interact among themselves and with all the components of the CPS.

1.4.2 Scalability

As CPS may involve large physical infrastructures like transportation systems or power grids, a large number of physical entities have to be suitably monitored and controlled. Furthermore, a lot of physical applications are subject to different operating conditions, and the number of signals/measures being produced and managed can increase or decrease over time. As a result, there is the need to search for solutions which can scale, allowing a system to operate correctly and efficiently even under unexpected load peaks. This is more and more important for highly-critical systems, such as power grids and hydraulic networks.

1.4.3 Reconfigurability

In its lifespan, a cyber-physical system can change over time. New elements or devices can be added, e.g. when an urban infrastructure covers a new area, also obsolete or no longer needed elements can be removed. Furthermore, existing "smart" elements can need a software update or be physically repaired. When these changes involve highly-critical systems, a total shut-down for a scheduled maintenance can be not possible. A CPS must to be reconfigurable with little or no interruptions in its operations, reducing inefficiencies to a minimum

1.4.4 Cooperation models

Many existing systems exploit a centralized controller, which collects the measures generated from sensors, processes them so as to produce an actuation plan, and then transmits the appropriate signals to all the actuators. When the system requires coordination of a large number of devices, time constrained processing, and bandwidth limitations, this model does not perform well. Decentralized cooperation models can be better exploited, where each device behaves autonomously on the basis of its local status and exchanges information with others. Swarm intelligence and bio-inspired strategies can be suitably exploited for controlling a system in a decentralized fashion, and for the design of distributed and decentralized clustering and pattern detection algorithms[1] that can support the autonomous decisions of the elements involved.

[1] Spezzano, G., **Vinci, A**. (2015). Pattern Detection in Cyber-Physical Systems. Procedia Computer Science, 52, 1016-1021.

1.4.5 Human in the loop

In many application domains the operation of a CPS involves a closely interaction with people. These interaction can be indirect, when the behaviour of a system changing the status of an environment affects the people behaviour (and viceversa), and direct, when these interactions are designed to be part of the behaviour of a system, as in the Smart Home Environment scenario. These interactions have to be properly modelled, and communication mechanisms between human and CPS need to be designed, using graphic applications as user interface directly embedded on systems devices, web site portals, or even exploiting existing human oriented systems as social networks[2].

1.5 Conclusions

In this chapter, Cyber Physical Systems were over-viewed, defined as smart systems that encompass computational and physical components, seamlessly integrated and closely interacting to sense the changing state of an environment. Technologies enabling this vision have been summed up, and possible innovative applications listed. Finally, CPSs requirements and challenges, involving heterogeneity, scalability, reconfigurability and decentralization have been detailed. In the next chapter, a novel platform for large scale CPSs will be presented, providing solutions to the issues summarized in this chapter.

[2] Giordano, A., Spezzano, G., Sunarsa, H., **Vinci, A.** (2015, May). Twitter to integrate human and Smart Objects by a Web of Things architecture. In Computer Supported Cooperative Work in Design (CSCWD), 2015 IEEE 19th International Conference on (pp. 355-361). IEEE.

2

A Smart Platform for Large-Scale Cyber-Physical Systems[1]

2.1 Introduction

The increasing use of smart devices and appliances opens up new ways to build applications that integrate the physical and virtual world into consumer-oriented context-sensitive Cyber-Physical Systems (CPS) [46, 74, 45] enabling novel forms of interaction between people and computers. CPS are combinations of physical entities controlled by software systems to accomplish specified tasks under stringent real-time and physical constraints.

The emerging cyber-physical world interconnects a vast variety of static and mobile resources, including computing/medical/engineering devices, sensor/actuator networks, swarm of robots etcetera. Examples of CPS applications include [84] traffic control, power grid, smart structures, environmental control, critical infrastructure control, water resources and so on. These systems could be pervasively instrumented with sensors, actuators and computational elements to monitor and control the whole system. Furthermore, these devices should be interconnected so as to communicate and interact with each others and with people.

This scenario is supported by recent technology advancement in the fields of communication, embedded systems and computer science.

[1] - Giordano, A., Spezzano, G., **Vinci, A.** (2014). Designing Cyber-Physical Systems for Smart Infrastructures: The Rainbow Platform. ERCIM News, 2014(97).
- Giordano, A., Spezzano, G., **Vinci, A.** (2014). Rainbow: An Intelligent Platform for Large-Scale Networked Cyber-Physical Systems. In Proceedings of 5th International Workshop on Networks of Cooperating Objects for Smart Cities (UBICITEC), Berlin (pp. 70-85).
- A. Giordano, G. Spezzano, **A. Vinci**, A Smart Platform for Large-Scale Networked Cyber-Physical Systems. In Management of Cyber Physical Objects in the Future Internet of Things: Methods, Architectures and Applications, Springer 2016. (to appear)

The networked cyber-physical world has a great potential for achieving tasks that are far beyond the capabilities of existing systems. However, the problem of effectively composing the services provided by cyber and physical entities to achieve specific goals remains a challenge [46, 74, 1]. Advanced models and architectures, autonomous resource management mechanisms, and intelligent techniques are needed for just-in-time assembly of resources into desired capabilities.

The complexity of a CPS, and the large number of elements involved, makes data analysis and operation planning a very difficult task. A currently used approach involves two layers: a *physical* layer and a *remote* (cloud) cyber layer. The physical layer sends sensed data to a remote server, which processes them and computes a suitable operation plan. Afterwards, the remote server sends the sequence of operations it must execute to each device on the physical layer. The reasoning is performed in the remote layer. This solution cannot be applied when there are constraints on *responsivity* time, that is, when a system needs to react fast to critical events that may overwhelm its integrity and functionality. Communication lag and remote processing can cause delays that a system simply cannot bear.

A wide variety of applications means a wide variety of devices. Currently, there is a plethora of different devices, each with its own particular functionalities and capabilities. There are simple devices without any computational unit as well as "smarter" devices with high computation power inside. There are devices with no operating systems and devices with simple or complex operating systems, such as tinyOS or Android. Our framework is designed to cope with this inherent heterogeneity.

To addressing the issues described above, our proposal moves on these main lines:

- Hiding the heterogeneity of CPS by introducing a *virtual object* layer.
- Moving the computation as close as possible to the physical resources in order to foster good performance and scalability.
- Introducing a distributed intelligence layer between the physical world and remote servers (cloud), which can execute complex tasks and horizontally/vertically coordinate the devices;
- Switching from a cloud-based model to a cloud-assisted one, where the intelligent intermediate level carries out almost all the real-time control tasks, whereas the remote cloud level remains in charge of non-real-time tasks such as offline data analysis or presentation. The information provided by the data analysis executed by the remote server are used by the intermediate level to optimize its operations and behaviour.

In this chapter we propose a three-tier architecture (Rainbow) that uses single-board computers such as the Raspberry PI to connect massive-scale networks of sensors. This architecture is composed by the *Cloud* layer, the *Intermediate* layer and the *Physical* layer. Sensors are partitioned into groups, each of which is managed by a single computing node. These computing nodes host multi-agent applications designed to monitor multiple conditions or activities within a specific environment.

We present a new integrated vision that allows the designing of a large-scale networked CPS based on the decentralization of control functions and the assistance of cloud services to optimize their behaviour. Decentralization will be obtained using a distributed multi-agent system in which the execution of a CPS application is carried out through agents' cooperation [33, 48, 49, 13]. The distributed multi-agent system lays the foundations for properly exploiting swarm intelligence concepts. Swarm intelligence [15, 41] systems are typically multi-agent systems made up of a population of simple agents interacting locally with one another and with their environment. The agents follow very simple rules, and although there is no central control structure dictating how individual agents should behave, local and to a certain degree random, interactions among such agents lead to the emergence of "intelligent" global behaviour, unknown to the individual agents. Natural examples of swarm intelligence include ant colonies, bird flocking, animal herding, bacterial growth, and digital infochemicals. Agents interacting with cloud services can exploit the analysis, predicting, optimization and mining scalable capabilities on historical data allowing applications to adjust their behaviour to best optimize their performance.

2.2 Related work

In the recent years, the world has witnessed a real revolution about people habit in terms of ability of exploiting high technology solution in everyday life. This new scenario opens up new challenges regarding how the physical stuff can be used and integrated with the preexisting digital world. In these so called Cyber-Physical systems, many physical components collaborate each other by means of network communications in order to sense and act upon the physical world. These physical components are enhanced by using computational resources which supplies them the "smartness" needed to cope with complex tasks as controlling the physical environment and supporting most of the everyday human activities [93]. This scenario is supported by recent technology advancement in the fields of communication technologies, embedded systems and computer science. On the communication technologies side, new protocols like EPC TDS and IPv6 ensure unique addressability for all the elements involved in a CPS, while connectivity technologies like IEEE 802.11, ZigBee, Umts and ZTE, ensure light and fast connection both among the devices and between the devices and the Internet. On the embedded systems side, the miniaturization and the constant improvement of energy efficiency of electronic components enables the environment to be easily instrumented with sensors, actuators and computing devices, while the presence on the market of cheap and general purpose single-board computers, like Raspberry PI [34] and BeagleBoard opens up to new approaches and application scenarios.

Finally, on the computer science side, the development of new techniques to analyse a massive volume of data, together with the advances in the fields of artificial and swarm intelligence, allows us to properly deal with even a large number of devices.

The inherent complexity of these kind of systems is highlighted in [84], where cyber-physical issues are summarized in the following topics

- integrate the physical components in the digital world;
- supply each physical component of its own computational capabilities;
- communication and networking issues;
- dynamic reconfiguration;
- human interactions;
- security and reliability.

In order to deal with architectural issues posed by CPS, and well highlighted in [46], several frameworks and architectures have been proposed. [26] proposes Web of Things based framework for CPS where physical sensors and actuators are universally identified (through URIs) and modelled as WoT resources. The resources can be accessed by means of remote RESTful invocations. The architecture is built using two key components, CPS Node and CPS Fabric. CPS nodes are directly connected with physical devices and interconnected each other using the CPS fabric, which consists of a set of network system functions (i.e. routing, admission control, and so on). An Intelligent Vehicle System is shown as a case of study.

in [51, 56, 38] some middlewares are proposed which implement the pervasive computing paradigm in the CPS context. Each one presents a different framework which allows programmers to design and develop applications by coding in an high level of abstraction. The code is automatically compiled and deployed to the suitable computing nodes where sensors and actuators are managed through a wireless network. In [51], the framework provides an object oriented model and a rule-based mechanism. The framework also introduces the concept of *item* which consists in physical entities dynamically detected by the system on the basis of the values measured by the sensors. Every time a physical item is identified by a specific set of rules, an associated software object is properly instanced. [38] proposes a similar approach but exploiting Matlab.

Several works deal with CPS issues adopting multi-agent paradigms. In [86] a multi-agent system is used together with an event-based mechanism. In [49], a framework is described which lies on the *semantic agent* concept while in [48] the multi-agent systems is integrated with a service-oriented architecture (SOA). This work also suggests how well-known and proven swarm intelligence techniques can be properly adopted for industrial purposes. At last, in [33], an agent-based middleware for cooperating smart-objects is proposed. An implementation using JADE is also provided where a topic-based publish/subscribe protocol is exploited for permitting cooperation among agents.

2.3 Rainbow architecture

Rainbow is a three-layer architecture designed in order to bring the computation (i.e the controlling part) as close as possible to the physical part. Since CPS foresees that physical entities are spread across a large (even geographic) area, the previous assumption implies the controlling part to be intrinsically distributed.

Our proposal foresees the use of a *distributed agent-based* layer in order to address the aforementioned issues. The agent paradigm has several important characteristics:

Autonomy. Each agent is self-aware and has a self-behaviour. It perceives the environment, interacts with others and plans its execution autonomously.

Local views. No agent has a full global view of the whole environment but it behaves solely on the basis of local information.

Decentralization. There is no “master” agent controlling the others, but the system is made up of interacting “peer” agents.

Through these basic features, multi-agent systems make it possible to obtain complex *emergent* behaviours based on the interactions among agents that have a simple behaviour. Examples of emergent behaviour could refer to the properties of adaptivity, fault tolerance, self-reconfiguration, etcetera. In general, we could talk about *swarm-intelligence* when an “intelligent” behaviour emerges from interactions among simple entities . There is a plethora of bio-inspired swarm intelligence approaches in the literature that could be properly adopted in the context of CPS. For instances, in section 3.3 we show a CPS where swarm intelligence is used to map noise pollution inside a city area, while in chapter 4 a fully decentralized approach for controlling an urban drainage network is detailed.

Besides involving complex interaction pattern, a cyber-physical application may requires only a simpler reactive behaviour from agents, where agents’ actions are triggered by new messages from other agents or new event occurrences from the physical systems. In section 3.2 an application scenario concerning a CPS for building automation is detailed, which use only agents having such kind of behaviour.

Rainbow architecture is shown in Figure 2.1. As it can be seen, the architecture could be divided into three layers. The bottom layer is the one that is devoted to the physical part. It encloses sensors and actuators, together with their relative computational capabilities, which are directly immersed in the physical environment.

In the Intermediate layer, sensors and actuators of the physical layer are represented as virtual objects (VOs). VOs offer to the agents a transparent and ubiquitous access to the physical part due to a well-established interface exposed as API. VO allows agents to connect directly to devices without care about proprietary drivers or addressing some kind of fine-grained technological

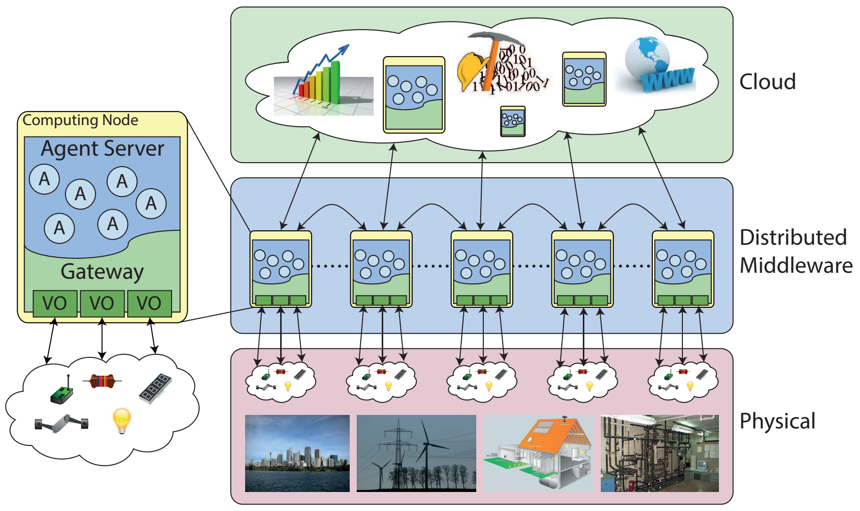


Fig. 2.1. Rainbow architecture.

issues. Each VO comprises "functionalities" directly provided by the physical part. Essentially, a VO exposes an abstract representation (i.e. *machine readble-description*) of the features and capabilities of physical objects spread in the environment. Functionalities exposed by different types of VOs can be combined in a more sophisticated way on the basis of event-driven rules which affect high-level applications and end-users.

In summary, as detailed in section 2.3.1, all the devices are properly wrapped in VOs which, in turn, are enclosed in distributed *gateway* containers. The computational nodes that host the gateways represent the middle layer of the Rainbow architecture. Each node also contains an agent server that permits agents to be executed properly. Gateways and agent servers are co-located in the same computing nodes in order to guarantee that agents exploit directly the physical part through VO abstraction. Instead of transferring data to a central processing unit, we actually transfer the process (i.e. fine-grain agent's execution) toward the data sources. As a consequence, less data needs to be transferred over a long distance (i.e. toward remote hosts) and local access and computation will be fostered in order to achieve good performance and scalability. Furthermore, since agents and Virtual Objects can be dynamically added or removed, if a system is well designed, it can be dynamically reconfigured and extended taking into account new physical objects and desired behaviour.

The upper layer of Rainbow architecture concerns the cloud part. This layer addresses all the activities that cannot be properly executed in the middle layer like, for instance, algorithms needing complete knowledge, tasks that require high computational resources or when a historical data storage is mandatory. On the contrary, all tasks where real time access to the physical part is required could be suitably executed in the middle layer.

2.3.1 Virtual Objects

We address issues about heterogeneity in CPS by introducing the Virtual Object (VO) concept. VO aims to hide heterogeneity by supplying a well-established interface permitting the physical parts to be suitably integrated with the rest of the system.

VO could be defined as a collection of physical entities like sensors and actuators, together with their computational abilities.

It can be composed of just a simple sensor or it can be a more complex object that includes many sensors, many actuators, computational resources like CPU or memory and so on.

In general, VO outputs can be represented by *punctual values* (e.g. the temperature at a given point of a room) or *aggregate values* (e.g. the average of moisture during the last 8 hours). Also, the values returned by VOs could be just the measurement of sensors or could be the result of complex computations (e.g. the temperature of a given point of space computed by means of interpolation of the values given by sensors spread across the environment).

Furthermore, a VO could supply actuation functionality by changing the environment on the basis of external triggers or internal calculus.

These different kinds of behaviour that VO can expose must be taken into account. VO is therefore conceived as a complex object that can read and write upon many simple physical resources. More in detail, we consider that each VO exposes different *functionalities*. Each functionality can be either sensing or actuating and can be refined by further parameters that dynamically configure it.

The previous assumption leads to the definition of *resource* as the following *triplet*:

$$[VOId, VOFunctionId, Params]$$

Where `VOId` uniquely identifies the VO, `VOFunctionId` identifies the specific functionality and `Params` is an ordered set of parameter values that configure the functionality.

For example let's consider a *Virtual Room* made of sensors for measuring different physical quantities inside a room such as *temperature*, *moisture*, *brightness* and so on. Suppose now you want to read from Smart Room the temperature in a given spatial point of the room. In that case the triplet could be:

$$[VirtualRoom, temperature, [x, y, z]]$$

Where `x`, `y` and `z` are the cartesian coordinate of the point of interest.

Using object oriented terminology, a Resource could be seen as a particular "instance" of a functionality of a given VO.

Besides read and write operations (i.e. sensing and actuation), it is provided for VOs to be able to manage events that occur in the physical part. To that scope, our proposed middleware includes a *publish/subscribe* component for managing events in each computing node. Each event is defined by a *logical rule* where one or more VOs could be involved.

Each rule is a *logical proposition* in which the *atomic predicates* can be of the following kinds:

- *resource < threshold (e.g. temperature < 300)*
- *resource > threshold*
- *boolean_resource (e.g. the_door_was_unlocked)*

Just an example of rule:

(temperature < 100 and brightness >500) or people > 3 or door_unlocked

The publish/subscribe manager component is in charge to parse the logical rule and generate a *binary tree* made as explained below: each node N of the tree corresponds to a logical proposition $N()$. given L and R, the child nodes of N, their associated logical propositions are respectively $L()$ and $R()$ so that it results either *N() = L() or R()* or *N() = L() and R()*. The radix of the tree corresponds to the entire rule while the leaves contain the atomic propositions that is passed to the suitable VOs. A binary tree representation example of a composed rule is shown in Fig. 2.2.

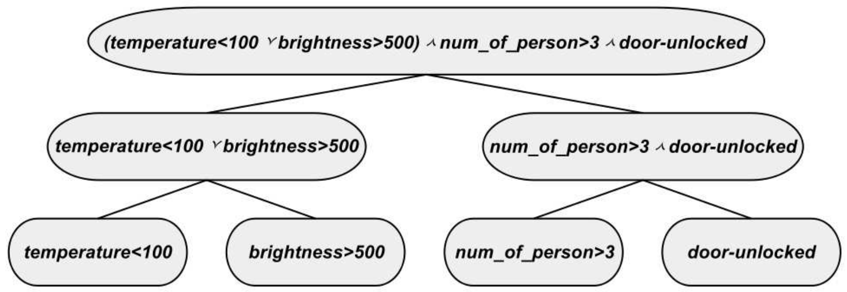

(temperature<100 ∨ brightness>500) ∧ num_of_person>3 ∧ door-unlocked
temperature<100 ∨ brightness>500
num_of_person>3 ∧ door-unlocked
temperature<100
brightness>500
num_of_person>3
door-unlocked

Fig. 2.2. Example of binary tree of a rule.

A VO is in charge to establish each time when the assigned atomic propositions are $true$ or $false$. The logical proposition of a given node is computed on the basis of the value of its child nodes. The root of the tree is recursively involved by this bottom-up computation. As soon as the value of the root node (i.e. the value of the entire rule) changes all the subscriber will be notified.

All the physical things linked to a computing node together with relative VOs is enclosed in the *Gateway* container. The Gateway exposes an interface to interact directly with the VOs.

Each gateway represents the "entry point" that agents can use to exploit VOs of the relative computing node.

In the following is described the interface of Gateway that will be used by the overlying layer:

```
interface GatewayInterface {
    void resourceNaming(String name, VOId voId, VOFunctionId functionId
        , VOFunctionParams params);
    VOResult check(String name);
    VOResult check(String name, VOFunctionParams params);
    VOResult acting(String name);
    VOResult acting(String name, VOFunctionParams params);
    void setRule(Rule rule, String idRule);
    void subscribe(String idRule, EventHandler handler);
}
```

The method `resourceNaming` assigns an identification `name` to a given resource supplied by a given VO. A resource is a specific instance of a *functionality* of a VO refined by some *parameters*. In other word, a resource is the above-mentioned triplet: $[VOId, VOFunctionId, Params]$. The `name` assigned to a resource via `resourceNaming` can be used in the other methods in order to simply identify the resource. Furthermore, the identification `name` of a resource is useful to compose the rules in a more human-readable fashion.

The method `check` reads the current value of the resource identified by `name` whereas `acting` triggers the actuation operation upon the resource identified by `name`. Both `check` and `acting` methods are of two kinds: the first take only `name` as parameter and refers to the resource as it is previously defined in `resourceNaming`; the second kind, instead, permits the parameters of the referred resource to be refined dynamically.

The method `setRule` permits a complex rule to be published (e.g. *(temperature < 100 and brightness >500) or number_of_ person > 3 or door_unlocked*) and to assigns an id (i.e. `idRule`) useful for subscribing the rule afterwards.

The method `subscribe` permits a previously published rule (identified by `idRule`) to be subscribed. The occurrence of the event identified by `idRule` will be notified to the `handler` passed as a parameter to the method.

2.3.2 Rainbow Multi Agent system

The Multi Agent component of the rainbow architecture is made up of the following entities: *Agents*, *Messages*, the *Agent Server* and the *Deployer*. Figure 2.3 shows these entities and how they interact among themselves and with the *Gateway*.

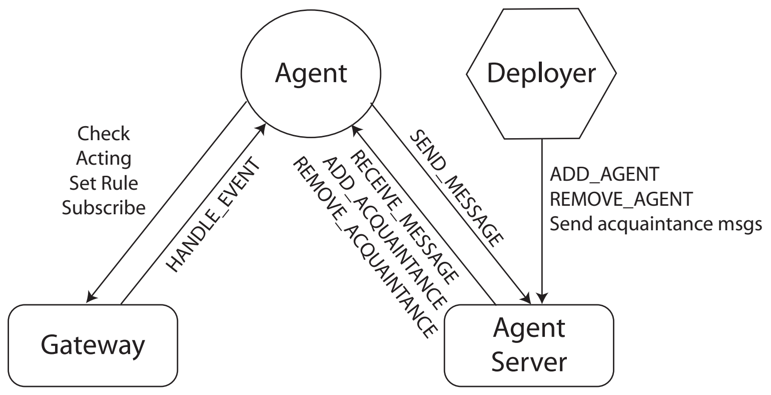


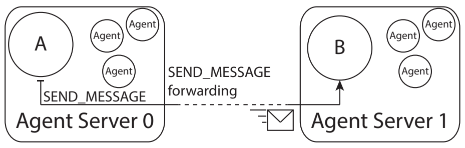
Fig. 2.3. Rainbow multi-agent entities.

The *Agent Server* is the container for the execution of agents. It offers functionalities concerning the life cycle of the agents as well as functionalities for agents' communication. Agent servers are arranged in a peer-to-peer fashion where each agent server hosts a certain number of agents and permits them to execute and interact among themselves in a transparent way. In other words, when an agent requests the execution of a functionality, its host agent server is in charge of redirecting transparently the request to the suitable agent server. In the following are listed the main functionalities each agent server exposes:

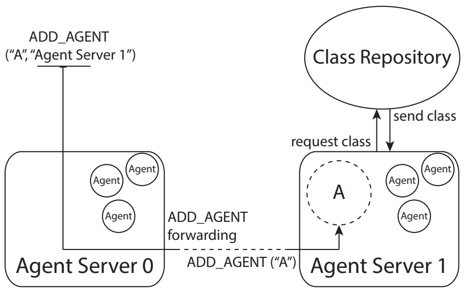
`SEND_MSG`. Through this functionality, the communication between agents is performed. The Agent Server is responsible for correctly delivering messages from the sender agent to the receiver one. If the sender and the receiver do not belong to the same agent server, the message is forwarded to the suitable "peer" agent server which is, in turn, engaged finally to deliver the message. The latter mechanism is showed in figure 2.4(a).

`ADD_AGENT`. It instances an agent to an agent server. Rainbow Multi Agent system is designed to permit agents to be dynamically loaded to the agent server they have to belong to. As in SEND_MSG operation, agent servers are in charge for exchanging information among themselves in order to guarantee the ADD_AGENT request to be delivered to the correct agent server. This mechanism is shown in figure 2.4(b). The latter figure also shows how the code is dynamically loaded exploiting *class repository server*. More in detail, when an ADD_AGENT request reaches the suitable agent server, if the agent code is not already available, the agent server automatically downloads it from a class repository.

`REMOVE_AGENT`. It removes an instance of an agent hosted by an agent server. This operation also exploits the "forwarding" mechanism described above.



(a) SEND_MSG



(b) ADD_AGENT

Fig. 2.4. Forwarding mechanism

A *Message* is the atomic element of communication between agents. It carries an application specific content together with informations about the sender agent and the receiver one.

Our architecture provides for specific kinds of message, that are the *acquaintance messages*. Those messages are used for establishing an acquaintance relationship among agents. The acquaintance message carries information about the location of a given agent (i.e. location of hosting agent server). The agent who receives the acquaintance message will use this information when it needs to send messages toward that destination. This kind of mechanism ensures agent behaviour to be completely independent w.r.t. the locations of agents it has to collaborate with.

For instance, let's consider that an agent is a computing node interconnected with others by means of a ring network. Each agent, therefore, can only interact with its previous agent nodes and its next one. Whenever further nodes must be connected to the ring network, only the acquaintance relationships have to be updated. In other words, a third entity can establish dynamically those acquaintance relationships without resorting to modifying, re-building or restarting any agent.

In rainbow architecture the entity which is in charge of sending acquaintance messages in order to establish the acquaintance network is called *Deployer*. Deployer could be an external process as well as an agent, it can run during the configuration phase as well as during application execution. The Deployer concept will be described in details in section 2.3.2.

An *Agent* is an autonomous entity which executes its own behaviour interacting with other agents via Agent Server. In addition, each agent can interact with the physical part exploiting functionalities exposed by the Gateway (i.e. using the Virtual Object abstraction).

The functionalities of an agent are exposed to its own Agent Server and Gateway. As said before, Agent Servers are in charge of the "forwarding" mechanism that eventually ends with the calling of these functionalities, while the Gateway is in charge of notifying the events that occur in the physical part. In the following are listed the main functionalities of an agent:

`RECEIVE_MESSAGE`. It is called when there is a Message to be delivered for the agent.
`HANDLE_EVENT`. It is called by the Gateway to notify that an event is occurred in the physical part.
`ADD_ACQUAINTANCE`. It is called when there is an acquaintance message to be delivered to the agent. The implementation of this functionality concerns the store of the acquaintance relationship between the agent itself and the agent identified inside the message.
`REMOVE_ACQUAINTANCE`. It is called for removing a previously stored acquaintance relationship.

The specific behaviour of an Agent is realized through the implementation of `RECEIVE_MESSAGE` and `HANDLE_EVENT` functionalities.

Dynamic Deployment and Roles

The deployment of the agents as well as the configuration of the acquaintance relationships and the start-up of the application are all actions performed by the so-called *Deployer*. An external process or even an agent can act as a Deployer. The deployment phase is typically executed just before the application can start properly; however, it is possible to act as Deployer even during application execution in order to update the configuration dynamically for hosting new features or adapting to foreseen and unforeseen changes in the environment. Deployer can be implemented centrally or in a distributed way. Basically, who acts as a Deployer operates using the `ADD_AGENT` functionality for deploying a new instance of an agent into an agent server, `REMOVE_AGENT` for removing a running agent from an agent server. Furthermore, Deployer is responsible for sending acquaintance messages that eventually end with calls to `ADD_ACQUAINTANCE` or `REMOVE_ACQUAINTANCE` on the specific agents. Finally, Deployer is also in charge of sending suitable "start" messages using `SEND_MSG` in order to start the application properly.

The acquaintance relationship is formally defined by a triplet: [A, B, R] where A and B are the agents involved in the relationship and R is a *Role* label. The triplet above means that agent A knows agent B and that B has the role R as acquaintance of A. During the execution, an agent exploits the Roles of its acquaintances to discriminate about how to interact with them.

As an instance, let's consider that each agent represents a physical person in a town. The relationship between two agents could have roles of neighbourhood and/or friendship. A deployer is in charge of configuring those relationships during the initial phase. In addition, as soon as a person changes home or starts a new friendship, the deployer has to re-arrange relationships dynamically among agents through sending acquaintance messages. During the execution of that system, each agent will use roles of neighbourhood and friendship to discriminate how to interact with other agents. For instance he/it can exchange information about its district with its neighbours while it invites its friends to a party.

2.3.3 Cloud layer

In the cloud part a set of Rainbow nodes are deployed and run on a cloud infrastructure. Such nodes lack of the Virtual Objects Gateway since, obviously, there are no physical entities connected. For this reason each node consists of the only agent server. The communication between the nodes connected with the physical part and the nodes in the cloud occurs by means of message exchange (see Section 2.3.2).

Agents located on the cloud nodes act as intermediary between the Rainbow MAS and cloud analytics services. The feature of adding new agents at runtime can be used to link new services in the cloud during the execution of the system (no reboot is needed).

The Rainbow cloud part is PaaS, namely it provides a software stack and a set of libraries for the application execution. Anyway, it also can be seen as SaaS because the Rainbow user, i.e. the application developer, can inject his application by remotely add the needed agents.

2.4 Conclusions

In this chapter we introduced Rainbow, an architecture that permits an easy development of large-scale cyber-physical applications. The novelty of Rainbow is that it relies on the adoption of a distributed multi-agent layer on top of the physical part that is, in turn, wrapped in suitable Virtual Objects. Rainbow aims to hide heterogeneity, cope with complexity and real-time issues. In the future, new intelligent, adaptive and decentralized algorithms will be explored for developing large-scale cyber-physical applications using Rainbow, such as those related to smart cities, power grid, water networks and so on.

In the following chapters a set of application scenarios that use Rainbow will be presented. Chapter 3 shows applications and solution for Smart Buildings, Ambient Assisted Living and Smart Cities scenarios. Chapter 4 is entirely focus on a CPS for an urban drainage systems, which implements a

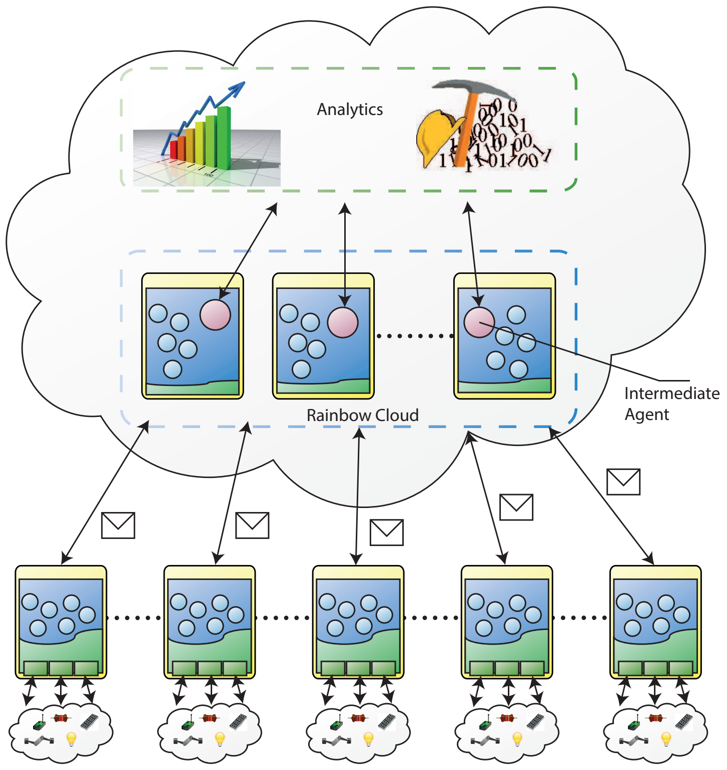


Fig. 2.5. Rainbow cloud layer.

novel decentralized approach so as to obtain a behaviour which is effective in reducing environmental impact of the drainage network.

3

Rainbow in action

3.1 Introduction

In this chapter we details the design of four CPS which exploit the features and capabilities provided by the Rainbow framework in different application scenarios.

The first one (Section 3.2) is a case study which aims to show our architecture from a practical perspective so as to better understand and figure out all the system's details and concerns a Building Automation scenario..

The second scenario (Section 3.3) concerns the use of the platform for mapping the noise pollution on an urban area, and shows how Rainbow can be exploited to run swarm intelligence algorithms in order to realize CPS applications owning properties such as adaptivity, fault tolerance, self-reconfiguration.

The third scenario (Section 3.4) proposes a solution for Ambient Assisted Living in Smart Home environment. The main purpose of this section is to point out how a knowledge discovery workflow can be realized involving all the three layers of the proposed architecture[1].

The last scenario (Section 3.5) is about the design of a Smart Street environment. Indeed, this scenario has been physically realized in the city of Cosenza (Italy).

3.2 Floor control

This section presents an application for monitoring and controlling a floor of a building hosting offices. Each floor contains a certain number of room.

[1] G. Fortino, A. Giordano, A. Guerrieri, G. Spezzano, **A. Vinci**. A Data Analytics Schema for Activity Recognition in Smart Home Environments. Proceedings of9th International Conference on Ubiquitous Computing and Ambient Intelligence (UCAMI 2015) (to appear)

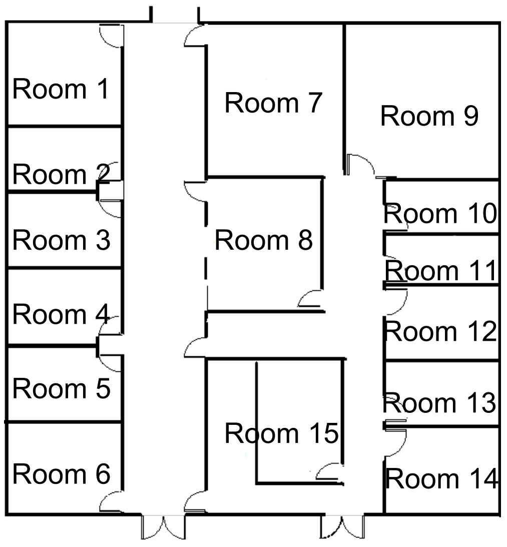


Fig. 3.1. Floor topology.

Figure 3.1 shows how a generic floor could be. In general, each room contains: doors, desks, chairs and adjustable brightness lights.

Each room is instrumented by some sensors and actuators listed below.

Sensors:

- sensors that detect the opening and closing of doors;
- sensors that detect when a person enters or leaves a room;
- proximity sensors detecting presence of the people in each zone of a room;
- a weight sensor for each chair in order to detect if the chair is currently used.

Actuators:

- adjustable brightness lights for all zones of a room;
- a display on each desk.

The use of the above described devices, for example, permits adjusting lights on the basis of people movements, writing informational messages on displays and so on.

Integration in Rainbow using Virtual Objects

In order to develop the controlling part in a object-oriented fashion, it is required to integrate the above described physical things with Rainbow middleware defining the suitable Virtual Object(VO). Each VO abstracts and wraps a certain number of sensors as well as actuators. For the sake of simplicity, in this example we chose to design VOs in a human-readable fashion: *virtual desk*, *virtual chair*, *virtual door* and *virtual wall*.

The functionalities exposed by these VOs are listed in table 3.1, 3.2, 3.3, 3.4. It is worth to note that each functionality of the virtual wall is parametric: the *zone* parameter specifies which area of the room is referred.

Functionality	**Type**	**Description**
lock	Sensing	Boolean, true if the door is closed
unlock	Sensing	Boolean, true if the door is open
entry	Sensing	Boolean, true when a person enter the room through the door
exit	Sensing	Boolean, true when a person exit the room through the door

Table 3.1. Virtual Door.

Functionality	**Type**	**Description**
proximity	Sensing	detects people near the chair
sitting	Sensing	Boolean: true when someone sits on the chair

Table 3.2. Virtual Chair.

Functionality	**Type**	**Description**
near people	Sensing	number of people in the zone (supplied by parameter)
add light	Acting	increase light brightness in the zone (supplied by parameter)
less_light	Acting	decrease light brightness in the zone (supplied by parameter)
light_off	Acting	set off light in the zone (supplied by parameter)

Table 3.3. Virtual Wall.

Functionality	**Type**	**Description**
proximity	Sensing	detects people near the desk
display	Acting	show a message supplied by parameter on the display

Table 3.4. Virtual Desk.

Each VO is located on the same computing node where the sensors and actuators that VO encloses are connected to. A computational node can generally host VOs that may refer to more than one room. Assuming than we have only three computational nodes available to monitor and control the whole floor, we can assign rooms to nodes as in figure 3.2.

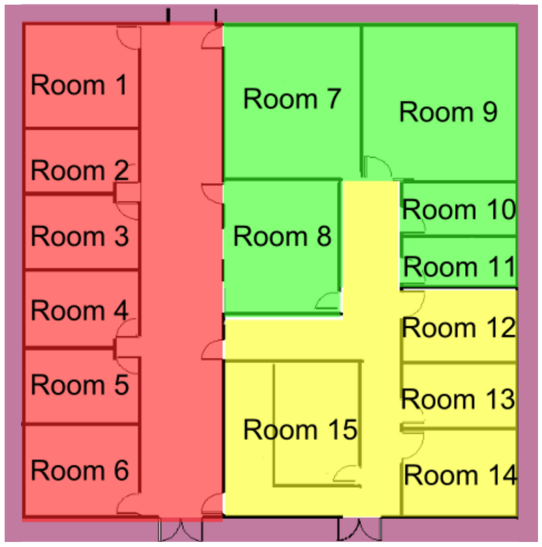


Fig. 3.2. Rooms assignment to computational nodes. Each different color identify a different node.

Multi-agent floor application

The application is designed for managing the floor and its rooms. For each room a energy-saving light-management is developed which considers people presence for suitably adjusting the brightness of the various zones of a room. This control management will also consider if the chairs are utilized or not in order to better adjust the lights. In addition, it permits a message to be displayed on a certain desk when needed. All those features are implemented in the *RoomAgent*. The code inside the RoomAgent is a typical object-oriented code where VOs are exploited as simple objects. The code is omitted in this chapter for sake of brevity.

Besides this room-wise features, the application is also designed for addressing issues concerning the entire floor (i.e. where more than one room is involved). For instance, it could be useful to know how many people are in the floor at a given time in order to properly manage the locking of the main door of the floor as well as to shut down all the lights where the floor is empty. In this example, instead, the knowledge of the number of people is used to notify a person when he is alone in the floor writing a message on the display of his desk.

The *FloorAgent* is designed for addressing the above mentioned issues. Summarizing, there is a *RoomAgent* per room and a unique *FloorAgent* as it is shown in figure 3.3.

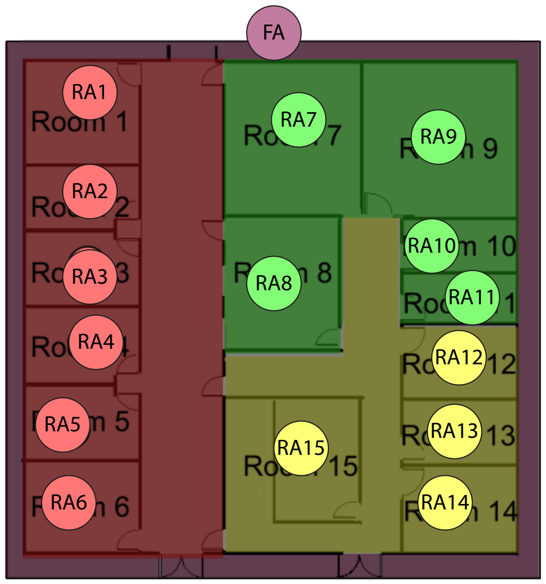


Fig. 3.3. Logical distribution of agents in the floor.

Deployment of the application

As mentioned before the *Deployer* is in charge to load the agents upon the agent servers, to establish acquaintance relationships among them and to start the application.

In our application, each *RoomAgent* must be located in the computing node where the VOs of the relative room belong to.

Conversely, the *FloorAgent* can be located everywhere in the system (it has not connection with any physical part), even in a remote cloud node. The process made by the *Deployer* is summarized in figure 3.4.

Agent interaction and acquaintance relationships

After loading each agent in the proper location, the *Deployer* sends acquaintance messages to each *RoomAgent* in order to let them know the *FloorAgent*.

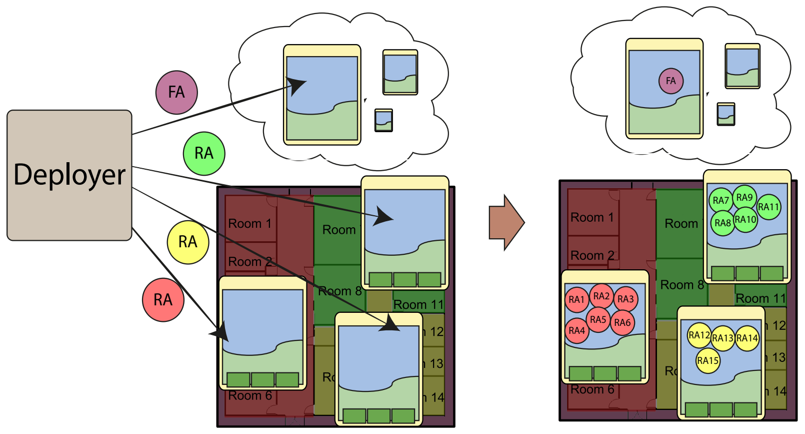


Fig. 3.4. Deployment of the agents and their physical distribution on the computing nodes.

Afterwards, each agent sends an acquaintance message to the *FloorAgent* in order to be known by it. This is an example of an agent that acts as *Deployer*. Once the deployment phase is completed, the application execution can start. When a person leaves a room, *RoomAgent* will be notified by the gateway and, consequently, will send a message carrying the number of people currently inside the room to the *FloorAgent*. The latter will update its people counter on receiving such a message. When it verifies that there is only one person in the floor, it will send a message to the relative *RoomAgent* that, in turn, will write a message on the desk display.

3.3 Noise pollution mapping

Many environments, such as airports, road works, factories, construction sites, and other environments producing loud noises, require effective noise pollution monitoring systems. Noise pollution is a common environmental problem that affects people's health by increasing the risk of hypertension, ischemic heart disease, hearing loss, and sleep disorders, which also influence human productivity and behavior [79]. For this reason the European Community passed the directive 2002/49/EC [28], which declares noise protection as one necessary objective to achieve a high level of health and environmental conservation. The directive imposes several actions to be made upon member states, including the mapping of noise in larger cities via noise maps. On the basis of these maps, the countries can formulate plans to counter the threat that is noise pollution.

Noise maps are mostly based on numerical calculations that have shown to give good estimates of long term averaged noise levels. However, such maps does not take into account the real-time variation of the noise levels.

Using the rainbow platform we designed an agent-based, self-organizing system for the real-time construction of noise maps and identification of the sources of noise.

Noise sensors are spread into the environment, linked to the computational nodes, and suitably wrapped inside the VOs. Each agent is directly associated with a VO representing a noise sensor. During the deployment phase, each agent is supplied by the knowledge of its neighbours (i.e. agent associated with a spatially near sensor).

We use a simple self-organizing algorithm, proposed by [13], to let sensor network to self-organize itself in a region partitioning based on similar sensing patterns (*noise levels*). Regions can grow or shrink according to the dynamic variation of noise levels. Organization in regions occurs by creating an overlay network made by agents connected by virtual weighted links. Agents belonging to the same region will have strong links, while agents belonging to different regions will have weak (or null) links.

In the following the details of the algorithm. Let s_i and s_j be two neighbour sensor agents. Let $n(s_i)$ and $n(s_j)$ the values of noise sensed by s_i and s_j, respectively. Let us assume that a distance function D can be defined for couples of v values. Region formation is then based on iteratively computing the value of a logical link $l(s_i, s_j)$ for each and every agent of the system as in following update_link procedure:

```
Update_link:
$if(D(n(s_i), n(s_j)) < T\{$
    $l(s_i, s_j) = min(l(s_i, s_j) + \Delta, 1)$
$\}else\{$
    $l(s_i, s_j) = max(l(s_i, s_j) - \Delta, 0)$
$\}$
```

Where: T is a threshold that determines whether the measured values are close enough for $l(s_i, s_j)$ to be re-enforced or, otherwise, weakened; and Δ is a value affecting the reactivity of the algorithm in updating link. Based on the above algorithm, it is rather clear that if $D(n(s_i), n(s_j))$ is lower than threshold T, $l(s_i, s_j)$ will rapidly converge to 1. Otherwise it will move towards 0. Transitively, two nodes s_h and s_k are defined in the same region if and only if there is a chain of agents such that each pair of neighbours in the chain are in the same region. From the Rainbow perspective, region information is stored adding/removing new acquaintance relationships among agents.

In order to properly map the noise pollution, it is necessary that each and every agent within a region is locally provided with information related to the overall status of the region. To this end, it is possible to integrate forms of diffusive gossip-based aggregation [40] within the described general scheme. The algorithm requires that the agents periodically exchange information with their neighbors about some local value, locally aggregate the value according to some aggregation function (e.g., maximum, minimum, average, etc.), and further exchange in the subsequent step the aggregated value.

3.4 Smart Home[2]

In the following, the design of a possible Smart Home Environment using the Rainbow platform is shown, which use the three layers of the architecture to execute activity recognition and discovery tasks.

Smart Homes [14] are advancing as a disruptive trend in the literature. Regarding Smart Homes, much progress has been made in the past few years on several fields: *Home Automation and Domotics* [5], *Energy Optimization* [81], *Distributed Sensing and Actuation* [32], *Activity Recognition* [68], *Ambient Assisted Living* [57] [29], *Indoor Positioning Systems* [70], and *Home Security* [73].

A very important and interesting application scenario for Smart Homes has emerged in the last few years. Given the growth in population age, more and more elderly people living alone require assistance 24/7. *Activity Recognition* [68] using data mining techniques can be successfully adopted for this purpose. Indeed, applying these techniques on data coming from the home environment permits us to recognize and analyse human activities in real time and, eventually, to send alarms to relatives or doctors of the monitored person.

Nowadays, Smart Homes is an important research field both in academia and in industry [14]. One of the major challenges in the realization of these applications is interoperability among various devices and deployments. Thus, the need for a new architecture - comprising smart control and actuation - has been identified by many researchers. In this context, the increase of an ageing population means a growth in the need to implement Activity Recognition [67] in Smart Homes. Several works have been done in this direction.

The authors of [39] introduce an activity recognition system for single person smart homes using Active Learning. They apply data mining techniques to cluster in-home sensor samplings so that each cluster represents a human activity. The users have to label each cluster with the corresponding activity so as to allow the system to learn how to recognize future activities. The system has been designed to detect not only single but also overlapping activities.

In [68] [21] the authors present an automatic approach to activity discovery and monitoring for assisted living in a real world setting. It does not use supervised methods for activity recognition, but it automatically discovers activity patterns. Moreover, the system is designed to discover variations in such patterns and is also able to handle real life data by dealing with different sensor problems.

[2] G. Fortino, A. Giordano, A. Guerrieri, G. Spezzano, **A. Vinci**. A Data Analytics Schema for Activity Recognition in Smart Home Environments. Proceedings of9th International Conference on Ubiquitous Computing and Ambient Intelligence (UCAMI 2015) (to appear)

The work in [85] shows a prototype used to forecast the behaviour and health of the elderly by monitoring the daily use of appliances in a smart home. Elderly people perform activities at regular intervals of time. When daily activities are performed regularly, it means that the elderly man/woman is in a state of good health. In this work, Predictive Ambient Intelligence techniques are used involving the extraction of patterns related to sensor activations.

In [19] the authors introduce an ontology-based hybrid approach to activity modelling that combines domain knowledge-based model specification and data-driven model learning. Central to the approach is an iterative process that begins with "seed" activity models created by ontological engineering. The "seed" models are deployed, and subsequently evolved through incremental activity discovery and model update. The rationale of this work is to provide (on single-user single-activity scenarios) generic activity models suitable for all users and then to create individual activity models through incremental learning.

The aim of this section is to show how the Rainbow platform can be used in the context of Smart Home Environment for Activity recognition tasks, allowing:

- cloud-assisted integration between environmental sensory data with data coming from wearable sensors which permits the better detection of human activities;
- the possibility of dynamically evolving the classification model by adding or updating classifiers as soon as new activities are discovered;
- an easy integration of different kinds of smart home applications with the activity recognition one.

3.4.1 Knowledge discovery workflow for home activity recognition.

Even though the Rainbow architecture can be effectively used in many smart home application domains, as the one detailed in section 3.2, this section is mainly focused on the activity recognition task because it can be suitably exploited to analyse human behaviour so as to realize and prevent dangerous situations which may occur in home environments. The activity recognition knowledge discovery workflow of the Rainbow architecture is presented in the following.

Figure 3.5 shows the four main tasks for the in-home activity recognition: *data acquisition*, *feature extraction*, *activity discovery*, *activity recognition*. The data acquisition task consists in collecting the sensor data from the home environment. The feature extraction task is focused on filtering the acquired data in order to extract the relevant information which is then rearranged and passed to the activity discovery and activity recognition tasks. The activity discovery task uses the features to discover new previously unknown activities in order to enlarge the knowledge thus dynamically producing new *classifiers*. These classifiers are eventually used by the activity recognition task which is

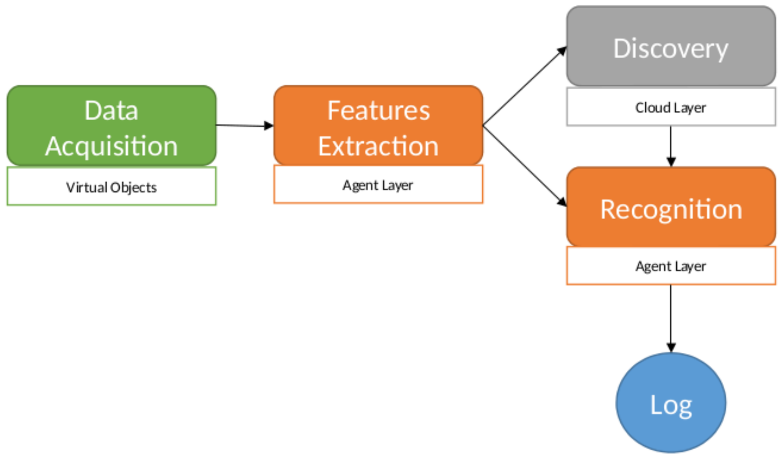


Fig. 3.5. Activity recognition tasks flow. Each task is executed in a different layer of the Rainbow architecture, as labeled.

in charge of processing the features and recognizing the high level activities in real time.

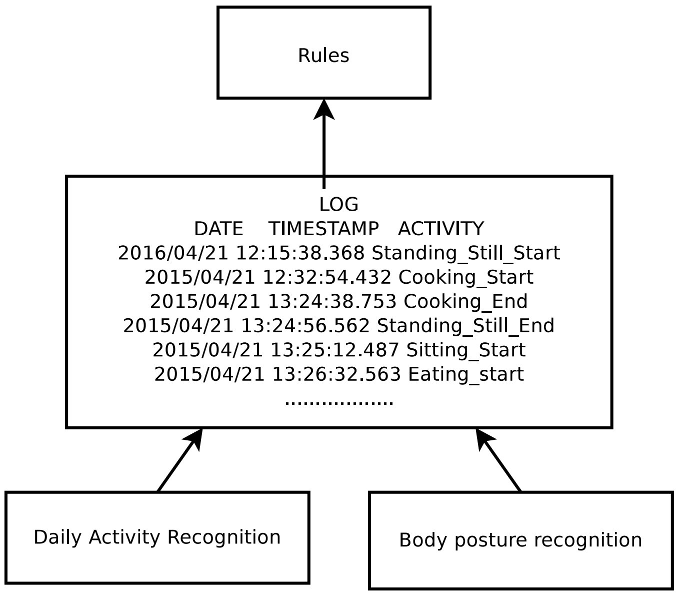
Each task is carried out in a different layer of the Rainbow architecture (Figure 3.5). In particular, the data acquisition task is assigned to the Virtual Objects layer, the feature extraction is carried out by *extractor agents*. The activity discovery task is assigned to the Rainbow cloud part. Indeed, it typically involves complex data mining algorithms so it can suitably exploit the large computational resources of the cloud technologies. The activity discovery task produces as output new *classifier agents* which are dynamically deployed on the agent servers. These agents run in the computational nodes physically placed in the home environment so they can classify the activities fast enough to fulfil the real time requirements.

3.4.2 Ambient assisted living

In the following, the setting scenario will be detailed, inspired by the CASAS project [21]. The setting concerns a single apartment, which is instrumented with various proximity sensors. Each sensor provides boolean information (on/off) on the presence of someone in its sensed area. The apartment comprises five rooms: a kitchen, a dining room, a living room, a bedroom and a bathroom. In addition, the person who lives in the apartment always wears wearable sensors consisting of three axis accelerometers. Each room hosts a fixed Rainbow node to which the environmental sensors are connected. A mobile Rainbow node, e.g., a smart-phone always carried by the person, can receive all the data of the wearable sensors.

Figure 3.6 shows the main logical flow involved in this application scenario. At the bottom, two activity recognition processes are designed to be continuously executed: the first uses the environmental sensors' data (i.e. the proximity sensors) to recognize daily activities, while the second uses the wearable sensors to recognize the body postures of the apartment inhabitant. These two tasks will feed a single log, composed of triplets ⟨ DATE, TIMESTAMP, ACTIVITY ⟩, as shown in Figure 3.6. In this log, an activity could be both a daily life activity, such as cooking or watching tv, or a body posture, such as standing still or sitting.

A third component has been designed to use this log in order to detect anomalies so as to produce alerts through the exploitation of a set of high level rules.



```
LOG
DATE    TIMESTAMP   ACTIVITY
2016/04/21 12:15:38.368 Standing_Still_Start
2015/04/21 12:32:54.432 Cooking_Start
2015/04/21 13:24:38.753 Cooking_End
2015/04/21 13:24:56.562 Standing_Still_End
2015/04/21 13:25:12.487 Sitting_Start
2015/04/21 13:26:32.563 Eating_start
.................
```

Fig. 3.6. Case study main flow

Mining on environmental sensors data

The use of the algorithms presented in [39] is proposed to recognize daily activities in the presented setting. A general schema of activity recognition has been given in Figure 3.5. The data generated from ambient sensors produce events represented by a quadruple:

$$\{DATE, TIMESTAMP, SENSORID, STATUS\}$$

which represents a status change (*on/off*) of the sensor identified by *SENSORID*, at the time *DATE,TIMESTAMP*.

The raw data are then processed by collecting consecutive sensor firings segmented per room. The result of this operation is a list of occupancy episodes, in the form of (*roomID, startTime, duration, usedSensors*), where *roomID* uniquely identifies a room, *startTime* is the beginning time of the occupancy episode, *duration* is how long the episode lasts, *usedSensors* is the set of the sensors that was used during a particular occupancy episode.

These episodes are the features that feed the activity discovery and recognition tasks.

The activity discovery task uses the frequent itemset mining Apriori algorithm for identifying the relevant occupancy episodes which are then clustered [39]. The output of this task is a set of clusters, which can be updated over time. Each cluster is represented by a tuple (*UsedSensors, MeanStartTime, MeanDuration*) which needs to be manually labelled.

The set of labelled clusters constitutes the model for the activity recognition task, where each new occupancy episode is assigned to the most similar cluster, and will be used to feed the previously introduced activity log (see Figure 3.6).

Mining on wearable sensors data

In order to achieve the previously described goal, wearable sensors are designed to send periodically to their Mobile Rainbow Node a set of features directly calculated on the accelerometer sensor nodes [31]. These features will be selected in the activity discovery/training phase through the SFFS [65] algorithm as already proposed in [31]. The chosen features act as input to the activity recognition system, designed in a Rainbow Agent, which will run a classifier recognizing the body postures and movements defined (i.e. *standing still, sitting, laying, walking, falling down*). Among the classification algorithms available in the literature, a K-Nearest Neighbor (KNN)-based classifier [23] will be used. The output of the KNN will be calculated every second to have in the Rainbow Agent a real-time update. Instead, the activity log introduced above will be fed everytime an update in the body posture is ready.

Rules

The activity log, fed by both the environmental and the wearable sensors activity recognition tasks, represents the knowledge base of an expert system (composed of a set of collaborating Cloud Agents) which is able to produce alerts on the basis of a set of high level rules.

In the following some simple inference rule examples are listed:

Condition One	**Condition Two**	**Alert**
Cooking	Lying	faint
Shower	Lying	faint
Night	Too long Standing Still	illness
Toileting	Too long sitting	illness
Any Activity	Falling down	accidental fall or faint

Whenever a Cloud Agent detects that one of these rules is satisfied, it will generate an alarm to notify an anomaly event. This feature of the system is particularly important in a Smart Home environment where elderly assistance is performed. A wider set of alarms can be defined at cloud level based on the specific habits of people living in a smart home.

3.5 Smart Street Cosenza

In the context of the Res-Novae[3] project, a smart street environment has been designed and implemented in the city of Cosenza (Italy), exploiting the features of the Rainbow platform. This smart environment is called *Smart Street Cosenza*. The goal is to furnish the city with an IT infrastructure which provides services to the citizens, and which can be further extended over time, either covering new areas or providing new features and functionality. The target areas for a first deployment are shown in Figure 3.5 and are located in the centre of the city. Specifically, the infrastructure covers: (A) a Bus Station, (B) a square, (C) the main commercial street of the city, (D) part of one of the main driveways and (E) the area around a commercial centre.

Over the chosen areas, a set of thirteen computational nodes (Figure 3.8(a)) and seventy wireless sensors nodes (Figure 3.8(b))are deployed. Each computational node consists of a Raspberry Pi mod.2 board, is mains powered, and hosts the Rainbow middleware. Each sensor node has a battery embedded and is powered by a solar panel. Sensor nodes can be of two types: *type A* nodes host noise, temperature, relative humidity and luminosity sensors, and *type B* nodes host air quality sensors, measuring the concentration of CO, CO_2, NO and O_3. All the sensor and computation nodes are connected using a wifi network infrastructure, also deployed in the same areas.

Currently, the described CPS infrastructure hosts an application devoted to real-time monitoring the status of the areas involved. Raw measures are collected by the computational nodes, then filtered and aggregated and finally sent to a remote server that stores the data in a DBMS. A web portal is available showing geo-referenced sensor measures as well as aggregated information indexes such as Simmer Summer Index (SSI), Humidex and Air Quality Index (AQI). This information is available to the citizens, who can use it to figure out how much the different areas are comfortable and healthy, to the city

[3] The RES-NOVAE - “Buildings, roads, networks, new virtuous targets for the Environment and Energy” project is funded by the Italian Government (PON 04a2_E)

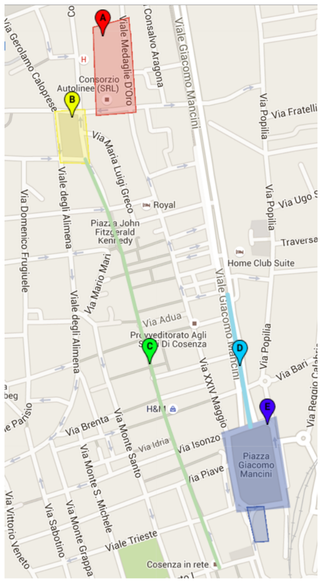


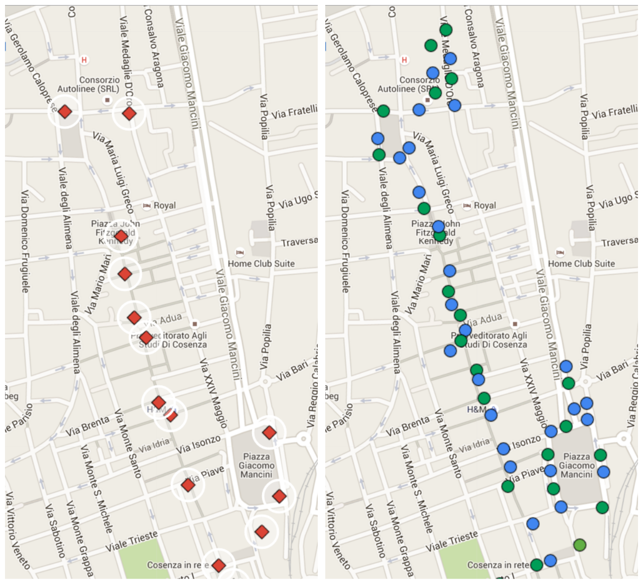
Fig. 3.7. Smart Street Cosenza: target areas.

administrator, who can monitor critical climatic or pollution events, and the research community, who can use it for further investigations and analysis.

3.6 Conclusions

In this chapter, solutions are shown for different applications scenarios in the context of Building Automation, Ambient Assisted Living and Smart Cities, so as to highlight how the features and the capabilities of the proposed framework can be exploited for the design and the implementation of cyber-physical systems.

In the following chapter, a novel CPS for managing an urban drainage network is detailed and discussed. The proposed CPS is capable to provide an optimized behaviour to the drainage network, which is capable of reducing effectively both urban flooding and combined sewer overflow phenomena, as well detailed in the experimental section.



(a) (b)

Fig. 3.8. Smart Street Cosenza. (a) Computational Nodes; (b) Sensors nodes. Green locations host one *type A* node and one *type B* node, blue locations hosts only a *type A* node.

4

A distributed real-time control system for mitigating CSO and flooding in urban drainage systems[1]

4.1 Introduction

Climate change and exponential reduction of pervious surfaces in urban catchments have led to an increase in the frequency and magnitude of two undesired phenomena which negatively affect human life, economic assets and the environment: (i) local *flooding* and (ii) *combined sewer overflows* (CSOs)[59][62].

Urban flooding occurs when the *urban drainage system* (UDS) becomes drastically overloaded during extreme rainy events, causing untreated combined sewage and storm water to back up into basements and to overflow from manholes onto surface streets. This phenomenon is generally worsened by obstructions in conduits and manholes due to an infrequent maintenance.

CSO [60][63] takes place when the *wastewater treatment plant* (WWTP) is not able to treat the wastewater delivered by the UDS. Specifically, the sewage and wet weather flows that exceed the WWTP treatment capacity. Specifically, the sewage and wet weather flows are conveyed through the UDS to the WWTP until the maximum treatment capacity is reached. The exceedance of the water flows are discharged directly into the receiving water bodies, such as rivers or lakes, without receiving any treatment. As a consequence, CSO is one of the major contributors to water pollution experienced in rivers, lakes etc.

In accordance with the European Water Framework Directive (WFD) and the EU Flood Directive (2007/60/EC), measures need to be adopted to manage stormwater volumes efficiently by reducing CSOs and preventing urban

[1] - Giordano, A., Spezzano, G., **Vinci, A.**, Garofalo, G., Piro, P. (2014). A Cyber-Physical System for Distributed Real-Time Control of Urban Drainage Networks in Smart Cities. In Internet and Distributed Computing Systems (pp. 87-98). Springer International Publishing.
- Garofalo G., Giordano A., Piro P., Spezzano G., **Vinci A.** A distributed real-time control system for mitigating CSO and flooding in urban drainage systems. Submitted to Environmental Modelling and Software Journal in 2015

areas from sewer flooding. For this purpose, offline storage facilities, which temporarily accumulate stormwater volumes, are widely used, even though they are often overly expensive due to the high construction and maintenance costs. In contrast, approaches aiming at temporarily accumulating stormwater volumes directly in the existing UDSs have been also developed thus avoiding large investments [78][10][12]. These approaches are supported by the fact that the UDSs are typically designed by taking into account a set of safety factors. In particular, conduits are intentionally designed to be larger than required in the case of typical network working conditions. Basically, the UDS is managed by a *real-time control* (RTC) system which requires the network to be embedded with sensors and actuators permitting the network to be real-time monitored and regulated so as to adapt to the different rainfall events [27][2]. In particular, physical layer information is collected by sensors and sent to the control component which elaborates an optimized actuation strategy so as to ensure the desired UDS working behaviour.

Many works in the literature focus on RTC based on a centralized approach. For instance, in [64] a centralized optimal control system has been implemented which decreased CSO volumes at four overflow sites by more than 85% for seven rainfall events. However, according to [64] most current RTC implementations are limited to local reactive or supervisory control. In [36] the centralized control is modelled as an optimization problem and is able to obtain a significant improvement in reducing CSOs in comparison with a conventional base case scenario. In [78] the centralized RTC is applied to some real cases with a global overflow volume reduction per site varying from 40% (Sute) to 100% (Dijon), and a global reduction per event close to 70% on average.

In those studies, as the control system is fully centralized, several issues arise due to the large amount of data to be read, managed and processed. Using a typical centralized monolithic approach, all sensory data are sent to a central unit that elaborates a suitable strategy based on a comprehensive network model thus producing commands for the actuation part. This approach tends to be very reliable, but it has some drawbacks: (i) it requires a complex mathematical model of the network (ii) all physical parts (sensors and actuators) need to be connected with and reachable by the remote central unit and (iii) communication times can be too long to capture dynamic changes on the physical part correctly [12].

Conversely, in this chapter a CPS implementing a distributed real-time control (DRTC) system is adopted, exploiting the Rainbow framework presented in Chapter 2, and specifically a gossip-based algorithm. The UDS is equipped with electronically moveable gates and a set of water level sensors spread across the network. All the gates are locally controlled by *Proportional Integrative Derivative* (PID) controllers which are globally orchestrated by the mentioned gossip-based algorithm thus achieving an optimal hydrodynamic behaviour in terms of CSO and flooding reduction. The case study is the UDS of the city of Cosenza (Italy), which is modelled by using the SWMM simulation software. The SWMM model permits an accurate simulation of the hydrological and hydraulic behaviour of the UDS during both dry and wet weather conditions. SWMM simulation software has been customized in order to allow it to be integrated with an external real-time control module. Experiments, conducted using a set of 15 extreme rainfall events recorded in the years 2011-2015, show an substantial reduction of both CSO and flooding when the proposed approach is exploited.

4.2 Methods

4.2.1 The structure of an urban drainage system

An *urban drainage system* (UDS) aims at collecting and delivering the combined wastewater (sewage and wet weather flows), coming from the urban catchment, to a wastewater treatment plant (WWTP). A UDS physically consists of a series of junctions, conduits, weirs and storage units [53], which, in most cases, forms a dendritic structure. Indeed, the network typically follows a tree structure, in which a series of sub-networks (branches), with smaller pipes, converges into a main network (trunk), consisting of larger collector conduits. Sequentially, each sub-network (branch) in turn follows a tree distribution. Finally, all conduits are linked to one final collector pipe, which ends with the *outfall* of the network, delivering the wastewater to the WWTP. Overflow structures are used to direct the combined wastewater volumes which exceed the capacity of the WWTP directly into receiving water bodies. When the UDS follows, instead, a fully looped structure, it can be converted into a tree network by a loop-by-loop cutting algorithm for modelling purposes [80].

4.2.2 Drainage network instrumentation

In order to achieve the proposed goals, the network requires the following equipment: (i) *sensors*, one water level sensor per conduit and a flow sensor on the outfall; (ii) *computational nodes*, which can host and execute the distributed control algorithm [2] ; (iii) down-hinged *moveable gates*, which can be real-time regulated electronically.

The computational nodes dynamically regulate the gates according to the information acquired by the sensors in the neighbour areas. Each computational node has a partial view of the network as it can read only from the sensors, and actuate only on the gates, which are located in its spatial neighbourhood, i.e. the sensors/gates it can physically reach. In addition, each computational node communicates only with its neighbour peer nodes. The

[2] the computational nodes can be custom-built devices or single-board computers such as *Raspberry pi* or *Beagleboard* which can be effectively distributed inside the network because they have low energy consumption and small size.

computational nodes are distributed throughout the network in order to cover all the points of interest, i.e. the sensors and the gates. The nodes read data from the sensors and collectively process the acquired information in order to trigger suitable actuations on the gates. The collective computation of the network of nodes supplies the gates with an "intelligent" behaviour.

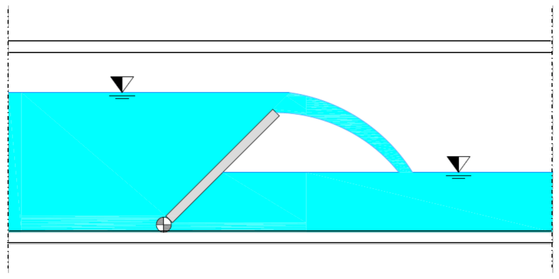


Fig. 4.1. The down-hinged movable gate

The electronically moveable gates are made up of mobile plates rotating around a horizontal hinge placed on the bottom of the conduit[3], as shown in Figure 4.1. The gate is completely closed when the plate rotates in a perpendicular position with respect to the flow direction. Conversely, the gate is fully open when the plate is parallel to the flow. When the gate is closed, the opening area is null and no flow rate is delivered from the node. An intermediate position of the gate corresponds to a partial opening degree. The gates allow utilization of the full storage capacity of the conduits by accumulating the excess stormwater volume in the less overloaded parts of the system [17].

4.2.3 Distributed control of urban drainage network

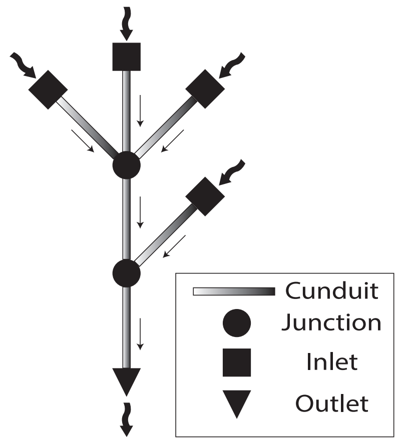
This section details our approach for reducing: (i) flooding phenomenon and (ii) CSO problem. The flooding issue is tackled by balancing the water level throughout the conduits of the network so as to reduce flooding in the most overloaded conduits at any given moment. In other words, balancing the water level means that the underloaded conduits are triggered to store additional water in order to help the overloaded conduits when they are about to overflow. The CSO problem is addressed by extending this strategy, taking into account also the flow at the outfall as will be better clarified in section 4.2.3.

Based upon the description reported in section 4.2.1, a drainage network can be formally seen as a graph (V, E) of nodes $v \in V$ connected by edges

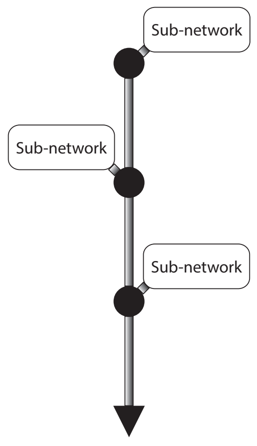
[3] The top-hinged gate was avoided to prevent high speed flow under the plate, which may re-suspend the sediment material at the bottom of the sewer pipes.

$e \in E$. More specifically V comprises *Junctions* $j \in J$, *Inlets* $l \in L$ and *Outlets* $o \in O$. E is made up of *Conduits* $c \in C$. Junctions are just intersection points for conduits. Inlets are nodes where runoff enters into the system. Outlets are the points of the network where water is discharged into a river, lake, reservoir and so forth. Conduits are pipes of different cross-sectional shapes where the water flows [30].

In the following we introduce an enhanced version of the model used in our approach for achieving the proposed goal of balancing the conduits' water level. This refined model is based on some other features which are inherent to drainage networks. Firstly, in a typical urban scenario, the whole drainage watershed can be broken down into several, not connected, networks in which each network comprises only one outfall. In addition, each network is likely to be modelled by a tree structure. Indeed, we can see a network as a main channel and a set of sub-networks which are connected, through junctions, to different points of the main channel. Each of these sub-networks can be defined recursively in the same way. Finally, it can be assumed that inlets are located in the "leaves" of the tree. A very simple drainage network is outlined in Figure 4.2(a).



(a) A simple drainage network scheme



(b) Recursive definition of drainage network

Fig. 4.2. Drainage network structure

On the basis of the previous considerations we formally define a drainage network as follows. Firstly, we define *Most Simple Drainage Network* ($MSDN = (c, l)$) as a network only made up of one conduit c ending with inlet node l. Then, we define a generic drainage network DN as either just an $MSDN$ or a couple (M, S) where M represents the main channel and S a set of DNs defined in the same way. A main channel M is an ordered set of conduits in which each conduit is linked with the next one through a junction. The last conduit optionally ends with an outlet node. Figure 4.2(b) shows graphically this recursive definition. Figure 4.3(a) and 4.3(b) show respectively: a case of a realistic network and the sub-networks, surrounded by dashed lines, as results from the above definition. We also define *Degree* of a DN as a function $Degree(DN)$ defined as:

$$Degree(DN) = \begin{cases} 0, & \text{if } DN \text{ is a } MSDN \\ 1 + max_{s \in S}(Degree(s)), & \text{with } DN = (M, S) \end{cases}$$

The recursive definition of a drainage network permits us to extend an optimization strategy conceived for simple networks (such as the one shown in Figure 4.2(a)) also for complex/general scenarios as will be better specified in the following. Basically, given the physical complex drainage network, we firstly generate a set of simpler logical networks, afterwards, we execute the "simple network" optimization strategy on each of these generated network and then we obtain the optimal behaviour to be actuated in the original network. For this purpose we firstly define, for a given network DN, the set $nets(DN)$ as follows:

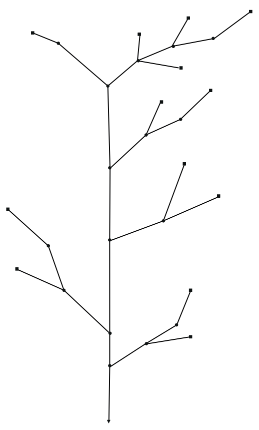
$$nets(DN) = \begin{cases} \emptyset, & \text{if } DN \text{ is a } MSDN \\ \{DN\} \cup \bigcup_{s_i \in S} nets(s_i), & \text{if } DN = (M, S) \end{cases}$$

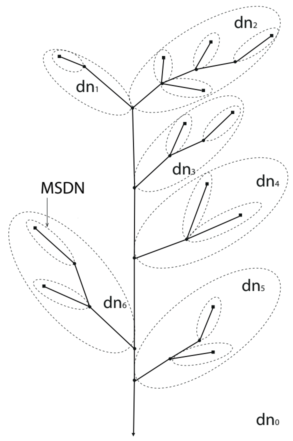
Then, starting from $nets(DN) = \{dn_i\}$ with $dn_i = (M_i, S_i)$, we generate the set $GN = \{gn_i\}$, in which each $gn_i = (M'_i, S'_i)$ is given by the following formulas:

$$M'_i = \begin{cases} M_i, & \text{if } i = 0 \\ M_i \cup o_i, & \text{elsewhere} \end{cases}$$

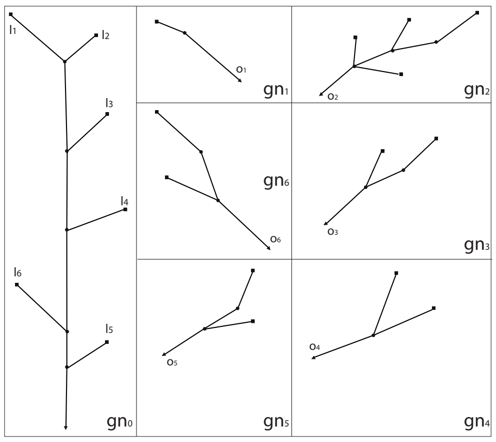
$$S'_i = \{msdn_k = (c_k, l_k) : \forall dn_k \in S_i\}$$

The intuitive idea concerns replacing the set of sub-networks $S_i = dn_k$ with a new set S'_i made up of only MSDNs. In particular, each $dn_k \in S_i$ corresponds to an $msdn_k = (c_k, l_k) \in S'_i$. M'_i is M_i plus o_i outlet node except for $i = 0$, since the top level sub-network already hosts an outlet node. As a result, starting from a complex network we generate a set of networks with 1 degree. Our approach relies on the assumption that the optimization algorithm, tailored for one degree networks, when executed simultaneously on all the generated networks permits us to find the global optimal behaviour for the original network. In other words, balancing the water level in all the generated networks means balancing the water level in the original network. This assumption holds only if the separated executions are kept consistently linked. Basically, it must be ensured that during the advancement of the execution the incoming flow of the generated inlet nodes l_k is set to be equal


(a) Network



(b) Sub-network



(c) Networks generated

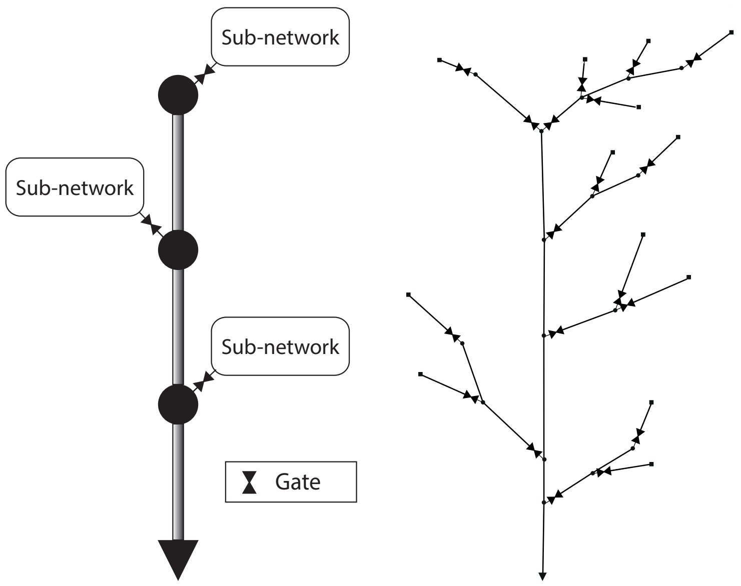
Fig. 4.3. Sub-networks in a realistic case

to the outcoming flow of the corresponding outlet nodes $o_{i=k}$ (i.e., the outlet nodes of the networks gn_k).

For instance, let's consider the network of Figure 4.3(a) and its set of generated networks $gn_{1..6}$, shown in Figure 4.3(c), where the balancing algorithm runs. The $i_{1..6}$ water levels, which correspond to the $o1..6$ water levels, are involved in both gn_0 balancing operation and $gn_{1..6}$ balancing operations at the same time. As a consequence, the balancing operations of all the generated networks, even if executed separately, results in a global balancing.

The optimal behaviour we want to actuate on the drainage network consists in balancing the water level throughout the conduits of the network. In the following we firstly describe how to place the gates inside the network, afterwards, we details the multi-agent algorithm which is executed on each generated network.

The gates are located at the points of the network where sub-networks are connected to the main channel. Figure 4.4(a) shows the logical places for inserting the gates, while Figure 4.4(b) shows the gates insertion in a case of a realistic network.



(a) Logical (b) Realistic case

Fig. 4.4. Gates positions

As said before, each computational node has a partial view of the network as it reads only from sensors located in its spatial neighbourhood, i.e. the sensors it can physically reach. In the same way, it can actuate only on its neighbour gates. On the basis of the previous considerations our proposal lies in using a *distributed agent-based* architecture [91]. The agent paradigm has several important characteristics:

Autonomy. Each agent is self-aware and has a self-behaviour. It perceives the environment, interacts with others and plans its execution autonomously.
Local views. No agent has a full global view of the whole environment but it behaves solely on the basis of local information.
Decentralization. There is no "master" agent controlling the others, but the system is made up of interacting "peer" agents.

Through these basic features, multi-agent systems make it possible to obtain complex *emergent* behaviours based on the interactions among agents that have a simple behaviour. Examples of emergent behaviour could refer to the properties of adaptivity, fault tolerance, self-reconfiguration, etcetera. In general, we could talk about *swarm-intelligence* [15] when an "intelligent" behaviour emerges from interactions among simple entities.

In the case of drainage networks, the property of fault tolerance is particularly useful since the system needs to continue to operate properly even if unexpected conditions occur, such as obstructions and blockages, which may reduce the hydraulic capacity of the system.

Our proposal considers one agent per gate. Each *gate-agent* runs on one of the computational nodes covering the specific gate, it can perceive the local water level and communicate with the neighbouring gate-agents in order to elaborate a proper actuation strategy for its gate. Another agent, called *outfall agent*, is logically associated with the outlet node, it behaves the same as other agents except for the actuation part, indeed, it is not associated with any gate. Figure 4.5 gives an intuitive idea of the agents' role in the generated networks. For each generated network, the algorithm consists in real-time balancing the water level perceived by the agents. Given that the conduits have different sizes, the water level is normalized with respect to the height of the conduit.

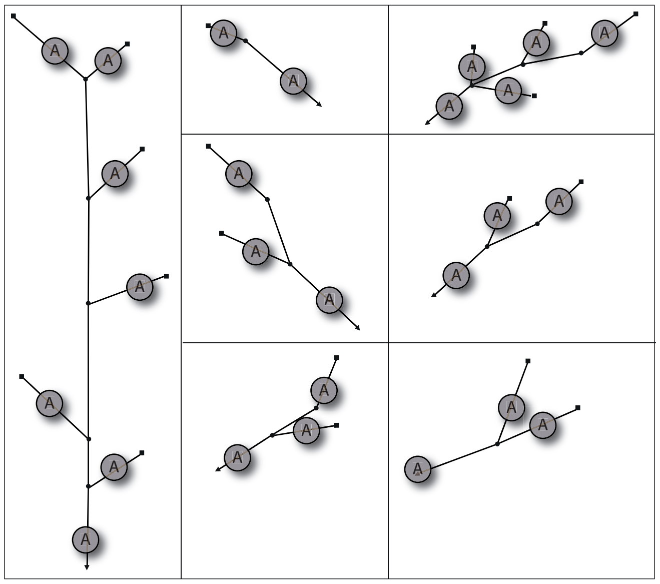


Fig. 4.5. Agents in generated networks

This water balancing is achieved by means of agents continuously executing two tasks:

Task 1. figuring out collectively the average of the water level in the network.
Task 2. each agent triggers its specific gate in order to bring the water level closer to that average.

Given that an agent has no global knowledge of the network, i.e. it does not know the water level in each point of the network, *Task 1* is accomplished by exploiting a gossip-based algorithm summarized in section 4.2.3. This kind of algorithm also supplies the previously mentioned fault-tolerance property. Anyway, even if we knew the optimal water level to set, we would not know how to tune the gate so as to achieve it. Indeed, the relationship between the actuation upon the gate (i.e. its opening degree) and the actual change of water level is determined by the structure of the whole network and the dynamics of the water flowing through the system, so it is very hard or even impossible to deduce a tractable mathematical model for it. For this reason *Task 2* is accomplished exploiting a *PID* controller as explained in section 4.2.3.

Task 1: Gossip-based aggregation

In a Gossip-Based Algorithm [40] there are many nodes interconnected through a network. Each node possesses some numerical values and can exchange information only with a limited set of peer nodes (i.e. its neighbourhood). The goal of this kind of algorithm concerns estimating global aggregate values such as average, variance, maximum and so forth, despite only local communication being possible.

Basically, in the case of average aggregated value, each agent maintains its current measured value and its local average (initially set to the measured value). The algorithm consists in continuously exchanging local averages among neighbour nodes. Each time a node receives the average of a neighbour node, it updates its local average (just applying average operator). Values exchanges and local computations are done continuously for enough *step*s so as to ensure that each local average, computed at every node, converges to the actual global average (the algorithm convergence is proved in [40]).

In our approach, we run the gossip-based algorithm for computing the average of the degree level as measured by all the agents of a generated network. When the algorithm converges, the estimated average value is exploited by each gate-agent for tuning its gate so as to bring water levels closer to that average (*Task 2*). Afterwards, a new iteration begins, consisting in: measuring the new values of the water level, running the gossip-based algorithm until the convergence is reached again, using the computed value for the actuation part, and so on and so forth. Running this process continuously ensures the fault-tolerance property mentioned before because even if an unforeseen event dramatically changes some structural properties (as in the case of: obstructions, blockages, damages etc.) the algorithm is able to pass smoothly from the previous computed optimal value to the new one.

Task 2: tuning gates through PID controllers

Once an agent knows the global water level through the previously described "gossip-based aggregation", there remains the problem of appropriately tuning its gate so as to reach that "desired" level.

This issue is addressed using the well-known controlling technique called Proportional Integral and Derivative (PID) control [9] which, indeed, can be used when you do not know an exact mathematical model of the system you want to control.

A PID controller is a control loop feedback mechanism where an error value is computed as the difference between a measured output of a process and the desired value (setpoint) (see Figure 4.6). The controller tries to minimize this error, appropriately tuning the actuator device.

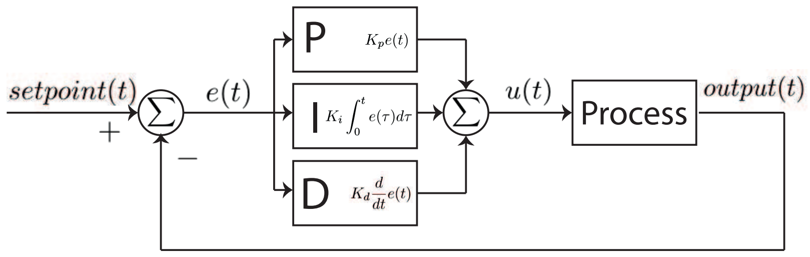


Fig. 4.6. PID controller

The setting of the actuator device is determined by three effects, suitably tuned by three parameters: the proportional one (P), the integral one (I) and the derivative one (D). P is determined by present error, i.e. the absolute error computed at the current evaluation. D measures the foreseen error, i.e. the expected error in the next step, computed deriving the error signal, while I represents the integral effect, a measures of the historical behaviour of the error signal. The following equation defines a general time-continuous PID controller.

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau \ + K_d \frac{d}{dt} e(t)$$

Where $e(t) = setpoint(t) - output(t)$; $u(t)$ is the controller output at time t, i.e. the actuation signal; K_p, K_i, K_d are three constants which refer respectively to the proportional, integral and derivative effects. These parameters are tuned adopting the well-known ZieglerNichols method [96].

In the case of this study, each gate of the drainage network is controlled by a PID implemented by the gate-agent. $u(t)$ represents the degree of opening of a gate, $output(t)$ is the actual water level of its related conduit and $setpoint(t)$ is the "desired" water level, i.e., the average computed by *Task 1*.

CSO Reduction

The approach described so far focuses only on reducing the flooding phenomenon. Anyway, the approach can be slightly enhanced so as to address the CSO problem as well. As described in the introductory section, CSO occurs when the water flow on the outfall node exceeds the capacity of the treatment plant. For this reason the CSO issue can be tackled by controlling the water flow on the outfall node. Basically, we introduce an additional gate placed at the outfall and the outfall-agent is extended as to control the flow of the new inserted gate. Keeping the flow level under a certain desired value results unavoidably in increasing the water level which can cause local flooding. Fortunately, the algorithm described before is able to ensure a balanced water level throughout the whole network. In other words, when the water level on the outfall increases due to the flow control, the rest of the gates are triggered to store more water thus helping the outfall to decrease properly its local water level.

The latter consideration suggests that increasing the number of gates can results in further improving the CSO reduction. The experimental section 4.3 will show that adding more gates can effectively improve the CSO reduction for the selected rainfall events.

4.2.4 SWMM

In this work the drainage network is simulated using the software StormWater Management Model (SWMM) provided by EPA [72], which is an open-source computer model for simulation of hydrodynamic water and pollutant transport in sewer systems. SWMM relies on a time stepped dynamic model, where the dynamic simulation is performed by numerically solving flow routing equations (dynamic wave) based on a fixed time step. The SWMM model also allows the modelling of water quality constituents, dry-weather pollutant build-up over different land uses in the contributing catchment and pollutant washoff from specific land uses during storm events. Figure 4.7 shows a snapshot of the software in execution.

The input data and parameters required for the SWMM model simulation of drainage flow hydrographs include physiographic characteristics of the catchment (e.g. the area and slope), physical characteristics of the sewer pipes (the diameter, length, slope and material) and the hydrological/hydraulic parameters such as the width of the subcatchments (i.e. the overland flow width).

The routing flow model used is the dynamic wave flow routing to predict non-steady flows through a general network of open channels, closed conduits and weirs. In contrast to simpler routing methods, this procedure can model such phenomena as backwater effects, flow reversals, pressurized flow, and entrance/exit energy losses [72]. The governing equations are the Saint Venant equations that can be expressed in the following form for flow along an individual conduit:

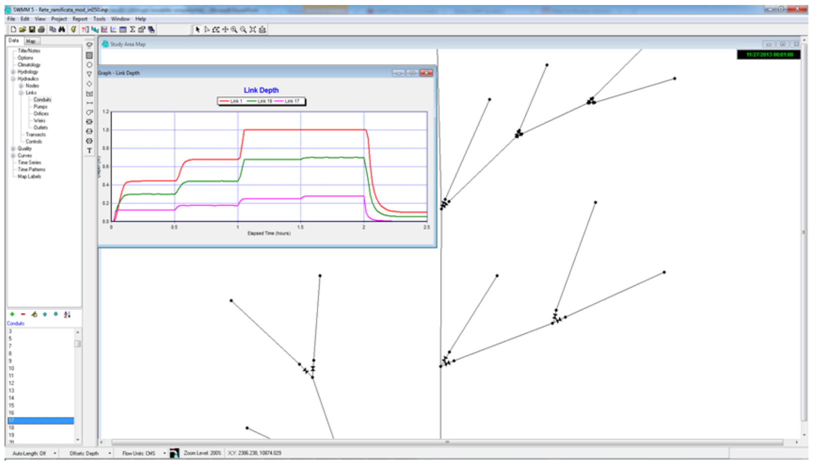


Fig. 4.7. Snapshot of the SWMM software

$$\frac{\partial A}{\partial t} + \frac{\partial Q}{\partial x} = 0 \qquad \text{(Continuity)}$$

$$\frac{\partial Q}{\partial t} + \frac{\partial Q^2/2}{\partial x} + gA\frac{\partial H}{\partial x} + gAS_f + gAh_L = 0 \qquad \text{(Momentum)}$$

where x is distance along the conduit, t is time, A is cross-sectional area, Q is flow rate, H is the hydraulic head of water in the conduit (elevation head plus any possible pressure head), S_f is the friction slope (head loss per unit length), h_L is the local energy loss per unit length of conduit, and g is the acceleration of gravity. The governing equations in their differential form are numerically solved using the modified *Euler method* (equivalent to a *2nd order Runge-Kutta method*).

SWMM Customization

Although SWMM permits some trivial real-time control of the parameters of the network (by defining some simple rules), for the purposes of the work, SWMM has been customized for permitting it to communicate in real time with a separate Java controller which implements the algorithm described in Section 4.2.3.

Figure 4.8 shows the architecture of the integration. SWMM has been enhanced by adding the *TCP connector* which is in charge of sending and receiving information to/from the external module during the advancement of a simulation.

More in detail, the connector enables the external module to select where the sensors are placed inside the network, i.e. it permits the physical variables which the external module is interested in to be chosen. Basically, at each step of SWMM simulation the values of the selected variables are collected

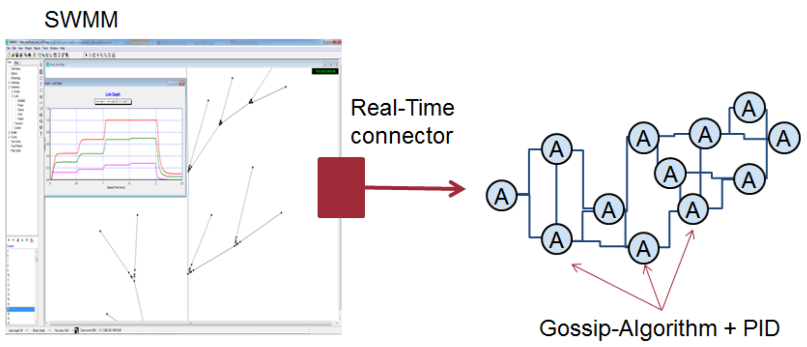


Fig. 4.8. SWMM customization

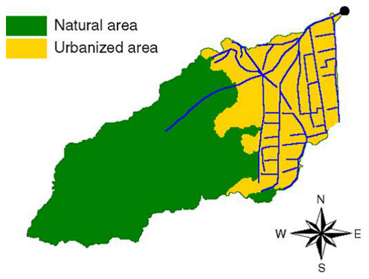
and sent toward the external module which, in turn, replies to SWMM with all the actuation, i.e. the opening degree of the gates, as computed by the multi-agent algorithm.

4.3 Results and discussion

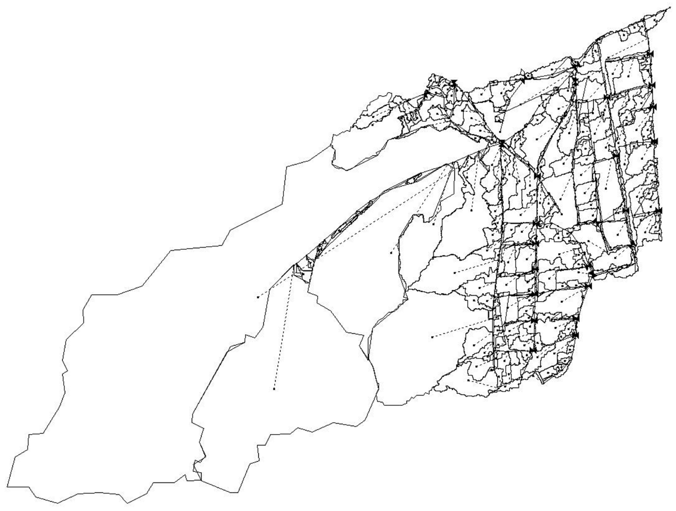
4.3.1 The case study: the sewer system of the city of Cosenza

In this study, the proposed approach is applied to an urban catchment in Cosenza, Italy, shown in Figure 4.9, which is densely populated (approximately 50,000 residents). The total surface of the catchment is 414 hectares, out of which 202 hectares are pervious, covered by vegetation, while the rest consists of paved areas. Buildings for residential housing and minor commercial and artisan enterprises are present on the catchment. Further details on the physical characteristics of the basin and the drainage system are reported in other publications [59]. Sewage and wet weather flows from the urban watershed, which do not exceed treatment capacity, are directly sent to the wastewater treatment plant (WWTP). The exceedance of wet weather flows is directly discharged, without receiving any treatment, into the Crati River. The dry weather flow of the urban catchment is approximately 0.23 m^3/s. The CSO occurs when the flow rate from the outfall of the drainage system is higher than 0.7 m^3/s as reported in [61][60] where other information about the untreated CSO features can be found.

The model used in this study was previously calibrated on the basis of several measurement campaigns [63]. Calibration parameters were: surface roughness of the impervious (N-Imperv) and pervious (N-Perv) catchment surfaces, and the depths of surface depressions on impervious (Dstore-Imperv) and pervious (Dstore-Perv) areas. The urban drainage network modelled in SWMM, as shown in Figure 1b, consists of 324 conduits with different shapes and sizes, i.e., (i) circular and egg-shaped pipes with diameters varying from



(a) Urban watershed in the city of Cosenza



(b) SWMM model of Urban drainage network in the city of Cosenza

Fig. 4.9. Urban drainage system of Cosenza, Italy

0.3 to 1.5 m and (ii) policentric pipes with a maximum depth of 3.20 m. The slope of the pipes varies from 0.5% to 6%. The connection points between urban watershed surfaces (roads and street paving) and conduits are represented by catch basins which collect and deliver surface runoff into the sewer system. There are in total 326 nodes which represent the catch basins. The total number of subcatchments is 296.

The moveable gate is modelled as a transverse weir with the opening area equal to the conduit section area.

4.3.2 Experimental setup

In this study, different scenarios have been analyzed to evaluate the performance of the DRTC as a function of the number of the moveable gates, linked to actuators, and placed across the system. The scenario without DRTC, which corresponds to the actual UDS, is called *scenario 0*. The other scenarios are controlled by the DRTC and differ according to the number of secondary pipes equipped with moveable gates. The number of moveable gates is 91 for *scenario 1*, 107 for *scenario 2*, 214 for *scenario 3* and 322 for *scenario 4*. Scenario 1 corresponds to the gates' placement described in section 4.2.3. In scenarios 2-4 an increasing number of gates is exploited which should result in a further CSO reduction as mentioned in section 4.2.3. The response of the UDS for all these scenarios is modelled for 15 independent rainfall events recorded in the weather station in Cosenza (Italy) during the years 2010-2015 with a time resolution of 1 minute. The hydrological characteristics of the rainfall events selected are reported in table 4.1. Rainfall events were considered independent if they are separated by an inter-event time of 6 hours [3].

Event	**Date**	**htot (mm)**	**Iav (mm/h)**	**Qmax (m3/s)**	**Qav (m3/s)**	**CSO Vol (m3)**	**LF Vol (m3)**
1	22-Jan-11	31	1.29	2.06	0.64	23530	348
2	08-Oct-11	48.6	2.022	6.3	1.07	66402	933
3	05-Dec-11	54	2.25	4.85	1.32	53351	1007
4	01-Dec-13	31.4	2.76522	4.74	0.713	26790	366
5	23-Nov-13	46.6	3.33912	5.58	1.081	50216	795
6	21-Jan-14	31.8	1.32	2.8	0.69	17092	362
7	01-Feb-14	14.5	1.74228	2.8	0.83	11591	345
8	01-Feb-14	16.2	1.01298	2.98	0.6	8167	
9	24-Mar-14	38.6	1.60836	4.27	0.87	33309	563
10	30-Jan-15	42.8	1.78332	5.45	1	37039	680
11	31-Jan-15	4.6	1.53906	2.89	0.78	18478	565
12	31-Jan-15	35.38	2.24742	3.2	1.173	9442	
13	01-Feb-15	20.8	1.29738	3.57	0.64	14692	281
14	01-Feb-15	8.32	1.01568	2.98	0.49	3489	
15	22-Feb-15	18.91	0.78822	1.99	0.405	6080	95

htot	Total rainfall depth
Iav	Average rainfall intensity
Qmax	Maximum flow rate
Qav	Average flow rate
CSO Vol	Combined sewer overflow volume
LF Vol	Local flooding volume

Table 4.1. Hydrological and hydraulic characteristics of the selected rainfall events.

4.3.3 Experimental results

In the following section, the findings obtained from the DRTC applied to the case study are described and discussed. Firstly, results in terms of CSO and Flooding reductions are presented for scenarios 1 and 4, using the set of 15 rainfall events; afterwards, the behaviour of the controlled network is analysed in details for 3 representative rainfall events taking into account all the scenarios (0-4). Finally, consideration about the overall performance of the developed approach are drawn in the light of the study's goals. The CSO and the local flooding volumes, as given by the SWMM simulator, are reported in table 4.1 for the 15 rainfall events selected and for scenario 0 (without DRTC) used in this section as the reference scenario. The CSO volumes, computed as the sum of overflow spilled into the river, are fairly significant, ranging from 3489 m^3 to 66402 m^3. The local flooding volumes, calculated as the sum of spilled volume for each manhole of the UDS, range from 95 to 1007 m^3. In table 4.2, the CSO reduction is computed for each rainfall event as the relative percentage difference between the CSO volume in the scenarios with DRTC and the reference scenario. In scenario 1, the CSO reduction varies from 2.7% to 83%, while in scenario 4 it varies from 13% to 99%, according to the rainfall events. The CSO drop is consistently higher in the scenario 4, demonstrating the beneficial effect provided by using a larger number of moveable gates.

Event	Date	CSO				Flooding			
		1		4		1		4	
		(m3)	%	(m3)	%	(m3)	%	(m3)	%
1	22-Jan-11	16742	28.9	10965	53	0	100	320	8.1
2	08-Oct-11	64614	2.7	58030	13	0	100	808	13.4
3	05-Dec-11	43600	18.3	34244	36	0	100	971	3.6
4	01-Dec-13	20968	21.7	15582	42	0	100	333	9.0
5	23-Nov-13	39950	20.4	31785	37	0	100	7	99.1
6	21-Jan-14	8167	52.2	2617	85	0	100	356	1.7
7	01-Feb-14	7865	32.1	4718	59	0	100	324	6.1
8	01-Feb-14	5776	29.3	3596	56				
9	24-Mar-14	25442	23.6	19497	41	0	100	518	8.0
10	30-Jan-15	30311	18.2	23962	35	0	100	664	2.4
11	31-Jan-15	13060	29.3	9223	50	0	100	532	5.8
12	31-Jan-15	8086	14.4	6629	30				
13	01-Feb-15	10423	29.1	6859	53	0	100	265	5.7
14	01-Feb-15	1269	63.6	338	90				
15	22-Feb-15	1035	83.0	44	99	0	100	90	5.3

Table 4.2. CSO and flooding reduction for scenarios 1 and 4

Regarding local flooding, the results are reported in Table 4.2, as well. The local flooding reduction is computed as the relative percent difference between the total flooding volumes from the scenarios with DRTC and the reference scenario. The DRTC in scenario 1 utterly prevents the UDS from local flooding, through the temporary stormwater detention provided in the less overloaded conduits. However, in the scenario 4 the risk of flooding is

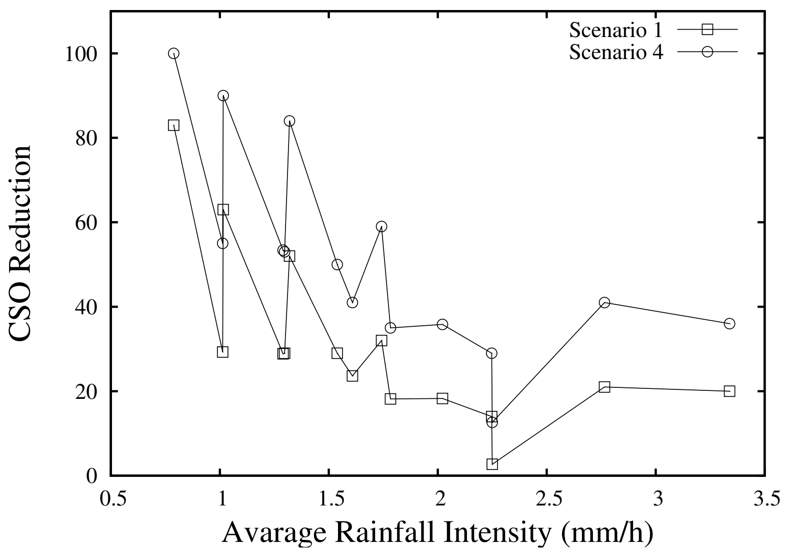


Fig. 4.10. CSO reduction vs. rainfall event average intensity.

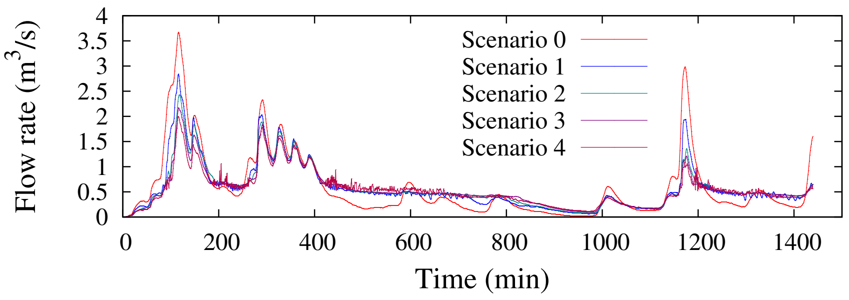
solely mitigated, with reductions, which vary from 2.4% to 13.4% for all the events, except for the event 23 November 2013, where a drop of 100% is obtained. These results are due to the fact that, when the moveable gates are allowed to close, the upstream conduits can be prone to surcharge. Generally, the DRTC is able to control these situations by maintaining the conduits, upstream of the moveable gates, under the surcharging conditions. However, when the number of moveable gates increases, the likelihood of overloaded conduits increases as well, resulting in spills and local overflows from the manholes. This demonstrates that it is necessary to define an optimal number of gates to control concurrently the risk of both CSO and local flooding, taking into account that the two phenomena may not be complementary. In other words, an optimal number of gates needs to be selected as a tradeoff between maximizing the wastewater storage, which, on the one hand, allows reduction of CSO, and minimization of upstream surcharging conduits, which, on the other hand, significantly can affect local flooding.

To investigate the effect of rainfall characteristics on the DRTC performance, the CSO reduction values for scenarios 1 and 4 are reported in Figure 4.10 as a function of the average rainfall intensity. As can be observed, the CSO reduction diminishes, as the average rainfall intensity increases. Specifically, for rainfall events with an average intensity less than 1.6 mm/h, the CSO reduction ranges from 100% to 40% for scenario 4 and from 85% to 25% for scenario 1. For events with an average intensity higher than 1.6 mm/h, the CSO reduction is around 30% on average, for scenario 4 and 10% on average, for scenario 4. These percentages correspond to the maximum storage capacity available in the different scenarios for detaining a portion of stormwater, in order to reduce CSOs, during rainy events. The trend observed is justified by the fact that very intense rainfall events such as, those on 11th October 2011 and 31st January 2015, with an average intensity of 2.2 and 2.24 mm/h, respectively, produce a large amount of runoff volume, which tends to utilize, almost completely, the whole capacity of the UDS. In such conditions, therefore, the likelihood of using the UDS as a temporary storage drastically decreases. Although the DRTC performance is affected by the hydrological characteristics of the events, the relative performance, computed as the difference between scenario 1 and scenario 4, turns out to be independent of rainfall characteristics. Indeed, the difference in the results between scenario 1 and scenario 4 are consistently around 20% and 30%, regardless of the average intensity.

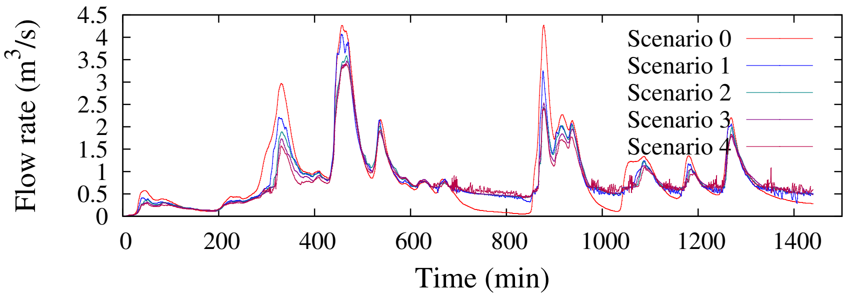
In Figure 4.11, the time distribution of flow rate, discharged in the collector pipe, modelled with SWMM, is plotted for the five different scenarios investigated and three selected events: 24th March 2014, 30th January 2015, 1st February 2015[4]. The hydrographs, obtained for the reference scenario, exhibit a peak flow rate of 4.27, 5.45 and 3.57 m^3/s, respectively for the three events. Instead, the hydrographs show lower peak flow rates as the number of moveable gates increases. Indeed, the higher the number of moveable gates, the higher the effect of flow equalization provided by the DRTC. This is because the presence of moveable gates controlled by the DRTC enhances two major factors: (i) the storage of stormwater volume in the less overloaded pipes, with the results of abating the peak flow rates; (ii) the gradual release of the stored volumes to the outfall, once the rain stops. For instance, in the hydrograph obtained on 1st February 2015, reported in Figure 4.11(a), it is possible to observe that, comparing scenario 0 and scenario 4, the peak flow rate is reduced by around 40%. Furthermore, once the first event is over, the flow rate in scenario 0 rapidly decreases, tending to the minimum value. The flow rate in scenario 4, instead, diminishes more gradually, remaining fairly higher than the previous one. This is because once the rainfall event stops, the UDS in scenario 4 releases the volume previously stored during the rainfall event. The UDS empties almost completely (except for the sewage contribution) at time 1000 min, when the second event begins. A similar behavior can be observed in Figures 4.11(b) and 4.11(c) for 24th March 2014 and 30th January 2015. Moreover, as described in Section 3.1, in the UDS of interest, the overflow occurs when flow rates are higher than 0.7 m^3/s. As can be observed from all the hydrographs reported in Figure 4.11, the DRTC tends, in fact, to maintain the flow rate in the collector pipe below this target value, with the beneficial consequence of reducing CSO.

Figures 4.12,4.13,4.14 show the reduction of local flooding and CSO as a function of the five scenarios for the events of 24th March 2014, 30th January

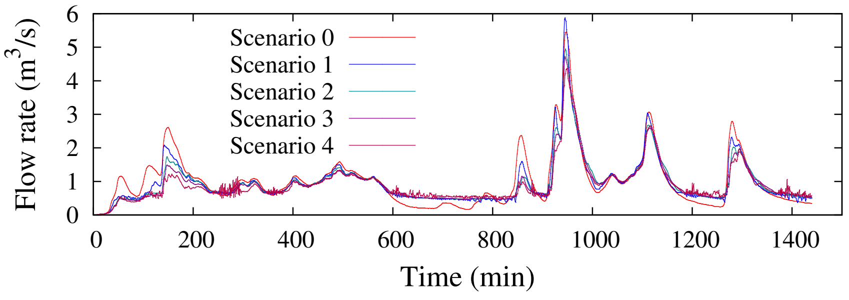
[4] There are two independent rainfall events on 1st February 2015. The experimental results are given considering both events



(a) 1st February 2015



(b) 24th March 2014



(c) 30th January 2015

Fig. 4.11. Flow Rate vs. Time using Scenarios 0-4

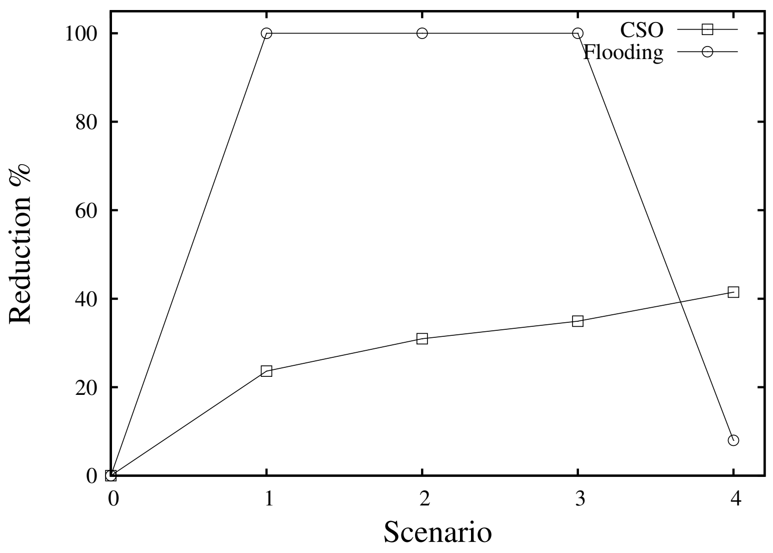


Fig. 4.12. CSO and Flooding reduction vs. Scenario using the rainfall event of 24th March 2014.

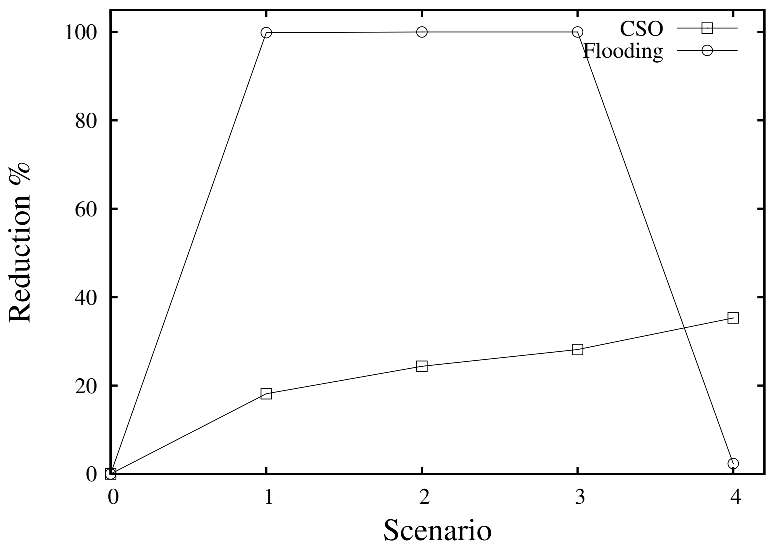


Fig. 4.13. CSO and Flooding reduction vs. Scenario using the rainfall event of 30th January 2015.

2015, 1st February 2015, respectively. In these figures, local flooding and CSO reductions are 0 for scenario 0, as it is the reference scenario. In Figure 4.12, it

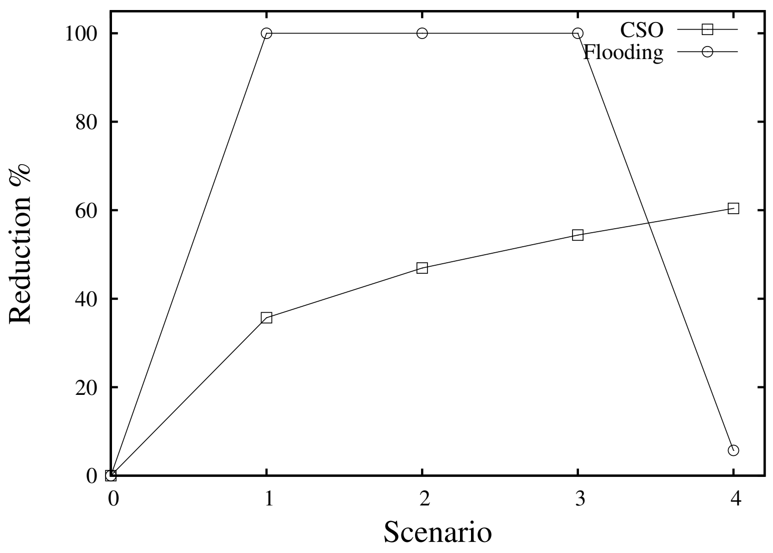


Fig. 4.14. CSO and Flooding reduction vs. Scenario using the rainfall event of 1st February 2015.

can be observed that for the event of 24th March 2014, scenario 1 reduces the CSO by 23.6%, with respect to scenario 0. As the number of gates increases, the reduction percentage rises, reaching a value of 41% for scenario 4. Regarding local flooding, the reduction is 100% for all the controlled solutions, except for scenario 4. Similar results are obtained for the other two selected events (see Figures 4.13 and 4.14). In particular, the flooding reduction trend is basically the same as the previous case, while CSO reduction span from 18.2% (scenario 1) to 35% (scenario 4) for the event of 30th January 2015, and from 35.7% to 60% for the event of 1st February 2015. Summing up, these findings suggest that scenarios 2 and 3 are the most convenient solutions, since they offer the highest overall performance in terms of both CSO reduction and local flooding removal.

4.4 Conclusions

In this chapter, a decentralized real time control (DRTC), based on a multi-agent paradigm and specifically a gossip-based algorithm, has been developed and integrated with the hydrodynamic simulation model, SWMM. The DRTC proposed has been applied to the urban drainage system (UDS) in the city of Cosenza, Italy. The UDS, modelled in SWMM, is equipped with a series of moveable gates, functioning as actuators, and sensors, which monitor water level in each conduit. The findings show that the DRTC algorithm proposed was able to balance the hydraulic capacity of the conduits within the system by utilizing the storage capacity of the less overwhelmed conduits during intense rainfall events. In other words, the DRTC algorithm was able to control the water level within the UDS successfully, ensuring a full utilization of the actual storage capacity of the system. The findings clearly demonstrated that the DRTC produced beneficial effects on the management of the UDS by substantially mitigating the risk of flooding and CSO.

Future research will test the response of the DRTC not only to varying hydrological inputs, but also to changing boundary conditions, for example, sudden pipe breakage or increased clogging in the catch basins, to demonstrate the high adaptability of the algorithm to different, even unforeseen, situations. Furthermore, the actual costs of the physical implementation and the energy consumption required will be further investigated.

Conclusions and outlook

The work behind this PHD thesis was devoted to the design and implementation of a novel cloud-assisted, agent-based framework for Cyber-Physical Systems, called Rainbow, which addresses requirements and issues of CPS such as scalability, physical devices heterogeneity, responsiveness and reconfigurability.

Rainbow hides heterogeneity providing the concept of Virtual Objects, and addresses the distributed nature of CPSs introducing a distributed multi-agent system on the top of the physical part. Rainbow aims to get the computation as close as possible to the physical devices, which are capable both of sensing and changing the physical environment in which they are immersed. The dynamic adaptivity requirement of CPSs is addressed by fostering the use of decentralized and swarm intelligence algorithms.

The effectiveness of the proposed framework has been shown through a set of application scenarios, in the contexts of Building Automation, Smart Homes and Smart Cities, thus showing how different applications can be easily designed and implemented using the proposed framework, as well as how decentralized and swarm intelligence algorithms can be effectively exploited to provide adaptivity capabilities to physical systems.

Finally, a CPS for an urban drainage system has been proposed. For this scenario, an original distributed real time control system was designed, exploiting a gossip-based algorithm. The proposed approach requires the urban drainage network to be equipped with a set of water level sensors and electronically movable gates, spread across the whole network. All the gates are locally controlled by PID controllers, which are globally orchestrated by the gossip-based algorithm, thus achieving an optimal hydrodynamic behaviour which is capable to effectively reduce combined sewer overflow and flooding phenomena. The experiments, which has been conducted using as case study the urban drainage network of the city of Cosenza (Italy), have demonstrated the validity of the approach.

Starting from this thesis contributions, further works will investigate other directions, as follows:

- It is interesting to find out whether the decentralized approach designed for the urban drainage network can be adapted or extended so as to deal with other scenarios involving physical systems consisting of large network, such as traffic control, water distribution, power grids.
- The knowledge discovery workflow presented in Section 3.4.1 can also be further investigated, finding new mining algorithms that fits with it and new application scenario, other then ambient assisted living, that can benefit from this approach.
- While the proposed framework was successfully exploited in many application scenarios, it can be further extended providing additional features and capabilities. A set of working decentralized coordination and cooperation algorithm, such as those which are used in the proposed application scenarios, can be provided as a library, so as to support developers in the design of systems. In addition, functionalities for efficient data stream aggregation and filtering can be also provided.

References

1. T Abdelzaher. Towards an architecture for distributed cyber-physical systems. In *Proceedings of the 2006 NSF workshop on cyber-physical systems*, 2006.
2. Stefan Achleitner, Michael Möderl, and Wolfgang Rauch. City drain©–an open source approach for simulation of integrated urban drainage systems. *Environmental modelling & software*, 22(8):1184–1195, 2007.
3. Barry J Adams and Fabian Papa. Urban stormwater management planning with analytical probabilistic models. *Canadian Journal of Civil Engineering*, 28(3):545, 2001.
4. Charu C Aggarwal. *Data streams: models and algorithms*, volume 31. Springer Science & Business Media, 2007.
5. A. Z. Alkar and U. Buhur. An Internet Based Wireless Home Automation System for Multifunctional Devices. *IEEE Trans. on Consum. Electron.*, 51(4):1169–1174, November 2005.
6. S Massoud Amin and Bruce F Wollenberg. Toward a smart grid: power delivery for the 21st century. *Power and Energy Magazine, IEEE*, 3(5):34–41, 2005.
7. Panos Antsaklis. Goals and challenges in cyber-physical systems research editorial of the editor in chief. *Automatic Control, IEEE Transactions on*, 59(12):3117–3119, 2014.
8. Kevin Ashton. That internet of things thing. *RFiD Journal*, 22(7):97–114, 2009.
9. Karl J Astrom. Pid controllers: theory, design and tuning. *Instrument society of America*, 1995.
10. Peter M Bach, Wolfgang Rauch, Peter S Mikkelsen, David T McCarthy, and Ana Deletic. A critical review of integrated urban water modelling–urban drainage and beyond. *Environmental Modelling & Software*, 54:88–107, 2014.
11. Paolo Baronti, Prashant Pillai, Vince WC Chook, Stefano Chessa, Alberto Gotta, and Y Fun Hu. Wireless sensor networks: A survey on the state of the art and the 802.15. 4 and zigbee standards. *Computer communications*, 30(7):1655–1695, 2007.
12. T Beeneken, V Erbe, A Messmer, C Reder, R Rohlfing, M Scheer, M Schuetze, B Schumacher, M Weilandt, and M Weyand. Real time control (rtc) of urban drainage systems–a discussion of the additional efforts compared to conventionally operated systems. *Urban Water Journal*, 10(5):293–299, 2013.

13. Nicola Bicocchi, Marco Mamei, and Franco Zambonelli. Self-organizing virtual macro sensors. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 7(1):2, 2012.
14. Ilse Bierhoff, Ad van Berlo, Julio Abascal, Bob Allen, Anton Civit, Klaus Fellbaum, Erkki Kemppainen, Noemi Bitterman, Diamantino Freitas, and Kristian Kristiansson. *Smart home environment.* COST, Brussels, 2007.
15. Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm intelligence: from natural to artificial systems.* Number 1. Oxford university press, 1999.
16. Manfred Broy, María Victoria Cengarle, and Eva Geisberger. Cyber-physical systems: imminent challenges. In *Large-Scale Complex IT Systems. Development, Operation and Management*, pages 1–28. Springer, 2012.
17. M Carbone, G Garofalo, and P Piro. Decentralized real time control in combined sewer system by using smart objects. *Procedia Engineering*, 89:473–478, 2014.
18. CL Philip Chen and Chun-Yang Zhang. Data-intensive applications, challenges, techniques and technologies: A survey on big data. *Information Sciences*, 275:314–347, 2014.
19. Liming Chen, C. Nugent, and G. Okeyo. An Ontology-Based Hybrid Approach to Activity Modeling for Smart Homes. *Human-Machine Systems, IEEE Trans. on*, 44(1):92–105, Feb 2014.
20. Diane J Cook and Sajal K Das. How smart are our environments? an updated look at the state of the art. *Pervasive and mobile computing*, 3(2):53–73, 2007.
21. Diane J. Cook, Narayanan Chatapuram Krishnan, and Parisa Rashidi. Activity Discovery and Activity Recognition: A New Partnership. *IEEE Trans. on Systems, Man, and Cybernetics*, 43(3):820–828, 2013.
22. Marcello Coppola, Babak Falsafi, John Goodacre, and George Kornaros. From embedded multi-core socs to scale-out processors. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 947–951. EDA Consortium, 2013.
23. T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Trans. Inf. Theor.*, 13(1):21–27, September 1967.
24. Erik Dahlman, Stefan Parkvall, and Johan Skold. *4G: LTE/LTE-advanced for mobile broadband.* Academic press, 2013.
25. Robert H Dennard, VL Rideout, E Bassous, and AR Leblanc. Design of ion-implanted mosfet's with very small physical dimensions. *Solid-State Circuits, IEEE Journal of*, 9(5):256–268, 1974.
26. Tharam S Dillon, Hai Zhuge, Chen Wu, Jaipal Singh, and Elizabeth Chang. Web-of-things framework for cyber–physical systems. *Concurrency and Computation: Practice and Experience*, 23(9):905–923, 2011.
27. G Dirckx, M Schütze, S Kroll, Ch Thoeye, G De Gueldre, and B Van De Steene. Rtc versus static solutions to mitigate csos impact. In *12nd International Conference on Urban Drainage, 2011b. Porto Alegre, Brazil*, 2011.
28. E Directive. The environmental noise directive (2002/49/eg). *Official Journal of the European Communities*, 2002.
29. A. Dohr, R. Modre-Opsrian, M. Drobics, D. Hayn, and G. Schreier. The internet of things for ambient assisted living. In *Information Technology: New Generations (ITNG), 2010 Seventh International Conference on*, pages 804–809, April 2010.
30. Florian Dötsch, Jörg Denzinger, Holger Kasinger, and Bernhard Bauer. Decentralized real-time control of water distribution networks using self-organizing

multi-agent systems. In *Self-Adaptive and Self-Organizing Systems (SASO), 2010 4th IEEE International Conference on*, pages 223–232. IEEE, 2010.

31. G. Fortino, R. Giannantonio, R. Gravina, P. Kuryloski, and R. Jafari. Enabling Effective Programming and Flexible Management of Efficient Body Sensor Network Applications. *Human-Machine Systems, IEEE Trans. on*, 43(1):115–133, 2013.
32. G. Fortino, A. Guerrieri, G.M.P. O'Hare, and A. Ruzzelli. A flexible building management framework based on wireless sensor and actuator networks. *Journal of Network and Computer Applications*, 35:1934 – 1952, 2012.
33. Giancarlo Fortino, Antonio Guerrieri, Michelangelo Lacopo, Matteo Lucia, and Wilma Russo. An agent-based middleware for cooperating smart objects. In *Highlights on Practical Applications of Agents and Multi-Agent Systems*, pages 387–398. Springer, 2013.
34. Raspberry Pi Foundation. Raspberry pi - teach, learn and make with raspberry pi, 2011.
35. David J Frank, Robert H Dennard, Edward Nowak, Paul M Solomon, Yuan Taur, and Hon-Sum Philip Wong. Device scaling limits of si mosfets and their application dependencies. *Proceedings of the IEEE*, 89(3):259–288, 2001.
36. Guangtao Fu, David Butler, and Soon-Thiam Khu. Multiple objective optimal control of integrated urban wastewater systems. *Environmental Modelling & Software*, 23(2):225–234, 2008.
37. Carles Gomez, Joaquim Oller, and Josep Paradells. Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology. *Sensors*, 12(9):11734–11753, 2012.
38. Timothy W Hnat, Tamim I Sookoor, Pieter Hooimeijer, Westley Weimer, and Kamin Whitehouse. Macrolab: a vector-based macroprogramming framework for cyber-physical systems. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 225–238. ACM, 2008.
39. Enamul Hoque and John A. Stankovic. AALO: Activity recognition in smart homes using Active Learning in the presence of Overlapped activities. In *6th International Conference on Pervasive Computing Technologies for Healthcare, PervasiveHealth 2012*, pages 139–146, 2012.
40. Márk Jelasity, Alberto Montresor, and Ozalp Babaoglu. Gossip-based aggregation in large dynamic networks. *ACM Transactions on Computer Systems (TOCS)*, 23(3):219–252, 2005.
41. James Kennedy, James F Kennedy, Russell C Eberhart, and Yuhui Shi. *Swarm intelligence*. Morgan Kaufmann, 2001.
42. Siddhartha Kumar Khaitan, James D McCalley, and Chen Ching Liu. *Cyber Physical Systems Approach to Smart Electric Power Grid*. Springer, 2015.
43. Hermann Kopetz. *Real-time systems: design principles for distributed embedded applications*. Springer Science & Business Media, 2011.
44. Gerd Kortuem, Fahim Kawsar, Daniel Fitton, and Vasughi Sundramoorthy. Smart objects as building blocks for the internet of things. *Internet Computing, IEEE*, 14(1):44–51, 2010.
45. Anis Koubâa and Björn Andersson. A vision of cyber-physical internet. 2009.
46. Edward Lee et al. Cyber physical systems: Design challenges. In *Object Oriented Real-Time Distributed Computing (ISORC), 2008 11th IEEE International Symposium on*, pages 363–369. IEEE, 2008.

47. Jay Lee, Edzel Lapira, Behrad Bagheri, and Hung-an Kao. Recent advances and trends in predictive manufacturing systems in big data environment. *Manufacturing Letters*, 1(1):38–41, 2013.
48. Paulo Leitão. Towards self-organized service-oriented multi-agent systems. In *Service Orientation in Holonic and Multi Agent Manufacturing and Robotics*, pages 41–56. Springer, 2013.
49. Jing Lin, Sahra Sedigh, and Ann Miller. Modeling cyber-physical systems with semantic agents. In *Computer Software and Applications Conference Workshops (COMPSACW), 2010 IEEE 34th Annual*, pages 13–18. IEEE, 2010.
50. Chung Laung Liu and James W Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM (JACM)*, 20(1):46–61, 1973.
51. Liqian Luo, Tarek F Abdelzaher, Tian He, and John A Stankovic. Envirosuite: An environmentally immersive programming framework for sensor networks. *ACM Transactions on Embedded Computing Systems (TECS)*, 5(3):543–576, 2006.
52. Friedemann Mattern and Christian Floerkemeier. From the internet of computers to the internet of things. In *From active data management to event-based systems and more*, pages 242–259. Springer, 2010.
53. M Moderl, M Kleidorfer, and W Rauch. Influence of characteristics on combined sewer performance. *Water Science & Technology*, 66(5):1052–1060, 2012.
54. Pieter J Mosterman and Justyna Zander. Cyber-physical systems challenges: a needs analysis for collaborating embedded software systems. *Software & Systems Modeling*, pages 1–12, 2015.
55. Shanmugavelayutham Muthukrishnan. *Data streams: Algorithms and applications*. Now Publishers Inc, 2005.
56. Ryan Newton, Greg Morrisett, and Matt Welsh. The regiment macroprogramming system. In *Proceedings of the 6th international conference on Information processing in sensor networks*, pages 489–498. ACM, 2007.
57. Nieves Pavón-Pulido, Juan Antonio López-Riquelme, Joaquín Ferruz-Melero, Miguel Ángel Vega-Rodríguez, and Antonio José Barrios-León. A service robot for monitoring elderly people in the context of ambient assisted living. *J. Ambient Intell. Smart Environ.*, 6(6):595–621, 2014.
58. Eldad Perahia and Michelle X Gong. Gigabit wireless lans: an overview of ieee 802.11 ac and 802.11 ad. *ACM SIGMOBILE Mobile Computing and Communications Review*, 15(3):23–33, 2011.
59. P Piro. Il bacino sperimentale urbano del canale liguori nella città di cosenza. osservazioni sperimentali quali-quantitative nel periodo 1995–2003. *Bios*, 2007.
60. P Piro, M Carbone, and G Garofalo. Distributed vs. concentrated storage options for controlling cso volumes and pollutant loads. *Water Practice & Technology*, 5(3), 2010.
61. P Piro, M Carbone, G Garofalo, and J Sansalone. Cso treatment strategy based on constituent index relationships in a highly urbanised catchment. *Water Science & Technology*, 56(12):85–91, 2007.
62. P Piro, M Carbone, G Garofalo, and J Sansalone. Size distribution of wet weather and dry weather particulate matter entrained in combined flows from an urbanizing sewershed. *Water, Air, and Soil Pollution*, 206(1-4):83–94, 2010.
63. Patrizia Piro and Marco Carbone. A modelling approach to assessing variations of total suspended solids (tss) mass fluxes during storm events. *Hydrological Processes*, 28(4):2419–2426, 2014.

64. Martin Pleau, Hubert Colas, Pierre Lavallée, Geneviève Pelletier, and Richard Bonin. Global optimal real-time control of the quebec urban drainage system. *Environmental Modelling & Software*, 20(4):401–413, 2005.
65. P. Pudil, J. Novovičová, and J. Kittler. Floating search methods in feature selection. *Pattern Recogn. Lett.*, 15(11):1119–1125, November 1994.
66. Ragunathan Raj Rajkumar, Insup Lee, Lui Sha, and John Stankovic. Cyber-physical systems: the next computing revolution. In *Proceedings of the 47th Design Automation Conference*, pages 731–736. ACM, 2010.
67. P. Rashidi and A. Mihailidis. A Survey on Ambient-Assisted Living Tools for Older Adults. *Biomedical and Health Informatics, IEEE Journal of*, 17(3):579–590, May 2013.
68. Parisa Rashidi and Diane J. Cook. Com: A method for mining and monitoring human activity patterns in home-based health monitoring systems. *ACM Trans. Intell. Syst. Technol.*, 4(4):64:1–64:20, October 2013.
69. Kahn RHKJM and KSJ Pister. Mobile networking for smart dust. In *ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 99), Seattle*, 1999.
70. Philipp Richter, Manuel Toledano-Ayala, Genaro M. Soto-Zarazúa, and Edgar A. Rivas-Araiza. A Survey of Hybridisation Methods of GNSS and Wireless LAN Based Positioning System. *J. Ambient Intell. Smart Environ.*, 6(6):723–738, November 2014.
71. Joel JPC Rodrigues and Paulo ACS Neves. A survey on ip-based wireless sensor network solutions. *International Journal of Communication Systems*, 23(8):963–981, 2010.
72. Lewis A Rossman and Water Supply. *Storm water management model, quality assurance report: dynamic wave flow routing.* US Environmental Protection Agency, Office of Research and Development, National Research Management Research Laboratory, 2006.
73. Lee Sang-hyun, Jeong-gi Lee, and Moon Kyung-il. Smart Home Security System Using Multiple ANFIS. *International Journal of Smart Home*, 7(3):121–132, May 2013.
74. Teodora Sanislav and Liviu Miclea. Cyber-physical systems-concept, challenges and research areas. *Journal of Control Engineering and Applied Informatics*, 14(2):28–33, 2012.
75. Loïc Schmidt, Nathalie Mitton, and David Simplot-Ryl. Towards unified tag data translation for the internet of things. In *Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology, 2009. Wireless VITAE 2009. 1st International Conference on*, pages 332–335. IEEE, 2009.
76. Bernhard Schulz. Lte transmission modes and beamforming. *Rohde and Schwarz White Paper*, 2011.
77. Edmund W Schuster, Stuart J Allen, and David L Brock. *Global RFID: the value of the EPCglobal network for supply chain management.* Springer Science & Business Media, 2007.
78. Manfred Schütze, Alberto Campisano, Hubert Colas, Wolfgang Schilling, and Peter A Vanrolleghem. Real time control of urban wastewater systemswhere do we stand today? *Journal of hydrology*, 299(3):335–348, 2004.
79. Immanuel Schweizer, Roman Bärtl, Axel Schulz, Florian Probst, and Max Mühläuser. Noisemap-real-time participatory noise maps. In *Proc. 2nd Intl*

Workshop on Sensing Applications on Mobile Phones (PhoneSense11), pages 1–5, 2011.

80. Yongwon Seo, Young-Ho Seo, and Young-Oh Kim. Behavior of a fully-looped drainage network and the corresponding dendritic networks. *Water*, 7(3):1291–1305, 2015.
81. Jordi Serra, David Pubill, Angelos Antonopoulos, and Christos Verikoukis. Smart HVAC Control in IoT: Energy Consumption Minimization with User Comfort Constraints. *The Scientific World Journal*, 2014.
82. Lui Sha, Tarek Abdelzaher, Karl-Erik Årzén, Anton Cervin, Theodore Baker, Alan Burns, Giorgio Buttazzo, Marco Caccamo, John Lehoczky, and Aloysius K Mok. Real time scheduling theory: A historical perspective. *Real-time systems*, 28(2-3):101–155, 2004.
83. Pervez Hameed Shaikh, Nursyarizal Bin Mohd Nor, Perumal Nallagownden, Irraivan Elamvazuthi, and Taib Ibrahim. A review on optimized control systems for building energy and comfort management of smart sustainable buildings. *Renewable and Sustainable Energy Reviews*, 34:409–429, 2014.
84. Jianhua Shi, Jiafu Wan, Hehua Yan, and Hui Suo. A survey of cyber-physical systems. In *Wireless Communications and Signal Processing (WCSP), 2011 International Conference on*, pages 1–6. IEEE, 2011.
85. N.K. Suryadevara, S.C. Mukhopadhyay, R. Wang, and R.K. Rayudu. Forecasting the behavior of an elderly using wireless sensors data in a smart home. *Engineering Applications of Artificial Intelligence*, 26(10):2641 – 2652, 2013.
86. Carolyn Talcott. Cyber-physical systems and events. In *Software-Intensive Systems and New Computing Paradigms*, pages 101–115. Springer, 2008.
87. Roberto Verdone, Davide Dardari, Gianluca Mazzini, and Andrea Conti. *Wireless sensor and actuator networks: technologies, analysis and design.* Academic Press, 2010.
88. Chonggang Wang, Tao Jiang, and Qian Zhang. *ZigBee® network protocols and applications.* CRC Press, 2014.
89. Roy Want. An introduction to rfid technology. *Pervasive Computing, IEEE*, 5(1):25–33, 2006.
90. Mark Weiser. The computer for the 21st century. *Scientific american*, 265(3):94–104, 1991.
91. Michael Wooldridge. *An introduction to multiagent systems.* John Wiley & Sons, 2009.
92. Jennifer Yick, Biswanath Mukherjee, and Dipak Ghosal. Wireless sensor network survey. *Computer networks*, 52(12):2292–2330, 2008.
93. S Ying et al. Foundations for innovation in cyber-physical systems. In *Workshop Report, Energetics Incorporated, Columbia, Maryland, US*, 2013.
94. G Michael Youngblood, Edwin O Heierman, Lawrence B Holder, and Diane J Cook. Automation intelligence for the smart environment. In *International Joint Conference On Artificial Intelligence*, volume 19, page 1513. Citeseer, 2005.
95. Di Zhang, Nilay Shah, and Lazaros G Papageorgiou. Efficient energy consumption and operation management in a smart building with microgrid. *Energy Conversion and management*, 74:209–222, 2013.
96. John G Ziegler and Nathaniel B Nichols. Optimum settings for automatic controllers. *trans. ASME*, 64(11), 1942.