N°d'ordre NNT : 2017LYSEI126

THESE DE DOCTORAT DE L'UNIVERSITÉ DE LYON
opérée au sein de
INSA LYON

Ecole Doctorale ED 512
Informatique et Mathématiques de Lyon
(INFOMATHS)

Spécialité de Doctorat :
Informatique

Soutenue publiquement le 15/12/2017, par :

Ozgun PINARER

Sustainable Declarative Monitoring Architecture: Energy optimization of interactions between application service oriented queries and wireless sensor devices - Application to Smart Buildings

Devant le jury composé de :

Présidente LIBOUREL, Thérèse	Professeur Emérite, Université de Montpellier	
BABAU, J-Philippe	Professeur, Université de Brest	Rapporteur
BASKURT, Atilla	Professeur, INSA Lyon	Directeur de thèse
BOUJU, Alain	Maître de Conférences HDR, Université de La Rochelle	Examinateur
GRIPAY, Yann	Maître de Conférences, INSA Lyon	Examinateur
PINET, François	Directeur de Recherche, IRSTEA Clermont-Ferrand	Rapporteur
SERVIGNE, Sylvie	Maître de Conférences, INSA Lyon	Examinatrice

Département FEDORA – INSA Lyon - Ecoles Doctorales – Quinquennal 2016-2020

SIGLE	ÉCOLE DOCTORALE	NOM ET COORDONNEES DU RESPONSABLE
CHIMIE	**CHIMIE DE LYON** **http://www.edchimie-lyon.fr** Sec. : Renée EL MELHEM Bât. Blaise PASCAL, 3e étage **secretariat@edchimie-lyon.fr** INSA : R. GOURDON	M. Stéphane DANIELE Institut de recherches sur la catalyse et l'environnement de Lyon IRCELYON-UMR 5256 Équipe CDFA 2 Avenue Albert EINSTEIN 69 626 Villeurbanne CEDEX **directeur@edchimie-lyon.fr**
E.E.A.	**ÉLECTRONIQUE, ÉLECTROTECHNIQUE, AUTOMATIQUE** **http://edeea.ec-lyon.fr** Sec. : M.C. HAVGOUDOUKIAN **ecole-doctorale.eea@ec-lyon.fr**	M. Gérard SCORLETTI École Centrale de Lyon 36 Avenue Guy DE COLLONGUE 69 134 Écully Tél : 04.72.18.60.97 Fax 04.78.43.37.17 **gerard.scorletti@ec-lyon.fr**
E2M2	**ÉVOLUTION, ÉCOSYSTÈME, MICROBIOLOGIE, MODÉLISATION** **http://e2m2.universite-lyon.fr** Sec. : Sylvie ROBERJOT Bât. Atrium, UCB Lyon 1 Tél : 04.72.44.83.62 INSA : H. CHARLES **secretariat.e2m2@univ-lyon1.fr**	M. Fabrice CORDEY CNRS UMR 5276 Lab. de géologie de Lyon Université Claude Bernard Lyon 1 Bât. Géode 2 Rue Raphaël DUBOIS 69 622 Villeurbanne CEDEX Tél : 06.07.53.89.13 **cordey@univ-lyon1.fr**
EDISS	**INTERDISCIPLINAIRE SCIENCES-SANTÉ** **http://www.ediss-lyon.fr** Sec. : Sylvie ROBERJOT Bât. Atrium, UCB Lyon 1 Tél : 04.72.44.83.62 INSA : M. LAGARDE **secretariat.ediss@univ-lyon1.fr**	Mme Emmanuelle CANET-SOULAS INSERM U1060, CarMeN lab, Univ. Lyon 1 Bâtiment IMBL 11 Avenue Jean CAPELLE INSA de Lyon 69 621 Villeurbanne Tél : 04.72.68.49.09 Fax : 04.72.68.49.16 **emmanuelle.canet@univ-lyon1.fr**
INFOMATHS	**INFORMATIQUE ET MATHÉMATIQUES** **http://edinfomaths.universite-lyon.fr** Sec. : Renée EL MELHEM Bât. Blaise PASCAL, 3e étage Tél : 04.72.43.80.46 Fax : 04.72.43.16.87 **infomaths@univ-lyon1.fr**	M. Luca ZAMBONI Bât. Braconnier 43 Boulevard du 11 novembre 1918 69 622 Villeurbanne CEDEX Tél : 04.26.23.45.52 **zamboni@maths.univ-lyon1.fr**
Matériaux	**MATÉRIAUX DE LYON** **http://ed34.universite-lyon.fr** Sec. : Marion COMBE Tél : 04.72.43.71.70 Fax : 04.72.43.87.12 Bât. Direction **ed.materiaux@insa-lyon.fr**	M. Jean-Yves BUFFIÈRE INSA de Lyon MATEIS - Bât. Saint-Exupéry 7 Avenue Jean CAPELLE 69 621 Villeurbanne CEDEX Tél : 04.72.43.71.70 Fax : 04.72.43.85.28 **jean-yves.buffiere@insa-lyon.fr**
MEGA	**MÉCANIQUE, ÉNERGÉTIQUE, GÉNIE CIVIL, ACOUSTIQUE** **http://edmega.universite-lyon.fr** Sec. : Marion COMBE Tél : 04.72.43.71.70 Fax : 04.72.43.87.12 Bât. Direction **mega@insa-lyon.fr**	M. Philippe BOISSE INSA de Lyon Laboratoire LAMCOS Bâtiment Jacquard 25 bis Avenue Jean CAPELLE 69 621 Villeurbanne CEDEX Tél : 04.72.43.71.70 Fax : 04.72.43.72.37 **philippe.boisse@insa-lyon.fr**
ScSo	**ScSo*** **http://ed483.univ-lyon2.fr** Sec. : Viviane POLSINELLI Brigitte DUBOIS INSA : J.Y. TOUSSAINT Tél : 04.78.69.72.76 **viviane.polsinelli@univ-lyon2.fr**	M. Christian MONTES Université Lyon 2 86 Rue Pasteur 69 365 Lyon CEDEX 07 **christian.montes@univ-lyon2.fr**

*ScSo : Histoire, Géographie, Aménagement, Urbanisme, Archéologie, Science politique, Sociologie, Anthropologie

“Give me a place to stand and I will move the earth.”
—Archimède

Acknowledgements

First and foremost, I would like to express my special appreciation and gratitude to my supervisors Professor Atilla Baskurt, Associate Professor Sylvie Servigne and Associate Professor Yann Gripay. I would like to thank you for encouraging my research. After all our meetings, I have felt that a new perspective has opened in my mind not only about my research but also about science and universe. Since we met, I have always felt lucky to have such experienced and talented advisors. Your character with endless patience and discipline has become a good role model for my further academic life. I am very lucky to have the chance to work with you.

I would also like to express my appreciation to my Turkish advisor Associate Professor Atay Ozgovde from Galatasaray University who gave me inspiration. You were patient, helpful, and supportive during my research.

I am also grateful to my special friends that I met during my stay in France. You are always a part of my family. Our joyful discussions and exchanging ideas gave me the courage and belief to continue my research. You always remind me to smile to the difficulties. I hope I could help you in your life as much as you do for me. I would like to thank you.

Last but not least my family. You have always believed in me. You are always only and only supportive without questioning. I would like to thank you for all of the sacrifices that you have made on my behalf. I am very lucky to have a family so great and so unique.

Abstract

Over the past decade, popular attention to smart building has increased. Smart building and sustainability are intertwined. Since traditional buildings are primary consumers of a significant portion of energy resources, we need smart buildings that are designed based on sustainable construction standards to consume less energy than traditional buildings and to minimize their impacts on the natural environment. Thus, smart buildings became one of the major application domains of pervasive environments. Smart building technology brings in some nice features such as security, comfort and accessibility. It can also enhance the energy consumption of the buildings. A basic smart building infrastructure consists of a set of wireless sensor devices. High energy consumption of these devices is the most challenging problem in this research area. These devices build the Wireless Sensor Network environment and are autonomous in terms of energy: their energy consumption determines their lifespan. Since the energy consumption has a strong impact on the lifetime of the service, there are several approaches in the literature. However, existing approaches are often fitted to a single monitoring application and rely on static configurations for sensor devices. A basic sensor device is responsible for data acquisition, transmission and reception (computation if requested).

In this thesis, we study the monitoring system of a smart building that supports multi-application in a pervasive environment. We focus on a sustainable architecture for multi-application monitoring systems that continuously adapt to application requirements, context and user configuration. From data point of view, a monitoring system as a set of applications that exploit sensor measures in real-time, where these applications are declaratively expressed as (service-oriented) continuous queries over sensor data streams. Hence a multiple application system requires handling several data stream requests with different data acquisition/transmission frequencies for the same wireless sensor device and supporting dynamic requirements of applications (e.g. high transmission frequencies for occupied rooms, lower frequency during night). Since a static configuration can not optimize the energy consumption of the monitoring system with regards to actual application requirements, we work on intethe raction between the application requirements and the wireless sensor devices. Thus as a solution, we propose ***S**mart-**S**ervice **S**tream-**o**riented **S**ensor **M**anagement (3SoSM)*, an approach to optimize interactions between application requirements and wireless sensor environment in real-time. Our `3SoSM` approach performs energy-aware dynamic sensor device configuration to lower energy consumption while fulfilling real-time application requirements. It is based on a real-time sensor configuration that is derived from a scheduling time pattern. This time pattern is build by the acquisition, transmission and reception actions of the sensor device and it creates a periodic agenda for each sensor device. With this approach, we expect to avoid unnecessary data measurements and to promote shorter/compressed data transmission when possible.

Besides, we present an optimization process to finalize the schedule time pattern. The aim of the optimization process is to search the optimal communication slots with which the sensor devices consume less energy than the other cases. During the optimization process, we pay attention to the constraints of each sensor device due to the given application requirements. Due to the constraint based behavior and to the tree structured topology of the network, the optimization process starts by propagating transmission and reception constraints based on latencies in a bottom-up process. Then, the algorithm tries to find the most energy efficient communication slots, and finally propagates those choices to lower-layers and

continues the algorithm in a top-down process in the network topology. The optimization function of that process is to minimize the number of communication slots (RX slots) in order to minimize the consumed energy. The proposed optimization algorithm is a sort of *greedy heuristic algorithm* is employed to determine the optimal slots for sensor devices. The core concept of greedy algorithm is to perform a short-sighted action in each step. It starts from the reception part of the schedule table of sink device and tries to group the receptions from the same sensor device. In each step, an optimal communication slot is found and the step is repeated until all the reception data is matched with a slot.

Furthermore, we present a set of experiments that we conducted with a wireless sensor network simulator to perform the presented scenarios. The results of each scenario is introduced and discussed. At the end of the experiments, the energy enhancement of different types of sensor devices are presented. With the approach presented in this thesis, we optimize the energy consumption and reduce the unnecessary communication cost. Obtained results on energy saving and lifetime extension depend on the application requirements and the context of the application.

Resume

La dernière décennie a montré un intérêt croissant pour les bâtiments intelligents. En effet, comme les bâtiments traditionnels sont les principaux consommateurs d'une partie importante des ressources énergétiques, le besoin de bâtiments intelligents a alors émergé. Ces nouveaux bâtiments doivent être conçus selon des normes de construction durables pour consommer moins et minimiser les impacts sur le milieu naturel. Les technologies de construction de bâtiments intelligents contribuent à une amélioration de la sécurité, du confort, de l'accessibilité voire de la consommation d'énergie des bâtiments eux-mêmes. Ces bâtiments intelligents sont devenus l'un des principaux domaines d'application des environnements pervasifs. En effet, une infrastructure basique de construction de bâtiment intelligent se compose notamment d'un ensemble de capteurs sans fil. Les capteurs basiques permettent l'acquisition, la transmission et la réception de données. A ces réseaux de capteurs s'ajoutent généralement des périphériques et un système de surveillance. La consommation d'énergie élevée de l'ensemble de ces appareils est un des problèmes les plus difficiles et fait donc l'objet d'études dans ce domaine de la recherche. Les capteurs sont autonomes en termes d'énergie ce qui implique que leur consommation d'énergie détermine leur durée de vie et donc la continuité de service. Etant donné que la consommation d'énergie a un fort impact sur la durée de vie du service, il existe plusieurs approches dans la littérature. Cependant, les approches existantes sont souvent adaptées à une seule application de surveillance et reposent sur des configurations statiques pour les capteurs.

Dans cette thèse, nous contribuons à la définition d'une architecture de monitoring déclaratif durable par l'optimisation énergétique des interactions entre requêtes applicative orientées service et réseau de capteurs sans fil. Nous avons choisi le bâtiment intelligent comme cas d'application et nous étudions donc un système de surveillance d'un bâtiment intelligent pouvant prendre en charge des multi-applications dans un environnement pervasif. Nous nous concentrons sur une architecture durable pour les systèmes de surveillance multi-applications qui s'adaptent en permanence aux exigences des applications, au contexte et à la configuration de l'utilisateur. Du point de vue logiciel, un système de surveillance peut être défini comme un ensemble d'applications qui exploitent les mesures des capteurs en temps réel. Ces applications sont exprimées dans un langage déclaratif sous la forme de requêtes continues (orientées service) sur les flux de données des capteurs. Par conséquent, un système de multi-applications nécessite la gestion de plusieurs demandes de flux de données suivant différentes fréquences d'acquisition/transmission de données pour le même capteur sans fil, avec des exigences dynamiques requises par les applications (par exemple, fréquences de transmission élevées pour les salles occupées, fréquence inférieure pendant la nuit). Etant donné qu'une configuration statique ne peut pas optimiser la consommation d'énergie du système de surveillance en ce qui concerne les exigences réelles de l'application, nous travaillons sur l'interaction entre les exigences de l'application et les capteurs sans fil. Ainsi, nous proposons une approche intitulée Smart-Service Stream-oriented Sensor Management (3SoSM) afin d'optimiser les interactions entre les exigences des applications et l'environnement des capteurs sans fil, en temps réel. Notre approche 3SoSM offre une configuration dynamique des capteurs pour réduire la consommation d'énergie tout en satisfaisant les exigences des applications en temps réel. Le cœur de cette configuration de capteurs repose sur un Schedule Time Pattern (pattern temporel de planification). Ce pattern temporel est construit par les actions d'acquisition, de transmission et de réception du capteur et crée un agenda

périodique pour chaque capteur. Cette approche permet d'éviter les mesures inutiles de données et de favoriser une transmission de données plus courte/compressée lorsque cela est possible.

De plus, nous présentons un processus d'optimisation pour raffiner le Schedule Time Pattern. L'objectif du processus d'optimisation est de rechercher les créneaux de communication optimaux dans lesquels les capteurs consomment moins d'énergie que dans les autres cas. En raison du comportement basé sur les contraintes et de la topologie du réseau structurée en arbre, le processus d'optimisation proposé est un processus "bottom-up", des capteurs feuilles jusqu'à la racine. La fonction d'optimisation de ce processus consiste à minimiser le nombre de créneaux de communication (créneau RX) afin de minimiser l'énergie consommée. L'algorithme d'optimisation proposé est une sorte d'algorithme glouton utilisé pour déterminer les créneaux optimaux pour les capteurs. Nous avons conduit un ensemble d'expérimentations effectuées avec un simulateur de réseau de capteurs sans fil qui ont permis de valider notre approche quant à l'optimisation de la consommation d'énergie des capteurs, et donc l'augmentation de la durée de vie de ces capteurs, en réduisant notamment les communications non nécessaires.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

We live in an era where our lives are surrounded by highly advanced technological devices. In recent years, advances in microelectronics are making it possible to produce new smarter devices each day. Smart devices can be defined as devices that perform various smart functions such as health monitoring, people tracking, driving directions, etc. Some of these devices can even be worn on eyes, wrist, ankles, etc.

The most common smart device used by people is the smartphone. A research made in 2014 depicts the total number of smartphone users worldwide from 2014 to 2020. The number of mobile phone users in the world is expected to pass the five billion mark by 2019. The number of smartphone users is forecast to grow from 1.5 billion in 2014 to around 2.5 billion in 2019, with smartphone penetration rates increasing as well [54, 120, 142].

Smartphones, smartwatches, smart glasses are only few examples of the popular devices in the technology market. Similar to the given statistics, there exist many market researches about the market size, usage and the future of such devices. Predictions based on these statistical analyses are the strongest indication that smart devices will be among the mainstream computational devices in the near future.

Advanced technology based smart devices form an environment (the environment we live in) in which each device may interact with the other devices and with the human. This interaction opens a new era in the technology and a research domain so called `Internet of Things (IoT)`. It is considered as the next wave in the era of computing [70]. A generic description commonly adopted by the researchers is that the `IoT` is the network of objects that surround us with the understanding that information and communication systems are invisibly embedded in the environment around us. Objects may embed various sensors, actuators, UI devices, etc. This invisible network covers smart connectivity, cloud computing, context-aware computation, etc. A schematic of the interconnection of objects is depicted in Figure 1.1 where the application domains are numerous, e.g., home/building, transportation.

In the domain of building, energy consumption is currently an important social and economic issue. During the last decade, statistical energy consumption analysis reported that the energy consumption of buildings and also CO_2 emission increased significantly. 2011 Annual Energy Review of the World Business Council for Sustainable Development announced that in the United States, buildings are responsible for 41% of energy consumption of the country [45]. Moreover, [134] indicates that energy consumption of buildings in European Union (EU) is 37% of the total energy budget. Surprisingly this consumption is higher than the energy consumption of industry and transport which are respectively 28% and 32%.

Figure 1.2 shows for the USA the percentage of energy consumption of buildings which is 39%. 21% of this consumption stands for the residential and 18% represents commercial buildings. For each of them, details of energy consumption are given. For the residential buildings, heating consumes the majority part of the energy (32%), for the commercial building lighting consumes 28% of the energy budget. These

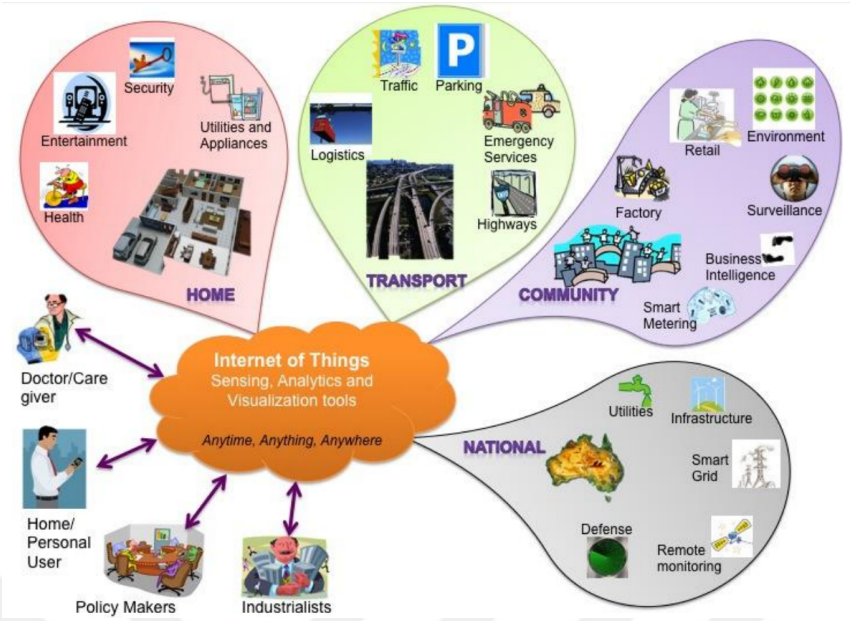


Fig. 1.1 Internet of Things Schematic showing the end users and application areas [70].

graphs and reports obviously show the necessity of monitoring and control system for buildings in order to monitor and reduce this high energy consumption.

The previous figure was introducing the sources of energy consumption in a building for the year 2011. Figure 1.3 presents the services that are the main energy consumers in a building for the year 2015 in the USA. Based on this data, 23% of the total energy is consumed by the space heating service. Lighting, space cooling, water heating services follow as the second major energy consumers in a building.

These statistical reports should be considered as a warning for the future. These results show the necessity of monitoring energy consumption of the buildings and of searching for solutions in order to reduce this high consumption. For controlling the energy consumption of a building, a building should be equipped with various wireless sensor devices (from a simple smart meter to more complex devices). These deployed wireless sensors form a computing environment that transforms a building into an intelligent building.

Institute for Building Efficiency [184] underlines that nowadays smart buildings besides having sensor devices for monitoring energy consumption, can provide useful services for the occupants (e.g. illumination, thermal comfort, air quality, physical security, sanitation etc). Nature of the buildings are changing, more dynamic work and living environments are preferred so that inhabitants can be actively supported and assisted by the building management system. If one facet is providing more comfort, more safety, the other facet should be providing this services with less money, less energy, and less environmental impact. Thus, intelligent buildings became recently one of the most popular research areas in computer science and information technology.

From the commercial point of view, in 2011, a market research company Frost & Sullivan published their statistical analysis report entitled “European market for building automation systems” [82] and stated that smarter and optimal energy efficiency buildings create a European market for building automation systems of $ 2 billion. This market generated $ 1.769 billion in 2010 and analysts expect that this value will reach to $ 2.123 billion in 2017. [82] believes that energy efficiency is the primary driver of the market for building automation systems.

Another significant report [74] indicates that 81% of the customers may prefer paying for energy

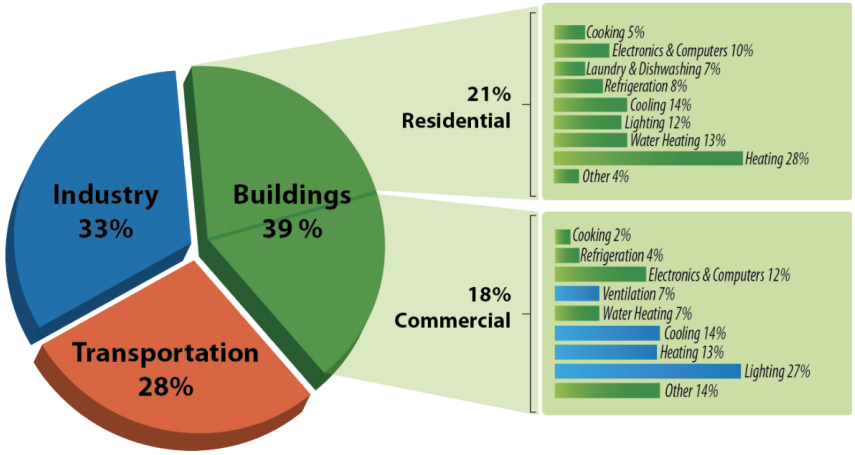


Fig. 1.2 Energy consumption of buildings in USA (2011) [45].

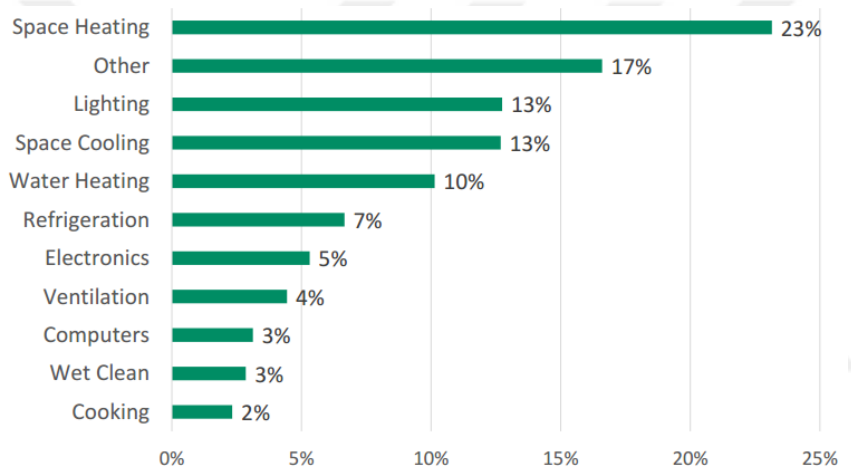


Fig. 1.3 Primary energy consumer services in U.S. (2015) [46]

management equipment if they could save up to 30% on their energy bill for smart energy home applications.

Nowadays, the global market for `Building Energy Management Systems (BEMS)` embraces new technologies and proposes efficient solutions such as `IoT` solutions, cloud-based environment management systems etc. Market reports clearly indicate that the customers are willing control and management systems that are much easier to use and that can provide user-friendly interactive dashboards (graphic-based information). For the economical side, a well-known market research and consulting team Navigant Research predicts global `BEMS` revenue (dominated by software and services) to reach nearly $ 11 billion by 2024, with the North American market representing about 27 percent [150]. All these reports point the necessity of Intelligent / Smart Building solutions, where users can interact easily with numerous different applications of `BEMS`.

Smart Buildings usually rely on a technical infrastructure with wireless sensor devices, organized in a `Wireless Sensor Network (WSN)`. Advantages of `WSN` are easier deployment, low installation cost, and lack of cabling, but the service lifetime of sensor devices depends on their limited battery. Smart Buildings also rely on a software architecture that can support multiple applications, like control of Heating, Ventilation and Air Conditioning (HVAC), control of lighting, security systems, etc.

For Smart Buildings, like for other different application domain (e.g., precision agriculture, health monitoring, air quality monitoring, security) there is a need of more and more complex monitoring architectures with numerous heterogeneous sensors that can support simultaneously various applications. Furthermore, the trend of Sustainable Development raises the issue of energy consumption of the monitoring architecture itself.

In this thesis, we are interested in Monitoring Architectures based on WSN where energy consumption of each sensor device impacts its service lifetime, and so the service lifetime of the whole monitoring system. Our objective is to optimize the energy consumption of sensors while the monitoring architecture is executing multiple applications.

1.1 Multi-Application Monitoring Architecture

A Multi-Application Monitoring Architecture can be considered as a part of a Pervasive Environment. Indeed, Pervasive Environment is a generic name of the field that aims to create a smart environment with embedded and networked computing devices while providing human users with seamless service access [148]. Every technological device we use in our daily life is a part of this environment. It can be either a flexible mobile device such as smartphones, smart watches that are immensely preferred by the majority of people, or a simple platform scale that we may use in our houses. Basically, all these interactions and network are based on the sensing, computing and communication features of the devices. These features provide a rich diversity of applications. These devices connect to worldwide networks and provide secure quick access to services [73, 193].

A Pervasive Environment consists of four fundamental features:

1. **Decentralization:** With the advanced technology in microelectronics, computing power is shifted from the server side to the client workstations. Since the computational capabilities of devices have been extremely improved in the last decade, pervasive computing may distribute the responsibilities and the tasks between the small devices. These devices cooperate to establish a dynamic network.

2. **Diversification:** The popular trend in the technology is moving from super computers to small diversified devices that meet the requirements of a specific user for specific tasks. These devices are small, lightweight and designed for special purposes. They provide only a few, or even just one type of information. Hence the Pervasive Environment requires numerous heterogeneous devices.

3. **Connectivity:** It stands for the devices that are integrated into information world without boundaries. These devices transmit information through a worldwide network. Achieving connectivity is based on the interoperability and the common communication standards, so that the devices may exchange and share information easily. Here the key point is not only the connection of devices but an exchange of application data as well.

4. **Simplicity:** Devices in the Pervasive Environment are designed for specific tasks. Even though these devices have high computational power, they have limited capabilities based on the given responsibility.

The most common subpart of a Pervasive Environment is the Wireless Sensor Network (WSN). Recent technological improvements in low power integrated circuits, microelectronics and wireless communications enable efficient, low cost, low power devices which are capable of local processing and wireless communication. Such devices are named wireless sensor devices (also called sensor nodes in WSN community). Each sensor device has three fundamental functionalities: sensing physical quantities, computation with limited amount of resources, communicate with the other devices in the environment

to transmit the network packets to other sensor devices and/or towards the base station (also called sink node) for data aggregation.

It is made up of four fundamental components: a sensing unit for sensing physical measure, a processing unit (microcontroller) to compute the acquired data (preprocessing task), a transceiver unit (radio module for the wireless communication), and a power unit (battery with limited energy budget). WSN are then able to monitor a given environment, to elaborate the acquired data, and to send the preprocessed data to the base station. With advancements in technology and sensors getting smaller, lighter, smarter, cheaper and more powerful, billions of wireless sensors are being deployed for diverse applications. A taxonomy of WSN applications is illustrated in Figure 1.4.

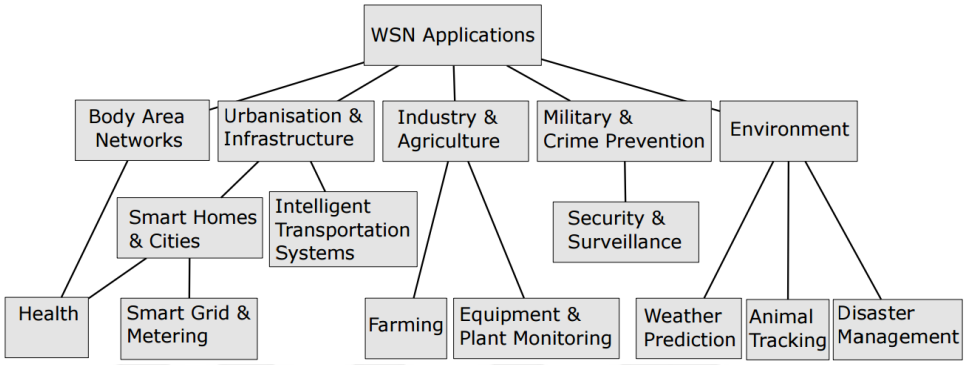


Fig. 1.4 Taxonomy of WSN applications [147].

Multiple wireless sensor devices form a wireless sensor network in which each sensor device senses the environment and communicates (exchanges) the data acquired from the monitored field through wireless links. These devices are spatially distributed and autonomous. All acquired data are gathered at the base station. The base station is the center of each sensor network. It aggregates all acquired data. In most cases, it stands for the central unit for the computation of the data. It is also called a sink device and in most applications, it is wired due to the necessity of power. The geometric properties and spatial relations of sensor devices and the sink device form the entire topology of the network. It indicates how the sensor devices are distributed in the environment. It also represents which device may communicate with which device. This topology is based on the communication ranges of the sensor devices. Thus a topology indicates the physical location of each sensor device and also the path to reach to the base station for each sensor device.

A wireless sensor device has a limited communication range. This communication range stands for the area in which two sensor device may communicate with each other wirelessly. If a sensor device is in this range of another sensor device, then they become neighbors and they may communicate with each other. If a sensor is out of this range of another device then they can not communicate. The communication within the network occurs due to the power produced by the sensor device to send the packet to its neighbor, thus the communication distance so called the range is directly proportional to the energy level of the device. In WSN terminology, transmission of a packet from one device to another is called “Hop”. It corresponds to the number of necessary transmission of a single data packet to reach to the base station (sink device). If the base station is in the communication range of a sensor device, then the sensor device may send the data packet to the base station. We call such type of communication: single hop communication. Otherwise, it should search a reachable neighbor that is located closer to the base station. This case is called multi-hop communication.

Proliferation of wireless devices with wide spectrum of sensing capabilities together with commercial availability has increased the applicability of wireless sensor concepts in practical system designs. Acquiring data from the environment covers numerous physical measures. Some of these physical

measures such as air pollution, water pollution monitoring are extremely important and provide significant information about the environment. WSN applications are widely preferred for hazardous environments such as fire, flood or landslide sensing. Such sensing features are called "Environmental sensing" and covers security applications as well such as military border security. Besides, condition monitoring is also one of the well-known WSN applications. It stands for the problems of structural condition monitoring such as the condition of the buildings, constructions, bridges, supply routes, etc. Health applications are another recent subjects recent subjects in WSN. Health status of especially elders, monitoring daily activities by wearable sensor devices are one of the most popular research benchmarks of WSN. Finally, one of the most important issues from the user perspective is the production performance monitoring, evaluation and improvement, that are achieved through WSNs.

From a more generic point of view, a wireless sensor device is a service provider and it provides a service to the application. The service mentioned here is restricted to the ability of the device. For instance, a temperature device is a service provider that may serve temperature service.

Moreover, a wireless sensor device may include more than one sensing unit (an unit for accelerometer, an unit for temperature etc), thus a wireless sensor device may acquire multiple physical quantities. We call such ability a multi-modality.

A wireless sensor device functions based on the given parameters. The key parameters of a wireless sensor device is the acquisition frequency and the transmission frequency. Acquisition frequency is the sampling rate (the number of samples per second). A higher sample frequency corresponds to a shorter interval between the samples. The high acquisition frequency is obviously more precise and gives much better understanding. The acquisition frequency can vary based on the application and the context. For instance, especially in health domain, the acquisition and transmission frequencies are higher than the environmental applications: 500-1000 Hz for ECG (electrocardiogram), EMG (electromyogram) applications. For daily living activities, usually 100-500 Hz of sampling rates are preferred for accelerometer, gyroscope, GSR (galvanic skin response) measures and for environmental monitoring the sampling frequencies are very low compared to the mentioned ones: 1-100 Hz for habitat monitoring applications.

1.2 Challenges for Multi-Application Monitoring Architecture

A typical pervasive monitoring system like a smart building depends on an infrastructure composed of hundreds of heterogeneous wireless sensor devices. Managing the energy consumption of these devices poses a challenging problem that affects the overall efficiency and usability.

Existing approaches for sensor energy consumption typically assume a single monitoring application and a static configuration for sensor devices during the execution. They do not benefit from real-time sensor configuration. Another critical challenge is to deal with multiple applications running simultaneously, potentially streaming data from the same sensor devices, but with different requirements (acquisition/ transmission frequencies). A related challenge is to consider dynamic application requirements: these requirements may change during the execution of applications, due to user reconfiguration and/or context change detected by some sensors. Besides, since such environments consist of thousands of multi-modal wireless sensor devices, the major challenge is the high energy consumption and short lifetime due to the limited battery life.

Moreover, existing studies on this research field try to balance the quality of service and the sensor or the user preferences. However, they do not mention architecture for multi-application monitoring systems and the dynamic user preferences. This gap forms the following question: considering a sustainable architecture for multi-application monitoring systems, how to configure sensor devices while fulfilling dynamic applications requirements in order to optimize the energy consumption?

1.3 Research Contributions

In this thesis, we are in the pervasive environment research domain and as a target application smart building applications are preferred. We study the monitoring system of a smart building that supports multi-application in a pervasive environment. We focus on a sustainable architecture for multi-application monitoring systems that continuously adapt to application requirements, context and user configuration. From data point of view, a monitoring system stands for a set of applications. These applications exploit various sensor measures in real-time and are declaratively expressed as (service-oriented) continuous queries over sensor data streams. Thus, a multiple application system requires handling several data stream requests with different data acquisition/transmission frequencies for the same wireless sensor device and supporting dynamic requirements of applications (e.g. high transmission frequencies for occupied rooms, lower frequency during night). Since a static configuration can not optimize the energy consumption of the monitoring system with regards to actual application requirements, we work on the interaction between the application requirements and the wireless sensor devices.

A detailed literature review shows that the existing approaches are mostly WSN-level techniques that do not tackle real-time dynamic interactions between application continuous queries and the physical environment, in a multi-application context where sensor devices are multi-modal and requirements are dynamic. Hence, this lack creates a gap between the computing environment and the physical environment, that can be both managed by pervasive applications.

To address this gap and propose a clear solution for multi-application monitoring systems, we design a sustainable multi-application monitoring system architecture for pervasive environments that collects application requirements for sensor data streams and optimizes them into sensor configurations. We propose an approach called ***S**mart- **S**ervice **S**tream-**o**riented **S**ensor **M**anagement (3SoSM)*, an approach to optimize interactions between application requirements and wireless sensor environment in real-time. Our 3SoSM approach performs energy-aware dynamic sensor device configuration to lower energy consumption while fulfilling real-time application requirements. It is based on a real-time sensor configuration that is derived from a scheduling time pattern. This time pattern is build by the acquisition, transmission and reception actions of the sensor device and it creates a periodic agenda for each sensor device. With this approach, we expect to avoid unnecessary data measurements and to promote shorter/compressed data transmission when possible. It is a global schedule formed by the sensor actions. We present a bottom-up process for the constraint propagation and propose an optimization algorithm based on a cost model to find the optimal communication time slots. As an outcome, optimized schedule forms sensor configuration oriented pattern SCO-PATTERN for each device.

Moreover, we introduce our implementation 3SoSM Gateway that supports the optimization process for multiple parameterized subscriptions to the same device to fulfill dynamic application requirements. Our approach is validated by the experiments using the SoCQ Engine and a modified WSNet simulator. Moreover, our approach gives opportunities to use real testbed data and simulation data which is not so common in the pervasive environment research domain. Impacts of our approach on energy consumption and on lifetime are presented and discussed. Smart building applications are our target application, however, this approach can be applicable to other pervasive environments as well.

To enhance the energy consumption, we propose an optimization process for the network communication and we present an optimization process to finalize the schedule time pattern. The aim of the optimization process is to search the optimal communication slots with which the sensor devices consume less energy than the other cases. During the optimization process, we pay attention to the constraints of each sensor device due to the given application requirements. Due to the constraint based behavior and to the tree structured topology of the network, the proposed optimization process is a top-down process, starts by propagating transmission and reception constraints based on latencies in a bottom-up process. Then, the algorithm tries to find the most energy efficient communication slots, and finally propagates those

choices to lower-layers and continues the algorithm in a top-down process in the network topology. The optimization function of that process is to minimize the number of communication slots (RX slots) in order to minimize the consumed energy. The proposed optimization algorithm is a sort of *greedy heuristic algorithm* employed to determine the optimal slots for sensor devices. The core concept of greedy algorithm is to perform a short-sighted action in each step. It starts from the reception part of the schedule table of sink device and tries to group the receptions from the same sensor device. In each step, an optimal communication slot is found and the step is repeated until all the reception data is matched with a slot. The proposed optimization process is validated by using a constraint modelling approach and tested on a constraint modelling tool. At the final step, we conducted experiments with a wireless sensor network simulator to perform a set of scenarios. The results of each scenario is introduced and discussed. At the end of the experiments, the energy enhancement of different types of sensor devices is presented.

1.4 Document Organization

Chapter 1 gives a detailed presentation of the research field especially multiple application monitoring architecture. The chapter also talks about the principal challenges of monitoring applications and focuses on smart building applications and monitoring systems. Besides, main research contributions are introduced.

In Chapter 2, we present that our research domain requires a multidisciplinary research and introduce the research domains that we touch upon. To present our work and our research field, we introduce definitions of important terms and key words in detail. Since then, our research motivation is given with the major challenges in the domain. Several existing energy enhancement approaches are presented and compared with our approach. Since our research domain covers multiple research areas related to the sensor devices, all these multidisciplinary researches are mentioned and the closest studies to ours are presented in this chapter. At the end of the chapter, a detailed conclusion is given in which analysed studies are compared with our approach based on different criteria such as pervasive environment, sensor data base principles etc. Multiple detailed tables are presented in order to compare the studies on sensor network processing and smart building management systems and to exhibit positive and negative points. These points are summarized and discussed at the end of this chapter.

Chapter 3 starts with the description of a declarative monitoring architecture and present a framework for managing pervasive environment systems. Besides, a detailed presentation of pervasive environment management system is given, followed by a framework to manage such systems. Moreover, the crucial item of such systems, application requirements, is defined. The core issue of our thesis, dynamic sensor configuration of a wireless device is also presented in this chapter. Then, the problematic that we focus is introduced. The given problematic is explained with several case studies and the importance of this work is given in this section.

The core of the thesis is presented in Chapter 4. Firstly query requirements and subscription requests are defined, and sensor configuration-oriented patterns are presented. These are the crucial items of our approach. The first version of our approach called `Device-Centered 3SoSM` is then presented with the `GeNoMe` process that generates a single sensor device configuration. Several case studies are given to clarify the usage of `Device-Centered 3SoSM` approach, especially the `GeNoMe` process. Furthermore, the `Network-Aware 3SoSM` approach is described. The network topology preferred in the study is explained, and we present the energy-aware optimization process. The generation of the final version of SCO-PATTERN is then described. All these descriptions are illustrated with various relevant figures following a running example. Since the optimization process requires an assessment, here we present the work we realized by using constraint modelling tool in order to validate our optimization algorithm.

In Chapter 5, we present our prototype that is developed based on the monitoring architecture for pervasive environments. The platform of our prototype has three main components: continuous

query engine SoCQ Engine, wireless sensor network simulator WSNet simulator and 3SoSM Gateway. A continuous query engine stands for the execution of queries over the relational pervasive environment, i.e. one-shot and continuous queries over XD-relations and service discovery queries. A wireless sensor network simulator refers to an environment on which protocols, schemes, models can be evaluated in a very large scale. WSNs simulators allow users to isolate different factors by tuning configurable parameters. Finally, to complete the overall architecture, 3SoSM Gateway is responsible for managing the interactions and bidirectional communication between the PEMS and the WSN.

In Chapter 6, we present our experiments, test scenarios and we introduce the results obtained by these experiments. Description of the experimental environment has already been presented in Chapter 5. To apply our approach and implementation of 3SoSM, we create various test scenarios targeting smart building applications. Each scenario is performed during one day (1440min). To evaluate our approach, we compare on scenarios with 2 types of architecture: Architecture with basic duty-cycle WSN devices, requiring a static configuration, architecture with 3SoSM approach where WSN devices can be dynamically configured with SCO-PATTERN generated by the 3SoSM Gateway. In the first part of the experimental part of our study, we describe our test scenarios. Then, we present the applications for our test scenarios and the application requirements (parameters defined in the application). We apply the Device-Centered 3SoSM approach and its implementation on these applications. After performing these scenarios, we present the obtained results and discuss the consequences of the Device-Centered 3SoSM approach. Afterwards, we again start with presenting the applications which are quite similar to the first ones (except the application requirements are more complex). We then apply the Network-Aware 3SoSM approach and its implementation on these applications. At the end, we expose the acquired results and debate the results.

Finally, Chapter 7 concludes our thesis. The contributions are summarized. The benefits of our novel approach are introduced. Concrete usage of our approach is presented. Moreover, general discussion and the future directions are given in this chapter. Our publications related to this thesis are listed in Appendix A. The journal article that gives an overview of our approach is also fully included in Appendix B.

Chapter 2

State of the Art

In Chapter 1, we introduced our research field and our target application that we focus on. As a target, we chose smart building applications that cover recent technologies like Internet of Things (IoT), pervasive environments and ubiquitous computing. Especially IoT provides a new breed of smart buildings: enabling operational systems that deliver more accurate and useful information for improving operations and providing the best experiences for tenants [81, 195].

A basic smart building application presents several useful features and opportunities to the users such as personalization, mobility, occupant comfort etc. To provide such features, the system should include various heterogeneous physical equipment with diverse technologies. Hence a basic smart building application consists of physical devices, wireless sensors, actuators, middlewares, applications, etc. These components and platforms compose the overall infrastructure of monitoring and building management systems [93].

A pervasive environment such as smart buildings requires a multidisciplinary research. It covers not only wireless sensor issues but also, sensor network processing tools, building management platforms and technologies. If one facet of the smart building monitoring system is the wireless sensor devices, the other facet is the management of the system and the data streams generated by the deployed equipment in the environment.

Among the given research domains, Wireless Sensor Networks (WSN) and the energy challenge on the wireless sensor devices dominate the literature [8, 20]. Since a smart building includes a set of wireless sensor devices deployed in the building, WSN imports its major challenge: high energy consumption and short lifetime due to the limited battery life (limited energy). There exists considerable literature dedicated to methods for reducing the energy consumption and extending the battery lifetime for wireless devices and platforms (embedded hardware systems included) [25, 59, 108]. These approaches, however, mostly belong to the WSN paradigm where implemented methods are component-based approaches. Energy healing techniques are commonly restricted to the hardware specifications [12]. Moreover, impacts of using these methods on the application performance are not considered adequately. Any possible remedy for the battery life challenge should be more generic, application and context aware, especially independent from the hardware constraints.

From the energy saving methods, putting the whole device or its various subcomponents into low energy state, also called the sleep state, is a long known technique to elongate the battery lifetime of small devices and sensors [12]. Especially, WSN and IoT paradigms incorporate a custom system design where known application behavior is made energy efficient by using sleep schedule on the wireless communication [72]. However, smart building systems depict a totally distinct case where the interaction between the system and the end user, in this thesis we propose a way to fulfill parameterized multiple application in real time. In that respect, to combat with the limited energy problem in smart building applications, we adopt a holistic approach where both the wireless sensor devices of the system and the

application requirements defined by the user are taken into account.

Along with WSN, there are other research domains that should be studied to understand better the pervasive environment concept: energy challenge in WSN, sensor network query processing, sensor-based Smart Building Management Systems (SBMS).

- Wireless Sensor technology brings in some nice features, it also enforces extra constraints in terms of form factor and weight. These constraints, in turn, heavily limit the computational properties and the battery lifetime of the sensor device. Limited processing ability and low data rate of sensor devices cannot guarantee high performance in some scenarios, especially for real-time applications. Short communication range can cause energy waste and network inefficiency since multi-hop communications are always required for data transport between the sensor devices and the base station so-called sink device. At this point energy consumption of a sensor device becomes crucial. This serious concern determines the lifespan of the sensor device [183]. It may be cost-prohibitive to replace exhausted batteries or even impossible in hostile environments. For this reason, energy efficiency for wireless devices is a central research theme. Several studies aim at minimizing the energy consumption, saving energy in order to decrease the total energy consumption of the system and to increase the battery lifetime of the device.

- The second part of the literature survey is the sensor management with sensor network query processing. With the recent advances in WSN research domain, WSN applications provide interaction with the environment at very high spatial and temporal densities. Such networks potentially enable observation over a large area in the real environment so called monitoring applications [169]. On the other hand, programming wireless sensor devices require specialized knowledge with some constraints such as a tight limit on code size and cumbersome debugging process. For instance, implementing a simple data collection application may require thousands of lines of code in an embedded programming language [79]. As a solution, [109, 192] propose that WSN devices can be programmed with considerably less effort with database paradigms. This approach opens a new research domain: Sensor Network Query Processor (SNQP).

- The last part inspects the studies on Smart Building Management Systems (SBMS). A SBMS is the name of the system that controls and manages sensor-based smart building application. A basic building management and automation/control systems rely on hundreds of heterogeneous sensor and also actuator devices that are located at various locations in the building [10, 37, 71]. Such monitoring systems with wireless devices (sensor and actuator) open a new branch in the WSN domain that is entitled Wireless Sensor and Actuator Networks (WSAN) [29, 152, 189].

While inspecting existing studies in the literature, for each study, we analyse which feature the study brings newly, discuss which fundamental abilities the study proposes. At the end, we categorize these studies for the SNQPs and SBMSs based on the key functionalities of such applications [125]:

1. Multi-application stands for the systems where more than one applications can request the same service with different parameters.

2. Context-awareness represents that user can define a set of conditions and the system can re-configure itself during the execution based on the updated conditions.

3. Energy-awareness indicates that the SNQP takes into account the energy consumption of the system while creating an execution plan.

4. Dynamic Application Requirements represents that the application needs for sensor data may vary during the execution.

5. Real-time sensor configuration indicates that the sensor device is configured during the execution. In this case, we consider the sensor configuration as dynamic acquisition and transmission settings. Even though duty cycling can be a way of configuration in WSN approaches, we take into account adaptable sensor adjustments based on application requirements.

6. Sensor Network Query Processor indicates the SNQP tool used in the study to manage the sensor devices in the pervasive environment.

7. Experimentation represents the experimental platform type used in the study (either physical real WSN devices or simulation environment).

At the end of the each section, we give a brief conclusion about the presented studies and make a close comparison with our study. We discuss our pros and cons compared to these studies.

2.1 Definitions of Key Terms

In Chapter 1, we have already presented the notions of the major terms such as pervasive environment and wireless sensor networks. However, before introducing the existing studies and presenting our approach, definitions, key terms used in this thesis should be introduced exhaustively.

2.1.1 Wireless Sensor Networks

When we mention pervasive environments, the underlying physical network infrastructures are commonly comprised of heterogeneous wireless sensor networks. A Wireless Sensor Network (WSN) is a set of wireless sensor devices that are connected and form a network. A wireless sensor device provides physical measures and tries to transmit the acquired data to the base station. It uses a routing protocol to define the path to reach the base station. It transmits the acquired data by a media access control (MAC) protocol and the radio communication protocol. Main abilities of such devices are already introduced.

The major problem of a WSN is the limited energy budget and the lifetime of the sensor device due to the energy consumption. There are many studies in the literature targeting different parts of a WSN: energy saving methods, improvement on routing protocols, efficient MAC (Medium Access Control) and communication protocols, security aspects, clustering techniques, localization techniques etc. Each listed parts have their own challenges. In our thesis, since we work on a pervasive environment equipped with hundreds of wireless sensor devices, we face the energy challenge of these devices.

Example 2.1.1 *A broad range of applications such as precision agriculture, environment monitoring, intrusion detection, target tracking etc. are facilitated through networking these sensor devices. Humidity, accelerometer and temperature are the most common physical measures acquired by the sensor devices. Especially precision agricultural applications are one of the most required in the terms of temperature, humidity (soil, leaf, ambient) and the wind (speed and direction).Table 2.1 gives a brief summary of WSN applications.*

2.1.2 Wireless Sensor and Actuator Networks

Basically, a Wireless Sensor and Actuator Network (WSAN) is a special type of WSN. It represents a group of sensors and actuators that are geographically distributed and interconnected by wireless networks. Wireless sensor devices acquire and transmit sensor data about the state of the physical world and the actuators react to this information by performing appropriate actions (decision can be taken on sensor device via computing feature (decentralized version) or on the base station (centralized

Table 2.1 WSN Applications based on different sensor types.

Sensor	Application
Temperature, Humidity, Wind, Pressure	Precision Agriculture [63, 88, 102, 119, 168, 196]
Accelerometer, Presence, Vibration	Motion detection, military applications [18, 20, 34]
CO2, O2 Emission	Respiration in humans and animals, pollution [6, 84, 129]
Infrared, Luminocity, Ultraviolet	Detect human presence, agriculture [2, 33, 130, 154, 178]
EEG, EMG, ECG, GSR	Health care, e-health applications [35, 139–141, 151]

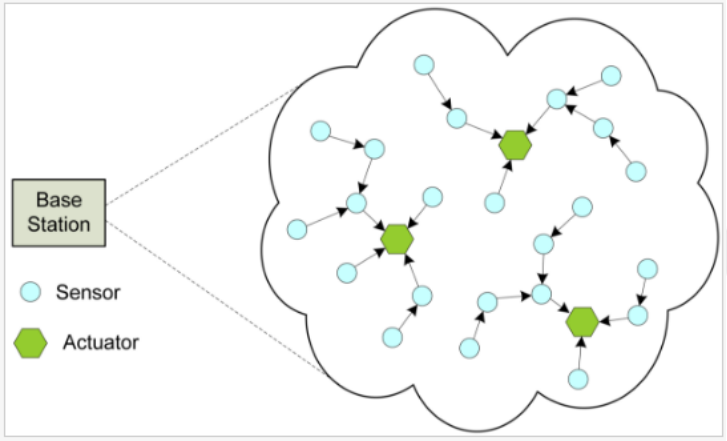


Fig. 2.1 An Example of Wireless Sensor and Actuator Network [187].

version)) [105, 177, 187]. An example of WSAN is illustrated in Figure 2.1. As seen from the illustrated figure, wireless sensor devices form a cluster-like structure to interact with the actuator. An actuator requires only a control signal and a source of energy. Hence, in such applications, actuators can be managed either by the sensor devices (on board computation is required) or by the base station where all acquired data are aggregated. For instance, a shutter window of a room can be controlled by an actuator and it can be managed either by the sensor devices located in that room or by the base station where al data are computed.

Example 2.1.2 *WSAN is a hybrid network. Like WSN, a WSAN also covers thousands of low-energy tiny sensors, but it includes actuator devices as well to realize appropriate action. Such networks mostly is preferred in homeland security applications. Since wireless sensor devices and actuators are in the same network, actuators apply/execute a physical action based on the data acquired by the sensor devices in the same environment. Smart building applications are a good example for WSAN applications (e.g. HVAC (Heating, ventilation, and air conditioning), lighting systems). Especially in a building automation, thousands of sensor or actuator devices are deployed throughout the building. The sensors acquire information about the environment and the actuators interact with the environment (e.g. controlling lights, heating, or door access).*

2.1.3 Monitoring Applications

Monitoring applications are very common in pervasive environment domain. The rapid evolution of wireless sensor network technology allows to monitor a variety of physical parameters and provides this data to relevant users. Thousands of heterogeneous diverse wireless sensor devices may be deployed to monitor a multitude of natural and man-made phenomena, i.e., habitat monitoring, wildlife monitoring,

patient monitoring, industrial process monitoring and control, battlefield surveillance, traffic control, and home automation etc.

The occupancy information is commonly preferred in monitoring applications. The majority of these studies analyse occupancy information in order to estimate the occupancy level owing to the fact that the ability to estimate the indoor occupancy level enables several useful services related to comfort, security, energy saving etc [179]. There are two primary methods to detect the occupancy of an environment: Firstly, the direct information of presence acquired from video cameras with dedicated people counting methods, optical tripwires, radio frequency identification (RFID). Secondly the indirect information that are generated in the environment by people presence such as wireless sensors for passive infra-red (PIR) motion detectors, the environmental parameters such as temperature, humidity etc, microphones. [178] proposes a methodology for the real-time evaluation of indoor occupancy, crowd detection through indirect environmental measurements (temperature, humidity etc.).

2.1.4 Context-Awareness

Context-awareness is also known as context-aware computing. It is one of the core feature of ubiquitous and pervasive computing systems. It stands for the systems that can sense their physical environment, and adapt dynamically their behavior accordingly. Context-aware systems are one of the fundamental and required component of any pervasive computing environment. Dey, a well-known researcher in this field, defines the concept of context as follows: "Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves" [4].

Besides, context information derives from the context and from the acquired measures. For instance, the data acquired by a GPS sensor device can be considered as raw sensor data. Once we put the GPS sensor data in such a way that it represents a geographical location, we call it context information.

Dey, based on many studies in this field, defines the term context-awareness as follows: "A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task" [4, 131].

2.1.5 Reconfigurable Sensor Device

We have already presented the sensor device. A sensor device includes several subcomponents: battery for energy, microcontroller for the computation, radio module for the wireless communication, sensing unit for the sampling physical measure from the environment. Fundamental abilities of each of these components have also been introduced.

Apart from that, a sensor device has a hardware and a software side. The listed components above form the hardware side of a wireless sensor device. On the other hand, a software side consists of an operating system such as TinyOS, Contiki etc [50, 96]. Such operating systems (OS) are specially designed for wireless sensor networks and they are typically less complex than general-purpose operating systems. They are mostly considered as embedded systems.

A wireless sensor device can be programmed using embedded programming languages (e.g. NesC [60]). Chosen programming model should be supported by the running OS. The programming model has a significant impact on the application development. In the literature, there are two highly adopted programming models: event driven programming and multi-threaded programming. Multi-threading is the application development model most familiar to a programmer, however, it is not considered well suited for resource constraint devices such as sensor devices. Event driven programming is considered more useful for computing devices equipped with limited resource.

Since most of these operating systems and programming languages are open source and free to use, it is possible to develop numerous embedded application running on the sensor device. The configuration of

a wireless sensor device defines the execution of a sensor device in terms of acquisition and transmission frequencies. Due to the problematic pointed in this thesis, we benefit from the reconfigurability of these sensor devices. Based on the existence of reconfigurable sensor devices [133], a sensor device may behave differently in real-time, i.e. it may change its acquisition and/or transmission frequencies dynamically. This configuration can be achieved on the software side which manages the hardware side of the sensor. For instance, apart from the computation, a basic wireless sensor device may execute the following actions: data acquisition, data transmission and data reception. The software on the sensor device decides when to acquire a physical measure, when to transmit the data over the channel, when to listen the radio channel to receive a packet. Thus, the hardware for instance a radio component goes into listening mode based on the command generated by the software. In order to configure a sensor device during the execution, a configuration packet should be sent to the relevant sensor so that it may configure itself and follow the new configuration.

2.1.6 Data Stream

Recently a new type of data-intensive computation mode has been widely recognized: applications where the data is not processed as persistent static relations but rather as transient dynamic data streams. These applications include financial analysis, network monitoring, telecommunications data management, traffic data monitoring, web applications, manufacturing, sensor networks etc. The data streams continuously arrive in multiple, rapid, time-varying, unpredictable and unbounded manners. Thus, a data stream is a real-time, continuous, ordered (explicitly by timestamp or implicitly by arrival time) sequence of items [166].

Traditional database systems expect all data to be managed within some form of persistent data sets. For many recent applications, where the data is changing constantly (often exclusively through insertions of new elements), the concept of a continuous data stream is more appropriate than a static data set. Several applications generate data streams naturally as opposed to data sets. Especially for recent applications, the concept of a data stream is more appropriate than a static traditional data set. In our case data stream is the continuous transmission of acquired physical measure.

2.1.7 Continuous Query

A continuous query (CQ) is a query that is issued once over a database DB, and then logically runs continuously over the data in DB until CQ is terminated. CQ lets users get new results from DB without having to issue the same query repeatedly. Continuous queries are best understood in contrast to traditional SQL queries over DB that run once to completion over the current data in DB [14].

It is a sort of a query which is re-evaluated continuously, produce a stream of answers over time, reflecting the evolution of the target data stream. Compared to traditional one-time queries which are run once to completion over the current data sets), continuous queries are queries that are issued once and then logically run continuously over the database.

Continuous querying allows you to subscribe to server-side events using SQL-type query filtering. With CQ, the client sends a query to the server side for execution and receives the events that satisfy the criteria. For example, in a region storing stock market trade orders, you can retrieve all orders over a certain price by running a CQ with a query like this:

$$SELECT * FROM /tradeOrder\ t\ WHERE\ t.price > 100.00$$

When the CQ is running, the server sends the client all new events that affect the results of the query. On the client side, listeners programmed by you receive and process incoming events. For this example

Table 2.2 Example SoCQ queries for a Smart Building.

CREATE RELATION TemperatureServices (ServiceID SERVICE, Location STRING, Temperature NUMBER VIRTUAL) USING (getTemperature[ServiceId]():(Temperature), temperature[ServiceId]():(Temperature) STREAMING) AS DISCOVER SERVICES PROVIDING PROPERTY Location STRING, METHOD getTemperature ( ) : (NUMBER), STREAM temperature ( ) : (NUMBER)	SELECT * ONCE FROM TemperatureServices WHERE Location = "501.340" USING getTemperature;	SELECT * STREAMING UPON insertion FROM TemperatureServices USING temperature[1];

query on /tradeOrder, you might program a listener to push events to a GUI where higher-priced orders are displayed. CQ event delivery uses the client/server subscription framework.

More complex examples of continuous queries are presented in Table 2.2. These queries are related to the smart building applications and have a SQL-Like syntax:

These queries are example of continuous queries used in Service oriented Continuous Query Engine SoCQ. It covers the classical continuous queries and also the relational queries. The first query is called as a discovery query. The query discovers the environment and provides a set of existing/available services in the environment. It represents the selection of a list of available services that match some criteria. In the given query, it explores the service providers in the environment that may serve temperature service.

The second query makes an exploration of sensor devices that can provide temperature measures from the given location. The last example of the continuous query is the stream query that starts data flow (temperature measure) from the temperature sensor devices located at given location.

2.2 Energy Challenge in WSN

Since sensor-based smart building systems are mainly derived from WSN systems, smart building technology and management systems are inspired from WSN approaches and hardware used in these systems. Thus, in the literature, the main source of knowledge on the energy consumption problem of wireless sensor devices of smart building applications comes from the WSN energy saving methods.

[12] gathers and classifies existing studies on energy consumption of wireless sensor devices in the literature and gives a brief presentation of each approach. The authors present that it is possible to classify existing approaches into three different subsets: energy saving by the usage of duty cycling, data-driven techniques and mobility-based methods. The taxonomy of energy saving approaches in WSN is presented in Figure 2.2.

2.2.1 Duty cycling approach

The common technique to conserve energy is using a duty cycling method. Duty cycling concerns not only sleep scheduling mechanism for power management but also topology control. Topology control and location-driven systems are used in multi-hop networks especially for updating the neighbor list of each sensor device in the environment. It can be considered as a type of periodical self-organization. Sleep scheduling for power management is one part of the duty-cycle approach. A number of solutions across multiple layers like energy-efficient MAC protocols [16, 91, 156, 171], communication strategies, operating systems and energy saving routing mechanisms [24] have been proposed to provide low-cost, high-quality and energy-efficient wireless access services [118, 126, 127].

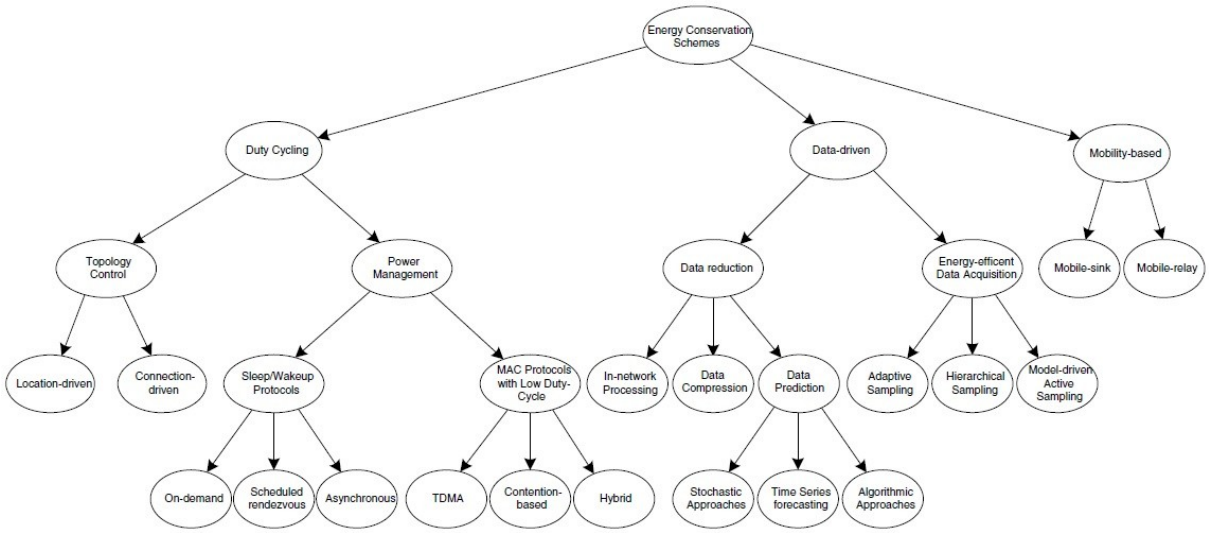


Fig. 2.2 Taxonomy of Energy Saving Approaches in WSN research domain [12].

In terms of sleep scheduling, [186] proposes scheduling algorithms to remove unnecessary listening cost and to reduce energy consumption for the state switching. The authors benefit from the clock synchronization, hence, each sensor device in the topology is required to wake up twice (one for receiving data packets from its lower-layer neighbors and one for transmitting data packets to its upper layer neighbors.) However, the scheduling mechanism is kept static during the execution and with this static behavior, any real-time change (so-called context awareness) in the environment cannot be handled.

2.2.2 Data-driven approach

The second common approach in the literature is based on the captured data (raw data). From this acquired/measured data, data prediction technique is one of the well-known data-driven approaches to predict the incoming data and to reduce the number of the data packet in the network. It builds a model of the sensed phenomenon. Application requests can be answered directly by sink device by using the model instead of the actually sensed data without requiring any unnecessary communication [2]. Another data-based energy saving method is the adaptive sampling technique. This technique adopts a holistic approach and uses model-based sampling. Such approaches aim to reduce the number of samples by exploiting spatiotemporal correlations between data [87]. Moreover, [146] proposes a platform for context continuous sensing and proposes a prediction approach through native semantic abstraction method.

Additionally, advanced onboard signal processing techniques are highly common for energy issues in WSN. The main objective of using complex onboard algorithms for data processing is the high energy cost of packet transmission (radio communication). Especially for special applications such as Wireless Body Area Network (WBAN) applications, data transmission frequency is higher than other application frequencies (e.g. 500 acquisitions in a second). [111] proposes compressed sensing approach for WBAN applications that process ECG, EMG signals. This technique intends to reduce the number of sent packet. From the same perspective, [113] proposes that complex signal processing techniques should be transformed into power efficient techniques to achieve more efficient signal processing in WBAN applications. [41] also presents that packet transfer should be limited as much as possible. So, the authors propose a novel distributed approach based on fuzzy numbers and weighted average operators for data aggregation and for data communication reduction.

Besides, data gathering approach is a well-known technique to enhance the energy consumption of the wireless sensor devices. For all the applications, data gathering is one of the primary operations

carried out in WSNs, where a base station collects all the data generated from each sensor through wireless communications. Data gathering is mainly for estimating network size, determining average system load, processing user queries etc. In general, data gathering can be further classified as data collection with aggregation and data collection without aggregation, referred to as data aggregation and data collection respectively. In data aggregation, specific aggregation functions are employed during the data gathering process, e.g., MAX, MIN, SUM, AVERAGE etc. In data collection, all the raw data produced at each node is gathered to the sink (base station) without any aggregation function. The studies commonly adopt data gathering algorithms based on a set of parameters such as network density, residual energy, the position of the sensor device within the network, network topology and workload etc [1, 5, 36, 75, 155].

Based on the literature, the data-driven techniques appear to be the most preferred approach. The researchers claim that the unnecessary communication cost can be reduced by the context awareness which is based on the acquired data [2]. However, these approaches focus on the energy saving of the deployed equipment and commonly ignore the users' needs from the system.

2.2.3 Mobility based approach

Another energy saving technique in WSN is using the mobility-based approach. In this approach, either the base station (sink device) or the sensor device has a mobility feature. Here, mobility can be considered as an alternative solution for energy-efficient data collection. The studies on this approach present that using a mobile sink device (i.e. the sink device travels between the sensor devices and aggregates acquired data) can achieve a network lifetime longer than using a sink device with static location. [181].

2.2.4 Discussion

As seen from these existing approaches in the literature, several energy saving methods are proposed, however, these approaches are mostly low-level techniques and deal only with high energy consumption of the physical wireless sensor devices. They do not consider the users' preferences (so-called application requirements) and the context information. Hence, they do not handle the real-time dynamic interaction with the users and the WSN environment. In addition, presented approaches are commonly fitted to a single application where in our case, a monitoring system of a smart building should serve various user demands and applications at the same time (so-called multi-application).

Besides these drawbacks, these approaches inspire us to use the measured data more efficiently. From the presented energy saving methods in the WSN research area, we adopt the fundamental principles of duty cycling and data-driven techniques. Putting the sensor device (or a specific component such as radio component) in sleep mode between two consecutive sensor actions will surely save energy.

Moreover, the existing studies in the literature adopt a static configuration for the wireless sensor devices. In such applications, adopted duty cycling method is being executed without any update, application parameters are not touched along the execution period. However, in our study, we focus on handling multiple applications and their dynamic application requirements. We aim to fulfill real-time application requirements with consuming energy as low as possible.

Hence, to achieve this aim, we require dynamic sensor configuration in real time. Thus, we benefit from the existence of dynamic re-configurable sensor devices. Besides, we adopt the topology and data-based approaches to create our schedule for each sensor device to have a global view of the network. With the inspiration of the data based approach, we create irregular sensor actions schedules.

2.3 Sensor Management with Sensor Network Query Processing

By definition, a Sensor Network Query Processing (SNQP also called sensorDB) can be expressed as a user-friendly interface for programming and running applications which translates instructions from a declarative programming language with high-level instructions to low-level instructions understood by the operating system [51, 61].

Since many researches have demonstrated that communication is the principle source of power consumption for wireless sensor devices, energy saving became one of the early objectives of developing a SNQP. A basic SNQP provides database-like access to sensor data through a simple querying interface. Besides, using queries and querying interfaces allow developers and the users to specify required service (e.g. temperature service) from the WSN environment without needing to know details such as how to contact the relevant sensing devices on sensor devices, how to deploy application logic, how to manage its execution and how to transmit results back to the user. The main functionality of such processors is to execute the continuous queries and to serve the sensor data to the relevant application. Moreover, the application requests can be processed, sensors can be managed by the SNQPs. Thus, SNQP is another important side of our multidisciplinary research domain.

Traditional database management systems expect all data to be managed within some form of persistent data sets. For many recent applications, the concept of a continuous data stream is more appropriate than a data set. Several applications naturally generate data streams as opposed to data sets: financial tickers, performance measurements in network monitoring and traffic management, web tracking and personalization, data feeds from sensor applications. Hence the researchers need stream-oriented processing to respond users' queries [15]. The engine that processes the Continuous Queries (CQ) is called Continuous Query Engine. With CQ, the client sends a query to the server-side for execution and receives the events that satisfy the criteria. When the CQ is running, the server sends the client all new events that affect the results of the query. On the client side, the clients receive and process incoming events. Clients can execute any number of CQs, with each CQ assigned any number of clients.

Early studies in this research domain focus on the data management techniques. The common adopted approach is to have an efficient data acquisition in order to serve users' demands. Hence preliminary studies result with basic continuous query engines such as TINA [162], COUGAR [22, 192], TinyDB [61, 109], MauveDB [42], PAQ [176], FunctionDB [174], Presto [99].

TinyDB is one of the preliminary sensor network query engines that inspires researchers to work on this domain. It supports events as a mechanism for initiating data collection. Such type of querying allows sensor devices transmitting the data only when the described event occurs. The authors present an energy-aware approach by applying event-based querying. Hence, the proposed system provides network longevity and enhancement of network traffic. For the implementation of TinyDB, TinyOS [96] as an operating system and NesC (network embedded systems C) [60] as a programming language are preferred.

Cougar [192] in terms of perspective is very similar to TinyDB. Unlikely to TinyDB, Cougar assumes that the central unit where the queries are parsed and optimized has the knowledge of the locations of the sensor devices in the environment.

Even though these studies are the primary tools, there is a big gap between the database and the data provider services. Thus, [67] points the lack of interaction between databases, data streams and services. That study overviews current trends in pervasive environment management and its complementary research domain, database and query processing systems. The authors target to bridge the gap between pervasive computing and database systems by studying pervasive computing from a data-centric point of view.

Several frameworks have been proposed to further enhance communication costs. [19] introduces a framework Selecting Representatives in a Sensor Network (SERENE), to provide models with better quality for sensor networks and to reduce energy consumption during the data aggregation. The authors present a clustering technique to select a subset of representative sensors, which will be queried

instead of the whole network. This approach allows to reduce the communication and the computation costs and to balance the energy consumption among sensor devices. This framework exploits data mining techniques as well to find the best quality representatives of all sensor devices in the environment.

[175] proposes an energy-efficient framework `Similarity-based Adaptive Framework (SAF)` for sensor network querying. Presented approach provides a mechanism to detect data similarities between sensor devices and organizes sensor devices in a clustered way to reduce additional communication cost. This mechanism is highly based on exploiting time series models, data similarity between sensors and the prediction measures.

[158] summarizes existing technologies (middlewares, frameworks, tools etc.) for managing network of heterogeneous sensor devices. It also presents the possible research directions and benchmarks of this domain: data-centric view of the pervasive systems; a homogeneous high-level interface to heterogeneous devices; support for highly dynamic networks; Plug and Play device addition mechanism. These major issues of the pervasive environment have been already taken into consideration by other studies like [67, 68]. [68] presents a language and infrastructure for data management for pervasive systems. Proposed middleware is confronted to the existing middlewares in the literature. Besides the middleware, a declarative SQL-like language has been introduced to provide a flexible interface that allows users to control each physical sensor device while masking the heterogeneity at the data level. The authors present an infrastructure called Perla for data management in pervasive systems, mainly oriented to monitoring applications, but also suitable as a support substratum for the deployment of autonomic systems.

[101] presents a framework, called `Acquisition Cost-Aware Query Adaptation (ACQUA)`, for acquisition cost aware continuous query processing. It explores an approach to enhance the energy consumption by reducing the volume of sensor data injected into the network. Even though it is implemented for smartphones, it covers ubiquitous computing issues as well. The main objective of this framework is to modify the order and the segments of data streams, to calculate and to update the acquisition cost function. The major advantage of this framework is the context-awareness and predicting activity accurately from diverse sensor streams (adaptive to dynamic changes). `ACQUA` uses a stream-oriented query model in which complex stream queries are expressed as an arbitrary conjunction or disjunction. With proposed approach, the algorithm assumes a specific energy cost associated with the acquisition of each individual sensor stream tuple.

[133] introduces the `Context Aware Sensor Configuration Model (CASCoM)` approach that allows non-IT experts to configure `IoT` middleware efficiently and effectively. The semantic technologies allow to capture user requirements and configure the sensor devices and data processing components accordingly by handling the low-level technical details without overwhelming the users. The proposed model has the capability to advise users on future sensor deployments in situations where user requirements cannot be satisfied using existing resources. `CASCoM` also allows to discover additional context information. Users are not required to know any underlying technical details. Instead, they are offered a user interface where they may select the additional context.

For the energy side, authors propose a cost model that calculates the cost of data acquisition based on sensor devices and data processing components combined. `CASCoM` selects the most optimized solution by default, though it allows advanced customization through context prioritization. Moreover, that model is integrated into an `IoT` middleware called `Global Sensor Networks (GSN)`. `CASCoM` has significantly increased the usability and capability of the `GSN` middleware. Hence, the study shows that it is possible to offer a sophisticated configuration model to support non-IT experts. Semantic technologies are used extensively to support this model. Ontologies are used to model sensor descriptions and data processing component descriptions.

Moreover, the authors of [131] introduce detailed analysis and evaluation of context-aware computing research efforts to understand how the challenges in the field have been tackled in desktop, web, mobile,

sensor networks, and pervasive computing paradigms. The authors also discuss the importance of context awareness in IoT. Besides, possible future challenges in this domain are pointed: context discovery, common standard interfaces for interoperability of techniques, optimal sensor selection for sensor-as-a-service, security and privacy. Hence, the paper presents various existing solutions (systems, middlewares, applications, techniques, and models) to solve different challenges in context-aware computing.

[26] describes the global view of network query processing and searches how to consume limited resources available in the network in an intelligent way without penalizing the application requirements. The authors claim that the essence of network query processing is to allow wireless sensor devices to execute queries and aggregation operators onboard. Thus, sensor data can be filtered and volume of data injected into the network can be reduced. For the issues related to the sensor networks and in-network aggregation, the authors propose an adaptive in-network aggregation operator, called `Adaptive aggregation algorithm for sensor networks (ADAGA)` for query processing for sensor devices. The main functionality of `ADAGA` [27] is to regulate/adjust sensor activities based on energy levels and memory usage of sensor devices. `ADAGA` aims to maximize sensor lifetime and the accuracy of query results. Moreover, authors present a query language for sensor networks, called `Sensor Network Query Language SNQL` that supports parameterized queries. Briefly, the authors present a novel adaptive query processing approach for `WSN` with notions of quality of query and novelty detection. As we propose in our own study, `ADAGA` is capable of processing query parameters like time window, sending interval, sensing interval, data window. `ADAGA` is more flexible compared to `TinyDB` and the other major `SNQPs`. It may be considered as the second generation of `SNQPs`. Even though it allows changing the send and sense periodicities dynamically, it does not allow a user to update or add new application requirements. The dynamic change in the sensor parameters is totally based on the context. Moreover, `ADAGA` do not support multiple application and multi-modality. This deficiency avoids system from supporting dynamic application requirements. Besides, it restricts itself to a single application.

Among enable technologies on `SNQP`, [56–58] are the closest studies to our approach. These studies deal with the conflict between the quality of service and the sensor acquisition/transmission settings. Authors propose an optimization of application requests and a generation of query execution plan while considering acquisition interval, buffering factor, network lifetime and delivery time. This study is rich in terms of performance metrics. Presented approach combines query and quality of service expectation to form `Quality of Service Aware Sensor Network Engine (QoSA-SNEE)`. Generated execution plan provides a global view of the network and a schedule that shows when to sleep, when to listen, when to receive a packet, when to transmit the packet. Moreover, the study uses a Steiner tree for the routing strategy [86]. A sample of agenda generated by the proposed approach is illustrated in Figure 2.3 [58]. In terms of sensor scheduling and parametrization of a query, this approach is highly close to ours. With its scheduling mechanism, the presented system has a full control on the topology. With given quality of service, a query is processed to find the optimal execution plan and fulfill the expectations. These features can be considered as the strongest points of `QoSA-SNEE`. Still, the creation of a query execution plan covers a single query with a single expectation. Moreover, multi-modality and multi-application features are not supported by `QoSA-SNEE`.

In our study, we use the Service-Oriented Continuous Query Engine (SoCQ) [68] as a continuous query engine to manage sensor data streams. `SoCQ` tool has a strong advantage compared to other propositions (especially `SNEE` [28, 56]) as it supports a multi-application feature. However, in the default version of `SoCQ`, it does support neither energy-awareness nor the real-time sensor configuration. Our approach provides these features and improvement to the initial approach. Details of the `SoCQ` will be presented in Chapter 5.

[79] presents the well-known `SNQPs` in Table 2.3. Jabeen et al. summarizes these `SNQPs` and classifies through database principles.

	TinyDB [109]	**Cougar** [191]	**Andulin** [89]	**SNQL** [27]	**SNEE** [56]	**QOSA-SNEE** [58]
Language	Extention of SQL	SQL	CQL	SQL	Extention of CQL	Extention of CQL
Devices equipped with Query Evaluation Engine	Yes	No	No	Yes	No	No
Support Event-based Queries	Yes	No	No	No	No	No
Support Historical Query over thedata store in flash memory	Yes	No	No	No	No	No
Metadata Refresh Automatically	Yes	Assumed network information to be up to date	Assumed network information to be up to date	Assumed network information to be up to date	Assumed network information to be up to date	Assumed network information to be up to date
Responds to excessive workload	Shedding tuples	Not Specified	Assumed results are produced with desired quality, hence no need to discard data	Modify send and sense interval dynamically	Assumed results are produced with desired quality, hence no need to discard data	Assumed results are produced with desired quality, hence no need to discard data
Support for join query	Supports Join operation using materialization points	Not Specified	Lack of documentation regarding query language	Not Specified	Supports Join operation without using materialization points	Supports Join operation without using materialization points
Support for Select, Project, and aggregation query	Yes	Yes	Yes	Yes	Yes	Yes
Query Optimization Goal	Prolong the lifetime of the network by reducing total energy consumption	Reducing total energy consumption	Prolonging the lifetime of network by reducing resource usage	Prolong the lifetime of the network by reducing power consumption	Prolong the lifetime of the network by reducing average energy consumption	QoS expectations of queries are minimization of delivery time, acquisition interval, energy consumption, and maximization of lifetime
Implementation	nesC [60], programming language for TinyOS [96]. Supports Mica motes with TinyOS 1.1	Simulations and implementations on WINS devices designed by Sensoria Corporation, which are large, Linux based devices	Operator implementation in C language. Evaluations based on MSBA2 boards and an ARM LPC2387 processor, a CC1100 transceiver, 512 KB Flash, 98 KB RAM. Running a Contiki-based OS	Executed on a sensor network simulator, developed in C++ and executed in a Pentium IV machine	nesC, programming language for TinyOS. Supports Mica 2 motes with TinyOS 1.x and TinyOS 2.0	nesC , programming language for TinyOS

Table 2.3 Classification of Well Known SNQPs in the Literature through Database Principles [79].

start time	site 10	site 5	site 22	site 17	site 11	site 24	site 18	site 12	site 6	site 4	site 3	site 2	site 1	site 0
0:00:00.000	$F1_1$		$F1_1$			$F1_1$				$F1_1$				
0:01:00.000	$F1_2$		$F1_2$			$F1_2$				$F1_2$				
	...		...			...				...				
0:45:00.000	$F1_{46}$		$F1_{46}$			$F1_{46}$				$F1_{46}$				
0:45:00.043	tx	rx												
0:45:00.478		tx												rx
0:45:00.913			tx	rx										
0:45:01.347				tx	rx									
0:45:01.782					tx									rx
0:45:02.216						tx	rx							
0:45:02.651							tx	rx						
0:45:03.085								tx	rx					
0:45:03.520									tx					rx
0:45:03.955										tx	rx			
0:45:04.389											tx	rx		
0:45:04.824												tx	rx	
0:45:05.258													tx	rx
0:45:05.703														F0

Fig. 2.3 A Sample of Agenda of a Network based on the Application Requirements [58].

Apart from the SNQPs, there are other studies that focus on the sensor data model and stream management. [145] introduces key metrics to evaluate a quality-aware data stream management systems: application requirements defined by the user and translation of these application requirements into system parameters by the infrastructure; sensor device density of sensor network which represents the architecture, topology and the optimal network size to avoid high computation cost of centralized approach; heterogeneity of sensor networks (various physical measures) for real-time environmental monitoring applications. Moreover, the study introduces possible monitoring system internal conflicts such as high accuracy vs timeless, timeless vs reliability, and reliability vs accuracy. In brief, the study discusses the existing techniques and possible solutions for effective and quality-aware sensor data management in distributed sensor networks. Interference of multiple application activities is pointed as a major challenge in data management especially when the sensor activities occur concurrently. In our study, similarly to this approach, we translate the multiple parameterized queries into system inputs that are used to manage the sensor data streams. Moreover, as a major distinct from the existing studies, we support dynamic application requirements, i.e. query parameters can be updated during the execution or a new condition/threshold can be added to the existing sensor streams.

2.4 Sensor-Based Smart Building Management

We spend most of our life in the buildings and enhancing living environment through the use of smart technology has a crucial role to improve the ambience in buildings and also working environments. To reduce energy consumption, management of the building (monitoring via control based wireless sensor networks and actuators) is required.

In the literature, existing studies mainly focus on design and data management sides. The studies inspected in this section are mostly interested in building management platforms and the applications [39, 49, 52, 103, 104]. Most of these studies are application based and adopt a static configuration for the system and the deployed wireless devices in the environment. User preferences are considered as a static

input of the system can not be modified during the execution process.

[154] presents a novel design for agent-based intelligent building control to make the users more comfortable in the building environment. With the agent based approach, user interaction is tried to be minimized. The authors have a different point of view from the other studies and consider a building as an intelligent agent that is itself recursively composed of other agents. Hence, the authors propose using agents which communicate with one another asynchronously. The agents are location based and each agent only observes and takes decisions about a small part of the environment. This approach facilitates decision making and learning in real-time. The study adopts basic set of fuzzy rules: static and dynamic rules. Static rules establish fixed boundaries for the system whereas dynamic rules are learned and modified continually. However, the approach is not energy aware and does not consider the configuration of sensor devices. Based on the sensor data, it interacts and controls the actuators in the environment.

[38] concentrates on the design of a basic intelligent building system that covers monitoring of energy consumption of the system, integrated building operations and occupant-aware building control. For all these building actions, the authors propose a high-level architecture. The study considers a commercial building with multiple offices and assumes that each office is equipped with a controller that controls the room's lighting level and the cooling/heating set points. The authors present a prototype system that covers automated response, policy-driven governance, occupant awareness. The proposed system architecture supports layered integration, automated response, policy-driven governance. The approach adopts flow-based sensor event processing and rule-based business event processing methods. As the other existing studies, this approach is application based as well and the `WSN` environment has a predefined fixed set of configuration. Moreover, the system is tested only on the simulation environment that does not always reflect the real world effects.

[157] is one of the few studies that work on reducing the energy consumption of the sensor devices. The authors propose to integrate tiny wireless sensor or actuator devices into an IP-based network in building control and monitoring systems. The study presents web services-based approach to integrate resource constrained sensor and actuator devices into IP-based networks. The key feature of this approach is the automatic service discovery. This approach, similarly to ours, considers each wireless sensor device as a service provider and when a new wireless sensor device is added/appeared in the environment or on contrary disappears, automatic service discovery becomes aware of this phenomenon and updates the available services. As a concrete advantage of this study compared to the existing approaches, the authors use a real testbed environment for the implementation and experimentation phase. The system is implemented on Pixie devices using TinyOS [96]. Moreover, a web interface is developed to provide a user-friendly interaction with the sensor devices. Still, the study shows an acceptable performance with limited computing power and memory constraints of the hardware platform, as the other studies, it does not benefit from the reconfigurable sensor devices and the proposed system does not support the multi-application.

[112] as [154] does, supports that agents are as important as wireless sensor devices for learning and prediction mechanism of the monitoring system. The authors present using supervised statistical machine learning techniques for labeling training dataset. With the deployed sensor agents, not only the current occupancy but also the prediction/estimation of future occupancy of an area are developed.

[7] introduces the key features of a basic intelligent building such as HVAC (Heating, Ventilation, Air Conditioning) services, light systems. The study deals with one of the basic smart building features: the occupancy of the rooms. The authors present the design and implementation of a low-cost and incrementally deployable occupancy detection system using battery operated wireless sensor devices. Even though the study presents the importance of processing occupancy information in an efficient way (as [77] does), the proposed approach adopts a static configuration for the wireless sensor devices and does not consider the energy consumption of the devices. Moreover, the main drawback of this study as

the other presented studies is targeting a single application.

In most of the existing approaches, the authors present a novel design and architecture to manage sensor devices and the building environment. Although they focus on the sensor data management in a various environment (not only smart building environment but also a smart city where big data challenges are also involved [32, 149]), their major concern is the energy consumption of the building. Thus, the amount of consumed energy by the deployed equipment are ignored or disregarded. However, as we do in our study, there are a few study which has a similar target with us: monitoring of energy consumption of architecture and deployed wireless sensor devices.

Smart meters are commonly adopted by the researchers to monitor the energy consumption of the building [48, 143, 153]. [143] introduces the necessity of having a monitoring/control system for a building and proposes deploying digital smart meters. The smart meters presented in this study are capable of communicating wirelessly. Differently from the other studies in the same research domain, the authors focus on the amount of energy consumed by the deployed equipment. However, the study does not propose any optimization of energy consumption. Still, the study presents a better understanding on how to monitor energy consumption of several devices.

One of the studies that adopt re-configuration of sensor devices is [98]. The study presents a fully configurable testbed for organizing, storing, retrieving, distributing, and visualizing data in an efficient and automatic way. The authors propose an energy efficient design in order to share a large amount of data, especially in a real-time manner. Even though the authors propose a system and device configuration, the proposed configuration does not cover a fully dynamic configuration thus, a dynamic multi-application requirements could not be handled during the execution process.

The majority of studies focus on either handling sensor data flows or intelligent building management systems (managing deployed wireless sensor devices, actuators, middleware to manage the environment decision center etc). These approaches, even if they focus on energy efficiency, adopt a static sensor configuration and are moderate in terms of application variety. Among the existing studies in the literature, [31] is the closest study to ours in terms of point of view and perspective. The study points the lack of energy monitoring systems for building environments due to the static system architecture and finite battery lifetime for wireless devices. The authors present an intelligent building design based on a self-adapting intelligent gateway. The study introduces a novel gateway that serves for service decisions, sensor device management, gathering environment information, pattern generation, provision of energy management services. The proposed architecture also includes a learning mechanism related to a pattern, thus an appropriate service can be served based on the learned knowledge. The study proposes an adaptive middleware in order to decrease the energy consumption of the wireless sensor devices. A novel self-adapting intelligent system is presented. According to the experiments explained in the study, energy saving of the overall system is approximately 16-24% which is quite efficient regarding the other studies in this domain.

Similarly to this study, [30] implicates event-triggered dynamic sensor configuration which is similar to our approach. The study proposes a `ZigBee-based intelligent self-adjusting sensor (ZiSAS)` and presents a situation-based self-adjusting scheme with an event-based self-adjusting sensor network. `ZiSAS` can autonomously reconfigure middleware, network topology, sensor density, and sensing rate based on the environmental situation. A context-aware feature and dynamic sensor management by sensor adjustment based on the occurred event is proposed. However, unlike our proposition, user preferences, application requirements, query mechanism or data stream processing are not handled.

A common point of these approaches is to propose an architecture and management system for an intelligent building in order to provide more comfort, more safety, and more security environment. Even though application parameters (user preferences) are processed and decision models are used, deployed wireless sensor devices are fixed and have static configuration during the execution process. Moreover, high energy problem of the monitoring architecture is not considered as a major issue. Even though few

studies propose dynamic system management while processing user preferences, the main drawback of these studies is being bounded by predefined building applications, more importantly, the application requirement-sensor configuration relation is not established. Besides, these approaches do not benefit from a potential reconfiguration of acquisition and transmission frequencies: sensor configuration stays static during the system lifetime.

The summary of the inspected studies and the classification of these studies based on the key functionalities of SNQP and SBMS is given in Table 2.4. This table provides a clear view for us to position our approach in the literature. When we compare our approach, we observe major advantages. Firstly, SoCQ engine is used for its SNQP features. It provides a multi-application mechanism which is one of the strongest distinguishing features. Our approach covers context-aware computing that brings dynamicity for the applications. The default version of SoCQ does not provide energy-awareness however, after the integration of our approach with SoCQ, it gains energy-aware feature which is essential in our research domain. The dynamic user configuration is the other strongest feature of our approach. The proposed system gives an opportunity to users to modify the application requirements during the execution, thus the system may re-manage the sensor subscriptions and the sensor streams. The real-time sensor configuration is the novel feature that we bring out. Based on the applications and their preferences, our approach proposes a real-time sensor configuration to fulfill the application requirements and to enhance energy consumption of the monitoring applications. Moreover, we manage the sensor streams by our continuous query engine SoCQ and the experimental phase of the monitoring applications covers physical real testbeds and simulation. Besides, the continuous query engine SoCQ, used in our approach, has an ability to manage the real testbed deployed in our laboratory building (LIRIS lab of INSA-Lyon).

2.5 Conclusion

In this chapter, we analyse the literature and observe the existing approaches in the domain. This literature survey uncovers the major challenges of the domain. The main purpose was to explore the existing approaches, to search the closest studies to ours that tackle the identified challenges.

From the main energy saving methods presented, in our study, we inspired ourselves of the duty cycling behavior. Although the duty cycling is a long known technique, using basic duty cycling method is not efficient in systems that support multiple application and dynamic changes. Still, in our study, we adopt the periodic behavior of duty cycle mechanism (go to one of the sleep modes just after executing a sensor action and stay in that mode till the next action) to execute our Schedule Time Pattern mechanism. Moreover, from the data driven approach, we adopt the adapting sensor action method to manage our sensor actions in an optimal way.

Among the enable technologies for SNQP presented in this chapter, [56, 58] are the closest studies to our approach. These studies consider not only the sensor acquisition/transmission settings but also the quality of service QoS, especially the conflict between these two points. Similarly to our point of view, the authors suggests a global execution plan for the network in order to satisfy application requirements. While creating this sensor based execution plan, acquisition interval, buffering factor, network lifetime and delivery time are considered as the key metrics. Presented approaches have to combine continuous query engine and an external complementary tool to define the quality of service expectations. However, in our study, we prefer using parameterized continuous queries managed by continuous query engine SoCQ that lead us to parameterized sensor subscription to data streams to represent application requirements.

Using a schedule based global execution plan as we do, provides a global view of the network. This plan defines the sensor actions and the time for each of these actions. With our Schedule Time Pattern, as Galpin et al. do [58], we have a full control on the topology and on the devices in the environment. The strongest points of this approach are the generation of an execution plan based on the given query and the chosen expectation that refers to a quality of service. Still, the approach proposed by Galpin et al. has

Table 2.4 Classification of Well-Known Smart Building Environment Systems through Pervasive Environment Principles.

Study	Multi-Application	Context-Awareness	Energy-Awareness	Dynamic User Config.	Real-time Sensor Config	SNQP	Experimentation
Sharaf et al. [162]	–	√	√	–	–	TINA	CSIM simulator
Yao et al. [192]	–	–	√	–	–	COUGAR	Simu
Madden et al. [109]	–	–	√	–	–	TinyDB	Simu
Rutishauser et al. [154]	–	√	–	–	–	–	Real Testbed
Deshpande et al. [42]	–	–	–	–	–	MauveDB	Simu on real data
Baralis et al. [19]	–	–	√	–	–	SERENE	Simu
Tulone et al. [175]	–	–	√	–	–	SEFA	Simu
Tulone et al. [176]	–	√	√	–	–	PAQ	Simu on real data
Doukas et al. [44]	–	√	–	–	–	–	Real Testbed
Thiagarajan et al. [174]	–	–	–	–	–	FunctionDB	C++ prototype
Brayner et al. [27]	–	√	√	–	√	ADAGA	Simu
Li et al. [99]	–	–	√	–	–	PRESTO	EmStar emulator
Chen et al. [38]	–	√	–	–	–	–	Simulation
Schor et al. [157]	–	√	√	–	–	–	Real Testbed
Gripay et al. [68]	√	√	–	√	–	SoCQ	Real testbed
Agarwal et al. [7]	–	√	–	–	–	–	Real Testbed
Galpin et al. [56]	–	√	√	–	–	SNEE	Real testbed
Schreiber et al. [158]	–	√	√	–	–	PERLA	Simu
Mamidi et al. [112]	–	√	–	–	–	–	Real Testbed
Byun et al. [30]	–	√	√	–	√	–	Real Testbed
Li et al. [98]	–	√	√	–	–	–	Real Testbed
Preisel et al. [143]	–	–	√	–	–	–	Real Testbed
Lim et al. [101]	–	√	√	–	–	ACQUA	Perl-based simulator

a few drawbacks such as network efficiently. This approach creates a global schedule for the network, and during the execution process, every sensor device (path to sink device) of the topology waits for each other, although, two sensor devices that do not have a common device (intersection device) may execute their actions at the same time without any network collision. In our study, to avoid such delay and to avoid a potential bottleneck or packet collisions, we prefer using latency information given by the user (as a part of an application requirement) for every measured data. Moreover, in Galpin et al.'s study, a static configuration is adopted for the wireless sensor devices in the environment during the execution process and the sensors may not be reconfigured. The presented system may also support only a single query with a single expectation, whereas we support multiple parameterized applications and handle dynamic changes during the execution process (either dynamic user preferences or a change in the context environment). Besides, in our thesis, we benefit from the dynamic reconfigurability of the sensor devices and we support multiple applications at the same time.

Moreover, the studies presented in this chapter are fitted to a specific application such as traditional WSN or Smart Building systems. However, a WSN or a smart building system should be considered as a pervasive environment: approaches to solve a problematic should not be limited to a single domain but should address the larger domain of pervasive environment. Here, we apply our approach to smart buildings but, it can be applicable to any sensor based environment. Furthermore, a sensor for us is not only a wireless device, but we consider it as a service provider that provide us a service such as a temperature measure. From this point of view, our approach can be used in many different domains.

Chapter 3

Monitoring Architecture for Smart Building Applications

In our thesis, we focus on monitoring applications for smart building environment. For smart building systems, we adopt a "declarative monitoring architecture", built upon a `Pervasive Environment Management System (PEMS)` as used by [68, 161]. In this chapter, firstly we give a description of a declarative monitoring application and present a sustainable declarative monitoring architecture for lower energy consumption taking into account the monitoring system itself. Afterwards, a framework for managing pervasive environment systems is proposed and presented exhaustedly.

3.1 Declarative Monitoring Architecture

The term declarative monitoring is recently adopted by studies about environmental monitoring and service oriented architectures [199]. It is mostly designed to monitor and dynamically reconfigure itself in real-time during the execution. This real-time is based on the changing state and conditions of the context (for a pervasive environment) being monitored. The dynamic monitoring system continuously changes and adjusts to the present state of the context [159]. The primary functionality of such systems is the real-time reconfigurability that brings context-awareness. The traditional monitoring systems are static: once the system is defined or programmed by the user, it does not change. In general terms, a static monitoring system employs a fixed set of checks and rules to evaluate the state of the system.

A declarative monitoring architecture is composed of several layers in which `Pervasive Environment Management System (PEMS)` is located at the core of the overall system. It interacts with the applications that are expressed as SQL-Like continuous queries, and sensor devices to manage sensor subscriptions to the sensor data streams. A basic declarative monitoring architecture is composed of three principal layers as illustrated in Figure 3.1:

- `Application layer`: where application requirements are defined, declaratively expressed as a set of continuous queries over distributed services [68]. This layer provides information to users from throughout the entire system;

- `PEMS` is the core of the monitoring architecture. The principal challenge in pervasive applications such as smart building systems is the integration of mixed devices into one common application. This heterogeneity results with various sensor data formed in different formats [80, 121, 144]. For this purpose, middleware represents an interesting approach to reduce the gap between high-level requirements of pervasive applications and the access to basic functionalities of sensor devices. This layer integrates non-conventional, dynamic and heterogeneous data sources and manages

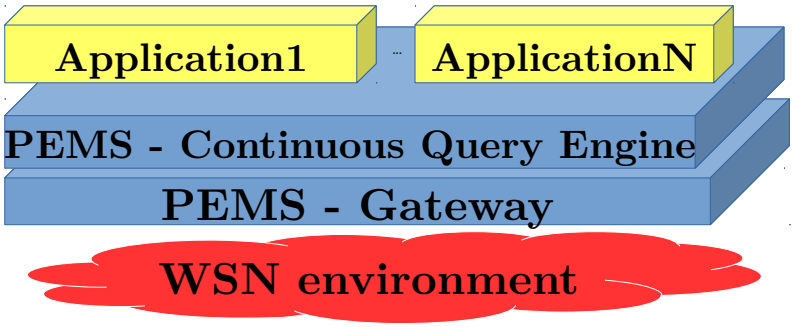


Fig. 3.1 Declarative Monitoring Architecture.

query executions; it includes a continuous query engine that interacts with services provided by the lower-layer through the logical gateway and manages queries coming from application layer [68, 69]. It represents a common ground to achieve interoperability when disparate components have to be integrated into a distributed system. PEMS layer includes also a gateway (PEMS Gateway) that stands for the interaction with the lower-layer;

- WSN environment is composed of devices geographically disposed within the building. They form together a building network using wireless technologies. Deployed wireless sensor devices acquire physical quantity measures, and can communicate with other sensor devices and physical gateways. In fact, from a generic point of view, this layer is composed of service providers (e.g. in the context of the smart buildings, a temperature sensor is a service provider and the service is the temperature measure) [9, 122, 132, 197].

3.2 Framework for Pervasive Environment Management System

3.2.1 Pervasive Environment Management System

As defined by [68], a PEMS is a service-enabled dynamic data management system that seamlessly handles network issues like service discovery and remote interactions. It supports the execution of service-oriented one-shot and continuous queries that application developers can easily devise to build pervasive applications. Based on this knowledge, a basic PEMS has to provide three main features:

1. the management of the distributed functionalities of the pervasive environment, i.e. the distributed services: service discovery and remote interaction techniques are needed to enable the integration of those functionalities into the PEMS;
2. the management of the dynamic data sources: database-like management is required to maintain a catalog of data containers and to handle their dynamic content;
3. the execution of queries over the relational pervasive environment, i.e. one-shot and continuous queries over dynamic data and service discovery queries: an implementation of the query operators, as well as query optimization techniques, are needed for this feature.

The primary functionality of any PEMS is to answer the major challenge of a pervasive environment: heterogeneous data sources and functionalities. Such systems should homogeneously manage heterogeneous data sources and functionalities. Traditional Data Stream Management Systems (DSMS) or ad hoc

programming are not addressing today's challenges. Thus, PEMS are created in order to ease the development of data-centric pervasive applications over pervasive environments. Such PEMS allow users to develop data-centric pervasive applications with insulating the users from the technological details [172].

3.2.2 Framework for PEMS

In this study, we adopt the SoCQ (Service-oriented Continuous Query) framework [68] that fulfill those features with its modular architecture. The SoCQ framework takes a data-oriented perspective on pervasive environments such as smart buildings. It provides a unified view and access to various and heterogeneous data resources, or services, available in the environment. XD-relations (eXtended Dynamic Relations) can represent standard relations, that may be updated, or data streams, that continuously produces data. Pervasive applications can then be created in a declarative fashion using service-oriented continuous queries over XD-relations. Queries may be one-shot queries (like standard SQL queries) or continuous queries (with a dynamic result, like a stream). Queries can also interact with distributed services: service discovery, method invocation, stream subscription. Furthermore, invocations and subscriptions can be finely parametrized.

For instance, a service discovery query can search for sensor services that provide a location, a method to get the current temperature, and a continuous stream of temperatures. The result is an XD-Relation with a ServiceID, a Location, and a virtual attribute for Temperature. Once relevant services are listed, a continuous query can subscribe to the temperature stream of every discovered service, to build a resulting data stream with Temperature values. If new services are discovered and/or some services become unavailable, the continuous query automatically adapts the set of stream subscriptions.

[172] introduces the related challenge of the design of pervasive environments. The authors present the necessity of having a PEMS to design and to develop a pervasive environment. They propose an approach to manage heterogeneous services in the environment and introduce P-Bench, a benchmark to assess the easiness of data-centric pervasive application development. According to the results obtained with this design, developing, deploying and updating pervasive applications is considerably easier when using a PEMS.

3.3 Application Requirements

In our study, application requirements stand for the requests for sensor data from monitoring applications configured according to user preferences and/or context. To handle these requests and manage them, a continuous query engine is highly required. This continuous query engine should be located at the PEMS layer of the monitoring architecture. For our monitoring architecture, we preferred SoCQ to handle multiple applications.

The definition of multiple applications is that the system may support several applications at the same time: these applications may request the same service from the same service providers with different parameters. Besides, applications requirements are mostly parameterized continuous queries. A parameterized continuous query stands for a query with defined parameters. Here, these parameters can be the database attributes (such as request service from a specific location), physical measures (such as requesting specific measure) or the frequency of sensor actions (such as data acquisition frequency). An example of a continuous query is illustrated in Table 3.1. The first query is a discovery query to find the service providers in the environment that may serve temperature service. In our target application (smart buildings), our service providers are the deployed wireless sensor devices and the services are temperature, presence, humidity, emission of CO_2 etc. This query builds a list of sensor devices that can provide temperature measures from given location with parameters: acquisition frequency, latency.

Table 3.1 Example SoCQ queries for a Smart Building.

```
CREATE RELATION TemperatureServices (
ServiceID SERVICE,
Location STRING,
Temperature NUMBER VIRTUAL
acquisition_periodicity INTEGER VIRTUAL,
latency INTEGER VIRTUAL
) USING (
getTemperature[ServiceId]():(Temperature),
temperature[ServiceId](acquisition_periodicity, latency):(Temperature) STREAMING
)
AS
DISCOVER SERVICES PROVIDING
PROPERTY Location STRING,
METHOD getTemperature ( ) : ( NUMBER ),
STREAM temperature (INTEGER, INTEGER ) : ( NUMBER )
```

ServiceID	**Location**	***Temperature***
sensor:01	501.337	*
sensor:03	501.340	*
sensor:17	502.321	*

```
SELECT *
STREAMING UPON insertion
FROM TemperatureServices
WITH acquisition_periodicity := 1800, latency :=300
USING temperature[1];
```

The second part of the query is the stream query that starts data flow (temperature measures) with given parameters from the temperature sensor devices. Here the acquisition frequency represents the temporal accuracy of measure and the latency stands for the maximum acceptable delay between the acquisition of data and its arrival to the base station.

Once the application requirements are collected by the PEMS framework, they are translated into parameterized subscriptions to the sensor streams. Moreover, the SoCQ framework supports real-time user configuration of applications and context-aware applications through queries that can dynamically combine data, streams and services. These major features allow users to insert complex queries with conditions over the data stream to trigger a new query during the execution process.

We believe that a concrete example may help us to clarify the application requirements and at the same time our problematic that we will discuss in the next section. Suppose that we have a smart building as a pervasive environment with deployed wireless sensor devices where applications may request basic physical measures from these sensors such as temperature, humidity, occupancy etc.

Example 3.3.1 *Two applications need to track the temperature of the each room of the building. Application 1 requests the temperature of each room every 30 minutes with latency 5min. Application 2 demands the temperature value of each room every 10 minutes with latency 4min.*

Here, there are two distinct applications that request the same physical measure (temperature service)

from the same subset of the sensor devices (all sensor devices that provide temperature service). These applications request temperature services with different parameters. We assume that the system has the real-time knowledge about each sensor device and their capabilities in terms of physical measure (e.g. sensor devices for temperature service). In this scenario, the acquisition period of Application 1 is a multiple of acquisition period of Application 2. This mathematical point makes the example quite easier to solve. Each application asks from the sensors to acquire and to send (i.e. no storage on the sensor device). Thus, we pose a critical question: how should our sensor devices be configured to fulfill these two application requirements at the same time ? There are two applications with different acquisition periodicities. The existing approaches (that support multiple applications) expect the worst case scenario and the sensor devices are configured according to that scenario. These approaches take into account the most frequent application requirement for both applications (e.g. in this example, common acquisition periodicity can be set to 10 min).

3.4 WSN Device Configuration

A wireless sensor device is a device in a sensor network that is capable of performing some processing, gathering sensory information and communicating with other connected devices in the network. The main components of a sensor node are a microcontroller (for computation), transceiver (for communication), external memory (for storage), a power source (batteries or capacitors) and one or more sensors (capture data from their environment). As previously introduced, a wireless sensor device like other electronic devices has a hardware and a software side. The listed components form the hardware side of a sensor device. Besides, the software defines the rules of the sensor device while running. For instance, a transceiver (for the communication) has several modes such as transmission, reception, idle, sleep etc. The software indicates in which mode the transceiver should be and when to switch mode.

A sensor configuration stands for the update of the software effect on the hardware. In other words, the software keeps the rules for running a device and a sensor configuration updates these rules. For instance, suppose that a configuration requests to change the acquisition frequency. Here the software updates the acquisition frequency that affects the hardware side (the sensor unit of the device). Another example can be given about the transceiver of the device. A configuration may request from the transceiver to go to listening mode after each transmission state. Hence, the microcontroller gets the sensor configuration (either at the deployment state or in the real-time), analyses and executes it. The application of the configuration is not necessarily on the hardware side (such as switching a mode, turning on/off). A configuration may also request a computation running on the microcontroller.

In many applications, sensor configuration affects the energy consumption of the sensor device. Thus, while configuring a wireless sensor device, either at the beginning or during the execution (in real-time), limited energy budget should be taken into account. Every major action (such as transmission, reception, computation etc) executed on the sensor device consumes a part of the energy budget and shorten the lifetime of the device.

An energy cost model is a mathematical algorithm or parametric equations used to estimate the costs of consumed energy for a device. An example of a power model for the Mica2 (third generation wireless sensor device used for enabling low-power, wireless, sensor networks) is presented in Table 3.2. Each component of the sensor device drains a current and consumes energy. Based on several measurements in the literature, the radio component (the transceiver) consumes approximately half of the energy budget. Hence, the existing studies on the energy challenge commonly focus on optimizing sensor communication to enhance energy. Table 3.3 presents the energy consumption of each component of the sensor device (Mika2) for various applications. These results are based on the measurements and may vary according to the preferred sensor device.

Table 3.2 Power Model for the Mica2 sensor device [164].

	Mode	Current
CPU	Active	8 mA
	Idle	3.2 mA
	ADC Noise Reduce	1 mA
	Power-down	103 μA
	Power-save	110 μA
	Standby	216 μA
	Extended Standby	223 μA
	Internal Oscillator	0.93 mA
Radio	Rx	7 mA
	Tx (-20 dBm)	3.7 mA
	Tx (-19 dBm)	5.2 mA
	Tx (-15 dBm)	5.4 mA
	Tx (-8 dBm)	6.5 mA
	Tx (-5 dBm)	7.1 mA
	Tx (0 dBm)	8.5 mA
	Tx (+4 dBm)	11.6 mA
	Tx (+6 dBm)	13.8 mA
	Tx (+8 dBm)	17.4 mA
	Tx (+10 dBm)	21.5 mA

Table 3.3 Energy Breakdown of Each Component for Various Applications (values are in mJ, Mica2 device is used) [164]. EEPROM: a read-only memory whose contents can be erased and reprogrammed using a pulsed voltage.

Application	CPU Idle	CPU active	Radio	Leds	Sensor	EEPROM	Total
Beacon	35.86	0.58	47.68	8.81	0	0	92.93
Blink	742.5	0.25	0	197.52	0	0	940.26
BlinkTask	742.5	0.27	0	197.52	0	0	940.28
CntToLeds	743.72	0.57	0	592.2	0	0	1336.49
CntToLedsAndRfm	741.9	1.61	1284.65	592.2	0	0	2620.37
CntToRfm	741.9	1.54	1284.65	0	0	0	2028.09
Oscilloscope	742.65	1.46	0	0	123.82	0	867.94
OscilloscopeRF	741.9	1.85	1268.76	0	123.95	0	2136.45
Sense	742.21	0.38	0	0	123	0	865.59
SenseLightToLog	741.9	0.81	1262.95	0	123.95	4.28	2133.89
SenseTask	742.21	0.42	0	0	123	0	865.62
SenseToLeds	743.72	0.73	0	0	124.25	0	868.7
SenseToRfm	741.9	1.77	1284.65	0	123.95	0	2152.27
Surge	727.28	1.5	1239.02	0	121.3	0	2089.09

3.5 Problem Statement

In this thesis, we study a monitoring architecture for pervasive environments. We mainly focus on a sustainable architecture for multi-application monitoring systems that continuously adapt to application requirements, context and user configurations. As an application example, we apply our approach on smart building applications.

As presented in the previous sections, a system may receive several requests at the same time from applications and that system should be capable of responding to all requests successfully. Here, the challenging point is that in our case (smart building as a pervasive environment), users may request several data streams with different parameters from the same wireless sensor device. Handling this situation puts our approach apart from the existing studies in this domain. The major challenge is to fulfill these application requests that include different parameters. The parameters are related to data constraints and sensor actions, more specifically to the data acquisition, transmission frequencies and latency. Based on the presented challenge, the key question is how to fulfill all the applications. Each application (expressed as a continuous query as described in application requirements) has its own requirements and set of parameters. While searching a feasible solution to the question, another major issue comes out, the chronic problem of any system based on wireless sensor device: energy consumption and limited lifetime. So the new question is how to support all applications and fulfill their requirements without penalizing wireless sensor device with high energy consumption. To manage these applications and their requirements, the application requirements should be transformed into acceptable working parameters for the sensor devices. With this perspective, we may enhance the energy consumption of the sensor devices in the environment and extend the lifetime of the monitoring system. In this thesis, our approach handles the application requirements that are expressed as parameterized continuous queries written in SQL-Like language and transforms into unique sensor configuration for each sensor device in the pervasive environment.

In this thesis, based on the existence of dynamically configurable wireless sensor devices [133], we propose a novel approach: an **energy-aware dynamic sensor configuration based on real-time application requirements to improve the energy consumption of the system**. To achieve, we present `Smart-Service Stream-oriented Sensor Management (3SoSM)`, a novel approach to **optimize interactions between application requirements and wireless sensor environment in real-time**. Our `3SoSM` approach relies on the energy-aware dynamic sensor device configuration to lower energy consumption while fulfilling real-time application requirements. With this approach, we expect to avoid unnecessary data measurements that may occur with static configuration and to promote shorter/compressed data transmission when possible.

Chapter 4

3SoSM Approach

In this chapter, we present the core of our approach. We focus on energy consumption by wireless sensor devices in pervasive monitoring system architectures, in particular for smart building infrastructures. We aim to design an energy-aware monitoring system that adapts to various application requirements, to context and to user configuration. Application requirements as declared by the application developer can be fulfilled in a multitude of ways: our approach translates these requirements as energy efficient schedules for wireless sensor devices. We propose our approach `Smart-Service Stream-oriented Sensor Management (3SoSM)` that allows to optimize the interaction between the application requirements and the `WSN` environment.

The key point of our approach is to configure dynamically wireless sensor devices to fullfill application requirements while enhancing energy consumption. To achieve this goal, we propose a novel approach by transforming subscription requests into suitable `Schedule Time Pattern` in order to configure wireless sensor devices with `sensor configuration-oriented pattern` (SCO-PATTERN). In fact, our approach has two versions: a version in which we focus on a single wireless sensor device, the `Device-Centered 3SoSM` approach and a version in which we take into account network, the `Network-Aware 3SoSM` approach. `Device-Centered 3SoSM` approach adopts device-centered point of view, thus it includes a local optimization process to manage the interaction mentioned above. `Network-Aware 3SoSM` approach requires a global view over the network topology and covers a network-aware optimization process for the energy-aware sensor management.

The global objective of our study is to manage multiple application requirements, transform these application requirements into a global schedule and then build individual SCO-PATTERNs to configure each device. Thus, we aim to optimize the subscription requests while fulfilling the application requirements.

4.1 Query Requirements & Subscription Requests

Before presenting the details of our approach, declarative monitoring architecture of the system should be recalled. At the top of the proposed architecture, we have multiple applications requesting services (physical measures such as temperature, humidity, occupancy etc) from the service providers (wireless sensor devices deployed in the environment). These applications form the `Application Layer` where a set of applications are located to exploit sensor measures in real-time. These applications are declaratively expressed as parameterized (service-oriented) continuous queries over sensor data streams. These queries are handled by the `Continuous Query Engine` that transforms these queries into subscriptions to the sensor devices in order to launch the data streams. There is a middleware Gateway located between the continuous query engine and the `WSN` environment. It stands for the interaction between the application requirements transformed into subscriptions and the environment that consists of wireless sensor devices. The gateway layer is responsible for managing the sensor devices. Each sensor receives its own `Schedule`

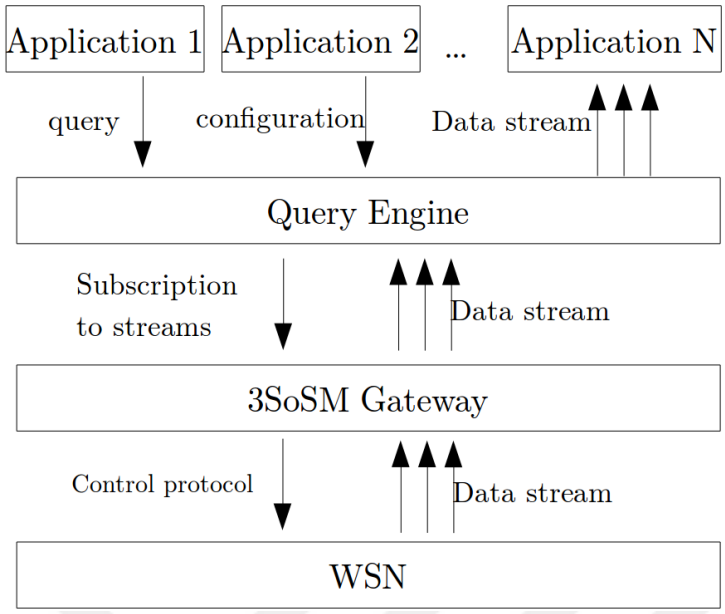


Fig. 4.1 Interactions between the Layers of the Monitoring Architecture.

`Time Pattern` from the gateway through control protocol and executes it. Moreover, the gateway receives the data streams from the sensor devices and forwards these streams to the query engine. Finally, the query engine serves the received data to the relevant applications on the top layer. The interactions between the layers of the monitoring architecture are illustrated in Figure 4.1.

Before giving the details of our approach, we first introduce concepts and notations that we use frequently in this chapter for both versions of our approach.

From our data-centric perspective, query requirements are application requirements when applications are defined by sets of continuous queries. Definition of application requirement is given in the previous chapter in Section 3.3.

A typical smart building is equipped with various wireless sensor devices $d_i \in D$, where D is the set of devices in the environment and d_i is a single wireless sensor device. Each sensor device may have multiple functionalities to acquire physical quantity measures $m_i \in M$, e.g., temperature, humidity, occupancy where m_i is a measure and M is the set of possible physical measures.

The input of our approach is the application requirements that are defined within queries in terms of data source requirements (targeted sensors d and measures m) and of temporal requirements: temporal window β, update periodicity p^{upd}, data acquisition periodicity p^{acq} and maximum latency λ. The unit for all these temporal parameters is the second (or millisecond, if required). Notation for the parameters of the query is summarized in Table 4.1. We use these notations throughout the remainder of this thesis.

To clarify the application requirements, a concrete example of an application can be as follows:

Example 4.1.1 *Application A1 computes the average temperature over the last 10 minutes, with an update every 5 minutes, with an accuracy of 1 second and a maximum latency of 1 minute.*

In terms of data management, requirements of such applications can be summarized by the following parameters:

1. temporal window: It introduces the time interval for calculating the result. For the given example, the application requests the average temperature value over the last 10 minutes. Hence, the temporal window for this request is 10 minutes;

Table 4.1 Overview of Notations for Query Requirements and Subscription Requests.

Notation	Stand for
d_i	a wireless sensor device
D	set of wireless sensor devices ($d_i \in D$)
m_i	a physical measure
M	set of physical measures ($m_i \in M$)
β	temporal window size ($\beta \in \mathbb{N}^*$)
p^{upd}	periodicity of result updates ($p^{upd} \in \mathbb{N}^*$)
p^{acq}	data acquisition periodicity ($p^{acq} \in \mathbb{N}^*$)
λ	maximum latency ($\lambda \in \mathbb{N}^*$)
s_i	a subscription to a wireless sensor device ($s_i = (d_i, m_i, p_i^{acq}, \lambda_i) \in D \times M \times \mathbb{N}^* \times \mathbb{N}^*$)

2. update periodicity stands for the refreshing rate of the result. For the given example, application demands to update the result every 5 minutes;

3. data acquisition periodicity represents the temporal accuracy of measure. For our example, the application requests data measurement every second (accuracy of 1 sec), i.e. the application indicates the precision for the measure.

4. maximum latency presents the maximum acceptable delay between the acquisition of data and its transmission to the `PEMS` layer for result calculation. For our example, the application specifies 1 minute of latency for the reception of an acquired physical measure;

The first two parameters (temporal window and the update periodicity) concern the computing of the result that takes place on the continuous query engine. These parameters are not related to the acquisition or data transmission in the network environment. For clarity, we do not present them in the remainder of this study. Besides, the other two query parameters are directly based on the sensor device and sensor actions. They represent respectively the acquisition periodicity of a physical measure and the maximum latency of an acquired measure.

Subscription requests are device-centered representation of the application requirements. We then represent application requirements on sensors at a given time instant by a set of parameterized subscription requests $S = \{s_1, s_2, ..., s_n\}$, where: $s_i = (d_i, m_i, p_i^{acq}, \lambda_i) \in D \times M \times \mathbb{N}^* \times \mathbb{N}^*$.

Example 4.1.2 *For example, suppose that there are two applications with the following requirements on 6 sensors:*
Application 1: Temperature (m_T) of sensors d_1, d_3, d_5 with an acquisition periodicity of 2 sec and a latency of 4 sec
Application 2: Humidity (m_H) of sensors d_3, d_4, d_5, d_6 with an acquisition periodicity of 5 sec and a latency of 3 sec

Based on the notation, all application requirements are represented by parameterized subscription requests:
$s_1 = \{(d_1, m_T, 2, 4), (d_3, m_T, 2, 4), (d_5, m_T, 2, 4)\}$
$s_2 = \{(d_3, m_H, 5, 3), (d_4, m_H, 5, 3), (d_5, m_H, 5, 3), (d_6, m_H, 5, 3)\}$

4.2 Sensor Configuration: SCO-Patterns

We propose a `Schedule Time Pattern` to represent a data acquisition/transmission schedule used to configure a physical device. We call this pattern `sensor configuration oriented pattern`

(SCO-PATTERN). This pattern consists of time-stamped events ($< timestamp,\ action >$ couples) and a length (or periodicity) of that pattern ℓ. These actions are enclosed by the length of the pattern. Time-stamped events are enclosed by time interval $[0;\ell[$. A SCO-PATTERN is denoted by: $\mathrm{P} = (\{(t_i, a_i)\}, \ell)$ with $\ell \in \mathbb{N}^*$, $t_i \in [0;\ell[$, $a_i \in \{A_m, T, R\}$. where A_m indicates the data acquisition of the physical measure m such as A_T for temperature, A_H for humidity, etc. The notation for representing a SCO-PATTERN is introduced in Table 4.2.

Example 4.2.1 *The* SCO-PATTERN *for a sensor device can be:*
$\mathrm{P}_{d_i} = (\{(0,\{A_H\}),(0.5,\{T\}),(5,\{A_H\}),(6.5,\{T\})\},10)$
where the sensor device d_i measures humidity at t=0sec, transmits the data at t=0.5sec, measures humidity at t=5sec, transmits the data at t=6.5sec.

$\mathrm{P}_{d_j} = (\{(0,\{A_T\}),(1.5,\{R\}),(2,\{A_T,R\}),(2.5,\{T\}),(4,\{A_T\}),(6,\{A_T\}),(6.5,\{R\}),(7,\{R\}),$
$(7.5,\{T\}),(8,\{A_T\}),(9,\{R\}),(9.5,\{T\})\},10)$
where the sensor device d_j measures temperature at t=0sec, it receives a data packet at t=1.5, measures temperature and receives a data packet at t=2sec etc., until t=10sec.

4.3 Device-Centered 3SoSM Approach

In this section, we present `Device-Centered 3SoSM` approach. This approach adopts a local point of view and includes local optimization process. Here, we focus on a single wireless sensor device and the subscriptions to that device. For this version of the approach, we do not take into account the multi-modality of a device or the network issues.

4.3.1 DOA-Patterns

Our approach aims to optimize a set of subscription requests for a sensor device into a `Schedule Time Pattern` defining the configuration of this device. `Device-Centered 3SoSM` approach relies on an intermediate `Data-Oriented Acquisition pattern` (DOA-PATTERN) before reaching to SCO-PATTERN. A DOA-PATTERN can represent a single subscription request and can also be merged other DOA-PATTERNs.

A DOA-PATTERN consists of a list of acquisition-latency events and the length ℓ of this pattern. An acquisition-latency event is a triple $< timestamp, action, latency >$, where action is always an acquisition. Based on our notation, a DOA-PATTERN ρ is denoted by: $\rho = (\{(t_i, A^{\lambda_i}_{m_i},)\},\ell)$, with $\ell \in \mathbb{N}^*$, $t_i \in [0;\ell[$. In fact, DOA-PATTERN is a sort of `Schedule Time Pattern` that only presents acquisition and latency information (TX/RX information are not involved).

The DOA-PATTERN is specific for each sensor device. It is periodic, and its length ℓ indicates the periodicity. A sensor device may have different acquisition periodicities from different applications. However, a common length is required: it is the lowest common multiple of acquisition periodicities from all subscription requests. Moreover, the length of the schedule should be greater than any latency of the subscription requests to properly handle data expiration time. The final length is the lowest multiple of this initial common length greater than any data expiration time. Then based on the periodicity of subscription requests, acquisition actions are defined. Furthermore, each acquisition action is tagged with its maximum latency (from the subscription request). Notations that represent the structure of the DOA-PATTERN and SCO-PATTERN are introduced in Table 4.2.

Example 4.3.1 *The common length of the patterns for the applications given in Example 4.2.1 is calculated as:*
Initial common length min $= LCM(2,2,2,5,5,5,5) = 10sec$
Maximum latency $\lambda max = MAX(4,4,4,3,3,3,3) = 4sec$
Final length schedule = 10 sec $> \lambda$ *max*

Table 4.2 Overview of Notations for DOA-Patterns and SCO-Patterns.

Notation	Stand for
ℓ	length of the pattern ($\ell \in \mathbb{N}^*$)
t_i	action time ($t_i \in [0, \ell[$)
a_i	sensor action ($a_i \in \{A_m, T, R\}$)
ρ	DOA-PATTERN ($\rho = (\{(t_i, A^{\lambda_i}_{m_i},)\}, \ell)$)
P	SCO-PATTERN ($\mathrm{P} = (\{(t_i, a_i)\}, \ell)$)

The pseudo code for determining the length of the schedule is given in Algorithm 1

Algorithm 1 Pseudo Code of Calculation of the Common Length.

```
1:  procedure CALCULATECOMMONLENGTH()
2:      list_p_acq[] = getAcquisitionPeriodicityFromQueries()  /*fetch acquisition periodicities from queries*/
3:      length = Math.LCM(list_p_acq)  /* calculate the LCM value of the acquisition periodicities */
4:      list_latency[] = getLatencyFromQueries()  /* fetch latency parameters from queries */
5:      max_latency = Math.MAX(list_latency)  /* find maximum of latencies */
6:      if max_latency > length then
7:          coefficient = 2
8:          while length < max_latency do
9:              length *= coefficient++
10:         end while
11:     end if
12: end procedure
```

Example 4.3.2 DOA-PATTERN *for d_3 (Sensor 3 of the given example) with subscriptions* $\{(d_3, m_T, 2, 4), (d_3, m_H, 5, 3)\}$*:*
$\rho = (\{(0, \{A^4_T, A^3_H\}), (2, A^4_T), (4, A^4_T), (5, A^3_H), (6, A^4_T), (8, A^4_T)\}, 10)$

4.3.2 GeNoMe Optimization Process

The goal of this algorithm is to generate a sensor device configuration, that we name SCO-PATTERN, that fulfills all the application requirements expressed by a set of subscription requests targeting this device (with their specific acquisition period and latency). Even though a sensor device can have multiple functionalities such as measuring temperature and humidity, in this section, we consider subscription requests only for a single physical measure. The multi-modal feature is involved and introduced in the complete version of our approach.

The generation process of the final SCO-PATTERN P_{final} is named `3SoSM GeNoMe` process (**Ge**nerate-**No**rmalize-**Me**rge) and illustrated in Figure 4.2. We now detail the 4 steps of this algorithm:

1. ***Generate DOA-Patterns***: First step of `GeNoMe` process is to represent the given subscription in terms of data acquisition. In this step, we generate an intermediate pattern that we call DOA-patterns. We form a DOA-PATTERN ρ_i for each subscription s_i with its parameters (p^{acq}, λ). This pattern is an intermediate step to reach the final pattern.

2. ***Normalize DOA-Patterns***: To be able to merge DOA-PATTERNs (next step), all the DOA-PATTERNs should have the same length, that will be the length of the final pattern. The `lowest common`

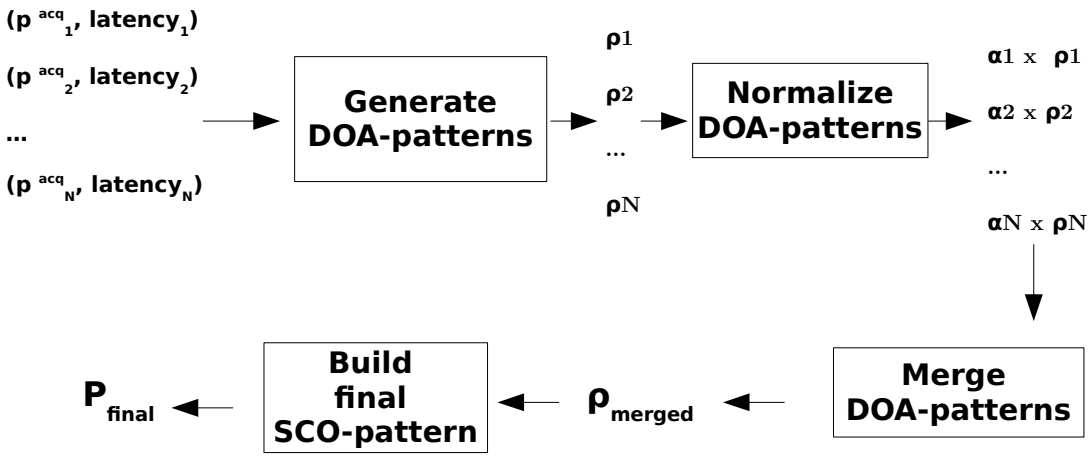


Fig. 4.2 Overview of the GeNoMe process.

`multiple method (LCM)` is used to calculate the common length, thus:
$\ell_{final} = k * LCM(\ell_i, \ell_j, \ldots)$ with respect to the constraint $\ell_{final} > MAX(\lambda_i, \lambda_j, ...)$.
Then, to reach the common length, a coefficient for each pattern is calculated: $\alpha_i = \ell_{final}/\ell_i \in \mathbb{N}^*$.
This coefficient indicates how many times each pattern should be repeated in order to reach the length of the final pattern, i.e., to normalize this pattern. Thus, each pattern is extended with its specific coefficient: $\alpha_i \times \rho_i$.

3. ***Merge DOA-Patterns***: We now merge the set of DOA-PATTERNs to obtain the merged DOA-PATTERN ρ_{merged}, with the normalized length calculated in the previous step. The lists of events are merged into a single list. If two events occur at the same timestamp, those events are themselves merged into a single event with a "merged" latency (the minimum latency of those two events). Event actions are still always acquisition actions (no transmission action).

4. ***Build final SCO-Pattern***: The DOA-PATTERN ρ_{merged} indicates data acquisition timestamps and latency, and the periodicity of that pattern. Data transmission information is not indicated yet in the DOA-PATTERN. As a final step, transmission actions are inserted on some events and latency values are then removed, in order to build a final SCO-PATTERN. Optimal transmission events are calculated based on the latency values of acquisition events, to fulfill maximum latency requirement for each acquisition event. As a first heuristic, starting from the first acquisition event, we search for the latest next event whose measure can be transmitted with all previous measures while respecting latency constraints. We then continue with the next acquisition event, until the end of the pattern.

4.3.3 Application to some Examples

Here, a concrete example that presents each step of the `GeNoMe` process for given two applications:

Example 4.3.3 *Suppose that there are two subscription requests to the same sensor device d_i for measuring temperature m_T. The first subscription requires data acquisition every 4 sec with latency 7 sec (p^{acq} =4sec, λ = 7sec).*

The second subscription requires data acquisition every 5 sec with latency 9 sec (p^{acq} =5sec, λ = 9sec). Regardless of window sizes β and result update periods p^{upd}, those subscriptions can be expressed as:
$s_1 = (d_i, m_T, \beta_i, p_i^{upd}, 4, 7)$ *and* $s_2 = (d_i, m_T, \beta_j, p_j^{upd}, 5, 9)$.

With our algorithm, we optimize this set of subscriptions into a sco-pattern:

1. *Generate* DOA-PATTERNS*: Subscription requests can be expressed as:*
   $\rho_1 = (\{4, A^7\}, 4)$ *and* $\rho_2 = (\{5, A^9\}, 5)$
2. *Normalize* DOA-PATTERNS*: Normalized length is LCM (4, 5) = 20 (20 > MAX(7,9)). Coefficients are* α_1 *= 20/4 = 5 for* ρ_1 *and* α_2 *= 20/5 = 4 for* ρ_2*. Then,* ρ_1 *should be repeated five times and* ρ_2 *should be repeated four times:*
   α_1 *x* ρ_1 *= 5 x* $\rho_1 = (\{(4, A^7), (8, A^7), (12, A^7), (16, A^7), (20, A^7)\}, 20)$
   α_2 *x* ρ_2 *= 4 x* $\rho_2 = (\{(5, A^9), (10, A^9), (15, A^9), (20, A^9)\}, 20)$
3. *Merge* DOA-PATTERNS*: The merged pattern has a length of 20, and only the last event (at 20) merges two events (with a latency of MIN (7, 9) = 7):*
   $\rho_{merged} = (\{(4, A^7), (5, A^9), (8, A^7), (10, A^9), (12, A^7), (15, A^9), (16, A^7), (20, A^7)\}, 20)$
4. *Build final* SCO-PATTERN: *Transmission actions are added to relevant events at 10, 16 and 20. All latency requirements are thus fulfilled:*
   $P_f = (\{(4, A), (5, A), (8, A), (10, AT), (12, A), (15, A), (16, AT), (20, AT)\}, 20)$

We remark that the final pattern requires only 18 transmission actions per 2-minutes (3 AT per 20s), whereas the two initial requests would require respectively 15 and 12, and a total of 24 AT considering a common AT every 40s.

To present the usage of the `GeNoMe` process, here we introduce four more examples of `GeNoMe` process on various subscriptions: for a single application, for two applications and for three applications.

Example 4.3.4 *With a single Application: Suppose that there is one subscription request for measuring temperature* m_T*. The subscription requires data acquisition every 5 sec with latency 8 sec (p^{acq} = 5sec,* λ *= 8sec). Subscription can be expressed as:* $s = (d, m_T, \beta, p^{upd}, 5, 8)$

`GeNoMe` process:

1. *Generate* DOA-PATTERNS*: Subscription requests can be expressed as:* $\rho = (\{5, A^8\}, 5)$
2. *Normalize* DOA-PATTERNS*: Normalized length is LCM (5) = 5 (*$5 \not> 8 \implies \ell = 10$*). Coefficient is* α*= 2:* α *x* ρ *= 2 x* $\rho = (\{(5, A^8), (10, A^8)\}, 10)$
3. *Merge* DOA-PATTERNS*: The merged pattern has a length of 10:*
   $\rho_{merged} = (\{(5, A^8), (10, A^8)\}, 10)$
4. *Build final* SCO-PATTERN*: Transmission actions are added to relevant events at t=10. All latency requirements are thus fulfilled:*
   $P_f = (\{(5, A), (10, AT)\}, 10)$

Example 4.3.5 *With two applications: Suppose that there are two subscription requests to the same sensor device* d_i *for measuring temperature* m_T*. The first subscription requires data acquisition every 5 sec with latency 3 sec (p^{acq} = 5sec,* λ *= 3sec). The second subscription requires data acquisition every 10 sec with latency 4 sec (p^{acq} = 10sec,* λ *= 4sec). Subscription can be expressed as:*

$s_1 = (d_i, m_T, \beta_i, p_i^{upd}, 5, 3)$
$s_2 = (d_i, m_T, \beta_j, p_j^{upd}, 10, 4)$.

GeNoMe process:

1. *Generate* DOA-PATTERNS*: Subscription requests can be expressed as:*
   $\rho_1 = (\{5, A^3\}, 5)$
   $\rho_2 = (\{10, A^4\}, 10)$

2. *Normalize* DOA-PATTERNS*: Normalized length is LCM (5, 10) = 10. Coefficients are* α_1 *= 10/5 = 2 for* ρ_1 *and* α_2 *= 10/10 = 1 for* ρ_2*. Then,* ρ_1 *should be repeated two times and* ρ_2 *should be repeated once:*
   α_1 *x* ρ_1 *= 2 x* $\rho_1 = (\{(5, A^3), (10, A^3)\}, 10)$
   α_2 *x* ρ_2 *= 1 x* $\rho_2 = (\{(10, A^4)\}, 10)$

3. *Merge* DOA-PATTERNS*: The merged pattern has a length of 10, and only the last event (at 10) merges two events (with a latency of MIN (3, 4) = 3):*
   $\rho_{merged} = (\{(5, A^3), (10, A^3)\}, 10)$

4. *Build final* SCO-PATTERN*: Transmission actions are added to relevant events at 10. All latency requirements are thus fulfilled:*
   $P_f = (\{(5, AT), (10, AT)\}, 10)$

Example 4.3.6 *With two applications: Suppose that there are two subscription requests to the same sensor device* d_i *for measuring temperature* m_T*. The first subscription requires data acquisition every 4 sec with latency 10 sec (*p^{acq} *= 4sec,* λ *=10sec). The second subscription requires data acquisition every 12 sec with latency 6 sec (*p^{acq} *= 12sec,* λ *= 6sec). Subscription requests can be expressed as:*
$s_1 = (d_i, m_T, \beta_i, p_i^{upd}, 4, 10)$
$s_2 = (d_i, m_T, \beta_j, p_j^{upd}, 12, 6)$.

GeNoMe process:

1. *Generate* DOA-PATTERNS*: Subscription requests can be expressed as:*
   $\rho_1 = (\{4, A^{10}\}, 4)$
   $\rho_2 = (\{12, A^6\}, 12)$

2. *Normalize* DOA-PATTERNS*: Normalized length is LCM (4, 12) = 12. Coefficients are* α_1 *= 12/4 = 3 for* ρ_1 *and* α_2 *= 12/12 = 1 for* ρ_2*. Then,* ρ_1 *should be repeated three times and* ρ_2 *should be repeated once:*
   α_1 *x* ρ_1 *= 3 x* $\rho_1 = (\{(4, A^{10}), (8, A^{10}), (12, A^{10})\}, 12)$
   α_2 *x* ρ_2 *= 1 x* $\rho_2 = (\{(12, A^6)\}, 12)$

3. *Merge* DOA-PATTERNS*: The merged pattern has a length of 12, and only the last event (at 12) merges two events (with a latency of MIN (10, 6) = 6):*
   $\rho_{merged} = (\{(4, A^{10}), (8, A^{10}), (12, A^6)\}, 12)$

4. *Build final* SCO-PATTERN*: Transmission actions are added to relevant events at 12. All latency requirements are thus fulfilled:*
   $P_f = (\{(4, A), (8, A), (12, AT)\}, 12)$

Example 4.3.7 *For triple applications: Suppose that there are three subscription requests to the same sensor device* d_i *for measuring temperature* m_T*. The first subscription requires data acquisition every 3*

sec with latency 5 sec (p^{acq} = 3sec, λ =5sec). The second subscription requires data acquisition every 4 sec with latency 10 sec (p^{acq} =4sec, λ = 10sec). The third subscription requires data acquisition every 12 sec with latency 5 sec (p^{acq} = 12sec, λ = 5sec). Subscription can be expressed as:
$s_1 = (d_i, m_T, \beta_i, p_i^{upd}, 3, 5)$
$s_2 = (d_i, m_T, \beta_j, p_j^{upd}, 4, 10)$
$s_3 = (d_i, m_T, \beta_k, p_k^{upd}, 12, 5)$.

`GeNoMe` *process:*

1. *Generate* DOA-PATTERNS*: Subscription requests can be expressed as:*
   $\rho_1 = (\{3, A^5\}, 3)$
   $\rho_2 = (\{4, A^{10}\}, 4)$
   $\rho_3 = (\{12, A^5\}, 12)$

2. *Normalize*DOA-PATTERNS*: Normalized length is LCM (3, 4, 12) = 12. Coefficients are α_1 = 12/3 = 4 for ρ_1 , α_2 = 12/4 = 3 for ρ_2 and α_3 = 12/12 = 1 for ρ_3 . Then, ρ_1 should be repeated four times, ρ_2 should be repeated three times and ρ_3 should be repeated once:*
   α_1 *x* ρ_1 = 4 *x* $\rho_1 = (\{(3, A^5), (6, A^5), (9, A^5), (12, A^5)\}, 12)$
   α_2 *x* ρ_2 = 3 *x* $\rho_2 = (\{(4, A^{10}), (8, A^{10}), (12, A^{10})\}, 12)$
   α_3 *x* ρ_3 = 1 *x* $\rho_3 = (\{(12, A^5)\}, 12)$

3. *Merge* DOA-PATTERNS*: The merged pattern has a length of 12, and only the last event (at 12) merges three events (with a latency of MIN (5, 10, 5) = 5):*
   $\rho_{merged} = (\{(3, A^5), (4, A^{10}), (6, A^5), (8, A^{10}), (9, A^5), (12, A^5)\}, 12)$

4. *Build final* SCO-PATTERN*: Transmission actions are added to relevant events at 8, 12. All latency requirements are thus fulfilled:*
   $P_f = (\{(3, A), (4, A), (6, A), (8, AT), (9, A), (12, AT)\}, 12)$

4.4 Network-Aware 3SoSM Approach

In this section, we present our complete version of our approach `3SoSM` named `Network-Aware 3SoSM` approach. This complete version needs a global view over the network topology to support network-aware optimization process. To present our `Network-Aware 3SoSM` approach, the network topology and the transmission constraints due to the subscription request are first introduced.

4.4.1 Network Topology and Transmission Constraints

To transmit data to the central base station (where the acquired measures are computed), we suppose that deployed wireless sensor devices construct a logical tree (tree topology) where each sensor has only one neighbor to deliver data packets. Each sensor device in this structure may have several lower-layer neighbors but has only one upper layer neighbor. This tree structure provides a uni-path for each sensor to reach the base station (sink device). In fact, our optimization algorithm relies on known and unique paths, as it does not consider a choice of network paths between two devices. However, the tree topology may be updated each time application requirements change, as it triggers a new optimization.

Moreover, we suppose that the base station has a full knowledge about the sensor devices' locations. Routing problems, finding the shortest or optimal path to the base station is a network issue that is not considered in this approach. Besides, the structure of tree has a direct effect on the energy consumption of the sensor device based on the network workload. Here, we suppose that the tree is well balanced

(i.e. balanced number of lower-layer neighbors for each sensor device) and the sensor devices in the environment are distributed fairly. We are also aware that if there is a link break in the uni-path on the active route then communication also breaks [163, 190]. There exist studies focusing on the routing on the tree based topologies in this domain as well [92, 107, 185].

Each sensor device has its own role in the topology based on the application and the location: Source (responsible for the data acquisition and transmission) and/or Relay (responsible for the reception and re-transmission) or Sink (base station, only responsible for the reception).

A tree topology has a layered form (Layer 0-1-2 etc.) based on the distance to the base station. Layer 0 stands for the root of the tree structure that represents the Sink device. It is the base station of the sensor network and aggregates the acquired sensor data. Layer 1 contains the sensor devices that are one hop away from the base station Sink. A sample tree topology is described in the following example.

To form a global schedule, we adopt a similar approach to Galpin et al. [58]. A predefined granularity determines the slots in the time space and the schedule is divided into time-slots. For the following example, granularity is set to 0.5 *sec* (e.g. 20 slots for 10 seconds). The time is divided into slots and each slot may be filled with sensor actions. The major constraint of the system is that all the acquired data must pass through the network topology up to the base station before the expiration of their "latency" in order to fulfill the application requirements. A transmission action during a single time slot stands for a transmission of all the data (acquired by itself and received from lower-layer neighbors) not yet sent, including acquisitions on the same time slot by the device. A transmission action on a sensor device requires a reception action at the same time slot on the sensor device that receives the data. Besides there are other constraints due to the radio protocol such as a device can only receive data from a single device at a time on the same time slot and a device cannot receive and send on the same time slot.

We propose an optimization algorithm to choose the optimal communication slots from the subscription requests. The optimization process starts by propagating transmission and reception constraints based on latencies and on the network topology in a bottom-up process, and then analyzes the possible reception actions of the base station and tries to find the most energy efficient communication slots, and finally propagates those choices to lower-layers and continues the analyze in a top-down process.

Example 4.4.1 *A tree topology with 6 multi-modal sensor devices and a base station is illustrated in Figure 4.3.*

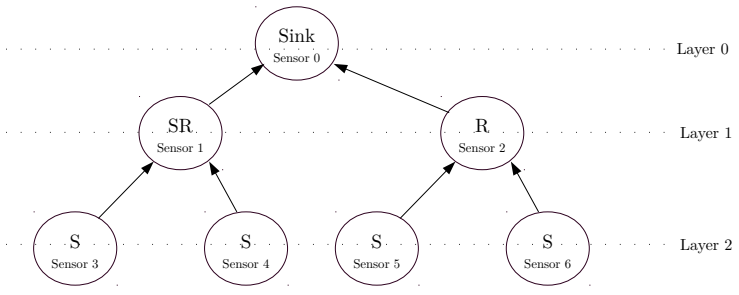


Fig. 4.3 A tree-structured topology and different sensor device roles:
Source (S), Relay (R), Source+Relay (SR), Sink.

Since our aim is to support multi-applicaiton and a sensor device may have different acquisition periodicities from different applications, a common length is highly required. This length is used as a length of the `Schedule Time Pattern` where every sensor action is assiged to a time slot. Besides, our `Network-Aware 3SoSM` approach reqires a global view over the network to manage the topology to achieve a network-aware optimization process for the energy-aware sensor management.

4.4.2 Schedule Model

We use a schedule table to represent the sensor action slots for Acq/Tx/Rx. Each sensor device has its own schedule and each schedule is composed of three parts: acquisition, transmission and reception part. Each line of the schedule table corresponds to an acquired measure: its acquisition for the acquisition part, its transmission action for the transmission part and its reception action for the reception part of the schedule.

Here, we present three principal sensor actions: data acquisition, data transmission and data reception. Data acquisition is executed on the microcontroller of the sensor device, whereas data transmission and reception actions are related to the radio component of the wireless device. Thus, during a single time slot, a sensor device may execute only one action of the subset $\{transmission,\ reception\}$. Besides, the sensor device may execute the data acquisition and transmission/reception during the same time slot, due to the fact that these actions are executed on the different components. Possible sensor actions during a single time slot are the following:

1. only data acquisition $\{A\}$
2. data acquisition and data transmission $\{A,\ T\}$
3. data acquisition and data reception $\{A,\ R\}$
4. only data transmission $\{T\}$
5. only data reception $\{R\}$

The acquisition part of the schedule represents the acquisition actions of a sensor device during the given length of schedule, indicates at what time the sensor acquires data. Only devices that acquire physical measures (source and source-relay devices) have an acquisition part. Figure 4.4 presents the acquisition part model that we adopt for our case. The first column lists the subscriptions for this sensor device with the notation:
`Acq of S[SensorID]: A[SubscriptionNumber].`
The second column indicates the physical quantity (temperature, humidity, etc.). In fact, the number of line of this part indicates the number of subscriptions for this sensor device. The rest of the part represents the acquisition moments during the given length. The notation to indicate the acquisition is the following:
`A[SubscriptionNumber]#[AcquisitionNumber].`

		Time (sec)					
Acquisition	Measure	t=0	t=1 x granularity	t=2 x granularity	...	....	Until t=pattern_length - granularity
Acq of S[sensorID]:A[SubscriptionNumber]	physical quantity	A[SubscriptioNumber]#[AcquisitionNumber]					

Fig. 4.4 Acquisition Part Model.

The transmission part of the schedule indicates the possible transmission slots for the sensor device for each acquired measure. According to the latency information λ, every sensor device has the initiative to keep the data (instead of sending it immediately) and send it at an optimal later moment under the condition that this acquired data arrives at the base station before the expiration of its latency. Figure 4.5 presents the transmission part model to fill the transmission part of the schedule. The transmission part lists all the transmission actions for each acquired and received data. Hence, the first column presents the transmission with the following notation:
`TX of S[SensorID]:A[SubscriptionNumber]#[AcquisitionNumber].`
The second column indicates the destination of the transmission with the notation: `S[SensorID]`. The rest of the part is filled with the latency information of the relevant data.

		Time (sec)					
Transmission	To	t=0	t=1 x granularity	t=2 x granularity	...	....	Until t=pattern_length - granularity
TX of S[SensorID]:A[SubscriptionNumber]#[AcquisitionNumber]	S[SensorID]	residual latency	residual latency – granulary	until latency expires			

Fig. 4.5 Transmission Part Model.

The reception part of the schedule represents the possible reception slots for each data coming from lower-layer neighbors (zero, one or several depending on the network topology). Figure 4.6 presents the model for the reception part where the first column indicates the source that acquired that data with the following notation:
`RX of S[SensorID]: A[SubscriptionNumber]#[AcquisitionNumber].`
Here, the SensorID stands for the sensor device that acquired the data with `[AcquisitionNumber]` for the subscription `[SubscriptionNumber]`. The second column indicates the sensor device that sends that data (not necessarily the source due to the multi-hop behavior). The rest of the part is filled with the latency information of the relevant data.

		Time (sec)					
Reception	From	t=0	t=1 x granularity	t=2 x granularity	...	....	Until t=pattern_length - granularity
RX of S[SensorID]:A[SubscriptionNumber]#[AcquisitionNumber]	S[SensorID]	residual latency	residual latency – granulary	until latency expires			

Fig. 4.6 Reception Part Model.

To present the scheduling mechanism and the energy-aware optimization process intelligibly, from this point, each step will be associated with the same example. Thus, each step of the process will be presented and applied in the same example to provide a continuation of the example. The example is the following:

Example 4.4.2 *For the example described in Example 4.1.2, the common length of schedules is 10 seconds (20 time slots). Time slots for acquisitions are:*
for Application 1 (Temperature, p^{acq}=2sec): t =0sec, t = 2sec, t = 4sec, t = 6sec, t = 8sec
and
for Application 2 (Humidity, p^{acq}=5sec): t = 0sec, t = 5sec.

4.4.3 Schedule Initialization

4.4.3.1 Acquisition Part

For a given sensor device, an acquisition part is generated from the acquisition periodicity p^{acq} defined in the subscriptions to this sensor device and the length of the schedule. Each subscription is represented as a line. Each acquired data is denoted by:
`A[SubscriptionNumber]#[AcquisitionNumber].`

Example 4.4.3 *Acquisition parts of the schedules for each sensor device of the topology are shown in Figure 4.7. Since there is not any subscription to Sensor 2 of the topology, it is a Relay device, hence it does not acquire data.*

The requirements of applications are transformed into subscriptions to the sensor devices. The list of the subscriptions to sensor devices is the following:

- *Sensor 1: One subscription (temperature $p^{acq} = 2sec$), acquisition part of the schedule is presented in Figure 4.7a.*
- *Sensor 2: No subscription, thus no acquisition part for this sensor device.*

		Time(sec)																			
Acquisition	Measure	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5
Acq of S1:A1	temperature	A1#1				A1#2				A1#3				A1#4				A1#5			

(a) Acquisition Part of the Schedule for Sensor 1.

		Time(sec)																			
Acquisition	Measure	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5
Acq of S3:A1	temperature	A1#1				A1#2				A1#3				A1#4				A1#5			
Acq of S3:A2	humidity	A2#1										A2#2									

(b) Acquisition Part of the Schedule for Sensor 3.

		Time(sec)																			
Acquisition	Measure	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5
Acq of S4:A1	humidity	A1#1										A1#2									

(c) Acquisition Part of the Schedule for Sensor 4.

		Time(sec)																			
Acquisition	Measure	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5
Acq of S5:A1	temperature	A1#1				A1#2				A1#3				A1#4				A1#5			
Acq of S5:A2	humidity	A2#1										A2#2									

(d) Acquisition Part of the Schedule for Sensor 5.

		Time(sec)																			
Acquisition	Measure	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5
Acq of S6:A1	humidity	A1#1										A1#2									

(e) Acquisition Part of the Schedule for Sensor 6.

Fig. 4.7 Acquisition Part of the Schedules for each Sensor Device based on the Acquisition Model.

- *Sensor 3: Two subscriptions (temperature $p^{acq} = 2sec$, humidity $p^{acq} = 5sec$), acquisition part of the schedule is presented in Figure 4.7b.*
- *Sensor 4: One subscription (humidity $p^{acq} = 5sec$), acquisition part of the schedule is presented in Figure 4.7c.*
- *Sensor 5: Two subscriptions (temperature $p^{acq} = 2sec$, humidity $p^{acq} = 5sec$), acquisition part of the schedule is presented in Figure 4.7d.*
- *Sensor 6: One subscription (humidity $p^{acq} = 5sec$), acquisition part of the schedule is presented in Figure 4.7e.*

Based on the given subscriptions, acquisition parts of these sensor devices are filled. Here, we believe that the multi-modality and multi-application features of our approach are more visible with the given subscriptions. There are two subscriptions to Sensor 3 and 5 for different physical quantities and with different requirements: The first line represents the subscription to the temperature service and the second line stands for the subscription to the humidity service (see respectively Figure 4.7b and Figure 4.7d).

4.4.3.2 Transmission Part

Information from the acquisition part is used to fill the transmission part of the schedule. For a given sensor device, each acquired data and each received data is represented as a line in the transmission part. A line is filled with the residual latency information starting from the time slot where the device acquires the data or from the time slot following the time slot where the device received the data, and until residual latency reaches 0 (in fact, 1 time slot before at 1 hop from the sink, 2 time slots at 2 hops, etc.). A Source sensor device fills its transmission part with the transmission of its own acquired data whereas a Relay device fills it with the received data.

$T = [e_{i,j}]$ is a 2D matrix with dimensions *m x n*. *m* is the number of data to transmit and *n* stands for the number of time slots. $e_{i,j}$ represents a potential transmission event of data *i* at time slot $j : e_{i,j} = \lambda_{i,j}$ is the residual latency, $e_{i,j} = 0$ means that data *i* can not be transmitted at that time slot.

Example 4.4.4 *Continuing the running example, transmission parts of the schedules for each sensor device are shown in Figure 4.8. For instance, Sensor 4 measures humidity at t=0sec (A1#1) and t=5sec (A1#2). For each acquired data, the latency is 3sec. This latency parameter gives an opportunity to Sensor 4 to keep the acquired data instead of sending it immediately. For the acquired data A1#1, Sensor 4 has 5 slots (t=0,0.5, 1, 1.5, 2 sec) to transmit that data before the latency is expired (see Fig. 4.8b). In case that Sensor 4 sends that data at t>2sec, the latency will be expired before reaching the Sink device due to the fact that Sensor 4 is located in Layer 2 (i.e. two hops away from the base station).*

For the Relay device Sensor 2, the transmission part given in Figure 4.8f, is filled with the data acquired by its lower-layer neighbors (Sensor 5 and 6). For the Source-Relay device, Sensor 1, the transmission part of the schedule given in Figure 4.8e, covers the transmission of its own data (see Figure 4.7a) and also data received from its lower-layer neighbors (i.e. Sensor 3 and 4) (see the transmission parts of these devices: Figure 4.8a and 4.8b).

4.4.3.3 Reception Part

For a given sensor device, the reception part is based on the transmission part of the schedules of lower-layer neighbors. Each line represents a data to receive and is linked to one line from the transmission part of the source device. Possible reception slots are filled with the data residual latency from the corresponding transmission time slot.

$R = [e_{i,j}]$ is a 2D matrix with dimensions *m x n* where *m* is the number of data to receive and *n* stands for the number of time slots. $e_{i,j}$ represents a potential reception event of data *i* at time slot $j : e_{i,j} = \lambda_{i,j}$ is the residual latency, $e_{i,j} = 0$ means that data *i* can not be received at that time slot.

Example 4.4.5 *Continuing the running example, reception parts of each sensor device (if exists) are shown in Figure 4.9. Only the devices that receive network packets have a reception part in their schedule. Since the Sensor 3,4,5,6 are the Source devices, they do not have reception part. Only intermediate devices (located in intermediate layers) and the Sink device have their own reception parts.*

For instance, reception part of Sensor 1 given in Figure 4.9a represents the possible reception slots for each acquired data coming from Sensor 3 and 4 with their residual latencies. For the Sink device (Sensor 0), reception part is composed by the data coming from the Sensor 1 and 2.

4.4.3.4 Overview of Schedule Initialization

In this schedule mechanism, each acquired measure is represented as a line in the schedule table. The critical point is the correlation between the reception and the transmission parts. The transmission part is based on its own reception part. However, the reception part is based on the transmission parts of the lower-layer neighbors.

Example 4.4.6 *Figure 4.10 presents the correlation between the transmission and the reception parts. Here, we focus on the transmission of the acquired data S4 : A1#1 (see Figure 4.7c). We have already introduced how to fill the transmission part of a Source device. Figure 4.10a presents the transmission part of the schedule for Sensor 4. The data S4 : A1#1 is acquired at t=0sec. Sensor 4 has 5 options (in terms of slot) to send this data to Sensor 1 before its latency is expired (at t=0, 0.5, 1, 1.5, 2sec). Now let's check the reception part of the schedule of Sensor 1. Figure 4.10b presents its reception part and*

		Time (sec)																		
Transmission	To	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5
TX of S3:A1#1	S1	4	3.5	3	2.5	2	1.5	1													
TX of S3:A1#2	S1					4	3.5	3	2.5	2	1.5	1									
TX of S3:A1#3	S1									4	3.5	3	2.5	2	1.5	1					
TX of S3:A1#4	S1													4	3.5	3	2.5	2	1.5	1	
TX of S3:A1#5	S1	2	1.5	1														4	3.5	3	2.5
TX of S3:A2#1	S1	3	2.5	2	1.5	1															
TX of S3:A2#2	S1											3	2.5	2	1.5	1					

(a) Transmission Part of the Schedule for Sensor 3.

		Time (sec)																		
Transmission	To	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5
TX of S4:A1#1	S1	3	2.5	2	1.5	1															
TX of S4:A1#2	S1											3	2.5	2	1.5	1					

(b) Transmission Part of the Schedule for Sensor 4.

		Time (sec)																		
Transmission	To	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5
TX of S5:A1#1	S2	4	3.5	3	2.5	2	1.5	1													
TX of S5:A1#2	S2					4	3.5	3	2.5	2	1.5	1									
TX of S5:A1#3	S2									4	3.5	3	2.5	2	1.5	1					
TX of S5:A1#4	S2													4	3.5	3	2.5	2	1.5	1	
TX of S5:A1#5	S2	2	1.5	1														4	3.5	3	2.5
TX of S5:A2#1	S2	3	2.5	2	1.5	1															
TX of S5:A2#2	S2											3	2.5	2	1.5	1					

(c) Transmission Part of the Schedule for Sensor 5.

		Time (sec)																		
Transmission	To	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5
TX of S6:A1#1	S2	3	2.5	2	1.5	1															
TX of S6:A1#2	S2											3	2.5	2	1.5	1					

(d) Transmission Part of the Schedule for Sensor 6.

		Time (sec)																		
Transmission	To	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5
TX of S1:A1#1	S0	4	3.5	3	2.5	2	1.5	1	0.5												
TX of S1:A1#2	S0					4	3.5	3	2.5	2	1.5	1	0.5								
TX of S1:A1#3	S0									4	3.5	3	2.5	2	1.5	1	0.5				
TX of S1:A1#4	S0													4	3.5	3	2.5	2	1.5	1	0.5
TX of S1:A1#5	S0	2	1.5	1	0.5													4	3.5	3	2.5
TX of S3:A1#1	S0		3.5	3	2.5	2	1.5	1	0.5												
TX of S3:A1#2	S0						3.5	3	2.5	2	1.5	1	0.5								
TX of S3:A1#3	S0										3.5	3	2.5	2	1.5	1	0.5				
TX of S3:A1#4	S0														3.5	3	2.5	2	1.5	1	0.5
TX of S3:A1#5	S0	2	1.5	1	0.5														3.5	3	2.5
TX of S3:A2#1	S0		2.5	2	1.5	1	0.5														
TX of S3:A2#2	S0												2.5	2	1.5	1	0.5				
TX of S4:A1#1	S0		2.5	2	1.5	1	0.5														
TX of S4:A1#2	S0												2.5	2	1.5	1	0.5				

(e) Transmission Part of the Schedule for Sensor 1.

		Time (sec)																		
Transmission	To	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5
TX of S5:A1#1	S0		3.5	3	2.5	2	1.5	1	0.5												
TX of S5:A1#2	S0						3.5	3	2.5	2	1.5	1	0.5								
TX of S5:A1#3	S0										3.5	3	2.5	2	1.5	1	0.5				
TX of S5:A1#4	S0														3.5	3	2.5	2	1.5	1	0.5
TX of S5:A1#5	S0	2	1.5	1	0.5														3.5	3	2.5
TX of S5:A2#1	S0		2.5	2	1.5	1	0.5														
TX of S5:A2#2	S0												2.5	2	1.5	1	0.5				
TX of S6:A1#1	S0		2.5	2	1.5	1	0.5														
TX of S6:A1#2	S0												2.5	2	1.5	1	0.5				

(f) Transmission Part of the Schedule for Sensor 2.

Fig. 4.8 Transmission Part of the Schedules for each Sensor Device based on the Transmission Model.

		Time (sec)																		
Reception	From	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5
RX of S3:A1#1	S3	4	3.5	3	2.5	2	1.5	1													
RX of S3:A1#2	S3					4	3.5	3	2.5	2	1.5	1									
RX of S3:A1#3	S3									4	3.5	3	2.5	2	1.5	1					
RX of S3:A1#4	S3													4	3.5	3	2.5	2	1.5	1	
RX of S3:A1#5	S3	2	1.5	1														4	3.5	3	2.5
RX of S3:A2#1	S3	3	2.5	2	1.5	1															
RX of S3:A2#2	S3											3	2.5	2	1.5	1					
RX of S4:A1#1	S4	3	2.5	2	1.5	1															
RX of S4:A1#2	S4											3	2.5	2	1.5	1					

(a) Reception Part of the Schedule for Sensor 1.

		Time (sec)																		
Reception	From	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5
RX of S5:A1#1	S5	4	3.5	3	2.5	2	1.5	1													
RX of S5:A1#2	S5					4	3.5	3	2.5	2	1.5	1									
RX of S5:A1#3	S5									4	3.5	3	2.5	2	1.5	1					
RX of S5:A1#4	S5													4	3.5	3	2.5	2	1.5	1	
RX of S5:A1#5	S5	2	1.5	1														4	3.5	3	2.5
RX of S5:A2#1	S5	3	2.5	2	1.5	1															
RX of S5:A2#2	S5											3	2.5	2	1.5	1					
RX of S6:A1#1	S6	3	2.5	2	1.5	1															
RX of S6:A1#2	S6											3	2.5	2	1.5	1					

(b) Reception Part of the Schedule for Sensor 2.

		Time (sec)																		
Reception	From	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5
RX of S1:A1#1	S1	4	3.5	3	2.5	2	1.5	1	0.5												
RX of S1:A1#2	S1					4	3.5	3	2.5	2	1.5	1	0.5								
RX of S1:A1#3	S1									4	3.5	3	2.5	2	1.5	1	0.5				
RX of S1:A1#4	S1													4	3.5	3	2.5	2	1.5	1	0.5
RX of S1:A1#5	S1	2	1.5	1	0.5													4	3.5	3	2.5
RX of S3:A1#1	S1		3.5	3	2.5	2	1.5	1	0.5												
RX of S3:A1#2	S1						3.5	3	2.5	2	1.5	1	0.5								
RX of S3:A1#3	S1										3.5	3	2.5	2	1.5	1	0.5				
RX of S3:A1#4	S1														3.5	3	2.5	2	1.5	1	0.5
RX of S3:A1#5	S1	2	1.5	1	0.5														3.5	3	2.5
RX of S3:A2#1	S1		2.5	2	1.5	1	0.5														
RX of S3:A2#2	S1												2.5	2	1.5	1	0.5				
RX of S4:A1#1	S1		2.5	2	1.5	1	0.5														
RX of S4:A1#2	S1												2.5	2	1.5	1	0.5				
RX of S5:A1#1	S2		3.5	3	2.5	2	1.5	1	0.5												
RX of S5:A1#2	S2						3.5	3	2.5	2	1.5	1	0.5								
RX of S5:A1#3	S2										3.5	3	2.5	2	1.5	1	0.5				
RX of S5:A1#4	S2														3.5	3	2.5	2	1.5	1	0.5
RX of S5:A1#5	S2	2	1.5	1	0.5														3.5	3	2.5
RX of S5:A2#1	S2		2.5	2	1.5	1	0.5														
RX of S5:A2#2	S2												2.5	2	1.5	1	0.5				
RX of S6:A1#1	S2		2.5	2	1.5	1	0.5														
RX of S6:A1#2	S2												2.5	2	1.5	1	0.5				

(c) Reception Part of the Schedule for Sink (Sensor 0).

Fig. 4.9 Reception Part of the Schedules for each Sensor Device based on the Reception Model.

		Time (sec)																		
Transmission	To	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5
TX of S4:A1#1	S1	3	2.5	2	1.5	1															
TX of S4:A1#2	S1											3	2.5	2	1.5	1					

(a) Transmission Part of the Schedule for Sensor 4 (Focus on Transmission of $S4 : A1\#1$).

		Time (sec)																		
Reception	From	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5
RX of S3:A1#1	S3	4	3.5	3	2.5	2	1.5	1													
RX of S3:A1#2	S3					4	3.5	3	2.5	2	1.5	1									
RX of S3:A1#3	S3									4	3.5	3	2.5	2	1.5	1					
RX of S3:A1#4	S3													4	3.5	3	2.5	2	1.5	1	
RX of S3:A1#5	S3	2	1.5	1														4	3.5	3	2.5
RX of S3:A2#1	S3	3	2.5	2	1.5	1															
RX of S3:A2#2	S3											3	2.5	2	1.5	1					
RX of S4:A1#1	S4	3	2.5	2	1.5	1															
RX of S4:A1#2	S4											3	2.5	2	1.5	1					

(b) Reception Part of the Schedule for Sensor 1 (Focus on Reception of $S4 : A1\#1$).

		Time (sec)																		
Transmission	To	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5
TX of S1:A1#1	S0	4	3.5	3	2.5	2	1.5	1	0.5												
TX of S1:A1#2	S0					4	3.5	3	2.5	2	1.5	1	0.5								
TX of S1:A1#3	S0									4	3.5	3	2.5	2	1.5	1	0.5				
TX of S1:A1#4	S0													4	3.5	3	2.5	2	1.5	1	0.5
TX of S1:A1#5	S0	2	1.5	1	0.5													4	3.5	3	2.5
TX of S3:A1#1	S0		3.5	3	2.5	2	1.5	1	0.5												
TX of S3:A1#2	S0						3.5	3	2.5	2	1.5	1	0.5								
TX of S3:A1#3	S0										3.5	3	2.5	2	1.5	1	0.5				
TX of S3:A1#4	S0														3.5	3	2.5	2	1.5	1	0.5
TX of S3:A1#5	S0	2	1.5	1	0.5														3.5	3	2.5
TX of S3:A2#1	S0		2.5	2	1.5	1	0.5														
TX of S3:A2#2	S0												2.5	2	1.5	1	0.5				
TX of S4:A1#1	S0		2.5	2	1.5	1	0.5														
TX of S4:A1#2	S0												2.5	2	1.5	1	0.5				

(c) Transmission Part of the Schedule for Sensor 1 (Focus on Transmission of $S4 : A1\#1$).

Fig. 4.10 Example of a Correlation between the Transmission Part and the Reception Part of the Schedule.

focuses the reception of the data $S4 : A1\#1$. Since the reception part of Sensor 1 is a direct reflexion of the transmission part of Sensor 4, we fill the same slots for possible reception time with the same latency information. Afterward, we fill the transmission part of schedule for Sensor 1. Figure 4.10c presents the transmission part for this sensor device. Here, we update the latency information while filling the cases. Suppose that Sensor 4 sends the data at t=0sec (λ = 3sec). In this case Sensor 1 receives the data at t=0sec (still λ = 3sec) and it may send that data starting from t=0.5sec. Sensor 1 has 4 options (4 slots) to send that data to Sink (at t=0.5, 1, 1.5, 2sec). We fill these cases as possibilities of transmission of this data with its updated latency. Sensor 1 should have sent that data until t=2.5sec. If Sensor 1 sends that data at t=2.5 sec, then the latency will expire before the data reaches to Sink device which causes an unwilling case.

Flowchart of the schedule generation for the presented example is illustrated in Figure 4.11. It introduces each step of the bottom-up process and the order of the steps. The given flowchart explains how the reception and transmission parts are filled. For instance, Sensor 1 fills its own transmission part with respect to the transmission part of Sensor 3 and 4, and its own acquisition part. Hence, we use a bottom-up computation method to propagate correctly the constraints (through data latencies) to the upper layers. The process starts from the leaves of the tree and passes to upper layer once it has computed all the nodes of that layer.

4.4.4 Searching Optimal Communication Slots

For each device, potential transmission and reception slots are now identified in their schedule table. The optimization process should now choose the most energy-efficient slots, in order to minimize the number of Tx/Rx slots for each device.

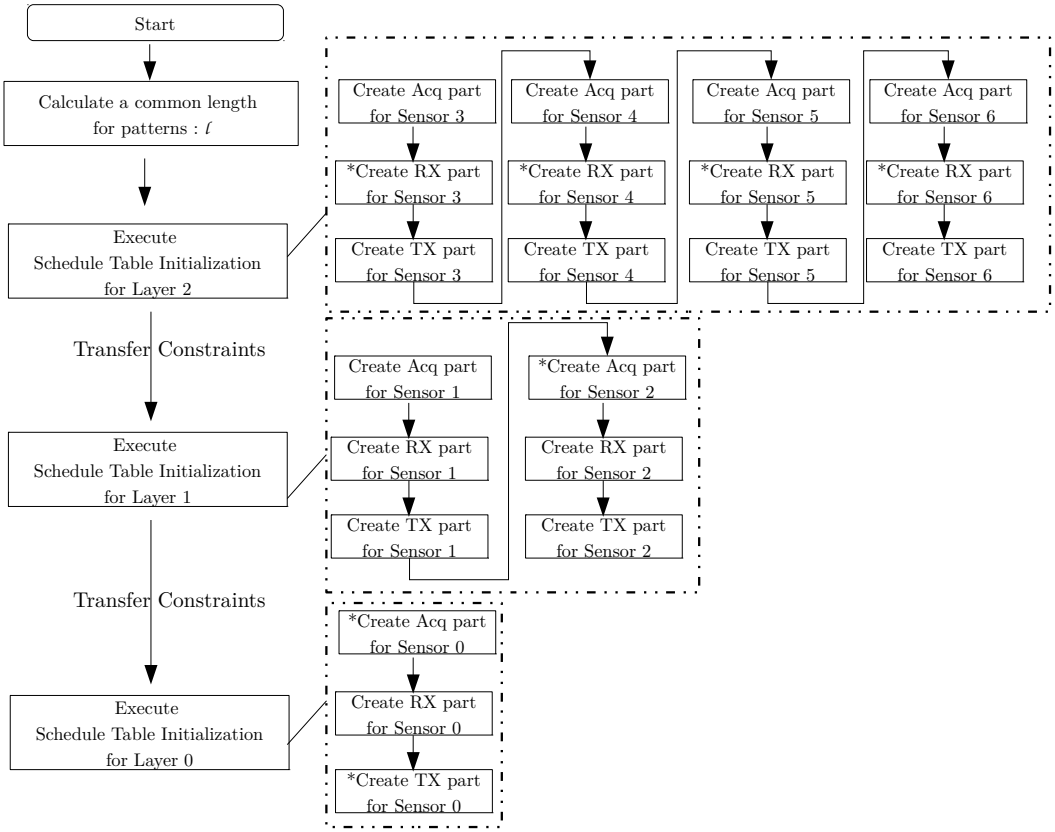


Fig. 4.11 Flowchart of the Schedule Mechanism of Network-Aware 3SoSM Approach.
Steps skipped due to the sensor role are marked with *.

In a tree-structured topology, it is highly possible to encounter a bottleneck effect that widely exists in WSN and leads to decrease the lifetime of the network [64, 94]. Therefore, we start to optimize the communication slots from the upper layers and we propose a top-down computation method to find the optimal energy-aware communication slots for each sensor.

A schedule is valid if each acquired data is transmitted to the base station before its latency expires. Here, we present a method for searching the optimal communication slots. This process is responsible for determining the optimal transmission/reception slots from the schedule for each sensor device in the network. This process analyzes reception part of the schedules and determines the optimal case. For choosing the optimal communication slots, a cost model is defined.

Cost Model

Our cost model is primarily based on the energy consumption: we focus on the energy consumption of each sensor device. It is clear that sending or receiving multiple data in a single time slot should consume less energy than sending or receiving in separate slots [76]. Hence, our cost model considers the number of transmission and reception slots as the main metric.

As a crucial remark, a sensor device may receive several data from the same sensor but cannot receive data from several sensors at the same time. Multiple packets can be received from different source devices on the condition that for each different sources, a different antenna is used (which is exceptional and

barely preferred due to the new challenges coming with) [21]. Here, we suppose that a sensor device in our environment has only one antenna for the communication. So that, the major issue is to minimize the communication slots and to maximize the number of packets sent/received during a single time slot while being sure that each acquired data will reach the base station before its latency expires.

The secondary metric of the cost model is the time to differentiate cases with the same number of transmission/reception slot. For these cases, our cost model will consider the residual latency. Since the tree-structured topology may cause a bottleneck effect (especially for the upper-layers of the tree), the latest possible reception or transmission slot is preferred as an optimal communication slot.

Searching the Optimal Solution

Based on the introduced metrics, the optimization algorithm is executed on the reception part of the schedule. We propose creating an overview matrix that presents a grouping of possible receptions during a single time slot for each lower-layer sender (neighbor that sends data). The overview matrix has the same length as the reception part and each line is allocated for a lower-layer sender. Each cell of that matrix represents the total number of possible reception from the relevant device during the relevant slot. Our algorithm analyzes the reception part of the schedules of each sensor device starting from the root of the topology which is the Sink device. This analysis uses a top-down computation method to process searching optimal solution based on our cost model.

The algorithm firstly searches every possibility and tries to figure out whether groupings of multiple transmission or reception can be possible or not. Since multiple data transmission to the same device or multiple receptions from the same device within the same time slot consumes less energy, our algorithm chooses the cases where data transmissions and receptions are grouped. Among the possible optimal cases (equal amount of consumed energy), our algorithm takes into account the time of the slots: Our algorithm decides the latest possible transmission or reception. Indeed, the algorithm analyses only the reception part of the schedules. Once the optimal communication slot is determined, relevant transmission slots of the lower-layer neighbors are automatically determined as well.

From the reception part of the schedule, the algorithm tries to group the receptions from the same source sensor device. For each time slot, it calculates the total number of possible receptions for each neighbor. The algorithm searches the latest slot in which maximum number of reception can occur. It searches the latest slot to create more opportunities for the lower-layer devices. Otherwise, it would force the lower-layer devices to send the data earlier.

Example 4.4.7 *The algorithm for searching optimal communication slot firstly analyses the reception part of the schedule and then creates an overview matrix. In fact, it gives the number of total possible receptions for each lower-layer sender. While creating an overview matrix for the optimization algorithm, it creates a list for managing the slot occupancy. The initial overview matrix based on the reception part of the Sensor 0 (sink device) and the initial slot occupancy list are given in Figure 4.12.*

	Time (sec)																			
From	**0**	**0.5**	**1**	**1.5**	**2**	**2.5**	**3**	**3.5**	**4**	**4.5**	**5**	**5.5**	**6**	**6.5**	**7**	**7.5**	**8**	**8.5**	**9**	**9.5**
S1	**3**	**6**	**6**	**6**	**5**	**6**	**4**	**4**	**3**	**4**	**4**	**6**	**5**	**6**	**6**	**6**	**3**	**4**	**4**	**4**
S2	**1**	**4**	**4**	**4**	**3**	**4**	**2**	**2**	**1**	**2**	**2**	**4**	**3**	**4**	**4**	**4**	**1**	**2**	**2**	**2**
slot occupancy	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Fig. 4.12 Initial Slot Occupancy List and the overview matrix. (Before starting the process)

Once the initial overview matrix is generated, the algorithm finds the latest slot where maximum number of reception is possible (firstly for Sensor 1 and then for Sensor 2). For the Sensor 1's round,

the algorithm chooses t=7.5sec since the latest maximum possible reception is 6. Once the optimal communication slot is found, slot occupancy list is updated and the slot for t=7.5sec becomes occupied. Then the algorithm repeats the same execution for Sensor 2. Proposed algorithm chooses t=7sec. Normally t=7.5sec is the latest maximum however t=7.5sec is already occupied thus the optimal communication slot for Sensor 2 is set to t=7sec. The status of the overview matrix and the slot occupancy after the first iteration are given in Figure 4.13:

	Time (sec)																			
From	**0**	**0.5**	**1**	**1.5**	**2**	**2.5**	**3**	**3.5**	**4**	**4.5**	**5**	**5.5**	**6**	**6.5**	**7**	**7.5**	**8**	**8.5**	**9**	**9.5**
S1	**3**	**6**	**6**	**6**	**5**	**6**	**4**	**4**	**3**	**4**	**4**	**6**	**5**	**6**	**6**	**6**	**3**	**4**	**4**	**4**
S2	**1**	**4**	**4**	**4**	**3**	**4**	**2**	**2**	**1**	**2**	**2**	**4**	**3**	**4**	**4**	**4**	**1**	**2**	**2**	**2**
slot occupancy	-	-	-	-	-	-	-	-	-	-	-	-	-	-	**X**	**X**	-	-	-	-

Fig. 4.13 Choice of optimal communication slot on the overview matrix and application on the Slot Occupancy List (At the end of 1st iteration) (for Sensor 1 and Sensor 2)

The first slot for the reception is determined. Before passing to the next iteration, the reception part should be updated. Since the t=7sec and t=7.5 sec are the fixed reception times, then the affected lines in the reception parts go out of consideration. For instance, line 3 represents the reception of A1#3 from Sensor 1. This data can be received from t=4sec to t=7.5sec. Since t=7.5sec is reserved for the reception from Sensor 1 then this data can be sent during that slot and the other possible slots get free. Once the reception part is updated, our algorithm repeats the same algorithm. The updated reception part and the next iteration over the updated overview matrix are given in Figure 4.14 and Figure 4.15:

		Time (sec)																			
Reception	From	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5
RX of S1:A1#1	S1	4	3.5	3	2.5	2	1.5	1	0.5												
RX of S1:A1#2	S1					4	3.5	3	2.5	2	1.5	1	0.5								
RX of S1:A1#3	S1									4	3.5	3	2.5	2	1.5	1	0.5				
RX of S1:A1#4	S1													4	3.5	3	2.5	2	1.5	1	0.5
RX of S1:A1#5	S1	2	1.5	1	0.5													4	3.5	3	2.5
RX of S3:A1#1	S1		3.5	3	2.5	2	1.5	1	0.5												
RX of S3:A1#2	S1						3.5	3	2.5	2	1.5	1	0.5								
RX of S3:A1#3	S1										3.5	3	2.5	2	1.5	1	0.5				
RX of S3:A1#4	S1														3.5	3	2.5	2	1.5	1	0.5
RX of S3:A1#5	S1	2	1.5	1	0.5														3.5	3	2.5
RX of S3:A2#1	S1		2.5	2	1.5	1	0.5														
RX of S3:A2#2	S1												2.5	2	1.5	1	0.5				
RX of S4:A1#1	S1		2.5	2	1.5	1	0.5														
RX of S4:A1#2	S1												2.5	2	1.5	1	0.5				
RX of S5:A1#1	S2		3.5	3	2.5	2	1.5	1	0.5												
RX of S5:A1#2	S2						3.5	3	2.5	2	1.5	1	0.5								
RX of S5:A1#3	S2										3.5	3	2.5	2	1.5	1	0.5				
RX of S5:A1#4	S2														3.5	3	2.5	2	1.5	1	0.5
RX of S5:A1#5	S2	2	1.5	1	0.5														3.5	3	2.5
RX of S5:A2#1	S2		2.5	2	1.5	1	0.5														
RX of S5:A2#2	S2												2.5	2	1.5	1	0.5				
RX of S6:A1#1	S2		2.5	2	1.5	1	0.5														
RX of S6:A1#2	S2												2.5	2	1.5	1	0.5				

Fig. 4.14 Updated Reception part after choosing communication slots for Sensor 1 and Sensor 2 Shadowed data are successfully assigned to relevant slots (At the end of 1st iteration).

	Time (sec)																			
From	**0**	**0.5**	**1**	**1.5**	**2**	**2.5**	**3**	**3.5**	**4**	**4.5**	**5**	**5.5**	**6**	**6.5**	**7**	**7.5**	**8**	**8.5**	**9**	**9.5**
S1	**3**	**6**	**6**	**6**	**5**	**6**	**4**	**4**	**2**	**2**	**2**	**2**	**0**	**0**	**0**	**0**	**1**	**2**	**2**	**2**
S2	**1**	**4**	**4**	**4**	**3**	**4**	**2**	**2**	**1**	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**
slot occupancy	-	-	-	**X**	-	**X**	-	-	-	-	-	-	-	-	**X**	**X**	-	-	-	-

Fig. 4.15 Choice of optimal communication slot on the overview matrix and application on the Slot Occupancy List (At the end of 2nd iteration) (for Sensor 1 and Sensor 2).

In that iteration, for Sensor 1, the latest maximum possible reception is at t=2.5sec and for Sensor 2, is t=1.5sec (again t=2.5sec got occupied by Sensor 1). At the end of the iteration, the reception part

is again updated and the overview matrix is regenerated. The updated reception part is illustrated in Figure 4.16:

		Time (sec)																		
Reception	From	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5
RX of S1:A1#1	S1	4	3.5	3	2.5	2	1.5	1	0.5												
RX of S1:A1#2	S1					4	3.5	3	2.5	2	1.5	1	0.5								
RX of S1:A1#3	S1									4	3.5	3	2.5	2	1.5	1	0.5				
RX of S1:A1#4	S1													4	3.5	3	2.5	2	1.5	1	0.5
RX of S1:A1#5	S1	2	1.5	1	0.5													4	3.5	3	2.5
RX of S3:A1#1	S1		3.5	3	2.5	2	1.5	1	0.5												
RX of S3:A1#2	S1						3.5	3	2.5	2	1.5	1	0.5								
RX of S3:A1#3	S1										3.5	3	2.5	2	1.5	1	0.5				
RX of S3:A1#4	S1														3.5	3	2.5	2	1.5	1	0.5
RX of S3:A1#5	S1	2	1.5	1	0.5														3.5	3	2.5
RX of S3:A2#1	S1		2.5	2	1.5	1	0.5														
RX of S3:A2#2	S1												2.5	2	1.5	1	0.5				
RX of S4:A1#1	S1		2.5	2	1.5	1	0.5														
RX of S4:A1#2	S1												2.5	2	1.5	1	0.5				
RX of S5:A1#1	S2		3.5	3	2.5	2	1.5	1	0.5												
RX of S5:A1#2	S2						3.5	3	2.5	2	1.5	1	0.5								
RX of S5:A1#3	S2										3.5	3	2.5	2	1.5	1	0.5				
RX of S5:A1#4	S2														3.5	3	2.5	2	1.5	1	0.5
RX of S5:A1#5	S2	2	1.5	1	0.5														3.5	3	2.5
RX of S5:A2#1	S2		2.5	2	1.5	1	0.5														
RX of S5:A2#2	S2												2.5	2	1.5	1	0.5				
RX of S6:A1#1	S2		2.5	2	1.5	1	0.5														
RX of S6:A1#2	S2												2.5	2	1.5	1	0.5				

Fig. 4.16 Updated Reception part after choosing communication slots for Sensor 1 and Sensor 2 Shadowed data are successfully assigned to relevant slots (At the end of 2^{nd} iteration).

Figure 4.17 presents the updated overview matrix and the determined optimal reception slots for the next iteration: t=9.5sec for Sensor 1 and t=5.5sec for Sensor 2.

	Time (sec)																			
From	**0**	**0.5**	**1**	**1.5**	**2**	**2.5**	**3**	**3.5**	**4**	**4.5**	**5**	**5.5**	**6**	**6.5**	**7**	**7.5**	**8**	**8.5**	**9**	**9.5**
S1	**2**	**2**	**2**	**2**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**2**	**2**	**2**
S2	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
slot occupancy	**-**	**-**	**-**	**X**	**-**	**X**	**-**	**-**	**-**	**-**	**-**	**X**	**-**	**-**	**X**	**X**	**-**	**-**	**-**	**X**

Fig. 4.17 Choice of optimal communication slot on the overview matrix and application on the Slot Occupancy List (At the end of 3^{rd} iteration) (for Sensor 1 and Sensor 2).

At the end of that iteration, the updated reception part is illustrated in Figure 4.18. All the data are successfully assigned with an optimal slot for reception.

		Time (sec)																		
Reception	From	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5
RX of S1:A1#1	S1	4	3.5	3	2.5	2	1.5	1	0.5												
RX of S1:A1#2	S1					4	3.5	3	2.5	2	1.5	1	0.5								
RX of S1:A1#3	S1									4	3.5	3	2.5	2	1.5	1	0.5				
RX of S1:A1#4	S1													4	3.5	3	2.5	2	1.5	1	0.5
RX of S1:A1#5	S1	2	1.5	1	0.5													4	3.5	3	2.5
RX of S3:A1#1	S1		3.5	3	2.5	2	1.5	1	0.5												
RX of S3:A1#2	S1						3.5	3	2.5	2	1.5	1	0.5								
RX of S3:A1#3	S1										3.5	3	2.5	2	1.5	1	0.5				
RX of S3:A1#4	S1														3.5	3	2.5	2	1.5	1	0.5
RX of S3:A1#5	S1	2	1.5	1	0.5														3.5	3	2.5
RX of S3:A2#1	S1		2.5	2	1.5	1	0.5														
RX of S3:A2#2	S1												2.5	2	1.5	1	0.5				
RX of S4:A1#1	S1		2.5	2	1.5	1	0.5														
RX of S4:A1#2	S1												2.5	2	1.5	1	0.5				
RX of S5:A1#1	S2		3.5	3	2.5	2	1.5	1	0.5												
RX of S5:A1#2	S2						3.5	3	2.5	2	1.5	1	0.5								
RX of S5:A1#3	S2										3.5	3	2.5	2	1.5	1	0.5				
RX of S5:A1#4	S2														3.5	3	2.5	2	1.5	1	0.5
RX of S5:A1#5	S2	2	1.5	1	0.5														3.5	3	2.5
RX of S5:A2#1	S2		2.5	2	1.5	1	0.5														
RX of S5:A2#2	S2												2.5	2	1.5	1	0.5				
RX of S6:A1#1	S2		2.5	2	1.5	1	0.5														
RX of S6:A1#2	S2												2.5	2	1.5	1	0.5				

Fig. 4.18 Updated Reception part after choosing communication slots for Sensor 1 and Sensor 2 Shadowed data are successfully assigned to relevant slots, All possible receptions are successfully distributed over slots.

The updated overview matrix is presented in Figure 4.19. The optimization process continues until all the cells of the overview matrix becomes zero which indicates that there is no more data to receive.

	Time (sec)																			
From	**0**	**0.5**	**1**	**1.5**	**2**	**2.5**	**3**	**3.5**	**4**	**4.5**	**5**	**5.5**	**6**	**6.5**	**7**	**7.5**	**8**	**8.5**	**9**	**9.5**
S1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
S2	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Fig. 4.19 Updated overview matrix at the end of the process. All data are assigned to relevant slots, no more data to assign.

The pseudo code for the optimization algorithm is given in Algorithm 2.

Once the optimal reception slots are determined by the optimization algorithm, the determined slots are applied to lower-layer senders: Transmission parts of these senders are updated and finalized. Then, the reception parts of the lower-layer senders are updated based on new constraints from their finalized transmission parts: data must be received before being transmitted.

The finalized parts of the schedules for each wireless sensor device (Sensor 0 to Sensor 6) are represented respectively in Figures 4.20, 4.21, 4.22, 4.23, 4.24, 4.25,4.26:

		Time (sec)																		
Reception	From	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5
RX of S1:A1#1	S1	4	3.5	3	2.5	2	1.5	1	0.5												
RX of S1:A1#2	S1					4	3.5	3	2.5	2	1.5	1	0.5								
RX of S1:A1#3	S1									4	3.5	3	2.5	2	1.5	1	0.5				
RX of S1:A1#4	S1													4	3.5	3	2.5	2	1.5	1	0.5
RX of S1:A1#5	S1	2	1.5	1	0.5													4	3.5	3	2.5
RX of S3:A1#1	S1		3.5	3	2.5	2	1.5	1	0.5												
RX of S3:A1#2	S1						3.5	3	2.5	2	1.5	1	0.5								
RX of S3:A1#3	S1										3.5	3	2.5	2	1.5	1	0.5				
RX of S3:A1#4	S1														3.5	3	2.5	2	1.5	1	0.5
RX of S3:A1#5	S1	2	1.5	1	0.5														3.5	3	2.5
RX of S3:A2#1	S1		2.5	2	1.5	1	0.5														
RX of S3:A2#2	S1												2.5	2	1.5	1	0.5				
RX of S4:A1#1	S1		2.5	2	1.5	1	0.5														
RX of S4:A1#2	S1												2.5	2	1.5	1	0.5				
RX of S5:A1#1	S2		3.5	3	2.5	2	1.5	1	0.5												
RX of S5:A1#2	S2						3.5	3	2.5	2	1.5	1	0.5								
RX of S5:A1#3	S2										3.5	3	2.5	2	1.5	1	0.5				
RX of S5:A1#4	S2														3.5	3	2.5	2	1.5	1	0.5
RX of S5:A1#5	S2	2	1.5	1	0.5														3.5	3	2.5
RX of S5:A2#1	S2		2.5	2	1.5	1	0.5														
RX of S5:A2#2	S2												2.5	2	1.5	1	0.5				
RX of S6:A1#1	S2		2.5	2	1.5	1	0.5														
RX of S6:A1#2	S2												2.5	2	1.5	1	0.5				

Fig. 4.20 Final Reception Part of the Schedule for Sink (Sensor 0) after Searching Optimal Communication Slot Process.

		Time(sec)																		
Acquisition	Measure	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5
Acq of S1:A1	temperature	A1#1				A1#2				A1#3				A1#4				A1#5			

(a) Acquisition Part of the Schedule for Sensor 1.

		Time (sec)																			
Reception	From	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5
RX of S3:A1#1	S3	4	3.5	3	2.5	2	1.5	1													
RX of S3:A1#2	S3					4	3.5	3	2.5	2	1.5	1									
RX of S3:A1#3	S3									4	3.5	3	2.5	2	1.5	1					
RX of S3:A1#4	S3													4	3.5	3	2.5	2	1.5	1	
RX of S3:A1#5	S3	2	1.5	1														4	3.5	3	2.5
RX of S3:A2#1	S3	3	2.5	2	1.5	1															
RX of S3:A2#2	S3											3	2.5	2	1.5	1					
RX of S4:A1#1	S4	3	2.5	2	1.5	1															
RX of S4:A1#2	S4											3	2.5	2	1.5	1					

(b) Finalized Reception Part of the Schedule for Sensor 1.

		Time (sec)																			
Transmission	To	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5
TX of S1:A1#1	S0	4	3.5	3	2.5	2	1.5	1	0.5												
TX of S1:A1#2	S0					4	3.5	3	2.5	2	1.5	1	0.5								
TX of S1:A1#3	S0									4	3.5	3	2.5	2	1.5	1	0.5				
TX of S1:A1#4	S0													4	3.5	3	2.5	2	1.5	1	0.5
TX of S1:A1#5	S0	2	1.5	1	0.5													4	3.5	3	2.5
TX of S3:A1#1	S0		3.5	3	2.5	2	1.5	1	0.5												
TX of S3:A1#2	S0						3.5	3	2.5	2	1.5	1	0.5								
TX of S3:A1#3	S0										3.5	3	2.5	2	1.5	1	0.5				
TX of S3:A1#4	S0														3.5	3	2.5	2	1.5	1	0.5
TX of S3:A1#5	S0	2	1.5	1	0.5														3.5	3	2.5
TX of S3:A2#1	S0		2.5	2	1.5	1	0.5														
TX of S3:A2#2	S0												2.5	2	1.5	1	0.5				
TX of S4:A1#1	S0		2.5	2	1.5	1	0.5														
TX of S4:A1#2	S0												2.5	2	1.5	1	0.5				

(c) Finalized Transmission Part of the Schedule for Sensor 1.

Fig. 4.21 Final Acquisition, Reception and Transmission Parts of the Schedule for Sensor 1 after Searching Optimal Communication Slot Process.

Algorithm 2 Pseudo Code of Optimization Process.

```
1:  procedure OPTIMIZE()
2:      /* optimization algorithm is executed on the reception part */
3:      /* fetch lower-layer senders */
4:      lower_layer_senders[] /* empty list */
5:      /* fill the lower-layer senders list */
6:      for i = 0; i < line; i++ do
7:          if NOT lower_layer_senders.contains(reception_part[i].sender) then
8:              lower_layer_senders.add(reception_part[i].sender)
9:          end if
10:     end for
11:     /* create a matrix for overview. dimensions: */
12:     overview[][] = new[lower_layer_senders.size][schedule_length]
13:     /* for each slot (column), count how many possible reception can occur from each sender */
14:     for j = 0; j < schedule_length; j++ do
15:         for i = 0; i < reception_part.size; i++ do
16:             if reception_part[i][j] != 0 then
17:                 overview[reception_part[i].sender][j]++
18:                 /* get sensorID from reception part and increment relevant overview line */
19:             end if
20:         end for
21:     end for
22:     slot_occupation[] = new[schedule_length]
23:     max = 0, index = 0
24:     while NOT isEmpty(overview) do /* until overview matrix becomes totally empty */
25:         for i = 0; i < lower_layer_senders.size; i++ do
26:             for j = 0; j < column_overview; j++ do
27:                 if overview[i][j] >= max AND slot_occupation[j] == NULL then
28:                     /* to find the latest max value for each sender */
29:                     max = overview[i][j]
30:                     index = j /* index of the slot that holds the maximum reception */
31:                 end if
32:             end for
33:             slot_occupation[index] = i /* reservation to that slot for the reception */
34:             for k = 1; k < line; k++ do
35:                 if reception_part[k].sender == lower_layer_senders[i] AND reception_part[k][index] != 0 then
36:                     /* once optimal reception slot is found, reception part should be updated before the next round */
37:                     for l = 0; l < schedule_length; l++ do
38:                         reception_part[k][l] = 0 /* delete the latency information on the relevant lines */
39:                     end for
40:                 end if
41:             end for
42:             overview = 0 /* before the next round, overview matrix should be updated */
43:             for m = 0; m < column; m++ do
44:                 for n = 0; n < column; n++ do
45:                     if reception_part[m][n] != 0 then
46:                         overview[reception_part[m].sender][n]++
47:                         /* get sensorID from reception part and increment relevant overview line */
48:                     end if
49:                 end for
50:             end for
51:         end for
52:     end while
53: end procedure
```

		Time (sec)																			
Reception	From	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5
RX of S5:A1#1	S5	4	3.5	3	2.5	2	1.5	1													
RX of S5:A1#2	S5					4	3.5	3	2.5	2	1.5	1									
RX of S5:A1#3	S5									4	3.5	3	2.5	2	1.5	1					
RX of S5:A1#4	S5													4	3.5	3	2.5	2	1.5	1	0.5
RX of S5:A1#5	S5	2	1.5	1														4	3.5	3	2.5
RX of S5:A2#1	S5	3	2.5	2	1.5	1															
RX of S5:A2#2	S5											3	2.5	2	1.5	1					
RX of S6:A1#1	S6	3	2.5	2	1.5	1															
RX of S6:A1#2	S6											3	2.5	2	1.5	1					

(a) Finalized Reception Part of the Schedule for Sensor 2.

		Time (sec)																			
Transmission	To	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5
TX of S5:A1#1	S0		3.5	3	2.5	2	1.5	1	0.5												
TX of S5:A1#2	S0						3.5	3	2.5	2	1.5	1	0.5								
TX of S5:A1#3	S0										3.5	3	2.5	2	1.5	1	0.5				
TX of S5:A1#4	S0														3.5	3	2.5	2	1.5	1	0.5
TX of S5:A1#5	S0	2	1.5	1	0.5														3.5	3	2.5
TX of S5:A2#1	S0		2.5	2	1.5	1	0.5														
TX of S5:A2#2	S0												2.5	2	1.5	1	0.5				
TX of S6:A1#1	S0		2.5	2	1.5	1	0.5														
TX of S6:A1#2	S0												2.5	2	1.5	1	0.5				

(b) Finalized Transmission Part of the Schedule for Sensor 2.

Fig. 4.22 Final Reception and Transmission Parts of the Schedule for Sensor 2 after Searching Optimal Communication Slot Process.

		Time(sec)																			
Acquisition	Measure	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5
Acq of S3:A1	temperature	A1#1				A1#2				A1#3				A1#4				A1#5			
Acq of S3:A2	humidity	A2#1										A2#2									

(a) Acquisition Part of the Schedule for Sensor 3.

		Time (sec)																			
Transmission	To	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5
TX of S3:A1#1	S1	4	3.5	3	2.5	2	1.5	1													
TX of S3:A1#2	S1					4	3.5	3	2.5	2	1.5	1									
TX of S3:A1#3	S1									4	3.5	3	2.5	2	1.5	1					
TX of S3:A1#4	S1													4	3.5	3	2.5	2	1.5	1	
TX of S3:A1#5	S1	2	1.5	1														4	3.5	3	2.5
TX of S3:A2#1	S1	3	2.5	2	1.5	1															
TX of S3:A2#2	S1											3	2.5	2	1.5	1					

(b) Finalized Transmission Part of the Schedule for Sensor 3.

Fig. 4.23 Final Acquisition and Transmission Parts of the Schedule for Sensor 3 after Searching Optimal Communication Slot Process.

		Time(sec)																			
Acquisition	Measure	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5
Acq of S4:A1	humidity	A1#1										A1#2									

(a) Acquisition Part of the Schedule for Sensor 4.

		Time (sec)																			
Transmission	To	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5
TX of S4:A1#1	S1	3	2.5	2	1.5	1															
TX of S4:A1#2	S1											3	2.5	2	1.5	1					

(b) Finalized Transmission Part of the Schedule for Sensor 4.

Fig. 4.24 Final Acquisition and Transmission Parts of the Schedule for Sensor 4 after Searching Optimal Communication Slot Process.

		Time(sec)																			
Acquisition	Measure	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5
Acq of S5:A1	temperature	A1#1				A1#2				A1#3				A1#4				A1#5			
Acq of S5:A2	humidity	A2#1										A2#2									

(a) Acquisition Part of the Schedule for Sensor 5.

		Time (sec)																			
Transmission	To	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5
TX of S5:A1#1	S2	4	3.5	3	2.5	2	1.5	1													
TX of S5:A1#2	S2					4	3.5	3	2.5	2	1.5	1									
TX of S5:A1#3	S2									4	3.5	3	2.5	2	1.5	1					
TX of S5:A1#4	S2													4	3.5	3	2.5	2	1.5	1	
TX of S5:A1#5	S2	2	1.5	1														4	3.5	3	2.5
TX of S5:A2#1	S2	3	2.5	2	1.5	1															
TX of S5:A2#2	S2											3	2.5	2	1.5	1					

(b) Finalized Transmission Part of the Schedule for Sensor 5.

Fig. 4.25 Final Acquisition and Transmission Parts of the Schedule for Sensor 5 after Searching Optimal Communication Slot Process.

		Time(sec)																			
Acquisition	Measure	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5
Acq of S6:A1	humidity	A1#1										A1#2									

(a) Acquisition Part of the Schedule for Sensor 6.

		Time (sec)																			
Transmission	To	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5
TX of S6:A1#1	S2	3	2.5	2	1.5	1															
TX of S6:A1#2	S2											3	2.5	2	1.5	1					

(b) Finalized Transmission Part of the Schedule for Sensor 6.

Fig. 4.26 Final Acquisition and Transmission Parts of the Schedule for Sensor 6 after Searching Optimal Communication Slot Process.

4.4.5 SCO-Pattern Generation

Once the three parts of the schedule are finalized after searching optimal communication slots for each device, the last step of the optimization algorithm is to merge these three parts to obtain an individual SCO-PATTERN for each sensor.

Example 4.4.8 *Here are the* SCO-PATTERN *for each sensor devices in the environment obtained at the end of the overall process:*

$P_1 = (\{(0,\{A_T\}),(1.5,\{R\}),(2,\{A_T,R\}),(2.5,\{T\}),(4,\{A_T\}),(6,\{A_T\}),(6.5,\{R\}),(7,\{R\}),$
$(7.5,\{T\}),(8,\{A_T\}),(9,\{R\}),(9.5,\{T\})\},10)$
$P_2 = (\{(0.5,\{R\}),(1,\{R\}),(1.5,\{T\}),(5,\{R\}),(5.5,\{T\}),(6,\{R\}),(6.5,\{R\}),(7,\{T\})\},10)$
$P_3 = (\{(0,\{A_T,A_H\}),(2,\{A_T,T\}),(4,\{A_T\}),(5,\{A_H\}),(6,\{A_T\}),(7,\{T\}),(8,\{A_T\}),(9,\{T\})\},10)$
$P_4 = (\{(0,\{A_H\}),(1.5,\{T\}),(5,\{A_H\}),(6.5,\{T\})\},10)$
$P_5 = (\{(0,\{A_T,A_H\}),(1,\{T\}),(2,\{A_T\}),(4,\{A_T\}),(5,\{A_H,T\}),(6,\{A_T,T\}),(8,\{A_T\})\},10)$
$P_6 = (\{(0,\{A_H\}),(0.5,\{T\}),(5,\{A_H\}),(6.5,\{T\})\},10)$

Once SCO-PATTERNs are completed, they are sent by the Gateway to sensor devices to configure them. Sensor devices then execute the given actions periodically, and data flow on the base station where continuous queries are computed. If an application changes its requirements, the configuration process starts again from the beginning. Flowchart of the overall process is illustrated in Figure 4.27.

4.4.6 Assessment of Optimization Process

Our optimization process is detailed in Section 4.4.4. This optimization aims to search the optimal communication slots so that wireless sensor devices may enhance energy, consume less energy than the other cases. Thus, we propose an optimization algorithm to search and determine the communication

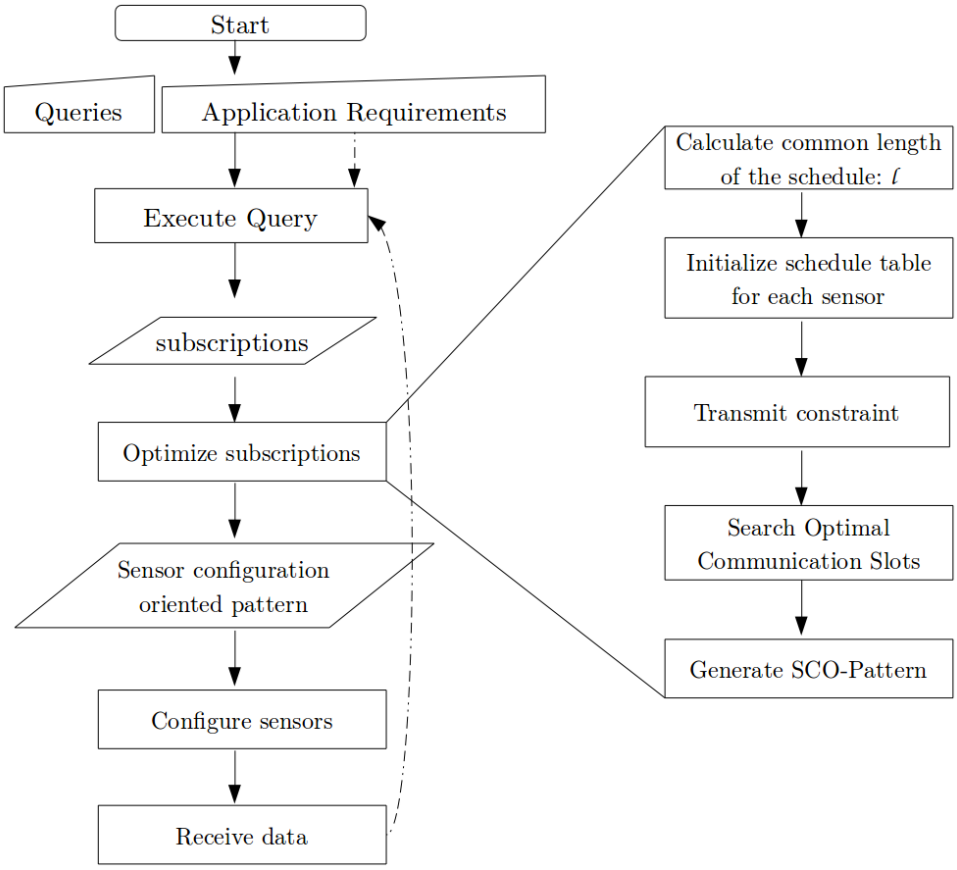


Fig. 4.27 Flowchart of the Overall Process.

slots by processing the schedule tables of each device of the network (reception part of the schedule table). While doing that, we should obey the constraint of each sensor device due to the given application requirements. Hence, the proposed optimization process starts by propagating transmission and reception constraints based on latencies that makes it a bottom-up process. Then, the algorithm tries to find the most energy efficient communication slots on the sink device, and finally propagates those choices to lower-layers and continues the algorithm in a top-down process in the network topology.

The details of the optimization is presented in the previous section. The application requirements and relevant constraints are specified, and the target of optimization is to minimize the number of communication slots (RX slots) to minimize the consumed energy. The proposed optimization algorithm is a sort of *greedy heuristic algorithm* to determine the optimal slots for sensor devices. The core concept of greedy algorithm is to perform a short-sighted action in each step [128, 167]. Here, it starts from the reception part of the schedule table of sink device and tries to group the receptions from the same sensor device. In each step, an optimal communication slot is found and the step is repeated until all the reception data is matched with a slot.

Greedy is an algorithmic paradigm that is generally preferred for optimization problems and solves the given problem piece by piece, always choosing the next piece that offers the most obvious and immediate benefit. An optimization problem such as searching optimal communication slots can be solved using Greedy if the problem has the following property: At every step, we can make a choice that looks best at the moment, and we get the optimal solution of the complete problem [40].

The most valuable advantage of this approach is that the rules to choose are easy to design and intuitive and easy to implement. Besides, it is a well-known optimization approach in the literature. Generally algorithms which seek a global optimal solution are computationally expensive, however greedy based algorithms are faster and less complex than general algorithms and ends deterministically.

On the other hand, as all local approaches, local view does not guarantee the optimal global solutions. Local optimization process does not implicate global optimality and may decreases their performance. Moreover, the solution quality depends on problem instance which makes difficult to have a robust algorithm for all problem instances. Finally local-based approaches should be run till the end to get a complete solution.

Besides, in `WSN` research area, the greedy heuristic method is a well-known technique to generate the data transmission paths for the sensor nodes to the base station (sink device), also to determine the location of the sink device in the topology. Since energy limitation is the key challenge in `WSN` applications, the transmission paths play the important roles. Properly designed transmission paths may provide the less amount of energy consumption and the shorter transmission delay [85, 106, 165].

In this thesis, we propose our optimization algorithm to search the optimal communication slot in which the sensor devices consume less energy than the other cases. The constraints of each sensor device due to the given application requirements are considered to search the optimal communication slots. However, to validate the proposed optimization process, it should definitively compared with an optimization tool in order to find out if the optimal solution that we propose is true or not.

To validate our approach, we prefer a constraint modelling technique since our optimization process is based on the constraints coming from the application requirements. The constraint modelling is a proven technology for solving complex combinatorial decision or optimisation problems of many disciplines, such as: scheduling; industrial design; aviation; banking; combinatorial mathematics; and the petrochemical and steel industries etc [55].

In our thesis, we prefer an open-source constraint modeling language `MiniZinc` [124] to model our constraint satisfaction and optimization problem in a high-level, solver-independent way. `MiniZinc` is a language designed for specifying constrained optimization and decision problems over integers and real numbers. With these features, `MiniZinc` seems the right choice for our case. `MiniZinc` is designed to interface easily to different backend solvers. It does this by transforming an input `MiniZinc` model

and data file into a FlatZinc model. FlatZinc models consist of variable declaration and constraint definitions as well as a definition of the objective function if the problem is an optimization problem. The translation from MiniZinc to FlatZinc is specializable to individual backend solvers, so they can control what form constraints end up in. In particular, MiniZinc allows the specification of global constraints by decomposition.

MiniZinc models can also contain another kind of variable called a decision variable. Decision variables are variables in the sense of mathematical or logical variables. Unlike parameters and variables in a standard programming language, the modeller does not need to give them a value. Rather the value of a decision variable is unknown and it is only when the MiniZinc model is executed that the solving system determines if the decision variable can be assigned a value that satisfies the constraints in the model and if so what this is. A simple constraint model implemented in MiniZinc from the MiniZinc documentation [114] is the following:

Example 4.4.9 *Suppose we color a map of Australia (made up of seven different states and territories). Each of state must be given a colour so that adjacent regions have different colors. The states of Australia is simply illustrated in Figure 4.28. The MiniZinc model for the relevant problem is given in Table 4.3. In the given example, the line "int : nc = 3;" specifies a parameter in the problem which is the number of colours to be used. In the example model, a decision variable is associated with each region, wa, nt, sa, q, nsw, v and t, which stands for the (unknown) color to be used to fill the region.*

Table 4.3 A MiniZinc model for coloring the states and territories in Australia and its output.

```
int: nc = 3; %Coloring Australia using nc colours

var 1..nc: wa;
var 1..nc: nt;
var 1..nc: sa;
var 1..nc: q;
var 1..nc: nsw;
var 1..nc: v;
var 1..nc: t;

constraint wa != nt;
constraint wa != sa;
constraint nt != sa;
constraint nt != q;
constraint sa != q;
constraint sa != nsw;
constraint sa != v;
constraint q != nsw;
constraint nsw != v;
solve satisfy;

output ["wa=", show(wa), "nt=", show(nt),
"sa=", show(sa), ", "q=", show(q),
"nsw=", show(nsw), "v=", show(v),
"t=", show(t)];
```

```
Compiling aust.mzn
Running aust.mzn
wa=3 nt=2 sa=1
q=3 nsw=2 v=3
t=1
----------
Finished in 26msec
```

Fig. 4.28 States of Australia.

For our optimization process, we implement our constraint on `MiniZinc` tool. For the cost function, we set the number of communication slots as a cost that stands for the energy consumption in the real world environment. However, there are many ways to express the objective function in our case. Since we adopt a tree structured topology, the balance of the tree is extremely important for us. For instance, in our model for the example given in this chapter, we propose three different objective function. The three objective function that can be applicable are the following:

1. Test 1: $solve\ minimize\ max(nbRxSlot\_s1 + nbTxSlot\_s1 + nbRxSlot\_s2 + nbTxSlot\_s2, nbRxSlot\_s0)$;
2. Test 2: $solve\ minimize\ max(nbRxSlot\_s0, max(nbRxSlot\_s1, nbRxSlot\_s2))$; % Minimize Maximum Rx
3. Test 3: $solve\ minimize\ max(nbRxSlot\_s1 + nbTxSlot\_s1, nbRxSlot\_s2 + nbTxSlot\_s2)$; % Minimize Maximum Rx+Tx (exclude S0)

where $nbRxSlot\_sX$ stands for the number of found optimal reception slots for Sensor X, $nbTxSlotsX$ represents the number of found optimal transmission slots for Sensor X. These three optimization function try to minimize the number of communication slots. Since it is a tree structured topology, it may cause an unbalance behavior. For instance an optimal solution for Sensor 0 may cause an additional communication cost for lower-layer neighbors or an additional communication cost for few branches. Hence, the balance of the branches of the tree topology is important in that sense and these three options may present us different optimal solution considering different objective functions. The output of the constraint model is given for each objective function in Table 4.4.

For instance $S0 = 6 + 0$ indicates that during the given time interval, Sensor 0 has 6 reception slots and it does not have any transmission slots (which is logically true since it is a base station and dedicated to data aggregation.), 2.5 (1) stands for the data reception at t=2.5sec from Sensor 1. For the given three solutions based on the different objective functions, *Test 1* and *Test 3* propose a better solution than the *Test 2*. *Test 1* and *Test 3* propose 6 reception slots for Sensor 0, 5 reception slots and 3 transmission slots for Sensor 1 and 2. However, *Test 2* proposes 6 reception slots for Sensor 1. From the energy point of view which is the major criteria, more communication requires more energy thus the results of *Test 1* and *Test 3* are optimal.

After executing constraint-based model, these obtained results should be compared with our results in terms of efficiency and quality in order to evaluate our approach presented in this thesis. Firstly the result of our optimization algorithm gives the same number of reception and transmission slots as an optimal solution for sensor devices (6 RX slots, 0 TX for Sensor 0 (see Figure 4.20), 5 RX slots, 3 TX slots for Sensor 1 (see Figure 4.21) and 5 RX slots, 3 TX slots for Sensor 2 (see Figure 4.22)).

Table 4.4 Output of MiniZinc for three different objective functions.

solve minimize max$(nbRxSlot\_s1 + nbTxSlot\_s1 + nbRxSlot\_s2 + nbTxSlot\_s2, nbRxSlot\_s0)$;
Compiling 3SoSM_Optimization.mzn
Running 3SoSM_Optimization.mzn
S0 = 6+0 [2.5 (1) | 2.5 (1) | 6.0 (1) | 6.0 (1) | 8.5 (1) | 2.5 (1) | 2.5 (1) | 6.0 (1) | 8.5 (1) | 8.5 (1) | 2.5 (1) | 6.0 (1) | 2.5 (1) | 6.0 (1) | 1.0 (2) | 5.5 (2) | 5.5 (2) | 6.5 (2) | 1.0 (2) | 1.0 (2) | 6.5 (2) | 1.0 (2) | 5.5 (2)]
S1 = 5+3 [2.0 (3) | 2.0 (3) | 5.5 (3) | 8.0 (3) | 8.0 (3) | 2.0 (3) | 5.5 (3) | 0.0 (4) | 5.0 (4)]
S2 = 5+3 [0.0 (5) | 4.0 (5) | 4.0 (5) | 6.0 (5) | 0.0 (5) | 0.0 (5) | 6.0 (5) | 0.5 (6) | 5.0 (6)]
==========
solve minimize max$(nbRxSlot\_s0, max(nbRxSlot\_s1, nbRxSlot\_s2))$; % Minimize Maximum Rx
Compiling 3SoSM_Optimization.mzn
Running 3SoSM_Optimization.mzn
S0 = 6+0 [2.5 (1) | 2.5 (1) | 6.0 (1) | 6.0 (1) | 8.5 (1) | 2.5 (1) | 2.5 (1) | 6.0 (1) | 8.5 (1) | 8.5 (1) | 2.5 (1) | 6.0 (1) | 2.5 (1) | 6.0 (1) | 1.0 (2) | 5.5 (2) | 5.5 (2) | 6.5 (2) | 1.0 (2) | 1.0 (2) | 6.5 (2) | 1.0 (2) | 5.5 (2)]
S1 = 6+3 [0.5 (3) | 2.0 (3) | 5.5 (3) | 8.0 (3) | 8.0 (3) | 0.5 (3) | 5.5 (3) | 0.0 (4) | 5.0 (4)]
S2 = 5+3 [0.5 (5) | 4.0 (5) | 4.0 (5) | 6.0 (5) | 0.5 (5) | 0.5 (5) | 6.0 (5) | 0.0 (6) | 5.0 (6)]
==========

solve minimize max$(nbRxSlot\_s1 + nbTxSlot\_s1, nbRxSlot\_s2 + nbTxSlot\_s2)$; % Minimize Maximum Rx+Tx (exclude S0)
Compiling 3SoSM_Optimization.mzn
Running 3SoSM_Optimization.mzn
S0 = 6+0 [2.5 (1) | 2.5 (1) | 6.0 (1) | 6.0 (1) | 8.5 (1) | 2.5 (1) | 2.5 (1) | 6.0 (1) | 8.5 (1) | 8.5 (1) | 2.5 (1) | 6.0 (1) | 2.5 (1) | 6.0 (1) | 1.0 (2) | 5.5 (2) | 5.5 (2) | 6.5 (2) | 1.0 (2) | 1.0 (2) | 6.5 (2) | 1.0 (2) | 5.5 (2)]
S1 = 5+3 [2.0 (3) | 2.0 (3) | 5.5 (3) | 8.0 (3) | 8.0 (3) | 2.0 (3) | 5.5 (3) | 0.0 (4) | 5.0 (4)]
S2 = 5+3 [0.5 (5) | 4.0 (5) | 4.0 (5) | 6.0 (5) | 0.5 (5) | 0.5 (5) | 6.0 (5) | 0.0 (6) | 5.0 (6)]
==========

These results prove that in the settings of our running example our optimization algorithm indeed finds optimal communication slots for the sensor devices in the network, validated by a constraint modelling tool (here, `MiniZinc`). Besides, the execution time is another key metric for the validation. Performance of the algorithm should be efficient enough in order to make fairful comparaison with the constrain models. The test code for comparing different objective functions is given in Tables 4.6, 4.7, 4.8. Table 4.5 presents the execution time of each model. The computer used for running the code is equipped with a processor Intel 3.40GHz (6 cores) and 6GB of RAM. It runs a 64bit Linux-based operating system: Ubuntu 16.04.2 LTS.

In comparison to the `MiniZinc` models, our algorithm based on the greedy principles finds the optimal solution quicker than the others. The tool `MiniZinc` executes the constraint model and tries to perform each possible case and at the end proposes us the optimal one, thus it is highly expected that the execution time of the tool is longer than our optimization process.

Table 4.5 Execution Time Comparaison of Models.

Model	**Execution Time**
Test 1	1h 47min 50sec
Test 2	10sec 98msec
Test 3	13sec 146msec
Our Optimization Algorithm	**19msec**

Table 4.6 MiniZinc test code (part 1).

```
% 3SoSM Test Final
int: length = 20;
int: nbDevices = 6;
set of int: SLOTS = 0..(length-1);

% S3 => Source
% S4 => Source

% S1
int: s1_Rx_size = 9;
array[1..s1_Rx_size] of 1..nbDevices: s1_Rx_Source = [ 3, 3, 3, 3, 3, 3, 3, 4, 4 ];
array[1..s1_Rx_size] of var SLOTS: s1_Rx_Slot;

% Acquisition + Latency on S3
constraint s1_Rx_Slot[1] >= 0 /\ s1_Rx_Slot[1] < 0+8; % AND
constraint s1_Rx_Slot[2] >= 4 /\ s1_Rx_Slot[2] < 4+8; % AND
constraint s1_Rx_Slot[3] >= 8 /\ s1_Rx_Slot[3] < 8+8; % AND
constraint s1_Rx_Slot[4] >= 12 /\ s1_Rx_Slot[4] < 12+8; % AND
constraint s1_Rx_Slot[5] >= 16 \/ s1_Rx_Slot[5] < ((16+8) mod length); % OR + Modulo
constraint s1_Rx_Slot[6] >= 0 /\ s1_Rx_Slot[6] < 0+6; % AND
constraint s1_Rx_Slot[7] >= 10 /\ s1_Rx_Slot[7] < 10+6; % AND

% Acquisition + Latency on S4
constraint s1_Rx_Slot[8] >= 0 /\ s1_Rx_Slot[8] < 0+6; % AND
constraint s1_Rx_Slot[9] >= 10 /\ s1_Rx_Slot[9] < 10+6; % AND
```

Table 4.7 MiniZinc test code (part 2).

```
% Reception on S1
constraint forall( i in 1..s1_Rx_size, j in 1..s1_Rx_size )
( s1_Rx_Slot[i] = s1_Rx_Slot[j] -> s1_Rx_Source[i] = s1_Rx_Source[j] );
% S2
int: s2_Rx_size = 9;
array[1..s2_Rx_size] of 1..nbDevices: s2_Rx_Source = [ 5, 5, 5, 5, 5, 5, 5, 6, 6 ];
array[1..s2_Rx_size] of var SLOTS: s2_Rx_Slot;

% Acquisition + Latency on S5
constraint s2_Rx_Slot[1] >= 0 /\ s2_Rx_Slot[1] < 0+8; % AND
constraint s2_Rx_Slot[2] >= 4 /\ s2_Rx_Slot[2] < 4+8; % AND
constraint s2_Rx_Slot[3] >= 8 /\ s2_Rx_Slot[3] < 8+8; % AND
constraint s2_Rx_Slot[4] >= 12 /\ s2_Rx_Slot[4] < 12+8; % AND
constraint s2_Rx_Slot[5] >= 16 \/ s2_Rx_Slot[5] < ((16+8) mod length); % OR + Modulo
constraint s2_Rx_Slot[6] >= 0 /\ s2_Rx_Slot[6] < 0+6; % AND
constraint s2_Rx_Slot[7] >= 10 /\ s2_Rx_Slot[7] < 10+6; % AND

% Acquisition + Latency on S6
constraint s2_Rx_Slot[8] >= 0 /\ s2_Rx_Slot[8] < 0+6; % AND
constraint s2_Rx_Slot[9] >= 10 /\ s2_Rx_Slot[9] < 10+6; % AND

% Reception on S2
constraint forall( i in 1..s2_Rx_size, j in 1..s2_Rx_size )
( s2_Rx_Slot[i] = s2_Rx_Slot[j] -> s2_Rx_Source[i] = s2_Rx_Source[j] );

% S0
int: s0_Rx_size = 23;
array[1..s0_Rx_size] of 1..nbDevices: s0_Rx_Source = [ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2,
2, 2 ];
array[1..s0_Rx_size] of var SLOTS: s0_Rx_Slot;

% Acquisition + Latency on S1
constraint s0_Rx_Slot[1] >= 0 /\ s0_Rx_Slot[1] < 0+8; % AND
constraint s0_Rx_Slot[2] >= 4 /\ s0_Rx_Slot[2] < 4+8; % AND
constraint s0_Rx_Slot[3] >= 8 /\ s0_Rx_Slot[3] < 8+8; % AND
constraint s0_Rx_Slot[4] >= 12 /\ s0_Rx_Slot[4] < 12+8; % AND
constraint s0_Rx_Slot[5] >= 16 \/ s0_Rx_Slot[5] < ((16+8) mod length); % OR + Modulo

% Transmission + Latency from S1
% Data from S3
constraint s0_Rx_Slot[6] > s1_Rx_Slot[1] /\ s0_Rx_Slot[6] < 0+8; % AND
constraint s0_Rx_Slot[7] > s1_Rx_Slot[2] /\ s0_Rx_Slot[7] < 4+8; % AND
constraint s0_Rx_Slot[8] > s1_Rx_Slot[3] /\ s0_Rx_Slot[8] < 8+8; % AND
constraint s0_Rx_Slot[9] > s1_Rx_Slot[4] /\ s0_Rx_Slot[9] < 12+8; % AND
constraint ( s1_Rx_Slot[5] > ((16+8) mod length) /\ (s0_Rx_Slot[10] > s1_Rx_Slot[5] \/ s0_Rx_Slot[10] <
((16+8) mod length) ) ) \/
( s1_Rx_Slot[5] < ((16+8) mod length) /\ s0_Rx_Slot[10] > s1_Rx_Slot[5] /\ s0_Rx_Slot[10] < ((16+8)
mod length) ); % OR + Modulo
constraint s0_Rx_Slot[11] > s1_Rx_Slot[6] /\ s0_Rx_Slot[11]< 0+6; % AND
constraint s0_Rx_Slot[12] > s1_Rx_Slot[7] /\ s0_Rx_Slot[12]< 10+6; % AND

% Acquisition + Latency on S4
constraint s0_Rx_Slot[13] > s1_Rx_Slot[8] /\ s0_Rx_Slot[13] < 0+6; % AND
constraint s0_Rx_Slot[14] > s1_Rx_Slot[9] /\ s0_Rx_Slot[14] < 10+6; % AND
```

Table 4.8 MiniZinc test code (part 3).

```
% Transmission + Latency from S2
% Data from S5
constraint s0_Rx_Slot[15] > s2_Rx_Slot[1] /\ s0_Rx_Slot[15] < 0+8; % AND
constraint s0_Rx_Slot[16] > s2_Rx_Slot[2] /\ s0_Rx_Slot[16] < 4+8; % AND
constraint s0_Rx_Slot[17] > s2_Rx_Slot[3] /\ s0_Rx_Slot[17] < 8+8; % AND
constraint s0_Rx_Slot[18] > s2_Rx_Slot[4] /\ s0_Rx_Slot[18] < 12+8; % AND
constraint ( s2_Rx_Slot[5] > ((16+8) mod length) /\ (s0_Rx_Slot[19] > s2_Rx_Slot[5] \/ s0_Rx_Slot[19] <
((16+8) mod length) ) ) \/
( s2_Rx_Slot[5] < ((16+8) mod length) /\ s0_Rx_Slot[19] > s2_Rx_Slot[5] /\ s0_Rx_Slot[19] < ((16+8)
mod length) ); % OR + Modulo
constraint s0_Rx_Slot[20] > s2_Rx_Slot[6] /\ s0_Rx_Slot[20]< 0+6; % AND
constraint s0_Rx_Slot[21] > s2_Rx_Slot[7] /\ s0_Rx_Slot[21]< 10+6; % AND

% Acquisition + Latency on S4
constraint s0_Rx_Slot[22] > s2_Rx_Slot[8] /\ s0_Rx_Slot[22] < 0+6; % AND
constraint s0_Rx_Slot[23] > s2_Rx_Slot[9] /\ s0_Rx_Slot[23] < 10+6; % AND

% Reception on S0
constraint forall( i in 1..s0_Rx_size, j in 1..s0_Rx_size )( s0_Rx_Slot[i] = s0_Rx_Slot[j] -> s0_Rx_Source[i]
= s0_Rx_Source[j] );
% Transmission vs Reception on S1
constraint forall( i in 1..s0_Rx_size)(s0_Rx_Source[i] = 1 -> forall(j in 1..s1_Rx_size)(s0_Rx_Slot[i] !=
s1_Rx_Slot[j]));
% Transmission vs Reception on S1
constraint forall( i in 1..s0_Rx_size)(s0_Rx_Source[i] = 2 -> forall(j in 1..s2_Rx_size)(s0_Rx_Slot[i] !=
s2_Rx_Slot[j]));

% count distinct
predicate unique_first(array[int] of var int: tab, int: index) = forall(j in 1..(index-1))(tab[index]!=tab[j]);
var int: nbRxSlot_s1 = sum(i in 1..s1_Rx_size)( if unique_first(s1_Rx_Slot,i) then 1 else 0 endif );
var int: sumRxSlot_s1 = sum(i in 1..s1_Rx_size)( if unique_first(s1_Rx_Slot,i) then s1_Rx_Slot[i] else 0
endif );
var int: nbTxSlot_s1 = sum(i in 1..s0_Rx_size)( if s0_Rx_Source[i] = 1 /\ unique_first(s0_Rx_Slot,i) then 1
else 0 endif );

var int: nbRxSlot_s2 = sum(i in 1..s2_Rx_size)( if unique_first(s2_Rx_Slot,i) then 1 else 0 endif );
var int: sumRxSlot_s2 = sum(i in 1..s2_Rx_size)( if unique_first(s2_Rx_Slot,i) then s2_Rx_Slot[i] else 0
endif );
var int: nbTxSlot_s2 = sum(i in 1..s0_Rx_size)( if s0_Rx_Source[i] = 2 /\ unique_first(s0_Rx_Slot,i) then 1
else 0 endif );

var int: nbRxSlot_s0 = sum(i in 1..s0_Rx_size)( if unique_first(s0_Rx_Slot,i) then 1 else 0 endif );
var int: sumRxSlot_s0 = sum(i in 1..s0_Rx_size)( if unique_first(s0_Rx_Slot,i) then s0_Rx_Slot[i] else 0
endif );

% Objective Functions
% test 1
solve minimize max(nbRxSlot_s1+nbTxSlot_s1+nbRxSlot_s2+nbTxSlot_s2,nbRxSlot_s0);
% test 2
solve minimize max(nbRxSlot_s0,max(nbRxSlot_s1,nbRxSlot_s2)); % Minimize Maximum Rx
% test 3
solve minimize max(nbRxSlot_s1+nbTxSlot_s1,nbRxSlot_s2+nbTxSlot_s2); % Minimize Maximum Rx+Tx
(exclude S0)
```

Table 4.9 MiniZinc test code (part 4).

```
output [ "Unique (S1) = ", show(nbRxSlot_s1), " ", show(s1_Rx_Slot), "" ];
output [ "S0 = ", show(nbRxSlot_s0), "+0 [ " ] ++ [ if i > 1 then " | " else "" endif ++ show(s0_Rx_Slot[i]/2)
++ " (" ++ show(s0_Rx_Source[i]) ++ ")" | i in 1..s0_Rx_size ] ++ [ " ] / " ++ show(sumRxSlot_s0) ++ " " ];
output [ "S1 = ", show(nbRxSlot_s1), "+", show(nbTxSlot_s1) , " [ " ] ++ [ if i > 1 then " | " else "" endif
++ show(s1_Rx_Slot[i]/2) ++ " (" ++ show(s1_Rx_Source[i]) ++ ")" | i in 1..s1_Rx_size ] ++ [ " ] / " ++
show(sumRxSlot_s1) ++ " " ];
output [ "S2 = ", show(nbRxSlot_s2), "+", show(nbTxSlot_s2) , " [ " ] ++ [ if i > 1 then " | " else "" endif
++ show(s2_Rx_Slot[i]/2) ++ " (" ++ show(s2_Rx_Source[i]) ++ ")" | i in 1..s2_Rx_size ] ++ [ " ] / " ++
show(sumRxSlot_s2) ++ " " ];
```

4.5 Summary of the 3SoSM Approach

The objective of our approach `3SoSM` is to transform the application requirements into a global schedule and then to build individual SCO-PATTERNs to configure each device. Application requirements are represented by a set of subscription requests $S = \{s_1, s_2, s_3, ...\}$. Hence the energy-aware optimization process can be expressed as a transformation function that gets a set of subscription requests (all subscription requests from all applications) and a network topology N and produces a SCO-PATTERN for each device: $f(\{s_1, s_2, .., s_n\}, N) = \{\mathrm{P}_{d_1}, \mathrm{P}_{d_2}, ..., \mathrm{P}_{d_m}\}$.

Since the communication costs are the significant part of the energy budget [140], the optimal solution is a valid solution that minimizes the number of Transmission / Reception actions in order to minimize the consumed energy of the sensor devices. Less communication means less consumed energy. To optimize the energy consumption of the devices, we try to minimize the number of transmission and reception actions of each device. In `Device-Centered 3SoSM` approach, we first construct DOA-PATTERNs that have a common length with only acquisition actions (Acq) for all the devices (and latency information tagged with the relevant measure). It is then necessary to insert the transmission (Tx) and reception (Rx) actions to obtain a solution, composed of the set of complete SCO-PATTERNs. A solution is valid if all acquired data traverse the network topology to the base station before the "latency" associated with that data expires.

`Device-Centered 3SoSM` approach supports a single-modality and does not take into account the topology of the network. It commonly uses DOA-PATTERN as an intermediate step and `GeNoMe` process to transform the subscription requests into SCO-PATTERN for each sensor device.

In the complete version of our approach, we present `Network-Aware 3SoSM` approach. Here, we take into account the network and topology, besides multi-modality is supported. Latency information given in the application requirements gains more importance than with the `Device-Centered` version. Moreover, in this complete version of our approach, the main idea is to create a `Schedule Time Pattern` for each sensor device. This pattern is a schedule composed of sensor actions, here with three types of actions: acquisition A, reception R, transmission T. Based on their roles, only certain actions are pertinent to sensor devices: acquire a measure and store it in a memory buffer, transmit all buffered measures one hop towards the sink (and empty the buffer), receive measures from a lower-layer device and store it in a memory buffer. Sensor devices can configure themselves with a `Schedule Time Pattern`, and then execute periodically the scheduled actions.

Chapter 5

Prototype Implementation

In this chapter we present the prototype that we developed based on a declarative monitoring architecture for pervasive environments. The general architecture is presented in Chapter 3. In order to validate our approach explained in Chapter 4 and conduct experiments, we deigned and implemented this prototype.

Our prototype enables users to interact with a pervasive environment through the Pervasive Environment Management System (PEMS) without worrying about low-level technical considerations like programming languages or network protocols. PEMS manages a relational pervasive environment, with its dynamic data sources and set of services, and can execute continuous queries over this environment. Our prototype can also communicate with sensor devices in a WSN, in particular with a WSN simulator.

5.1 Platform

Our prototype has three main components: continuous query engine, wireless sensor network simulator and 3SoSM Gateway. A continuous query engine stands for the execution of queries over the relational pervasive environment, i.e. one-shot and continuous queries over data, streams and services, and service discovery queries. A wireless sensor network simulator refers to an environment on which protocols, schemes, models can be evaluated in a very large scale. Running real experiments on a testbed with physical devices is costly and difficult. Besides, repeatability is largely compromised since many factors affect the experimental results at the same time. It is hard to isolate a single aspect [194]. Including a WSN simulator in our prototype allow users to isolate different factors by tuning configurable parameters. Finally, to complete the overall architecture, the 3SoSM Gateway is responsible for managing the interactions and bidirectional communication between the PEMS and the simulated WSN.

5.1.1 Continuous Query Engine

In this study, we use **S**ervice-**o**riented **C**ontinuous **Q**uery (SoCQ) Engine [68] as a continuous query engine. Benefits of the SoCQ framework as a pervasive environment management system was already introduced in Section 3.2. It takes a data-oriented perspective on the pervasive environment: it provides a unified view and access to the various and heterogeneous resources available in the environment. Pervasive applications can then be created in a declarative way using service-oriented continuous queries over such an environment.

Within the SoCQ framework, XD-Relations (eXtended Dynamic Relations) represents standard relations, that may be updated, or data streams, that continuously produce data. The definition of XD-Relations can also include virtual attributes and binding patterns that together enable queries to interact with distributed services: service discovery, method invocation, stream subscription. Queries may be one-shot queries (like standard SQL queries) or continuous queries (with a dynamic result, like

a stream). Furthermore, invocation of service methods and a subscription to service streams can be parametrized.

SoCQ Engine is a service-oriented continuous query engine implemented in Java. A Data Description Language (DDL) allows to define XD-Relations, and a SQL-like query language allows to specify one-shot and continuous queries over XD-Relations [69]. A user interface controls the query engine: users can visualize XD-Relations and their content, and launch one-shot/continuous queries.

We illustrate SoCQ in the context of Smart Building. Figure 5.1a shows an example of a discovery query (creating a new XD-Relation) to explore temperature sensor devices in the environment. The result of the given discovery query is presented in Figure 5.1b where the discovery query searches for sensor services that provide a location, a method to get the current temperature, and a continuous stream of temperature measures. The resulting XD-Relation TemperatureServices has one ServiceID attribute, a location attribute, and a virtual attribute for temperature (displayed as '*' in the table).

Figure 5.2a introduces a parameterized continuous query to subscribe to temperature services from all temperature service providers. The result of the query is given in Figure 5.2b. While executing, this continuous query subscribes to the temperature stream of every discovered service with given query parameters to build a resulting data stream. If new services are discovered and/or some services become unavailable, the continuous query automatically adapts the corresponding stream subscriptions.

Figure 5.3a introduces another parameterized continuous query. The given query requests temperature service from a specific location. Here, the query defines a location constraint (WHERE Location=501.342). The query engine checks the location attribute of the services and filters the relevant sensor devices that are located at the given location. The result of the query is presented in Figure 5.3b. The query engine establishes subscriptions to the temperature services of the relevant sensor devices that are located in the given location and starts streaming measures from these sensor devices according to given parameters.

5.1.2 WSN Simulator

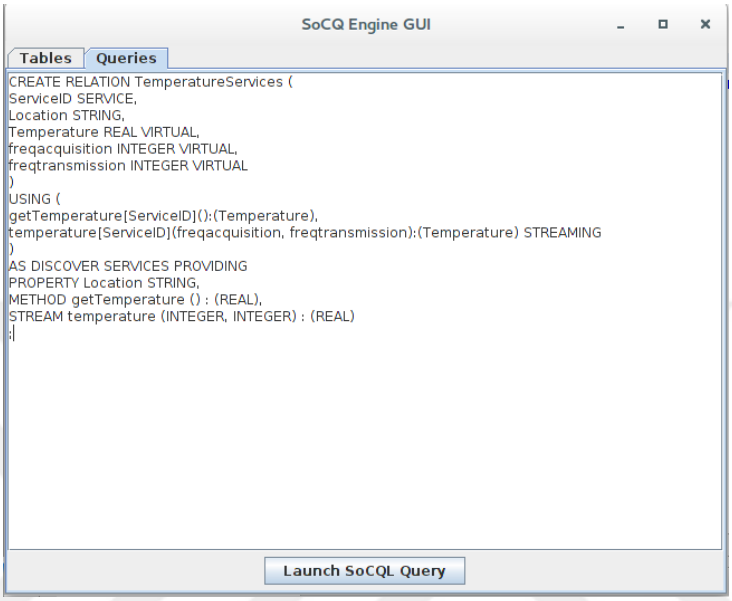
5.1.2.1 WSN Simulation Environment

Simulator tools allow researchers to verify new ideas and compare the proposed solutions in a virtual environment helping to avoid unnecessary, time-consuming or expensive hardware implementations. In a simulation, all of the influencing factors and also the algorithms that are to be investigated are modeled and examined in an artificial software environment with a high degree of abstraction. This allows repeatability, tight control, large scale and cost effective tests, possibly with heterogeneous operating systems and programming languages [194].

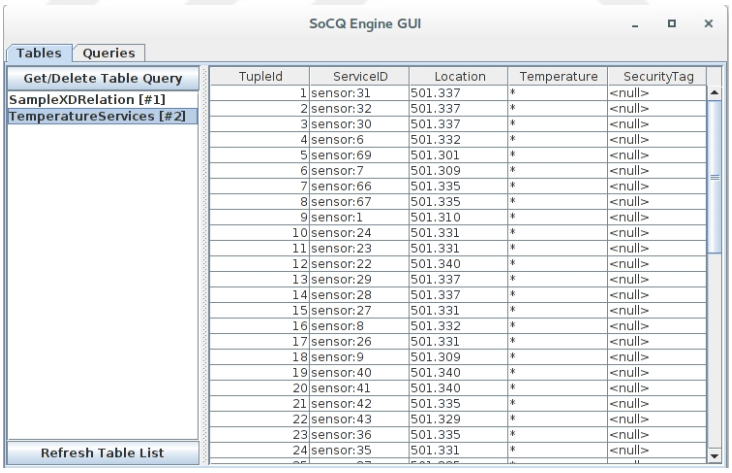
Since it provides a higher level of abstraction, a simulator is a good choice at the earlier phase of design and development. It involves lower cost, e.g. it can simulate a network with thousands of nodes. Simulations always require certain assumptions about the real world. These may turn out to be wrong or too coarse to capture all aspects that influence the performance of algorithms and protocols. Some important characteristics such as radio propagation or energy consumption, are inherently hard to model accurately in simulators [83].

In smart buildings, real physical wireless sensor devices are deployed for controlling and monitoring issues. In this research domain, even though a real testbed is extremely valuable for researchers to verify their approaches, testbeds may not be available for certain sensor network applications. Furthermore, testbeds are limited in scope and more importantly are not feasible in many cases due to cost, time and inaccessible terrain challenges [78]. Besides, real world parameters are often out of the experimenter's hands. This makes it difficult to repeat experiments and to fully understand and correctly interpret outcomes. Deploying testbeds supposes a huge effort and still, sensor devices can fail at any time due to hardware, software, or communication reasons.

As a trade-off solution, even if only a coarse energy consumption evaluation can thus be expected due

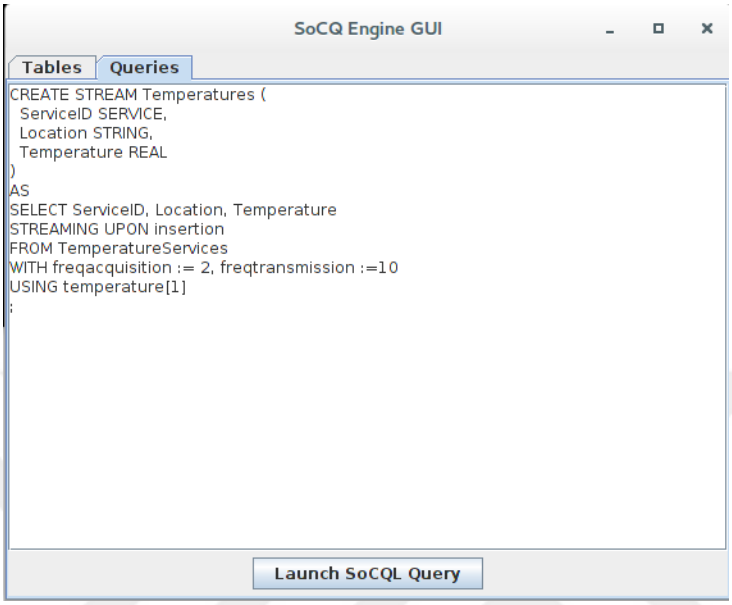


(a) SoCQ Discovery Query to explore temperature service providers.

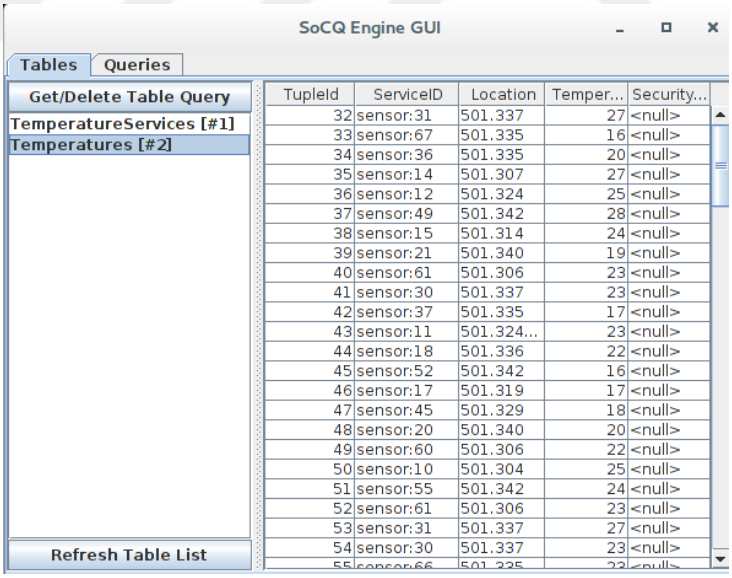


(b) SoCQ GUI Result of the Discovery Query showing discovered temperature sensor devices.

Fig. 5.1 Example 1 of SoCQ Query and Result.

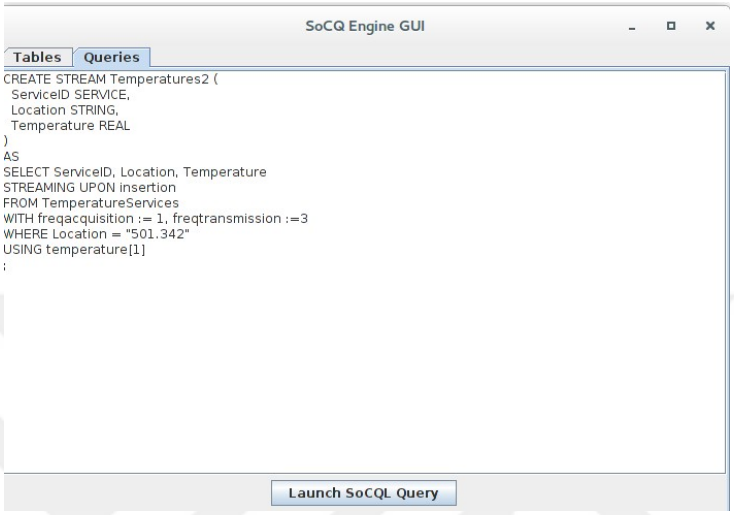


(a) SoCQ Parameterized Continuous Query to start stream temperature service.

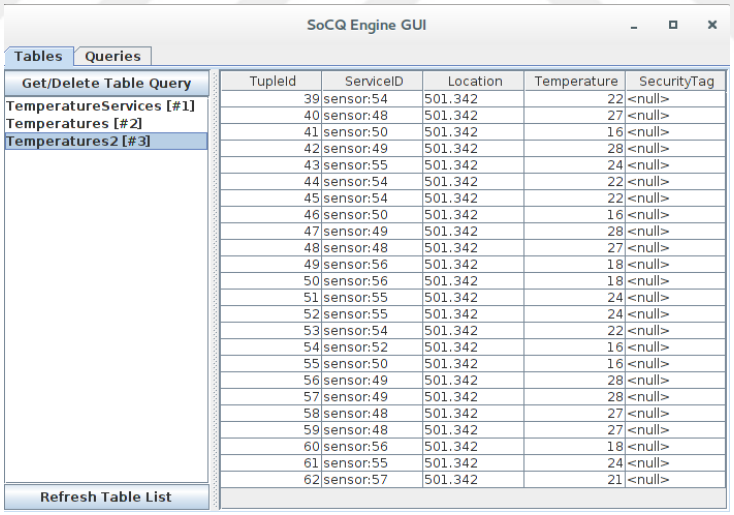


(b) SoCQ GUI Result of the Parameterized Continuous Query showing stream temperature service.

Fig. 5.2 Example 2 of SoCQ Parameterized Continuous Query and Result.



(a) SoCQ Parameterized Continuous Query to start stream temperature service from a specific location.



(b) SoCQ GUI Result of the Parameterized Continuous Query showing stream temperature service from given location as a constraint.

Fig. 5.3 Example 3 of SoCQ Parameterized Continuous Query and Result.

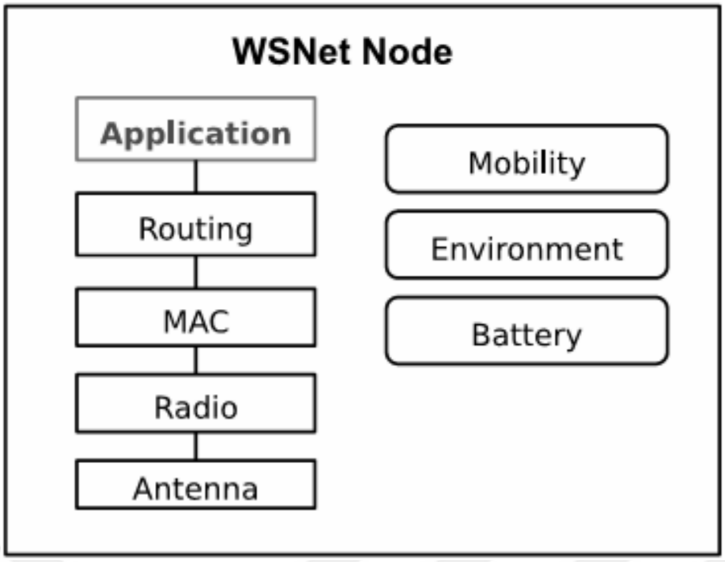


Fig. 5.4 Modular Architecture of a WSNet Node.

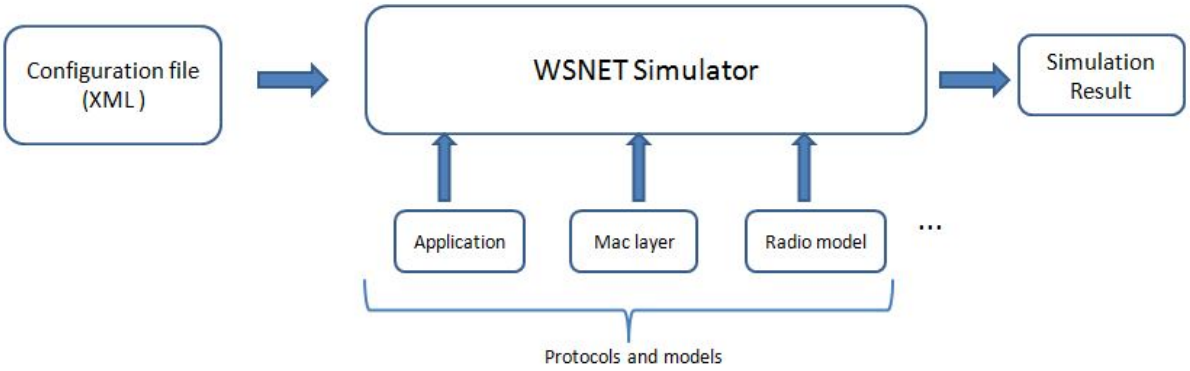


Fig. 5.5 I/O of WSNet Simulator.

to the lack of realistic low-level models, the simulation based approach is widely accepted and commonly used in the analysis of WSNs [97, 170].

5.1.2.2 Modified WSNet

In this work, WSN simulator WSNet [53] is chosen. WSNet is a modular event-driven simulator targeted to WSN written in C. Its major features are the scalability, extensibility and modularity for the integration of new protocols/hardware models and a precise radio medium simulation. The simulated nodes are built as an arbitrary assembly of blocks which represent either a hardware component, a software component or a behavior/resource of the node, such as Mobility, Energy source, Application, Routing protocols, Mac protocols, Radio interface, Antenna...

Figure 5.4 describes a node architecture that can be created in WSNet. There are already various modules that can be used for each part: support for complex nodes architecture (MIMO systems, multiple radio/antenna interface support), support for energy consumption simulation, support for various propagation models, support for propagation delays, etc. Modules are attached on run time and a configuration file is used to control the WSNet. I/O of the simulator is given in Figure 5.5.

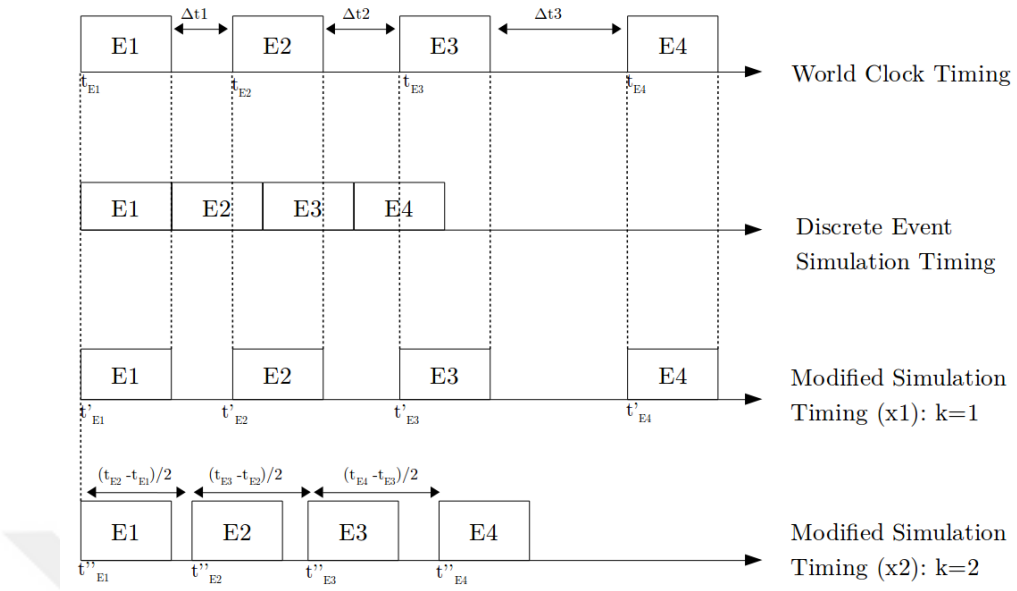


Fig. 5.6 Scheduling Events in Different Time Domains.

The simulated nodes (sensor devices), implemented in C, are built as an assembly of generic blocks taken from the `WSNet` library (e.g., hardware module, radio modules and protocols) and/or specific blocks developed for a particular experiment (e.g., specific application simulating a temperature sensor). A *Simulation Configuration* file (an XML file) defines the models and the protocols that are used in the wireless sensor network simulation, and the simulated environment properties such as simulation duration, the dimension of the area, the number of sensors, the position of each sensor (3-axis), attributes of each sensor device etc. At the beginning of each simulation, the simulator parses this configuration file as an input in order to execute the chosen models and protocol.

`WSNet` is a `discrete event simulator (DES)`. `DES` models the operation of a system as a discrete sequence of events in time. Each event occurs at a particular instant in time and marks a change of state in the system. Between consecutive events, no change in the system is assumed to occur, thus the simulation can directly jump in time from one event to the next. This is the key factor of being fast for the simulations.

However, in our settings the `Pervasive Environment Management System (PEMS)` works on real-time. For instance, a user may request sensor packets for the last 5 min: 5 min for the query engine is on the world clock timing, however, in 5 min, a `WSN` simulation may generate millions of sensor packets which is not realistic. That's the major drawback of using a `WSN` simulator with an external tool which works on real-time. To avoid time scheduling difference, **we modified the time scheduler** of `WSNet` simulator: we introduced a time factor (speed factor on time) k to run experiments in real-time or k-times faster ($\times 10$, $\times 20\ldots$) [136–138].

Figure 5.6 presents the influence of the chosen time domain on the execution process. Suppose that there are four events to execute: $\{E1, E2, E3, E4\}$. The time to execute these events are set on the world clock timing: $\{t_{E1}, t_{E2}, t_{E3}, t_{E4}\}$. In a simulation environment, with a basic `DES` based simulator, these events are executed consecutively without spending time as shown in the figure (on the discrete event simulation timing scale): the simulator does not wait for the next event time and directly jumps at the next event execution time. With the modification of the scheduler system (adding a speed factor on time k), we make it work on real-time (k=1) or accelerate the timing (e.g., k=2 as shown on the figure).

5.1.3 3SoSM Gateway

In the 3SoSM architecture (cf. Figure 4.1 in Chapter 4), the Gateway is a technical bridge between two environments: it manages interactions and bidirectional communication between the PEMS and the WSN. We implemented the 3SoSM principles in a 3SoSM Gateway. It is implemented in Java and interacts with the SoCQ Engine and WSNet. 3SoSM Gateway has two primary modules and two secondary modules:

- Service Manager is a module that manages sensor network devices and provided services. For every sensor discovered by a Sensor Discovery process, a sensor service should be registered to the system in order to be operated. When an application requests services from the system, the Service Manager module is checked and presents existing services. Subscriptions to sensor devices (services) are also initated in this module by interacting directly with SoCQ Engine. However, management of these subscriptions is handled in the module called Subscription Manager.

- Subscription Manager is a crucial module where we implement the optimization process of our 3SoSM approach. From our perspective, we consider each sensor data stream as a subscription to a service and this service is provided by a wireless sensor device. This module is responsible for the real-time analysis of application requirements and generates a SCO-PATTERN for each sensor device according to the set of parametrized subscriptions for that device.

- Communication Interface is responsible for bidirectional communication between the Gateway and the WSN environment, through the base station of WSN (called sink device, that aggregates sensor data). Thus, all sensor measurements pass to 3SoSM Gateway through this interface.

- Sensor Discovery is a module that is responsible for searching available wireless sensor devices (service providers) in WSN environment. Discovered sensor devices are presented to the system with their attributes and measure types. In the case of using a WSN simulator instead of real physical wireless sensor devices, this module checks topology and sensor configuration files of the WSN simulator and fetches relevant information about virtual simulated sensors.

With an integrated prototype, we can execute multiple SoCQ queries that dynamically subscribe to streams provided by WSNet nodes. The Gateway transparently optimizes the subscriptions and configures the WSNet nodes with SCO-PATTERNs.

5.2 Experimental Environment

During the experimental phase of our study, we simulate a real testbed: Project SoCQ4Home [66, 160], in which continuous query engine SoCQ is in use. This project tackles the general issue of the reduction of energy consumption for buildings. Through instrumentation with numerous and various sensors, such instrumented buildings can provide a lot of information about themselves while being occupied: effective usage of rooms, thermal behavior during days and nights, etc. Instrumented buildings can also provide additional services to their users, then become smart buildings. Figure 5.7 illustrates the SoCQ4Home project and presents the 3D projection of the testbed. The SoCQ4Home project focuses on the monitoring of instrumented buildings with four axes:

1. Middleware platform for easy deployment of sensors and sensor networks in buildings

2. Homogeneous modeling of monitoring systems to support efficient storage of heterogeneous sensor data

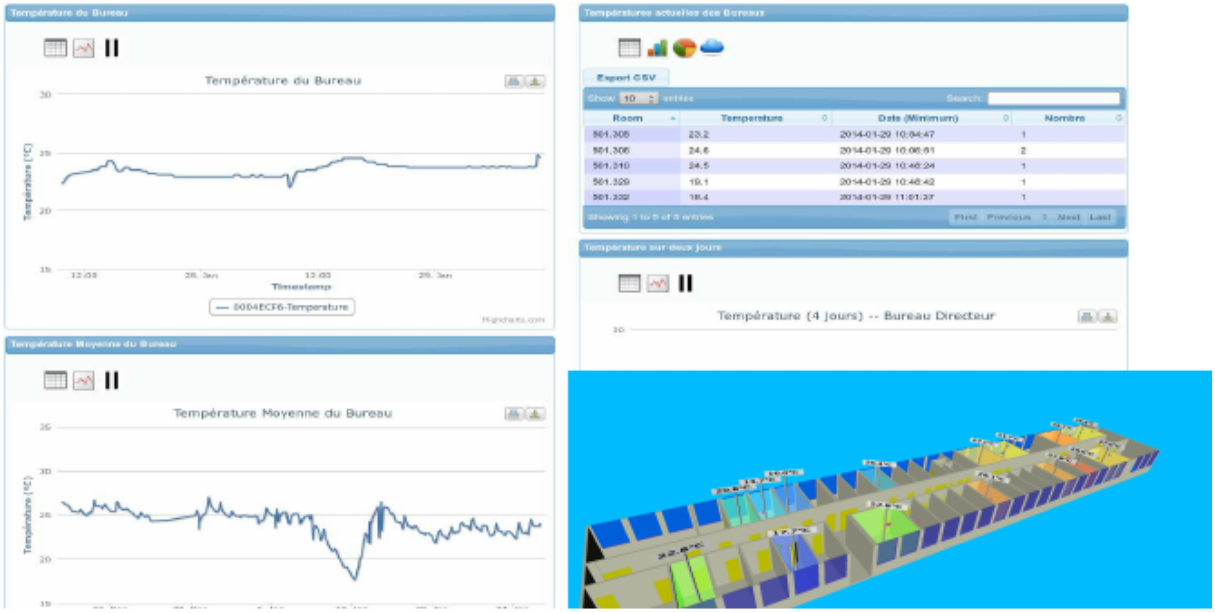


Fig. 5.7 SoCQ4Home DashBoard and 3D projection of sensor devices [160].

3. Declarative query language to support multidimensional data analysis and information discovery for monitoring applications
4. Declarative query language to support real-time interactions between devices and users for smart building applications

The simulations are performed on the `modified WSNet` simulator. The computer used for the experiments is equipped with a processor Intel 3.40GHz (6 cores) and 6GB of RAM. It runs a 64bit Linux-based operating system: Ubuntu 16.04.2 LTS.

5.3 Experimental Setup

We simulate one part of the topology of our physical platform `SoCQ4Home` deployed in our LIRIS laboratory [65, 66]: 70 simulated sensor devices are located at specific positions over a floor of the building (10m $\times$ 60m $\times$ 4m). The network topology is illustrated in Figure 5.8. The deployed sensor devices have fixed positions during the simulation and we consider that they have enough energy until the end of the simulation.

We adopt the most known pervasive environment communication protocol in `WSN`: `Zigbee IEEE 802.15.4` [173, 198], simulated with a UDG propagation model[1], 35m transmission range, modulated by BPSK [2] and basic radio module states for devices. For the routing model, the neighbor list for each device is predefined.

[1] Wireless ad hoc and sensor networks are most often modeled by Unit Disk Graphs [23], abbreviated by UDGs, a geometric graph $G = (V, E)$ in which the vertex set V is a set of n points in R^d, where d is the dimension, and the edge set E consists of m pairs from V . Let $dist(u,v)$ be the Euclidean distance between the nodes u and v: $dist(u,v) = \sqrt{(u_x v_x)^2 + (u_y v_y)^2 + (u_z v_z)^2}$. Two nodes u and v are considered adjacent if the Euclidean distance between them is less than or equal to 1 unit [3].

[2] Binary Phase-Shift Keying (BPSK) is a digital modulation scheme that conveys data by changing (modulating) the phase of a reference signal (the carrier wave). BPSK modulator is the basic component for generating advanced modulation formats such as polarization-division-multiplexed quadrature phase-shift keying. The modulation is impressed by varying the sine and cosine

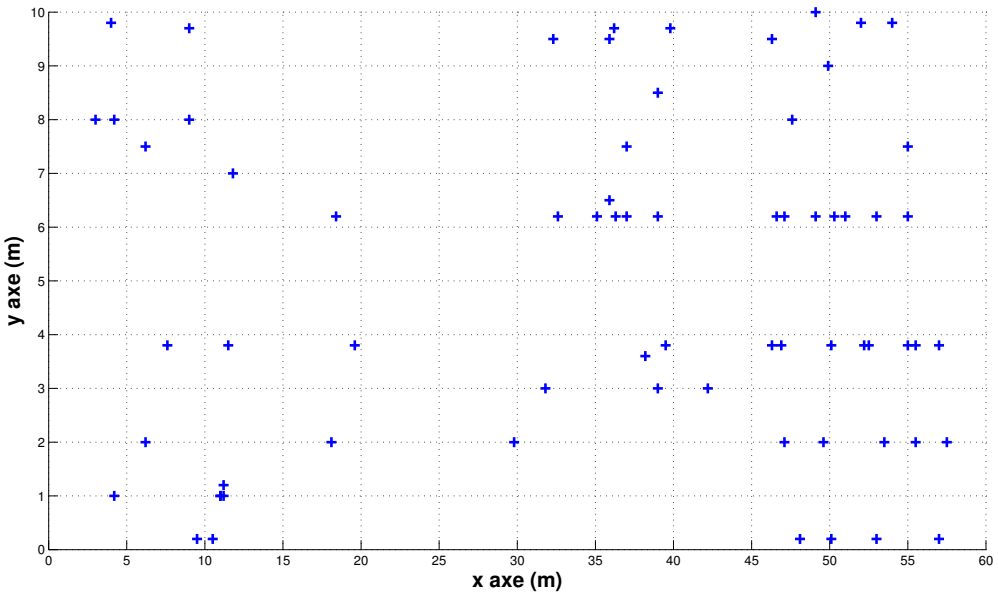


Fig. 5.8 2D Network Topology for WSNet Simulation.

Each sensor device, after sending a data packet, goes into `listen` state and listens to the channel in case there is a device trying to communicate with it (either a sensor device trying to reach the `Sink` device or `3SoSM Gateway` sending a configuration packet). For the radio model, we adopt basic states `Idle, RX, TX, Sleep` which stand for respectively idle state, packet reception or channel listening state, packet transmission state and the sleep mode [139, 140].

During the simulation, energy consumption of each sensor device is analyzed. Calculation of energy consumption is formalized in Equation 5.1 where $\mathscr{E}$, $\mathtt{V}_{drain}$, $\mathtt{I}_i$ and Δt_i stand for respectively energy consumption of sensor device component (CPU and radio), drain voltage, current of state `i` and time passed in state `i`. Numerical application of energy consumption is based on hardware datasheets of micro-controller and radio transmitter [47].

Table 5.1 summarizes the simulation setups.

$$
\begin{aligned}
&\mathscr{E}_{sensor} = \mathscr{E}_{CPU} + \mathscr{E}_{radio} \\
&\mathscr{E}_{CPU} = V_{drain} \times I_{active} \times \Delta t_{active} \\
&\mathscr{E}_{radio} = \mathscr{E}_{Idle} + \mathscr{E}_{TX} + \mathscr{E}_{RX} + \mathscr{E}_{Sleep} \\
&\mathscr{E}_{radio} = V_{drain} \times I_{Idle} \times \Delta t_{idle} + V_{drain} \times I_{TX} \times \Delta t_{TX} \\
&+ V_{drain} \times I_{RX} \times \Delta t_{RX} + + V_{drain} \times I_{sleep} \times \Delta t_{sleep}
\end{aligned}
\tag{5.1}
$$

inputs at a precise time. BPSK modulation is the most robust of all the PSKs: it takes the higher level of noise or distortion to make the demodulator reach an incorrect decision. The BPSK modulation is a technique widely used in various wireless standards such as CDMA, WiMAX (16d, 16e), WLAN 11a, 11b, 11g, 11n, Satellite, DVB, Cable modem etc [17, 100].

Table 5.1 Summary of the Simulation Parameters.

Parameter	Value
Number of sensor devices	69 sensor devices + 1 sink device
Area size	10m x 60m x 4m
Propagation model	Unit Disk Graph
Transmission range	35m
Topology	(x,y,z) coordinates given for each device
Modulation model	bpsk
Antenna model	omnidirectional
Radio model	802_15_4_902_bpsk (Zigbee model) [188]
MAC model	802_15_4_902_bpsk_u_csma_ca [11]
Routing model	Predefined static neighbor list

5.4 Conclusion

In this chapter, we have presented the prototype to perform experiments. Our prototype respects the the declarative monitoring architecture for pervasive environments. Based on this architecture, we build our prototype with three main components:

1. a continuous query engine to manage the PEMS: We use Service-oriented Continuous Query SoCQ Engine [68]. It is implemented in Java and provides a unified view and access to the various and heterogeneous resources available in the environment.

2. a wireless sensor network simulator: We use a modified WSNet simulator [53]. It is a modular event-driven simulator targeted to WSN written in C. It allows users to define/develop models, protocols, and to choose the simulated environment properties such as simulation duration, the dimension of the area, the number of sensors, the position of each sensor (3-axis), attributes of each sensor device etc.

3. a gateway: It is is a technical bridge between two environments: it manages interactions and bidirectional communication between the PEMS and the WSN. We implemented the 3SoSM principles in a 3SoSM Gateway. It is implemented in Java and interacts with the SoCQ Engine and WSNet.

Besides, the experimental environment is presented: we inspired ourselves from the topology use in Project SoCQ4Home [66, 160]. The experimental setup is also defined: chosen models and protocols for the WSN environment are presented, and the technical characteristics of the computer used to run the simulations are given.

Chapter 6

Experiments

In this chapter, we present our experiments with test scenarios and we introduce the results obtained by these experiments. The experimental environment has already been described in the previous chapter. To validate our approach and implementation of 3SoSM, we create various test scenarios targeting smart building applications. We compare with 2 types of architecture:

1. Architecture with basic duty cycle WSN devices, requiring a static configuration
2. Architecture implementing the 3SoSM approach where WSN devices can be dynamically configured with SCO-PATTERN generated by the 3SoSM Gateway.

We first describe our 4 test scenarios in the context of smart buildings. Then, we present the applications for our test scenarios and the application requirements (parameters defined in the application). We evaluate the Device-Centered 3SoSM approach with the first 2 scenarios: we describe the application requirements, and present the results of the experimentations. We then evaluate the Network-Aware 3SoSM approach with 4 scenarios: we again describe the application requirements and present the results of the experimentations. Each scenario is performed during one day, and we focus on the consumed and remaining energy on the WSN devices.

6.1 Test Scenarios

To observe the performance of our approach (Device-Centered 3SoSM and Network-Aware 3SoSM) and to evaluate it, we design diverse test scenarios. The test scenarios are related to basic smart building features such as temperature, occupancy, luminosity, CO_2 emission etc. Scenario 1 and 2 are related to the monitoring of temperature in rooms. Scenario 3 focuses on CO_2 emission in rooms, and Scenario 4 on light level.

During the experiments, we use the prototype that we presented in Chapter 5. 70 simulated sensor devices are located at specific positions over a virtual floor of the building (10m $\times$ 60m $\times$ 4m) and form the network topology illustrated in Figure 5.8. Each scenario is performed during one day (1440 minutes) on WSNet simulator with constant $\alpha = 5$ as a time factor (cf. Section 5.1.2.2).

6.1.1 Scenario 1: Comfort Temperature

It is well known that one of the objectives of smart buildings is to provide high-level comfort to customers. For indoor temperature control systems, users define their personal comfort zone by setting the parameters of minimum temperature and maximum temperature [Tmin, Tmax] [180]. This comfort zone can be either generated by a combination of existing control system or by the variations of outdoor. In this scenario, we

adopt adaptive comfort temperature models that are included in Chartered Institute of Building Engineers (CIBSE) (2006) guidelines and European Committee for Standardization (CEN) standard EN 15251. Proposed comfort temperature is:

$17.8 + 0.33 \times T_o < T_c < 22.6 + 0.09 \times T_o$

where $T_c = C + \kappa \times T_o$ (*CIBSE 2007 and CEN Standard EN 15252-2007*). T_o and T_c stand for respectively outdoor temperature (°C) and indoor comfort temperature (°C), C and κ are constants. Temperature data is generated on the sensor side by the simulated wireless sensor devices.

In this scenario, the objective is to manage multiple application requirements that request the same service (temperature service) from the system with different subscription requests. During the execution process, there are two applications: one application requests the temperature of each room with some parameters (acquisition frequency and transmission frequency), the other one focuses on the rooms that are not in the comfort range and requests temperature of such rooms with different parameters.

6.1.2 Scenario 2: Room Occupancy

One of the key features of a smart building environment is the occupancy information. Knowing whether or not someone is in a room, or even in the house at all is vital. Accurate occupancy information in buildings can enable several useful applications. The most known sensor device for occupancy is the PIR (Passive infra-red sensor) motion sensor with 360 degree viewing angle lens for maximum efficiency in different room settings. PIR sensors allow you to sense motion, and almost always detect whether a human has moved in or out of the sensors range. PIRs are basically made of a pyroelectric sensor which can detect levels of infra-red radiation. Everything emits some low level radiation, and the hotter something is, the more radiation is emitted [116].

Even though the basic component for the occupancy detection is the motion sensor device, they have many disadvantages: When someone is sitting still for a while no motion is ever reported and it ends up with the lights turning off in the room. This results in having to wave arms in the air to make the motion detector reacts. Thus, a motion sensor is commonly associated with other sensors such as door status, light status, CO_2 emission etc to be more accurate [33, 130].

In this scenario, the objective is still to manage multiple application requirements but this time, these applications may benefit from multi-modality of the sensor devices and request services (e.g. temperature and presence services) with different subscription requests. During the execution process, there are two applications: one application requests the temperature of each room with some parameters, the other one focuses on the occupied rooms and requests temperature of occupied rooms more frequently. In fact, this application needs occupancy information as well and it extracts occupancy information from motion detection sensor devices.

6.1.3 Scenario 3: CO_2 Emission in a Room

As introduced previously, occupancy information is not only provided by the motion sensors. Non-terminal based detection systems such as carbon-dioxide (CO_2) sensors can also provide occupancy information. Humans naturally exhale CO_2 on a constant basis making it present in varying amount in spaces. Each activity, as depicted in Figure 6.1, results in a different body metabolic rate and CO_2. CO_2 sensors measure the concentration of gases in a space in parts-per-million (PPM). The measurement of the amount of CO_2 in a space can be used in many other services as well [13, 123]. It is well adopted especially by the HVAC systems to provide information on the indoor air quality [62].

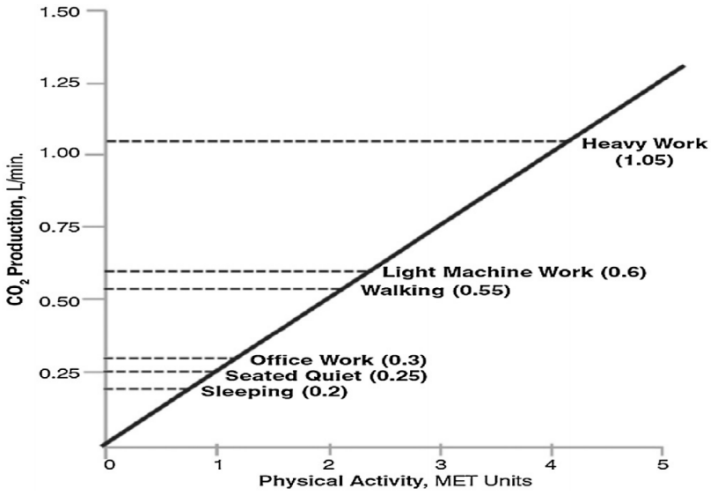
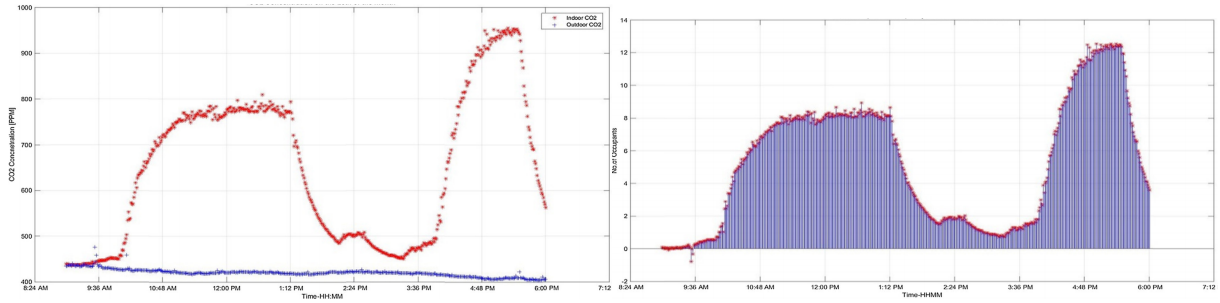


Fig. 6.1 CO_2 Production and Metabolic Activity [43].

The number of occupants in the space was also estimated from the measured space CO_2 using the steady state relation provided in [182]:

$$No.Occupants = \frac{SA \,.\, (N - Ci))}{G \,.\, 10^6} \tag{6.1}$$

where N is the space CO_2 concentration at the present time step measured in ppm, SA is the supply airflow rate, C_i is the CO_2 concentration in the supply air measured in ppm, G is the CO_2 generation rate per person measure in cfm [90]. An example of CO_2 concentration measurements and estimation of number of occupants based on the CO_2 are illustrated respectively in Figure 6.2a and Figure 6.2b.



(a) Space CO_2 Concentration Measurement. [90].

(b) CO_2 Estimated Number of Room Occupants. [90].

Fig. 6.2 Example of Estimating Number of Room Occupants based on CO_2 Concentration.

This scenario as the previous one requires temperature and presence information, however, as a distinct from the Scenario 2, here occupancy information is expected to be provided by CO_2 sensors. Thus, during the execution process, there are two applications: one application requests the temperature of each room with some parameters, the other one focuses on the occupied rooms and requests temperature of occupied rooms more frequently. In fact, this application needs occupancy information as well and it requires CO_2 emission measures from CO_2 sensor devices.

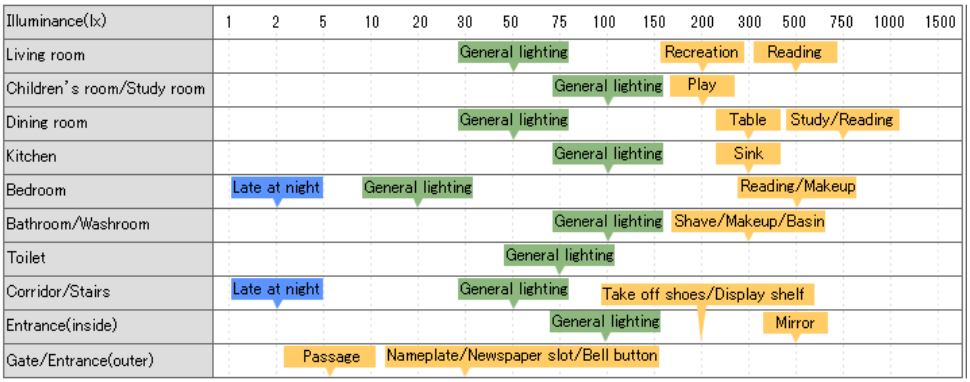


Fig. 6.3 Recommended Luminosity Levels for Different Tasks [117].

6.1.4 Scenario 4: Luminosity of a Room

Lighting service is an essential provision for any workplace especially for offices. It is preferable to provide a correct luminosity over the entire workplace by combining both natural and artificial lighting. Localised lighting services may be required in certain cases to cut costs and improve luminosity. Right lighting service helps us to see and to recognise the objects around us and also the hazards. It can also reduce visual strain and discomfort [95, 117].

On the other hand, poor or insufficient luminosity level may affect occupants' performance and health as poor visibility increases the chances of errors being made. Poor lighting can cause several problems such as:

- Insufficient light - not enough (too little) light for the need.
- Glare - too much light for the need.
- Improper contrast.
- Poorly distributed light.
- Flicker.

Nowadays, approximately 80% of the information is processed visually. This important portion leads to the fact that the luminosity level of the environment has an extremly significant impact on the adequate processing of information [44, 110]. Figure 6.3 presents the recommended luminosity levels for different daily activities realized in different rooms.

In this scenario, the objective is to fulfill the application requirements that both request luminosity information from the relevant sensor devices with different parameters. In fact, this scenario does not require the multi-modality of the sensor devices, as applications request only luminosity services.

6.2 Experiments with Device-Centered 3SoSM Approach

In this section, we present the experiments in which we applied our `Device-Centered 3SoSM` approach. We defined application requirements and implemented applications as a set of service-oriented continuous queries for the `SoCQ` Engine. We conducted experiments on Scenario 1 and 2. Results in terms of energy consumption are presented and discussed.

6.2.1 Scenario 1: Comfort Temperature

For the first scenario, we assume that there are two applications:

- Application 1: App 1 requests temperature service from the system. It asks to monitor the temperature degree of each room of the building with an acquisition periodicity of 10 sec and a transmission periodicity of 60 sec: (p^{acq}=10, p^{tx}=60).
- Application 2: App 2 has the same requirements as the as App1 (temperature service, with p^{acq}=10, p^{tx}=60) but it also requests to track more frequently temperature degree for rooms that are out of comfort temperature interval with parameters (p^{acq}=1, p^{tx}=5).

As shown from the description of the applications, Application 2 requests the same service as the Application 1. Unlike Application 1, Application 2 also asks to focus on rooms that are out of comfort temperature interval and it requests the track the temperature degree of these rooms (if any) more frequently than the others.

Here the key point is the dynamicity of the building environment. A temperature of a room may always be in the comfort temperature interval. In this case, the condition defined by the Application 2 may never be valid and Application 2 tracks the temperature degree of each room as Application 1 does (since they have the same parameters). To fulfill the given condition of Application 2, the system manages a dynamic list of the rooms that are out of the comfort temperature interval.

Before the subscription to the relevant services, a `SoCQ` query is executed in order to discover sensor devices that have temperature services. Once relevant services are determined and listed, subscription request to services is sent to `PEMS` for sensor data streaming. Our `Device-Centered 3SoSM` optimization process is executed on the `3SoSM` Gateway. SCO-PATTERNs are generated for the relevant sensors and are sent to launch the sensor data streams. Once the first application starts being executed, the second application is launched. Second application requests more frequently temperature of rooms which temperature is not in comfort temperature interval.

Table 6.1 SoCQ Query to Discover Temperature Sensors

```
CREATE RELATION TemperatureServices (
ServiceID SERVICE,
Location STRING,
Temperature NUMBER VIRTUAL,
Periodicity INTEGER VIRTUAL,
Latency INTEGER VIRTUAL
) USING (
getTemperature[ServiceID]():(Temperature),
temperature[ServiceID](Periodicity, Latency):(Temperature) STREAMING
) AS
DISCOVER SERVICES PROVIDING
PROPERTY Location STRING,
METHOD getTemperature ( ) : ( NUMBER ),
STREAM temperature (INTEGER, INTEGER ) : ( NUMBER )
```

ServiceID	**Location**	***Temperature***	***Periodicity***	***Latency***
sensor:01	501.337	*	*	*
sensor:03	501.340	*	*	*
sensor:17	502.321	*	*	*
sensor:04	501.301	*	*	*
sensor:28	501.303	*	*	*
sensor:60	502.305	*	*	*

Table 6.2 Parametrized Subscription Queries for Scenario 1.

```
/* Q1: Parametrized subscription query of Application 1 and Application 2 */
CREATE STREAM Temperatures (
ServiceID SERVICE,
Location STRING,
Temperature REAL,
Periodicity INTEGER VIRTUAL,
Latency INTEGER VIRTUAL
) AS
SELECT ServiceID, Location, Temperature
STREAMING UPON insertion
FROM TemperatureServices
WITH Periodicity := 10, Latency := 60
USING temperature[1];
```

```
/* Q2: Additional parametrized subscription query of Application 2 */
CREATE STREAM OutofComfortRoomsTemperatures (
ServiceID SERVICE,
Location STRING,
Temperature REAL,
Periodicity INTEGER VIRTUAL,
Latency INTEGER VIRTUAL
) AS
SELECT ServiceID, ReachToComfort.Location, Temperature
STREAMING UPON insertion
FROM ReachToComfort, TemperatureServices
WITH Periodicity := 1, Latency := 5
WHERE ReachToComfort.Location = TemperatureServices.Location
USING temperature[1];
```

Table 6.1 presents the discovery query to find temperature sensors in the environment and Table 6.2 presents the parametrized SQL-Like queries (Q1 and Q2) for launching sensor data streams with relevant parameters. As a result, both applications receive the temperature data flow from each room with given parameters. In addition to this, Application 2 also receives the data flow from rooms in which temperature degree is out of comfort interval. If the room temperature is always in the comfort temperature interval, Application 2 can not get the additional data flow. This functionality shows the context-awareness and the dynamicity features of our approach `3SoSM`.

As shown in the given queries, for all temperature sensor devices, both applications requests data acquisition each 10 seconds with a maximum data transmission latency of 60 seconds. Then, the second application requests the same subscriptions with the same parameters, and also requests subscriptions with different parameters (data acquisition each 5 seconds, with a maximum data transmission latency of 5 seconds) to sensors that are in a room which temperature is not in comfort interval. Since data transmission is the most expensive action in sensor side (in terms of energy consumption), we expect higher energy consumption for sensor devices that are in rooms where temperature is not in comfort temperature interval.

In this scenario, we consider the worst case for duty cycle: the acquisition periodicity is set to 1 sec and the transmission periodicity is set to 5 sec. Figure 6.4 shows the evolution of remaining energy of single wireless temperature sensor device with using most common duty cycle method and with our

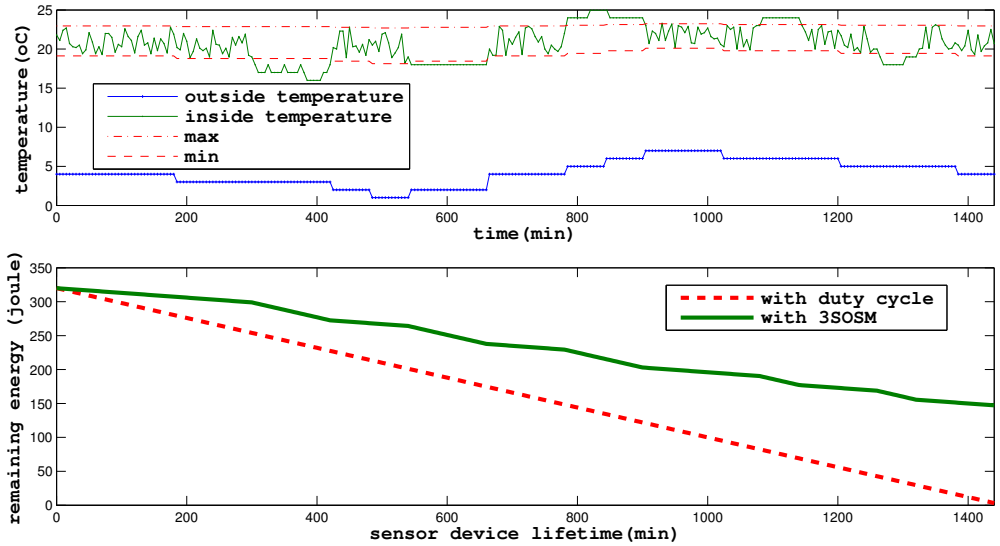


Fig. 6.4 Temperature of a Room and Evolution of Remaining Energy of a Sensor Device during 24 h (Applying Device-Centered 3SoSM - Scenario 1).

3SoSM approach. These graphs focus on a single sensor device that measures temperature and presents the slots where the additional requests of Application 2 is launched or dismissed. Upper graph presents outdoor, indoor temperatures and also dynamic comfort temperature intervals (dashed curves) based on the comfort temperature model (given in the scenario description). Thus, regions where indoor temperature is outside of comfort temperature interval are clearly visible. Lower graph shows the decrease of energy budget and lifetime of that sensor device that is located in that room. Energy consumption increases while room temperature is outside of comfort temperature interval. According to the temperature measures, while t∈[300;420], t∈[540;660], t∈[780;900], t∈[1080;1140] and t∈[1260;1320], indoor temperature is outside of comfort temperature boundaries: special condition of Application 2 is then valid, thus, data transmission frequency is increased according to application requirement.

Results show that the energy consumption of that sensor device is higher with duty cycle than with dynamic re-configuration: as a consequence, the sensor device dies earlier. With a static configuration, it nearly dies at the end of the day, whereas it still has energy with dynamic re-configuration. The initial energy level of a sensor device is here set on purpose to emphasize the lifetime difference between both cases. Based on the given scenario parameters, we achieve to reduce additional communication cost: For instance, while the temperature is inside the given comfort range, without our approach, a single temperature sensor sends 720 data packets to the base station during one hour (transmission periodicity is 5 sec, 12 transmissions per minute). With the 3SoSM Gateway, a number of the transmitted packets is only 60 (transmission periodicity is 60 sec, 1 transmission per minute) for the same time period.

Our approach can increase the lifetime of a sensor device. Let's consider a basic 9V alkaline battery ($\simeq$ 18.8kJ) connected to our sensor device. With given parameters, the sensor can survive only 59.5 days with a static configuration, however, its lifetime can be extended up to 109.1 days with dynamic re-configuration. Here, 3SoSM lengthens sensor lifetime by $\simeq$ 83%. Obviously, service lifespan and the energy savings depend on the application requirements and context (here, actual temperature).

Table 6.3 SQL-Like Queries for Scenario 2.

```
CREATE RELATION OccupiedRooms (
Location STRING
)
AS
SELECT Location
FROM Occupancy [30]
WHERE Occupancy = TRUE
GROUP BY Location
;
```

```
CREATE STREAM OccupiedRoomsTemperatures (
ServiceID SERVICE,
Location STRING,
Temperature REAL,
Periodicity INTEGER VIRTUAL,
Latency INTEGER VIRTUAL
)
AS
SELECT ServiceID, TemperatureServices.Location,
Temperature
STREAMING UPON insertion
FROM OccupiedRooms, TemperatureServices
  WITH Periodicity := 1, Latency := 4
WHERE OccupiedRooms.Location =
TemperatureServices.Location
USING temperature[1]
;
```

6.2.2 Scenario 2: Room Occupancy

For the second scenario, we suppose that there exist two applications:

- Application 1: App 1 requests temperature service from the system. It asks to monitor the temperature degree of each room of the building with an acquisition periodicty of 15 sec and a transmission periodicity 60 sec: (p^{acq}=15, p^{tx}=60).
- Application 2: App 2 has the same requirements as the as App1 (temperature service, with p^{acq}=15,p^{tx}=60) but it also requests to track more frequently temperature degree for occupied rooms with parameters (p^{acq}=1, p^{tx}=4).

Here, we observe that *Application 1* requests the temperature service from the monitoring system with given application requirements. *Application 2* also requests temperature service from the monitoring system, but with a special constraint. It intends to track the temperature of occupied rooms more frequently than the other rooms. To support this application, occupancy and temperature services are needed.

For the duty cycle approach, we adopt a static configuration. In this case, the system uses the worst case situation: occupation may always be true for every location during the simulation. Based on this constraint, the system adopts these configurations at the beginning: p^{acq}=1, p^{tx}=4.

Relevant queries are given in Table 6.3. For detecting the occupied rooms, we use a time window (line `FROM Occupancy [30]`) over the data stream that stands for the computation only on data of the last 30 seconds.

To respond to *Application 1*, a set of `SoCQ` queries is implemented: Firstly a discovery query to discover sensor devices that may serve temperature service and to create a relation with these devices, secondly a stream query to subscribe these devices with given application requirements. For Application 2, an additional set of `SoCQ` queries is implemented. Firstly, a discovery query to explore sensor devices that may serve occupancy services. Then, a stream query for the subscription to the devices that provide occupancy service. From the occupancy stream, we extract the location attribute of the occupied rooms. Afterwards, we list the temperature sensor devices that are located in these rooms. Finally, for these

devices, we implement a new stream query to subscribe to the temperature service with given application requirements. Since these are continuous queries, once a room is not occupied any more or an unoccupied room become occupied, lists are refreshed and new subscriptions or unsubcriptions are handled in real-time by our 3SoSM approach. If the room is no longer occupied, App2 unsubscribes from these devices and if a room becomes occupied, then App 2 subscribes to the sensor devices in that room.

Briefly, App 1 is launched in every condition and it receives temperature data from relevant devices. However, at least one room should be occupied so that App 2 may receive temperature data more frequently. Here, there are two subscriptions with different application requirements to the same set of temperature sensor devices located in occupied rooms.

Figure 6.5 shows the occupancy of a room and the evolution of energy budget of a wireless sensor device that is located in that room. Influence of using 3SoSM approach on energy consumption is presented in this graph. In the upper graph, occupancy of a classroom is simulated. For periods t∈[480;590], t∈[600;720], t∈[840;950] and t∈[960;1080], wireless sensor device indicates that the classroom is occupied, thus according to the given applications, the second subscription request with high data transmission frequency is launched. Lower graph presents the evolution of the remaining energy of the sensor device. It presents that sensor device that adopts duty cycle method runs out of energy and stops functioning at the end of the day whereas sensor device that uses dynamic reconfiguration lasts longer than the other. During the simulation, we obtained 46.8% of energy saving.

To better compare our approach with the duty cycle method, we now assume that the room is always occupied during the entire simulation. In theory, if a room is always occupied than the configuration of the sensor devices in that room should be the same for both cases thus the energy consumption for both cases should be nearly the same.

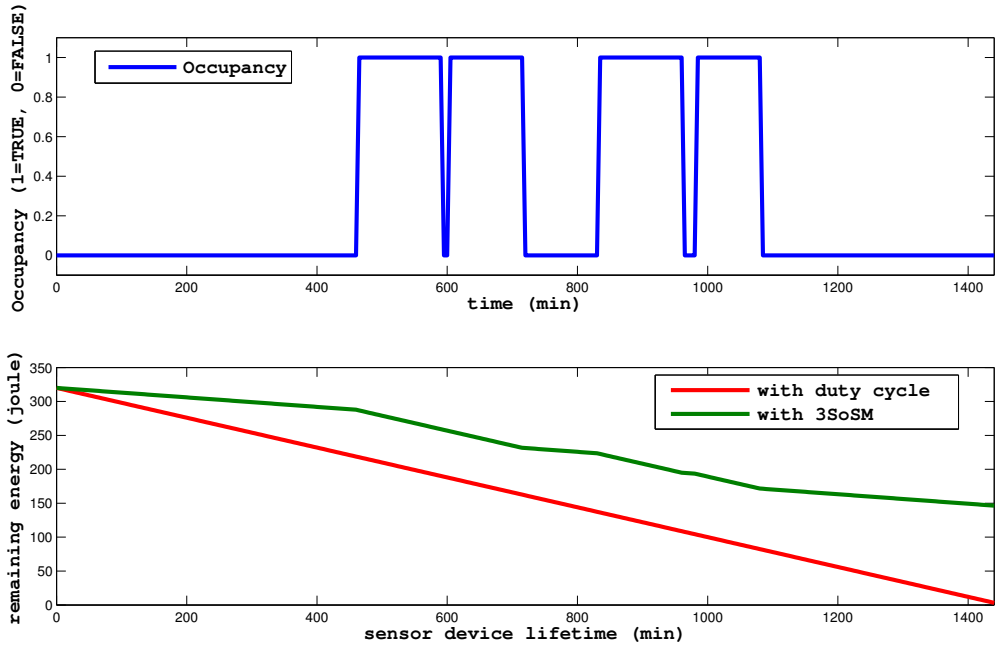


Fig. 6.5 Occupancy of a Room and Evolution of Remaining Energy of a Sensor Device during 24 h (Applying Device-Centered 3SoSM - Scenario 2).

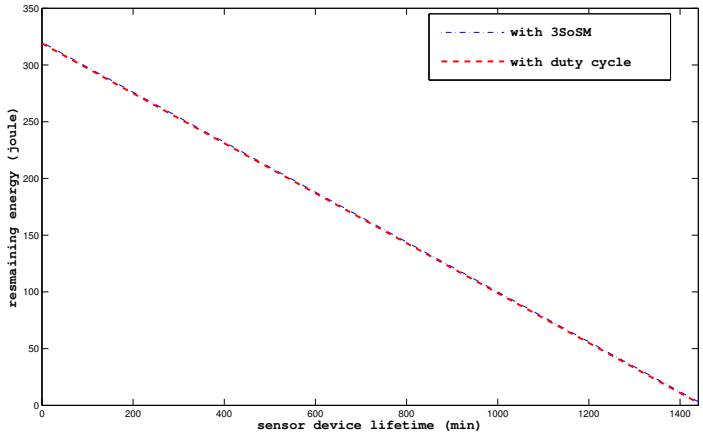


Fig. 6.6 Evolution of Remaining Energy of a Sensor Device during 24 h (Applying Device-Centered 3SoSM - Scenario 2).

Figure 6.6 shows the remaining energy of the sensor devices for both cases. The graph proves that sensor devices (one with duty cycle method, the other one with `3SoSM` approach) consume approximately the same amount of energy. For the given application parameters, both of the sensor devices die at the end of the worst case scenario.

6.3 Experiments with Network-Aware 3SoSM Approach

In this section, we present the experiments in which we applied the `Network-Aware 3SoSM` approach. Here, we consider the 4 scenario already described. In these experiments, we benefit from the new features of our approach: latency information, and network topology. Latency information gives opportunity to keep acquired and/or received sensor data in a buffer and to send them at a proper time slot if possible instead of sending them immediately. In this case, sensor devices try to group the sensor data and to send this set of data at one time. Besides, during these experiments, for each new subscription or unsubscription a new topology is created and a role is assigned to each sensor device in the environment: Source, Relay or Source-Relay (see Section 4.4.1).

6.3.1 Scenario 1: Comfort Temperature

In this scenario, there exist two applications:

1. Application 1: App 1 requests temperature service from the system. It asks to monitor the temperature degree of each room of the building with an acquisition periodicity of 20 sec and a latency of 60 sec: (p^{acq}=20, λ=60).

2. Application 2: App 2 requests temperature service as well. It requests to monitor the temperature degree of each room with parameters (p^{acq}=20, λ=60). But it also requests to track more frequently temperature degree of the rooms that are out of the comfort range (p^{acq}=10, λ=20).

For the duty cycle approach, we adopt a static configuration with the following parameters for the sensor device: (p^{acq}=10, λ=20), corresponding to the worst case scenario (temperature may always be out of comfort range). Figure 6.7 presents the status of the temperature. Like in the previous comfort range scenario, we create a comfort temperature interval. If the temperature of the room is inside the range then

it is represented as 0, if the temperature of the room is outside of the comfort temperature range then it is represented as 1 in the figure.

Figure 6.8 presents the average energy consumption of the source-relay sensor devices. We illustrate only the source-relay sensor devices since this type of sensors consumes more than the others due to their workload and have a significant effect on the lifetime of the network.

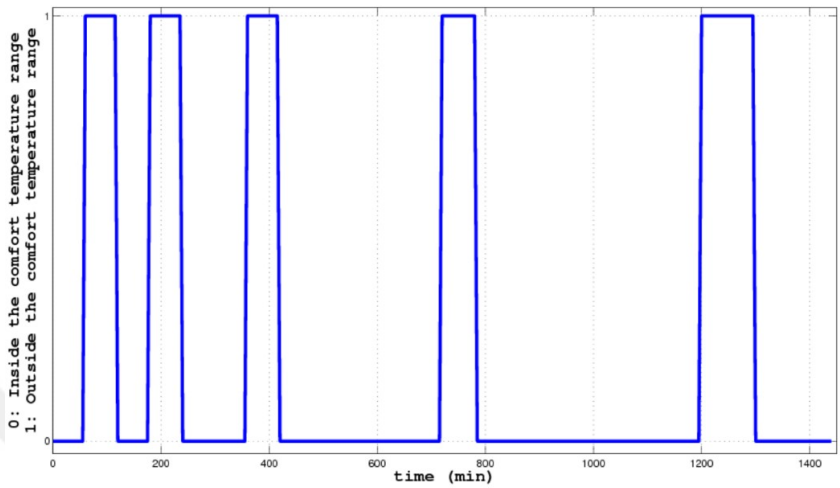


Fig. 6.7 Status of the Temperature (0 for inside, 1 for outside the comfort temperature range) (Applying Network-Aware 3SoSM - Scenario 1).

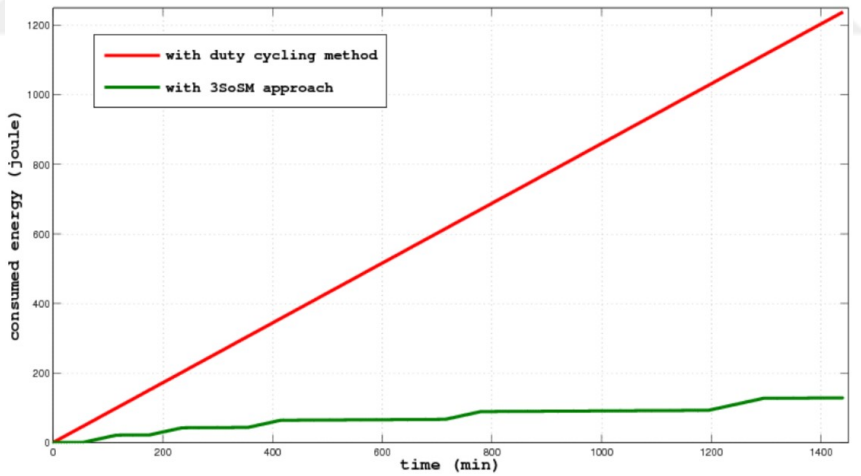


Fig. 6.8 Average energy consumption of source-relay sensor devices of the network (Applying Network-Aware 3SoSM - Scenario 1).

With the given application requirements, the most energy consuming sensor type (source-relay sensor device) burns 6,6% of its energy at the end of the day. However, with the `Network-Aware 3SoSM` approach, the same sensor device consumes only 0,7% of its energy budget during the same period.

Here, with our `Network-Aware 3SoSM` approach, we try to avoid unnecessary data measurements and to promote less expensive data transmission for sensor devices, Figure 6.9 illustrates the average number of generated packets during the last 5 min by the running sensor devices in the environment. It presents the big gap between two cases in terms of generated packets.

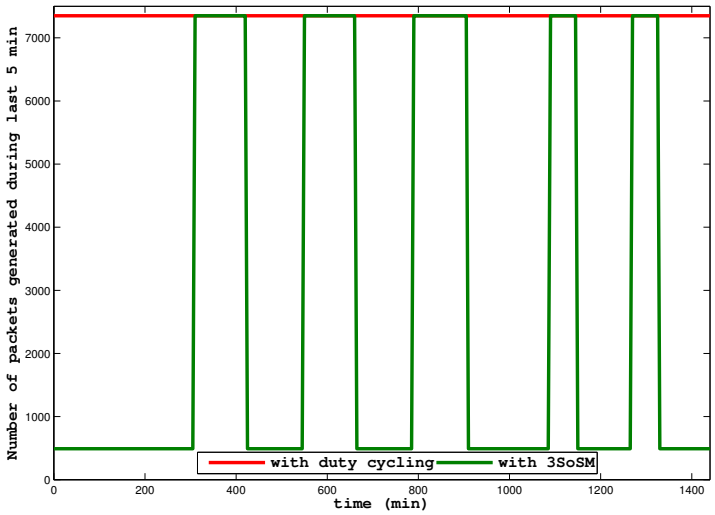


Fig. 6.9 Number of Generated Packets during the last 5 min
(Applying Network-Aware 3SoSM - Scenario 1).

With the duty cycle method, that sensor has a lifetime 15 days. On the other side, with the `Network-Aware 3SoSM` approach, its lifetime is extended to 146 days. These results, although they may vary based on the given parameters, illustrate the benefits of the dynamic behavior for the sensor devices and the interaction between the `PEMS` and the sensor devices.

6.3.2 Scenario 2: Room Occupancy

In this scenario, there exist two applications:

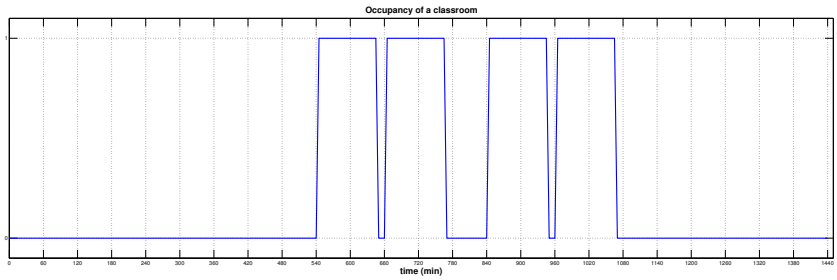
1. Application 1: App 1 requests temperature service from the system. It asks to monitor the temperature degree of each room of the building with an acquisition periodicity of 15 sec and a latency of 60 sec: (p^{acq}=15, λ=60).

2. Application 2: App 2 requests temperature service as well. It requests to monitor the temperature degree of each room with parameters (p^{acq}=15, λ=60). But it also requests to track more frequently temperature degree only for occupied rooms (p^{acq}=1, λ=4).

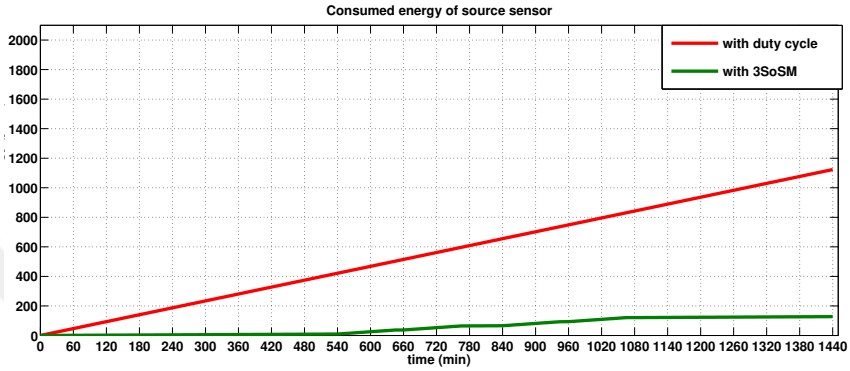
Here, there are two subscriptions to the same set of temperature sensor devices located in occupied rooms with different application requirements. For the duty cycle method, we adopt the static configuration: (p^{acq}=1, p^{tx}=4), corresponding to the worst case scenario (rooms may always be occupied).

Firstly, the room is not occupied, hence the `3SoSM Gateway` creates a schedule and configures the relevant sensor devices according to that situation. Once the room is tagged as occupied by the received occupancy data stream, the `3SoSM Gateway` re-creates a schedule and re-configures the relevant sensor devices. For each contextual change in the environment, the `3SoSM Gateway` creates a new schedule and configures the sensor device to adapt to the recent situation. Thus, adapting the system to the dynamic context avoids sensor devices from unnecessary data acquisitions and transmissions. Hence, the economy of energy can be achieved with this context-awareness.

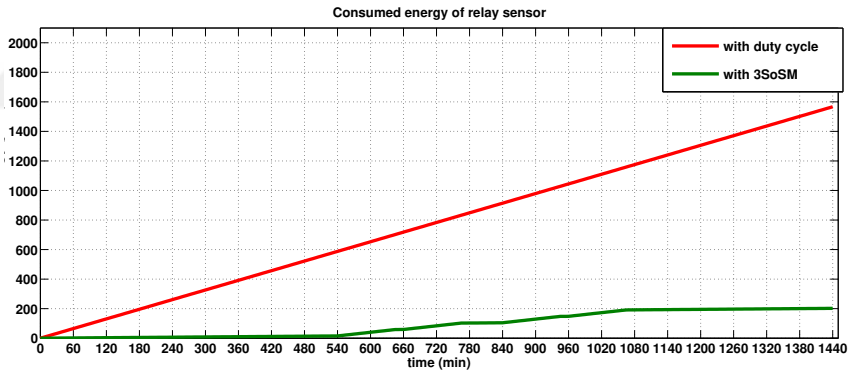
Figure 6.10a indicates the occupancy information of a selected classroom of our testbed. Figure 6.10b presents the average energy consumption of the source sensor devices. Figure 6.10c and 6.10d present respectively average energy consumption of the relay and source-relay sensor devices. For Figures 6.10b,



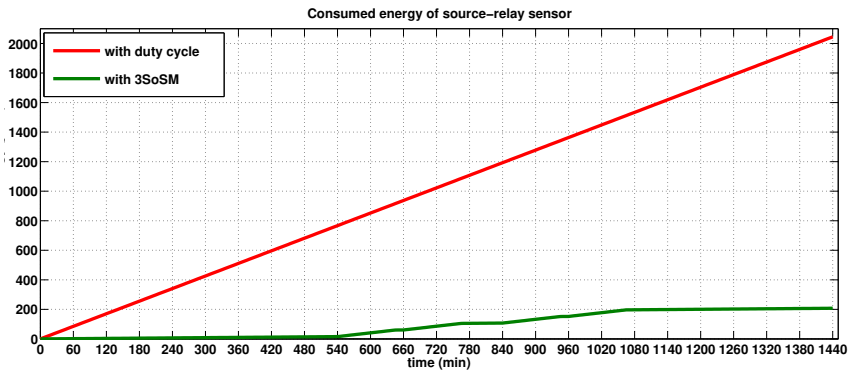
(a) Occupancy of a room (24h).



(b) Consumed energy of a source device (24h).



(c) Consumed energy of a relay device (24h).



(d) Consumed energy of a source-relay device (24h).

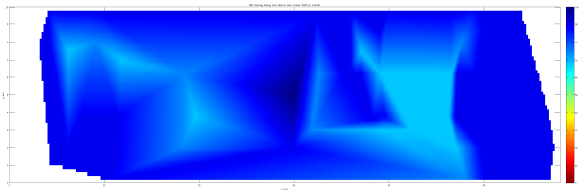
Fig. 6.10 Occupancy information and Energy consumption of different sensor types with duty cycle method and with our approach 3SoSM (Applying Network-Aware 3SoSM - Scenario 2).

6.10c, 6.10d, the upper lines concern the energy consumption of devices that adopt duty cycle method and the lower lines concern the energy consumption of devices that adopt 3SoSM approach. The area between the two lines shows the economy of energy achieved by our 3SoSM approach.

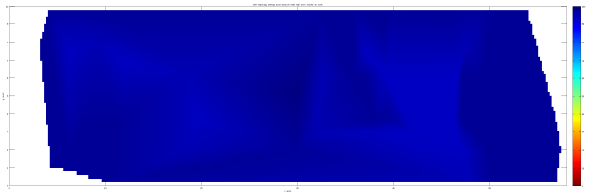
For the comparison between the energy consumptions of different sensor types, we can remark that source sensor consumes less energy than the other type of sensors. Since source sensor is responsible for transmitting only its own data, hence it does not consume energy for receiving and transmitting another sensor's data. The relay sensor devices consume more energy than the source device. These relay devices do not sample physical measures but have a role to forward other sensors' data towards the sink device. Relay sensors receive and transmit their neighbors' data (from lower-layer). The most consuming sensor type is the source-relay sensor devices. These sensors sample physical measures and transmit not only their data but also transmit received data as well. Hence, source-relay sensor devices consume more energy than the rest of the network devices. As a remark, the sink device is not considered in this energy consumption calculation.

Evolution of the energy consumption for the WSN can be observed with heat maps as well. Figure 6.11 represents the heatmap of the network based on the energy level of the sensor devices at different time of the simulation, with duty cycle method and with our Network-Aware 3SoSM approach. According to the given topology (see Figure 5.8 in Chapter 5), Figure 6.11a and Figure 6.11b present the energy levels of the sensor devices at $t = 8h$ with the duty cycle method and our Network-Aware 3SoSM approach. In the same way, Figure 6.11c and Figure 6.11d introduce the energy levels at t=16h. At the end of the day, energy levels of the network with duty cycle method is given in Figure 6.11e. Here we observe the critical zones (red zones) that signifies the sensor devices without energy. These are the zones where source-relay sensor devices are mostly located. On the other hand, Figure 6.11f presents the energy level of the sensor devices with 3SoSM approach at the end of the day. Due to the energy consumption for the duty cycle approach of this scenario, source-relay sensor devices start running out of energy at the end of the day (see Figure 6.11e). However, for the 3SoSM approach, for the same sensor devices, it takes around 8 days to consume their total energy. Figure 6.11g and Figure 6.11h present the heatmap of the network at the end of 8 days. Remark: Since the heatmap coloring algorithm [115] calculates the average remaining energy of 3 nearest sensor devices (for this context) and the sink device has unlimited energy source, in Figure 6.11g, the area where the sink device is located keeps being blue instead of red.

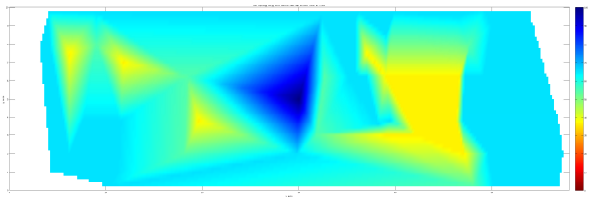
With the settings of this scenario, we observe that our approach 3SoSM enhances the lifetime of the sensor devices significantly. For a source, relay and source-relay, with given application requirements, 3SoSM extends the lifetime of the sensor devices by respectively 7.8 times (source device), 6.8 times (relay device) and 8.9 times (source-relay device).



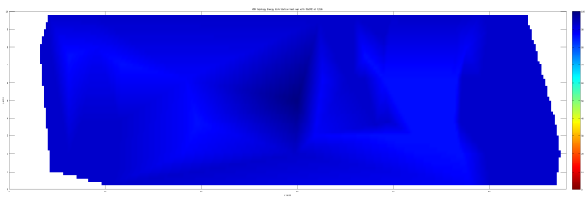
(a) Heat map of remaining energy at t=8h with duty cycle.



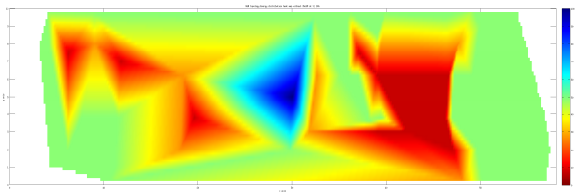
(b) Heatmap of remaining energy at t=8h with 3SoSM.



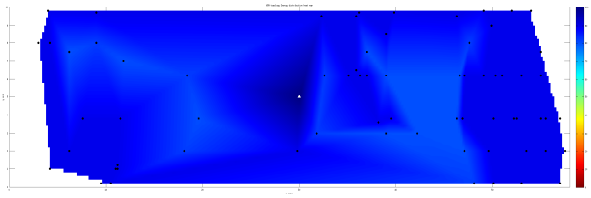
(c) Heatmap of remaining energy at t=16h with duty cycle.



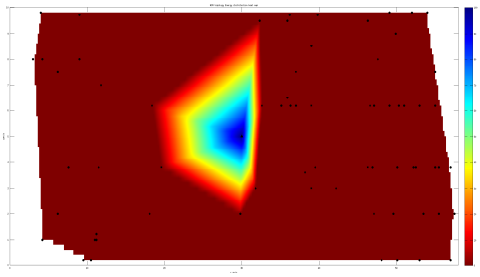
(d) Heatmap of remaining energy at t=16h with 3SoSM.



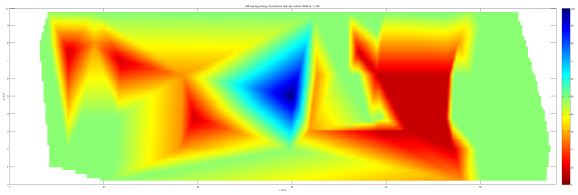
(e) Heatmap of remaining energy at t=24h with duty cycle.



(f) Heatmap of remaining energy at t=24h with 3SoSM.



(g) Heatmap of remaining energy at 8^{th} day with duty cycle.



(h) Heatmap of remaining energy at 8^{th} day with 3SoSM.

Fig. 6.11 Heatmap of remaining energy with duty cycle and with 3SoSM (Applying Network-Aware 3SoSM - Scenario 2).

6.3.3 Scenario 3: CO_2 Emission in a Room

As described in Section 6.1.3, an occupancy information can be detected/calculated by several methods: infra-red sensors, motion detection sensors, door status sensors, CO_2 emission. In the previous scenario, we used the occupancy sensor device to detect the presence in a classroom. Now we collect data from the CO_2 sensor device to monitor occupancy.

In this scenario, there exist two applications:

1. Application 1: App 1 requests temperature service from the system. It asks to monitor the temperature degree of each room of the building with an acquisition periodicity of 30 sec and a latency of 120 sec: (p^{acq}=30, λ=120).

2. Application 2: App 2 requests temperature service as well. It requests to monitor the temperature degree of each room with parameters (p^{acq}=30, λ=120). But it also requests to track more frequently temperature degree of occupied rooms (p^{acq}=15, λ=60). Here the occupancy information comes from the CO_2 emission sensor devices.

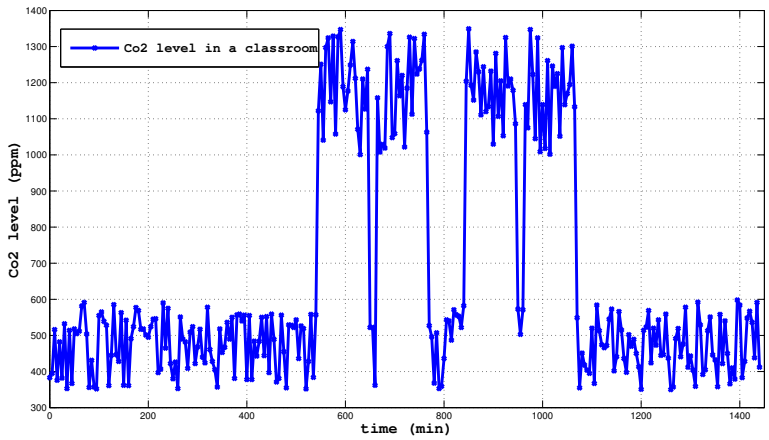


Fig. 6.12 CO_2 emission of a classroom environment during a day (24h)
(Applying Network-Aware 3SoSM - Scenario 3).

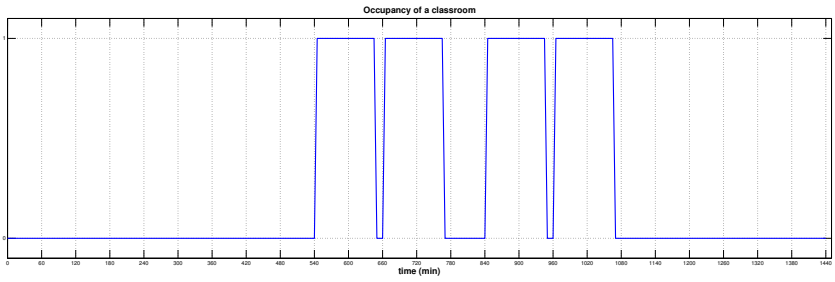


Fig. 6.13 Occupancy of a room (24h)
(Applying Network-Aware 3SoSM - Scenario 3).

Figure 6.12 presents the CO_2 level of a classroom during a day (24h). It is directly related to the occupancy information illustrated in Figure 6.13. Based on these information, the additional request of

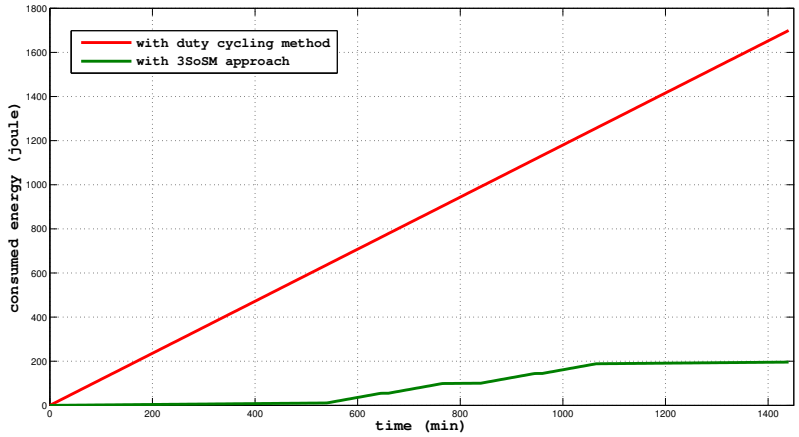


Fig. 6.14 Average energy consumption of source-relay sensor devices of the network (Applying Network-Aware 3SoSM - Scenario 3).

Application 2 is launched when the CO_2 level passes the threshold value (1000 ppm [135]) that indicates that the classroom is occupied.

Figure 6.14 presents the average energy consumption of source-relay sensor devices in the network. Here, we present only the results of the most consuming sensor type. We have presented the results of each sensor type in the network in the previous experiment, and it is explained that the source-relay sensor devices are the most consuming devices in the network.

In this scenario with the given application requirements, the lifetime of the source-relay sensor devices is calculated as 11 days with the static configuration model (p^{acq}=15, p^{tx}=60) adopted by duty cycle method. On the other hand, with the dynamic behavior, lifetime is extended to 95 days.

6.3.4 Scenario 4: Luminosity of a Room

In this scenario, we analyse the luminosity (light level) of a classroom. Importance of the light level in an environment and the minimum light required for different situations are presented in Section 6.1.4.

In this scenario, there exists two applications:

1. Application 1: App 1 requests the light level from the system. It asks to monitor the luminosity of each room of the building with an acquisition periodicity of 30sec and a latency of 100sec: (p^{acq}=30, λ=100).

2. Application 2: App 2 requests the light level as well. It requests to monitor the luminosity of each room with parameters (p^{acq}=20, λ=80).

Here, we observe that the two applications request the same service from the same sensor set (sensors that may measure luminosity of that room) with different acquisition periodicity and latency. Figure 6.15 presents the luminosity of a classroom in terms of lux. With the given application requirements, average energy consumption of source-relay sensor devices is illustrated in Figure 6.16.

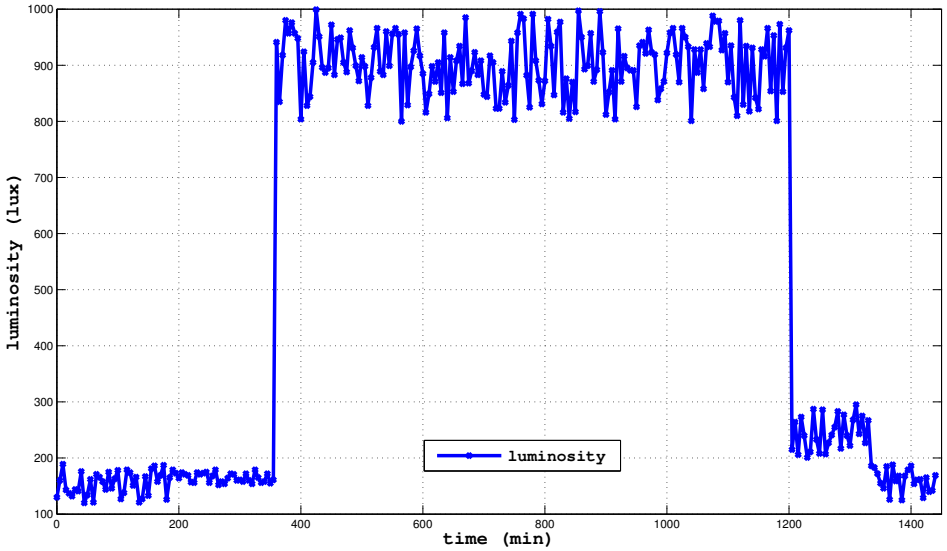


Fig. 6.15 Luminosity of a classroom during a day (24h)
(Applying Network-Aware 3SoSM - Scenario 4).

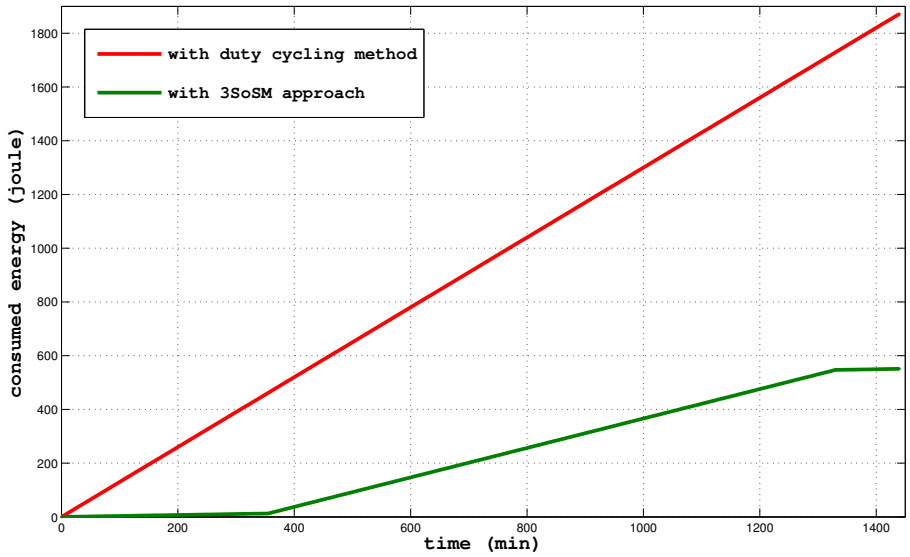


Fig. 6.16 Average energy consumption of source-relay sensor devices of the network
(Applying Network-Aware 3SoSM - Scenario 4).

In terms of lifetime, with the static configuration (p^{acq}=20, p^{tx}=80) adopted by duty cycle method, lifetime of the system is calculated as 10 days. On the other hand, with our approach, lifetime of the system is extended to 34 days.

6.4 Conclusion

The performed experiments and their results show that by applying our approach to given application requirements, we optimize the energy consumption and reduce the unnecessary communication cost. Obtained results on energy saving and lifetime extension depend on the application requirements and the context of the application (physical measures). With the current settings, the most unfavorable case is the situation where every room is occupied during the entire simulation time (for the scenario 2 for instance). Even in that case, `Network-Aware 3SoSM` achieves a concrete energy enhancement since the regular duty cycle approach does not benefit from latency requirement which provides a tangible energy saving by allowing grouped transmission of multiple acquired data.

Besides, the details of each performed scenarios with `Network-Aware 3SoSM` are summarized in Table 6.4: application requirements are given and the average lifetime of source-relay sensor devices are presented for each case, with and without our approach. Moreover a brief summary about the lifetime extension for each scenario is illustrated in Figure 6.17.

With more populated topologies and with massive subscription requests, a discussion about the optimization process appears: in which conditions our approach may fail to find the optimal slots on the schedules (since a slot is reserved for a single device)? In these cases, granularity should be decreased to have enough slots for the sensor devices.

Table 6.4 Summary of the Performed Experiments.

Scenarios	**Application Parameters**						**Lifetime (day)**	
	Duty-cycle		**Application 1**		**Application 2**		**Duty cycle**	**Network-Aware 3SoSM**
	acquisition periodicity (sec)	**transmission periodicity (sec)**	**acquisition periodicity (sec)**	**latency (sec)**	**acquisition periodicity (sec)**	**latency (sec)**		
#1 Comfort Temperature Range	10	20	20	60	10	20	9	90
#2 Temperature of Occupied Rooms	1	4	15	60	1	4	15	146
#3 Occupancy based on CO_2	15	60	30	120	15	60	11	95
#4 Luminosity of Rooms	20	80	30	100	20	80	10	34

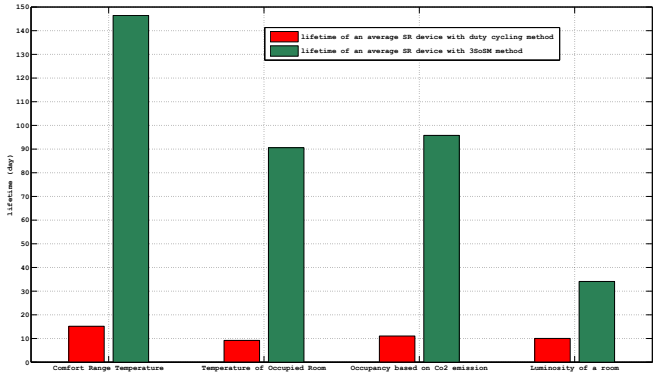


Fig. 6.17 Average lifetime extensions of Source-Relay devices for each scenario.

Chapter 7

Conclusion and Discussion

7.1 Conclusion and Contributions

This thesis presents a novel approach for the energy enhancement of multi-application monitoring systems for smart buildings. We focus on energy efficiency in wireless sensor networks, managing sensor data streams and optimizing the real-time dynamic interactions between application continuous queries and the physical environment. Our work highlights the existing studies in this field, clarifies the gap and identifies new directions in how to manage multi-application context. This dissertation has investigated and implemented a dynamic energy-aware sensor configuration in multi-application monitoring systems for pervasive environments. As a target, we choose smart building applications that cover recent technologies like Internet of Things (IoT), pervasive environments and ubiquitous computing.

In this thesis, we work on a pervasive environment equipped with hundreds of wireless sensor devices. A pervasive environment such as smart buildings requires a multidisciplinary research. It covers not only wireless sensor issues but also, sensor network processing tools, building management platforms and technologies. If one facet of the smart building monitoring system is the wireless sensor devices, the other facet is the management of the system and the data streams generated by the deployed equipment in the environment.

Here, we deal with one of the major challenges of smart building technology: how to optimize/reduce the energy consumption of monitoring architecture itself while managing sensor data streams to increase the lifetime of the system. A detailed literature review presents that existing studies do not tackle the energy consumption of the monitoring system itself but they propose approaches on the energy consumption of the building. Even the most closest studies, they also commonly adopt static configurations for sensor devices. Since application requirements are dynamic on the context, a dynamic sensor configuration is absolutely required. Besides these approaches are mostly `WSN`-level techniques that do not tackle real-time dynamic interactions between application continuous queries and the physical environment. However, in a such multi-application context, wireless sensor devices are multi-modal and application requirements are dynamic. Hence, we show up a gap between the computing environment and the physical environment, that can be both managed by pervasive applications.

To address this gap and propose a clear solution for smart building management systems, we design a sustainable multi-application monitoring system architecture for pervasive environments that collects application requirements for sensor data streams and optimizes them into sensor configurations. We propose energy-aware dynamic sensor device re-configuration (as a result of an optimization process) to lower energy consumption while fulfilling real-time application requirements.

We also present a sustainable declarative monitoring architecture. We adopt `Pervasive Environment Management Systems (PEMS)` principles to separate application development and optimization of device interaction. As a solution, we present our approach `Smart-Service Stream-oriented Sensor`

Management (3SoSM) that is based on a network schedule mechanism. It is a global schedule formed by the sensor actions. We present a bottom-up process for the constraint propagation and propose an optimization algorithm based on a cost model to find the optimal communication time slots. As an outcome, optimized schedule forms sensor configuration-oriented pattern (SCO-PATTERN) for each device.

We introduce our implementation 3SoSM Gateway that supports the optimization process for multiple parameterized subscriptions to the same device to fulfill dynamic application requirements. Our approach is validated by the experiments using the SoCQ Engine and a modified WSNet simulator. Moreover, our approach gives opportunities to use real testbed data and simulation data which is not so common in the pervasive environment research domain. Impacts of our approach on energy consumption and on lifetime are presented and discussed. Smart building applications are our target application, however, this approach can be applicable to other pervasive environments as well.

To position our approach in the literature among with the most known existing studies on smart building applications, we recall Table 7.1. Now, we update the table and the last line of the table is allocated for our 3SoSM approach. Compared to existing studies, our 3SoSM approach provides a multi-application mechanism and allows dynamicity for user configurations and network-aware sensor management while optimizing the sensor communication for enhancing energy consumption by real-time sensor configuration.

7.2 Discussion and Perspectives

From our point of view, there are some points need to be discussed. Firstly the chosen network has a highly important role in this work. Since our approach tries to create a valid Schedule Time Pattern for each sensor device in the pervasive environment, the routing, the paths for each device to the sink device are defined at the beginning of the execution and are rebuilt when a new subscription or unsubscription occurs. In other words, the network topology illustrated in Figure 5.8 (See Section 5.3) presents all the existing sensor devices in the environment. However, every sensor device in that network do not have to engage for each application given by the user. Each sensor device has its own feature and ability to measure physical quantities: temperature, humidity, CO_2, luminosity, occupancy, light blind control and door/window signal control. For instance an application that requires only the temperature measures will trigger only the sensors that are capable of measuring temperature (trigger sensor device that can provide temperature service). Based on the sensor device locations, other sensor devices can only be a relay device if needed. Thus, during the execution of such application a new topology will be used.

Suppose that there is a new application that requests luminosity measures from the environment. Hence, the 3SoSM re-executes and creates a new Schedule Time Pattern for the sensor devices. The major change will occur on the topology side. Due to the requirements of the second application, sensor devices that can provide luminosity service will involve into the network and a new topology is created. The source devices may become a source-relay, relay devices may become a source-relay or a relay device can go into the sleep mode since a new active path is found with the new application requirement. Here, we observe a dynamic user preferences or a change in the context environment) and a dynamic reconfigurability of the sensor devices that gives us opportunity to support multi-application monitoring systems. Besides, there is also the dynamicity of the network, regeneration of the topology based on the context. Since our target application is the smart buildings, smart buildings have a limited area to manage and limited number of devices to deploy (generally base station in the middle of the area). Moreover, the number of deployed sensor devices can not be huge and network problem can hardly be occurred. With the experiments and case studies presented in Section 6, we perform different cases with different applications (with different application requirements and measures). At the background of these cases, topology of the network varies due to the physical measures.

Table 7.1 Classification of Well-Known Smart Building Environment Systems through Pervasive Environment Principles.

Study	Multi-Application	Context-Awareness	Energy-Awareness	Dynamic User Config.	Real-time Sensor Config	SNQP	Experimentation
Sharaf et al. [162]	–	√	√	–	–	TINA	CSIM simulator
Yao et al. [192]	–	–	√	–	–	COUGAR	Simu
Madden et al. [109]	–	–	√	–	–	TinyDB	Simu
Rutishauser et al. [154]	–	√	–	–	–	–	Real Testbed
Deshpande et al. [42]	–	–	–	–	–	MauveDB	Simu on real data
Baralis et al. [19]	–	–	√	–	–	SERENE	Simu
Tulone et al. [175]	–	–	√	–	–	SEFA	Simu
Tulone et al. [176]	–	√	√	–	–	PAQ	Simu on real data
Doukas et al. [44]	–	√	–	–	–	–	Real Testbed
Thiagarajan et al. [174]	–	–	–	–	–	FunctionDB	C++ prototype
Brayner et al. [27]	–	√	√	–	√	ADAGA	Simu
Li et al. [99]	–	–	√	–	–	PRESTO	EmStar emulator
Chen et al. [38]	–	√	–	–	–	–	Simulation
Schor et al. [157]	–	√	√	–	–	–	Real Testbed
Gripay et al. [68]	√	√	–	√	–	SoCQ	Real testbed
Agarwal et al. [7]	–	√	–	–	–	–	Real Testbed
Galpin et al. [56]	–	√	√	–	–	SNEE	Real testbed
Schreiber et al. [158]	–	√	√	–	–	PERLA	Simu
Mamidi et al. [112]	–	√	–	–	–	–	Real Testbed
Byun et al. [30]	–	√	√	–	√	–	Real Testbed
Li et al. [98]	–	√	√	–	–	–	Real Testbed
Preisel et al. [143]	–	–	√	–	–	–	Real Testbed
Lim et al. [101]	–	√	√	–	–	ACQUA	Perl-based simulator
3SoSM	√	√	√	√	√	SoCQ	Real Testbed and Simulation

Besides, there exist few network issues that may cause problem for our approach. Since we define `Schedule Time Pattern` for each sensor in the environment, the sink device distributes all these patterns to the sensor devices. Thus, the question that should be asked is the time synchronization of these sensor devices. Each `Schedule Time Pattern` indicates the sensor actions with the timestamp values, however during the propagation of these patterns and during the execution, a delay may cause a problem in the long term. We accept that any problem about the time synchronization about the sensor devices may cause an unwilling reaction. It is well-known that the wireless sensors in an environment operate independently, their local clocks may not be synchronized with one another. This can cause difficulties when trying to integrate and interpret information sensed at different devices. Therefore, clock synchronization protocols different from the conventional protocols are extremely needed and currently in the `WSN` research field, there is a huge interest towards developing energy efficient clock synchronization protocols to provide a common notion of time. Hence, in our case we think that time synchronization is itself a very deep research area and in our thesis we unwillingly neglect the time synchronization problem.

This thesis put forward our approach to optimize interactions between application requirements and the wireless sensor environment in real-time. The key point of the thesis is the dynamicity of the application requirements/context and the reconfiguration of the sensor devices. In the wireless sensor device market, cheap sensor devices do not let applications to configure themselves during the execution. The only solution for them is the resetting the device which requires human action. However, more expensive sensor devices allow itself to configure themselves according to an out-coming command like our `Schedule Time Patterns` in real-time and then they start executing their new actions. Such sensor devices are our core material for the approach and due to this ability, we are able to manage multi-modal sensor devices and dynamic requirements.

References

[1] Abbasi, A. A. and Younis, M. (2007). A survey on clustering algorithms for wireless sensor networks. *Computer Communications*, 30(14):2826–2841.

[2] Abbate, S., Avvenuti, M., Corsini, P., Light, J., and Vecchio, A. (2010). *Monitoring of human movements for fall detection and activities recognition in elderly care using wwireless sensor network: A survey*. InTech.

[3] Abdallah, A. E., Fevens, T., and Opatrny, J. (2013). 3d local algorithm for dominating sets of unit disk graphs. *Ad Hoc & Sensor Wireless Networks*, 19(1-2):21–41.

[4] Abowd, G., Dey, A., Brown, P., Davies, N., Smith, M., and Steggles, P. (1999). Towards a better understanding of context and context-awareness. In *Handheld and ubiquitous computing*, pages 304–307. Springer.

[5] Abughalieh, N., Le Borgne, Y.-A., Steenhaut, K., and Nowé, A. (2010). Lifetime optimization for wireless sensor networks with correlated data gathering. In *Proceedings of the 8th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt'10)*, pages 266–272. IEEE.

[6] Agarwal, L., Dixit, G., Jain, A., Pandey, K., and Khare, A. (2016). Energy efficient pollution monitoring system using deterministic wireless sensor networks. In *Proceedings of the International Congress on Information and Communication Technology*, pages 301–309. Springer.

[7] Agarwal, Y., Balaji, B., Gupta, R., Lyles, J., Wei, M., and Weng, T. (2010). Occupancy-driven energy management for smart building automation. *Journal of Power*, 50(100):150.

[8] Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (2002). Wireless sensor networks: a survey. *Computer Networks*, 38(4):393–422.

[9] Alamri, A., Ansari, W. S., Hassan, M. M., Hossain, M. S., Alelaiwi, A., and Hossain, M. A. (2013). A survey on sensor-cloud: architecture, applications, and approaches. *International Journal of Distributed Sensor Networks*, 9(2):917923.

[10] Allen, K., Connelly, K., Rutherford, P., and Wu, Y. (2017). Smart windows-dynamic control of building energy performance. *Energy and Buildings*.

[11] Alvi, A. N., Naqvi, S. S., Bouk, S. H., Javaid, N., Qasim, U., and Khan, Z. A. (2012). Evaluation of slotted csma/ca of ieee 802.15. 4. In *Proceedings of 7th International Conference on Broadband, Wireless Computing, Communication and Applications (BWCCA'12)*, pages 391–396. IEEE.

[12] Anastasi, G., Conti, M., Di Francesco, M., and Passarella, A. (2009). Energy conservation in wireless sensor networks: A survey. *Ad hoc networks*, 7(3):537–568.

[13] Ansanay-Alex, G. (2013). Estimating occupancy using indoor carbon dioxide concentrations only in an office building: a method and qualitative assessment. In *Proceedings of REHVA World Congres Energy Efficient, Smart and Healthy Build(CLIMA'13)*, pages 1–8.

[14] Babu, S. (2009). *Continuous Query*, pages 492–493. Springer US, Boston, MA.

[15] Babu, S. and Widom, J. (2001). Continuous queries over data streams. *ACM Sigmod Record*, 30(3):109–120.

[16] Bachir, A., Dohler, M., Watteyne, T., and Leung, K. K. (2010). Mac essentials for wireless sensor networks. *IEEE Communications Surveys & Tutorials*, 12(2):222–248.

[17] Bahl, N., Sharma, A. K., and Verma, H. K. (2014). On the energy utilization for wsn based on bpsk over the generalized-k shadowed fading channel. *Wireless Networks*, 20(8):2385–2393.

[18] Ball, M. G., Qela, B., and Wesolkowski, S. (2016). A review of the use of computational intelligence in the design of military surveillance networks. In *Recent Advances in Computational Intelligence in Defense and Security*, pages 663–693. Springer.

[19] Baralis, E. and Cerquitelli, T. (2006). Selecting representatives in a sensor network. In *Proceedings of the 14th Italian National Conference on Advanced Data Base Systems (SEBD'06)*, pages 351–360.

[20] Bekmezci, I. (2009). *Wireless sensor networks: A military monitoring application*. VDM Verlag.

[21] Bellalta, B., Faridi, A., Staehle, D., Barcelo, J., Vinel, A., and Oliver, M. (2013). Performance analysis of csma/ca protocols with multi-packet transmission. *Computer Networks*, 57(14):2675–2688.

[22] Bonnet, P., Gehrke, J., and Seshadri, P. (2001). Towards sensor database systems. In *Proceedings of 2nd International Conference on Mobile Data Management (MDM'01)*, pages 3–14. Springer.

[23] Bose, P., Morin, P., Stojmenović, I., and Urrutia, J. (2001). Routing with guaranteed delivery in ad hoc wireless networks. *Wireless Networks*, 7(6):609–616.

[24] Boughanmi, N. and Song, Y. (2008). A new routing metric for satisfying both energy and delay constraints in wireless sensor networks. *Journal of Signal Processing Systems*, 51(2):137–143.

[25] Boukerche, A. (2008). *Algorithms and protocols for wireless sensor networks*, volume 62. John Wiley & Sons.

[26] Brayner, A., Coelho, A. L., Marinho, K., Holanda, R., and Castro, W. (2014). On query processing in wireless sensor networks using classes of quality of queries. *Information Fusion*, 15:44–55.

[27] Brayner, A., Lopes, A., Meira, D., Vasconcelos, R., and Menezes, R. (2008). An adaptive in-network aggregation operator for query processing in wireless sensor networks. *Journal of Systems and Software*, 81(3):328–342.

[28] Brenninkmeijer, C. Y., Galpin, I., Fernandes, A. A., and Paton, N. W. (2008). A semantics for a query language over sensors, streams and relations. In *Proceedings of 25th British National Conference on Databases (BNCOD'08)*, pages 87–99. Springer.

[29] Bressan, N., Bazzaco, L., Bui, N., Casari, P., Vangelista, L., and Zorzi, M. (2010). The deployment of a smart monitoring system using wireless sensor and actuator networks. In *Proceedings of 1st IEEE International Conference on Smart Grid Communications (SmartGridComm'10)*, pages 49–54. IEEE.

[30] Byun, J., Jeon, B., Noh, J., Kim, Y., and Park, S. (2012). An intelligent self-adjusting sensor for smart home services based on zigbee communications. *IEEE Transactions on Consumer Electronics*, 58(3):794–802.

[31] Byun, J. and Park, S. (2011). Development of a self-adapting intelligent system for building energy saving and context-aware smart services. *IEEE Transactions on Consumer Electronics*, 57(1):90–98.

[32] Calvillo, C. F., Sánchez-Miralles, A., and Villar, J. (2016). Energy management and planning in smart cities. *Renewable and Sustainable Energy Reviews*, 55:273–287.

[33] Candanedo, L. M. and Feldheim, V. (2016). Accurate occupancy detection of an office room from light, temperature, humidity and co 2 measurements using statistical learning models. *Energy and Buildings*, 112:28–39.

[34] Cao, L., Zheng, G., and Shen, Y. (2016). Research on design of military ammunition container monitoring system based on iot. In *Prognostics and System Health Management Conference (PHM-Chengdu), 2016*, pages 1–4. IEEE.

[35] Castillejo, P., Martinez, J.-F., Rodriguez-Molina, J., and Cuerva, A. (2013). Integration of wearable devices in a wireless sensor network for an e-health application. *IEEE Wireless Communications*, 20(4):38–49.

[36] Chakraborty, A., Chakraborty, K., Mitra, S. K., and Naskar, M. K. (2009). An energy efficient scheme for data gathering in wireless sensor networks using particle swarm optimization. *Journal of Applied Computer Science*, 6(3):9–13.

[37] Chasta, R., Singh, R., Gehlot, A., Mishra, R. G., and Choudhury, S. (2016). A smart building automation system. *International Journal of Smart Home*, 10(8):91–98.

[38] Chen, H., Chou, P., Duri, S., Lei, H., and Reason, J. (2009). The design and implementation of a smart building control system. In *IEEE International Conference one-Business Engineering (ICEBE'09)*, pages 255–262. IEEE.

[39] Cheng, M.-Y., Chiu, K.-C., Lien, L.-C., Wu, Y.-W., and Lin, J.-J. (2016). Economic and energy consumption analysis of smart building–mega house. *Building and Environment*, 100:215–226.

[40] Cormen, T. H. (2009). *Introduction to algorithms*. MIT press.

[41] Croce, S., Marcelloni, F., and Vecchio, M. (2008). Reducing power consumption in wireless sensor networks using a novel approach to data aggregation. *The Computer Journal*, 51(2):227–239.

[42] Deshpande, A. and Madden, S. (2006). Mauvedb: supporting model-based user views in database systems. In *Proceedings of International Conference on Management of Data (SIGMOD'06)*, pages 73–84. ACM.

[43] Dougan, D. S. and Damiano, L. (2004). Co2-based demand control ventilation: Do risks outweigh potential rewards? *ASHRAE journal*, 46(10):47.

[44] Doukas, H., Patlitzianas, K. D., Iatropoulos, K., and Psarras, J. (2007). Intelligent building energy management system using rule sets. *Journal of Building and Environment*, 42(10):3562–3569.

[45] EIA (2011). The world business council for sustainable development. *Annual Energy Review 2010*.

[46] EIA (2015). The world business council for sustainable development. *Annual Energy Review 2015*.

[47] EnOcean (2008). Manual, enocean tmc120 user, scavenger transceiver module-stm 300 and stm 300c datasheet. Technical report. Accessed: 2017-02-02.

[48] Fang, X., Misra, S., Xue, G., and Yang, D. (2012). Smart grid-the new and improved power grid: A survey. *IEEE Communications Surveys & Tutorials*, 14(4):944–980.

[49] Farias, C., Soares, H., Pirmez, L., Delicato, F., Santos, I., Carmo, L. F., Souza, J., Zomaya, A., and Dohler, M. (2014). A control and decision system for smart buildings using wireless sensor and actuator networks. *Transactions on Emerging Telecommunications Technologies*, 25(1):120–135.

[50] Farooq, M. O. and Kunz, T. (2011). Operating systems for wireless sensor networks: A survey. *Sensors*, 11(6):5900–5930.

[51] Fathallah, K., Frihida, A., and Hadj-Alouane, N. B. (2013). Wireless sensor network queries processor a survey. In *International Conference on Information Processing and Wireless Systems (IPWIS'13)*, pages 21–24.

[52] Fortino, G., Guerrieri, A., O'Hare, G. M., and Ruzzelli, A. (2012). A flexible building management framework based on wireless sensor and actuator networks. *Journal of Network and Computer Applications*, 35(6):1934–1952.

[53] Fraboulet, A., Chelius, G., and Fleury, E. (2007). Worldsens: development and prototyping tools for application specific wireless sensors networks. In *International Conference on Information Processing in Sensor Networks (IPSN'07)*, pages 176–185. IEEE.

[54] Friess, P. (2013). *Internet of things: converging technologies for smart environments and integrated ecosystems*. River Publishers.

[55] Frisch, A. M., Jefferson, C., Hernández, B. M., and Miguel, I. (2005). The rules of constraint modelling. In *IJCAI*, pages 109–116.

[56] Galpin, I., Brenninkmeijer, C. Y., Gray, A. J., Jabeen, F., Fernandes, A. A., and Paton, N. W. (2011). Snee: a query processor for wireless sensor networks. *Distributed and Parallel Databases*, 29(1-2):31–85.

[57] Galpin, I., Brenninkmeijer, C. Y., Jabeen, F., Fernandes, A. A., and Paton, N. W. (2009). Comprehensive optimization of declarative sensor network queries. In *International Conference on Scientific and Statistical Database Management (SSDBM'09)*, pages 339–360. Springer.

[58] Galpin, I., Fernandes, A. A., and Paton, N. W. (2013). Qos-aware optimization of sensor network queries. *The VLDB Journal*, 22(4):495–517.

[59] García-Hernández, C. F., Ibarguengoytia-Gonzalez, P. H., García-Hernández, J., and Pérez-Díaz, J. A. (2007). Wireless sensor networks and applications: a survey. *International Journal of Computer Science and Network Security (IJCSNS'07)*, 7(3):264–273.

[60] Gay, D., Levis, P., Von Behren, R., Welsh, M., Brewer, E., and Culler, D. (2003). The nesc language: A holistic approach to networked embedded systems. In *Acm Sigplan Notices*, volume 38, pages 1–11. ACM.

[61] Gehrke, J. and Madden, S. (2004). Query processing in sensor networks. *IEEE Pervasive computing*, 3(1):46–55.

[62] Gibson, D. and MacGregor, C. (2013). A novel solid state non-dispersive infrared co2 gas sensor compatible with wireless and portable deployment. *Sensors*, 13(6):7079–7103.

[63] Gómez-Candón, D., De Castro, A., and López-Granados, F. (2014). Assessing the accuracy of mosaics from unmanned aerial vehicle (uav) imagery for precision agriculture purposes in wheat. *Precision Agriculture*, 15(1):44–56.

[64] Gou, H. and Yoo, Y. (2010). Distributed bottleneck node detection in wireless sensor network. In *IEEE 10th International Conference on Computer and Information Technology (CIT'10)*, pages 218–224. IEEE.

[65] Gripay, Y. (2009a). Socq project: A framework for pervasive environments. Technical report. Accessed: 2017-02-02.

[66] Gripay, Y. (2009b). Socq4home project. Technical report. Accessed: 2017-02-02.

[67] Gripay, Y., Laforest, F., and Petit, J.-M. (2009a). Managing pervasive environments through database principles: A survey. In *Advances in Data Management*, pages 277–298. Springer.

[68] Gripay, Y., Laforest, F., and Petit, J.-M. (2009b). Socq: A framework for pervasive environments. In *International Symposium on Pervasive Systems, Algorithms, and Networks (ISPAN'09)*, pages 154–159. IEEE.

[69] Gripay, Y., Laforest, F., and Petit, J.-M. (2010). A simple (yet powerful) algebra for pervasive environments. In *Proceedings of the 13th International Conference on Extending Database Technology (EDBT'10)*, pages 359–370. ACM.

[70] Gubbi, J., Buyya, R., Marusic, S., and Palaniswami, M. (2013). Internet of things (iot): A vision, architectural elements, and future directions. *Future generation computer systems*, 29(7):1645–1660.

[71] Gungor, V. C., Lu, B., and Hancke, G. P. (2010). Opportunities and challenges of wireless sensor networks in smart grid. *IEEE transactions on industrial electronics*, 57(10):3557–3564.

[72] Han, H., Jo, J., Son, Y., and Park, J. (2015). Smart sleep care system for quality sleep. In *Information and Communication Technology Convergence (ICTC'15)*, pages 393–398. IEEE.

[73] Hansmann, U., Merk, L., Nicklous, M. S., and Stober, T. (2013). *Pervasive computing handbook*. Springer Science & Business Media.

[74] Hatler, M., Gurganious, D., and Chi, C. (2012). Smart home energy systems & cloud services, a market dynamics report. Technical report.

[75] Hong, S.-H. and Kim, B.-K. (2010). An efficient data gathering routing protocol in sensor networks using the integrated gateway node. *IEEE Transactions on Consumer Electronics*, 56(2).

[76] Hu, S. and Han, J. (2014). Power control strategy for clustering wireless sensor networks based on multi-packet reception. *IET Wireless Sensor Systems*, 4(3):122–129.

[77] Huang, Q. and Mao, C. (2017). Occupancy estimation in smart building using hybrid co2/light wireless sensor network. *Journal of Applied Sciences and Arts*, 1(2):5.

[78] Imran, M., Said, A. M., and Hasbullah, H. (2010). A survey of simulators, emulators and testbeds for wireless sensor networks. In *International Symposium in Information Technology (ITSim'10)*, volume 2, pages 897–902. IEEE.

[79] Jabeen, F. and Nawaz, S. (2015). In-network wireless sensor network query processors: State of the art, challenges and future directions. *Information Fusion*, 25:1–15.

[80] Jagadish, H., Gehrke, J., Labrinidis, A., Papakonstantinou, Y., Patel, J. M., Ramakrishnan, R., and Shahabi, C. (2014). Big data and its technical challenges. *Communications of the ACM*, 57(7):86–94.

[81] Jang, S. H., Nam, K. W., and Jung, Y. G. (2016). Smart building block toys using internet of things technology. *International Journal of Advanced Culture Technology*, 4(2):34–37.

[82] Jayanth, N. (2011). Marché européen des systèmes immotiques. Technical report.

[83] Jemai, A., Mastouri, A., and Eleuch, H. (2011). Study of key pre-distribution schemes in wireless sensor networks: case of brosk (use of wsnet). *Applied Mathematics and Information Sciences*, 5:655–667.

[84] Kadri, A., Yaacoub, E., Mushtaha, M., and Abu-Dayya, A. (2013). Wireless sensor network for real-time air pollution monitoring. In *Communications, Signal Processing, and their Applications (ICCSPA), 2013 1st International Conference on*, pages 1–5. IEEE.

[85] Kalpakis, K., Dasgupta, K., and Namjoshi, P. (2003). Efficient algorithms for maximum lifetime data gathering and aggregation in wireless sensor networks. *Computer Networks*, 42(6):697–716.

[86] Karl, H. and Willig, A. (2007). *Protocols and architectures for wireless sensor networks*. John Wiley and Sons.

[87] Kho, J., Rogers, A., and Jennings, N. R. (2009). Decentralized control of adaptive sampling in wireless sensor networks. *ACM Transactions on Sensor Networks*, 5(3):19.

[88] Kim, K., Lee, S., Yoo, H., and Kim, D. (2014). Agriculture sensor-cloud infrastructure and routing protocol in the physical sensor network layer. *International Journal of Distributed Sensor Networks*, 10(3):437535.

[89] Klan, D., Karnstedt, M., Hose, K., Ribe-Baumann, L., and Sattler, K.-U. (2011). Stream engines meet wireless sensor networks: cost-based planning and processing of complex queries in anduin. *Distributed and Parallel Databases*, 29(1):151–183.

[90] Labeodan, T., Zeiler, W., Boxem, G., and Zhao, Y. (2015). Occupancy measurement in commercial office buildings for demand-driven control applications-a survey and detection system evaluation. *Energy and Buildings*, 93:303–314.

[91] Lampin, Q., Barthel, D., Auge-Blum, I., and Valois, F. (2012). Sari-mac: The self adapting receiver initiated mac protocol for wireless sensor networks. In *IEEE 8th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob'12)*, pages 12–18. IEEE.

[92] Le, H. K., Henriksson, D., and Abdelzaher, T. (2007). A control theory approach to throughput optimization in multi-channel collection sensor networks. In *Proceedings of the 6th International Conference on Information Processing in Sensor Networks (IPSN'07)*, pages 31–40. ACM.

[93] Le, M.-H. and Ploix, S. (2016). Robust anticipative energy management system: Application of a smart building platform. *Building Services Engineering Research and Technology*, page 0143624416669832.

[94] Lee, K.-H., Kim, J.-H., and Cho, S. (2016). Power saving mechanism with network coding in the bottleneck zone of multimedia sensor networks. *Computer Networks*, 96:58–68.

[95] Lege, R. P., Hasegawa, S., Ishio, H., Takahashi, T., Hyodo, K., Matsunami, S., Ishii, Y., Iwata, K., Kojima, T., and Miyao, M. (2017). Measuring the effects of lighting on the readability of electronic devices. *Journal of the Society for Information Display*.

[96] Levis, P., Madden, S., Polastre, J., Szewczyk, R., Whitehouse, K., Woo, A., Gay, D., Hill, J., Welsh, M., Brewer, E., et al. (2005). Tinyos: An operating system for sensor networks. In *Ambient intelligence*, pages 115–148. Springer.

[97] Lévy-Bencheton, C., Ammar, D., Villemaud, G., and Risset, T. (2011). Multi-mode relay simulations: an energy evaluation on wsnet. In *IEEE Radio and Wireless Symposium (RWS'11)*, pages 42–45. IEEE.

[98] Li, C., Meggers, F., Li, M., Sundaravaradan, J., Xue, F., Lim, H., and Schlueter, A. (2013). Bubblesense: wireless sensor network based intelligent building monitoring. *Proceedings of the International Conferences for Sustainability (ICT4S'13)*, pages 159–166.

[99] Li, M., Ganesan, D., and Shenoy, P. (2009). Presto: feedback-driven data management in sensor networks. *IEEE/ACM Transactions on Networking, TON*, 17(4):1256–1269.

[100] Li, Q., Ding, R., Liu, Y., Baehr-Jones, T., Hochberg, M., and Bergman, K. (2015). High-speed bpsk modulation in silicon. *IEEE Photonics Technology Letters*, 27(12):1329–1332.

[101] Lim, L., Misra, A., and Mo, T. (2013). Adaptive data acquisition strategies for energy-efficient, smartphone-based, continuous processing of sensor streams. *Distributed and Parallel Databases*, 31(2):321–351.

[102] Lindblom, J., Lundström, C., Ljung, M., and Jonsson, A. (2016). Promoting sustainable intensification in precision agriculture: review of decision support systems development and strategies. *Precision Agriculture*, pages 1–23.

[103] Lu, B., Habetler, T. G., Harley, R. G., and Gutiérrez, J. A. (2005). Applying wireless sensor networks in industrial plant energy management systems. part ii. design of sensor devices. In *IEEE Sensors*, pages 6–pp. IEEE.

[104] Lu, B., Habetler, T. G., Harley, R. G., Gutierrez, J. A., and Durocher, D. B. (2007). Energy evaluation goes wireless. *IEEE Industry Applications Magazine*, 13(2):17–23.

[105] Lu, C., Saifullah, A., Li, B., Sha, M., Gonzalez, H., Gunatilaka, D., Wu, C., Nie, L., and Chen, Y. (2016). Real-time wireless sensor-actuator networks for industrial cyber-physical systems. *Proceedings of the IEEE*, 104(5):1013–1024.

[106] Lu, Z., Li, W. W., and Pan, M. (2015). Maximum lifetime scheduling for target coverage and data collection in wireless sensor networks. *IEEE Transactions on vehicular technology*, 64(2):714–727.

[107] Luo, D., Zhu, X., Wu, X., and Chen, G. (2011). Maximizing lifetime for the shortest path aggregation tree in wireless sensor networks. In *Proceedings IEEE Computer and Communications Societies IEEE Annual Joint Conference (INFOCOM'11)*, pages 1566–1574. IEEE.

[108] Madan, R. and Lall, S. (2006). Distributed algorithms for maximum lifetime routing in wireless sensor networks. *IEEE Transactions on Wireless Communications,*, 5(8):2185–2193.

[109] Madden, S. R., Franklin, M. J., Hellerstein, J. M., and Hong, W. (2005). Tinydb: an acquisitional query processing system for sensor networks. *ACM Transactions on database systems*, 30(1):122–173.

[110] Mainetti, L., Patrono, L., and Vilei, A. (2011). Evolution of wireless sensor networks towards the internet of things: A survey. In *19th International Conference on Software, Telecommunications and Computer Networks (SoftCOM'11)*, pages 1–6. IEEE.

[111] Mamaghanian, H., Khaled, N., Atienza, D., and Vandergheynst, P. (2011). Compressed sensing for real-time energy-efficient ecg compression on wireless body sensor nodes. *IEEE Transactions on Biomedical Engineering*, 58(9):2456–2466.

[112] Mamidi, S., Chang, Y.-H., and Maheswaran, R. (2012). Improving building energy efficiency with a network of sensing, learning and prediction agents. In *Proceedings of The 11th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'12)*, volume 1, pages 45–52. The International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS).

[113] Marnane, W., Faul, S., Bleakley, C., Conway, R., Jones, E., Popovici, E., de la Guia Solaz, M., Morgan, F., and Patel, K. (2010). Energy efficient on-sensor processing in body sensor networks. In *The 32th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC'10)*, pages 2025–2029. IEEE.

[114] Marriott, K. and Stuckey, P. J. (2016). A minizinc tutorial. Technical report.

[115] MathWorks (2011). Create heatmap chart. Technical report. Accessed: 2017-10-10.

[116] Micko, E. S. (2010). Pir motion sensor. US Patent 7,755,052.

[117] Mochizuki, E. and Koike, K. (2010). Field survey on actual conditions of light environment in mid-scale office buildings in japan. *Journal of Light & Visual Environment*, 34(3):157–164.

[118] More, A. and Raisinghani, V. (2014). Random backoff sleep protocol for energy efficient coverage in wireless sensor networks. In *Advanced Computing, Networking and Informatics-Volume 2*, pages 123–131. Springer.

[119] Mulla, D. J. (2013). Twenty five years of remote sensing in precision agriculture: Key advances and remaining knowledge gaps. *Biosystems engineering*, 114(4):358–371.

[120] Mumtaz, S. and Rodriguez, J. (2014). *Smart device to smart device communication*. Springer.

[121] Nachabe, L., Girod-Genet, M., and El Hassan, B. (2015). Unified data model for wireless sensor network. *IEEE Sensors Journal*, 15(7):3657–3667.

[122] Nan, C., Tila, F., and Kim, D. H. (2016). Design and implementation of service provider for integration of heterogeneous sensor networks. *International Journal of Control and Automation*, 9(1):47–54.

[123] Nassif, N. (2012). A robust co 2-based demand-controlled ventilation control strategy for multi-zone hvac systems. *Energy and buildings*, 45:72–81.

[124] Nethercote, N., Stuckey, P., Becket, R., Brand, S., Duck, G., and Tack, G. (2007). Minizinc: Towards a standard cp modelling language. *Principles and Practice of Constraint Programming–CP 2007*, pages 529–543.

[125] Nguyen, T. A. and Aiello, M. (2013). Energy intelligent buildings based on user activity: A survey. *Energy and Buildings*, 56:244–257.

[126] Obaidat, M. S., Anpalagan, A., and Woungang, I. (2012). *Handbook of green information and communication systems*. Academic Press.

[127] Oller, J., Demirkol, I., Casademont, J., Paradells, J., Gamm, G. U., and Reindl, L. (2016). Has time come to switch from duty-cycled mac protocols to wake-up radio for wireless sensor networks? *IEEE/ACM Transactions on Networking*, 24(2):674–687.

[128] Ozturk, U. A. and Norman, B. A. (2004). Heuristic methods for wind energy conversion system positioning. *Electric Power Systems Research*, 70(3):179–185.

[129] Pavani, M. and Rao, P. T. (2016). Real time pollution monitoring using wireless sensor networks. In *Information Technology, Electronics and Mobile Communication Conference (IEMCON), 2016 IEEE 7th Annual*, pages 1–6. IEEE.

[130] Pedersen, T. H., Nielsen, K. U., and Petersen, S. (2017). Method for room occupancy detection based on trajectory of indoor climate sensor data. *Building and Environment*, 115:147–156.

[131] Perera, C., Zaslavsky, A., Christen, P., and Georgakopoulos, D. (2014a). Context aware computing for the internet of things: A survey. *IEEE Communications Surveys & Tutorials*, 16(1):414–454.

[132] Perera, C., Zaslavsky, A., Christen, P., and Georgakopoulos, D. (2014b). Sensing as a service model for smart cities supported by internet of things. *Transactions on Emerging Telecommunications Technologies*, 25(1):81–93.

[133] Perera, C., Zaslavsky, A., Compton, M., Christen, P., and Georgakopoulos, D. (2013). Semantic-driven configuration of internet of things middleware. In *9th International Conference on Semantics, Knowledge and Grids (SKG'13)*, pages 66–73. IEEE.

[134] Pérez-Lombard, L., Ortiz, J., and Pout, C. (2008). A review on buildings energy consumption information. *Journal of Energy and buildings*, 40(3):394–398.

[135] Petty, S. (2014). Summary of ashrae's position on carbon dioxide (co2) levels in spaces.

[136] Pinarer, O., Gripay, Y., Servigne, S., and Ozgovde, A. (2016a). Energy enhancement of multi-application monitoring systems for smart buildings. In *International Conference on Advanced Information Systems Engineering*, pages 131–142. Springer.

[137] Pinarer, O., Gripay, Y., Servigne, S., and Ozgovde, A. (2016b). Real-time multi-application based sensor flux management. In *24th Signal Processing and Communication Application Conference (SIU'16)*, pages 765–768. IEEE.

[138] Pinarer, O., Gripay, Y., Servigne, S., Ozgovde, A., and Baskurt, A. (2017). Dynamic energy-aware sensor configuration in multi-application monitoring systems. *Pervasive and Mobile Computing*.

[139] Pinarer, O. and Ozgovde, A. (2015a). Application specific dynamic sleep scheduling. In *23th Signal Processing and Communications Applications Conference (SIU'15)*, pages 1765–1768. IEEE.

[140] Pinarer, O. and Ozgovde, A. (2015b). Improving the energy efficiency of wearable computing units using on sensor fifo memory. *International Journal of e-Education, e-Business, e-Management and e-Learning*, 5(2):105.

[141] Pinarer, O., Parmaksiz, A., Avci, I., Arslan, B., and Ozgovde, A. (2012). Wireless health monitoring of multiple patients on android phone with embedded computation. *Biomedical Engineering*, 9(1):569.

[142] Portal, T. S. (2014). Number of smartphone users worldwide from 2014 to 2020. Technical report. Accessed: 2017-03-10.

[143] Preisel, M., Diaz, A., and Wimmer, W. (2013). Energy consumption of smart meters. *Journal of Information and Communication Technologies*, page 37.

[144] Pu, Z., Li, Z., Ash, J., Zhu, W., and Wang, Y. (2017). Evaluation of spatial heterogeneity in the sensitivity of on-street parking occupancy to price change. *Transportation Research Part C: Emerging Technologies*, 77:67–79.

[145] Qin, Z., Han, Q., Mehrotra, S., and Venkatasubramanian, N. (2014). Quality-aware sensor data management. In *The Art of Wireless Sensor Networks*, pages 429–464. Springer.

[146] Rawassizadeh, R., Tomitsch, M., Nourizadeh, M., Momeni, E., Peery, A., Ulanova, L., and Pazzani, M. (2015). Energy-efficient integration of continuous context sensing and prediction into smartwatches. *Sensors*, 15(9):22616–22645.

[147] Rawat, P., Singh, K. D., Chaouchi, H., and Bonnin, J. M. (2014). Wireless sensor networks: a survey on recent developments and potential synergies. *The Journal of supercomputing*, 68(1):1–48.

[148] Raychoudhury, V., Cao, J., Kumar, M., and Zhang, D. (2013). Middleware for pervasive computing: A survey. *Pervasive and Mobile Computing*, 9(2):177–200.

[149] Reddy, K., Khaladkar, M. R., Khedekar, M. A., Khare, M. P., and Rajput, M. M. (2016). Building smart cities based on web architecture and using iot. *Imperial Journal of Interdisciplinary Research*, 3(1).

[150] Research, N. (2016). Building energy management system revenue. Technical report. Accessed: 2017-10-04.

[151] Rezaei, F., Shrestha, P. L., Hempel, M., Rakshit, S. M., and Sharif, H. (2015). Covert communication in wireless sensor networks for e-health applications. In *Technological Breakthroughs in Modern Wireless Sensor Applications*, pages 352–371. IGI Global.

[152] Rezgui, A. and Eltoweissy, M. (2007). Service-oriented sensor-actuator networks [ad hoc and sensor networks]. *IEEE Communications Magazine*, 45(12):92–100.

[153] Rieken, D. W. and Walker II, M. R. (2011). Ultra low frequency power-line communications using a resonator circuit. *IEEE Transactions on Smart Grid*, 2(1):41–50.

[154] Rutishauser, U., Joller, J., and Douglas, R. (2005). Control and learning of ambience by an intelligent building. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 35(1):121–132.

[155] Samundiswary, P., Sathian, D., and Dananjayan, P. (2010). Secured greedy perimeter stateless routing for wireless sensor networks. *arXiv preprint arXiv:1009.0585*.

[156] Saraswat, J. and Bhattacharya, P. P. (2013). Effect of duty cycle on energy consumption in wireless sensor networks. *International Journal of Computer Networks & Communications*, 5(1):125.

[157] Schor, L., Sommer, P., and Wattenhofer, R. (2009). Towards a zero-configuration wireless sensor network architecture for smart buildings. In *Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, pages 31–36. ACM.

[158] Schreiber, F. A., Camplani, R., Fortunato, M., Marelli, M., and Rota, G. (2012). Perla: A language and middleware architecture for data management and integration in pervasive information systems. *IEEE Transactions on Software Engineering*, 38(2):478–496.

[159] Seiffert, R. and Wong, S. (1998). Dynamic monitoring architecture. US Patent 5,787,409.

[160] Servigne, S., Gripay, Y., Deleuil, J.-M., Jay, J., and Mebrouk, R. (2014). Modélisation générique de données capteurs hétérogènes: De la captation de données à la compréhension de phénomènes. In *Conférence Internationale en Géomatique et Analyse Spatiale (SAGEO'14)*, pages 1–14.

[161] Servigne, S., Gripay, Y., Pinarer, O., Samuel, J., Ozgovde, A., and Jay, J. (2016). Heterogeneous sensor data exploration and sustainable declarative monitoring architecture: Application to smart building. *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci*, IV-4/W1:97–104.

[162] Sharaf, M. A., Beaver, J., Labrinidis, A., and Chrysanthis, P. K. (2003). Tina: a scheme for temporal coherency-aware in-network aggregation. In *Proceedings of the 3rd ACM International Workshop on Data Engineering for Wireless and Mobile Access (MobiDE'03)*, pages 69–76. ACM.

[163] Sharma, D., Verma, S., and Sharma, K. (2013). Network topologies in wireless sensor networks: A review 1.

[164] Shnayder, V., Hempstead, M., Chen, B.-r., Allen, G. W., and Welsh, M. (2004). Simulating the power consumption of large-scale sensor network applications. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 188–200. ACM.

[165] Snasel, V., Kong, L., Tsai, P., and Pan, J.-S. (2016). Sink node placement strategies based on cat swarm optimization algorithm. *J. Netw. Intell*, 1(2):52–60.

[166] Song, G. (2016). *Méthodes parallèles pour le traitement des flux de données continus*. PhD thesis, Paris Saclay.

[167] Song, M., Chen, K., He, Z., and Zhang, X. (2014). Optimization of wind farm micro-siting for complex terrain using greedy algorithm. *Energy*, 67:454–459.

[168] Srbinovska, M., Gavrovski, C., Dimcev, V., Krkoleva, A., and Borozan, V. (2015). Environmental parameters monitoring in precision agriculture using wireless sensor networks. *Journal of Cleaner Production*, 88:297–307.

[169] Stavropoulos, T. G., Koutitas, G., Vrakas, D., Kontopoulos, E., and Vlahavas, I. (2016). A smart university platform for building energy monitoring and savings. *Journal of Ambient Intelligence and Smart Environments*, 8(3):301–323.

[170] Stetsko, A., Stehlik, M., and Matyas, V. (2011). Calibrating and comparing simulators for wireless sensor networks. In *IEEE 8th International Conference on Mobile Adhoc and Sensor Systems (MASS'11)*, pages 733–738. IEEE.

[171] Stojmenovic, I. and Lin, X. (2001). Power-aware localized routing in wireless networks. *IEEE Transactions on Parallel and Distributed Systems,*, 12(11):1122–1133.

[172] Surdu, S., Gripay, Y., Scuturici, V.-M., and Petit, J.-M. (2013). P-bench: benchmarking in data-centric pervasive application development. In *Transactions on Large-Scale Data-and Knowledge-Centered Systems XI*, pages 51–75. Springer.

[173] Tennina, S., Koubâa, A., Daidone, R., Alves, M., Jurčík, P., Severino, R., Tiloca, M., Hauer, J.-H., Pereira, N., Dini, G., et al. (2013). *IEEE 802.15. 4 and ZigBee as enabling technologies for low-power wireless systems with quality-of-service constraints*. Springer Science and Business Media.

[174] Thiagarajan, A. and Madden, S. (2008). Querying continuous functions in a database system. In *Proceedings of International Conference on Management of Data (ACM SIGMOD'08)*, pages 791–804. ACM.

[175] Tulone, D. and Madden, S. (2006a). An energy-efficient querying framework in sensor networks for detecting node similarities. In *Proceedings of the 9th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM'06)*, pages 191–300. ACM.

[176] Tulone, D. and Madden, S. (2006b). Paq: Time series forecasting for approximate query answering in sensor networks. In *Wireless Sensor Networks*, pages 21–37. Springer.

[177] Verdone, R., Dardari, D., Mazzini, G., and Conti, A. (2010). *Wireless sensor and actuator networks: technologies, analysis and design*. Academic Press.

[178] Viani, F., Polo, A., Robol, F., Oliveri, G., Rocca, P., and Massa, A. (2014). Crowd detection and occupancy estimation through indirect environmental measurements. In *Antennas and Propagation (EuCAP), 2014 8th European Conference on*, pages 2127–2130. IEEE.

[179] Viani, F., Robol, F., Polo, A., Rocca, P., Oliveri, G., and Massa, A. (2013). Wireless architectures for heterogeneous sensing in smart home applications: Concepts and real implementation. *Proceedings of the IEEE*, 101(11):2381–2396.

[180] Wang, Z., Yang, R., Wang, L., Green, R. C., Dounis, A., et al. (2011). A fuzzy adaptive comfort temperature model with grey predictor for multi-agent control system of smart building. In *IEEE Congress on Evolutionary Computation (CEC'11)*, pages 728–735. IEEE.

[181] Wang, Z. M., Basagni, S., Melachrinoudis, E., and Petrioli, C. (2005). Exploiting sink mobility for maximizing sensor networks lifetime. In *Processing of Hawaii International Conference on System Sciences (HICSS'05)*, pages 287a–287a. IEEE.

[182] Warden, D. (2004). Supply air co2 control of minimum outdoor air for multiple space systems. *ASHRAE journal*, 46(10):26.

[183] Willig, A. (2006). Wireless sensor networks: concept, challenges and approaches. *Elektrotechnik und Informationstechnik*, 123(6):224–231.

[184] World Resources Institute, B. E. I. (2011). Institute for building efficiency. Technical report. Accessed: 2017-10-04.

[185] Wu, Y., Fahmy, S., and Shroff, N. B. (2008). On the construction of a maximum-lifetime data gathering tree in sensor networks: Np-completeness and approximation algorithm. In *Proceedings IEEE Computer and Communications Societies IEEE Annual Joint Conference (INFOCOM'08)*, pages 356–360. IEEE.

[186] Wu, Y., Li, X.-Y., Li, Y., and Lou, W. (2010). Energy-efficient wake-up scheduling for data collection and aggregation. *IEEE Transactions on Parallel and Distributed Systems*, 21(2):275–287.

[187] Xia, F. (2008). Qos challenges and opportunities in wireless sensor/actuator networks. *Sensors*, 8(2):1099–1110.

[188] Xia, F., Vinel, A., Gao, R., Wang, L., and Qiu, T. (2011). Evaluating ieee 802.15. 4 for cyber-physical systems. *EURASIP Journal on Wireless Communications and Networking*, 2011(1):596397.

[189] Xia, F., Zhao, W., Sun, Y., and Tian, Y.-C. (2007). Fuzzy logic control based qos management in wireless sensor/actuator networks. *Sensors*, 7(12):3179–3191.

[190] Xie, S. and Wang, Y. (2014). Construction of tree network with limited delivery latency in homogeneous wireless sensor networks. *Wireless Personal Communications*, 78(1):231–246.

[191] Yao, Y. and Gehrke, J. (2002). The cougar approach to in-network query processing in sensor networks. *ACM Sigmod record*, 31(3):9–18.

[192] Yao, Y., Gehrke, J., et al. (2003). Query processing in sensor networks. In *The Conference on Innovative Data Systems Research (CIDR'03)*, pages 233–244.

[193] Ye, J., Dasiopoulou, S., Stevenson, G., Meditskos, G., Kontopoulos, E., Kompatsiaris, I., and Dobson, S. (2015). Semantic web technologies in pervasive computing: A survey and research roadmap. *Pervasive and Mobile Computing*, 23:1–25.

[194] Yu, F. and Jain, P. R. (2011). A survey of wireless sensor network simulation tools. *Journal of Washington University, Department of Science and Engineering*.

[195] Yu, J., Kim, M., Bang, H.-C., Bae, S.-H., and Kim, S.-J. (2016). Iot as a applications: cloud-based building management systems for the internet of things. *Multimedia Tools and Applications*, 75(22):14583–14596.

[196] Yu, X., Wu, P., Han, W., and Zhang, Z. (2013). A survey on wireless sensor network infrastructure for agriculture. *Computer Standards & Interfaces*, 35(1):59–64.

[197] Zhu, C., Li, X., Leung, V. C., Yang, L. T., Ngai, E. C.-H., and Shu, L. (2017). Towards pricing for sensor-cloud. *IEEE Transactions on Cloud Computing*.

[198] Zigbee (2006). Zigbee specification. Technical report. Accessed: 2017-03-10.

[199] Żmuda, D., Psiuk, M., and Zieliński, K. (2010). Dynamic monitoring framework for the soa execution environment. *Procedia Computer Science*, 1(1):125–133.

Appendix A

List of Publications

A.1 Journal Articles

1. Ozgun Pinarer, Yann Gripay, Sylvie Servigne, Atay Ozgovde and Atilla Baskurt. "Dynamic energy-aware sensor configuration in multi-application monitoring systems." Pervasive and Mobile Computing (2017).
   DOI: 10.1016/j.pmcj.2017.08.005
   HAL: https://hal.archives-ouvertes.fr/hal-01585263 (PDF available)

A.2 Conference Papers

1. Ozgun Pinarer, Yann Gripay, Sylvie Servigne and Atay Ozgovde. "Energy Enhancement of Multi-application Monitoring Systems for Smart Buildings." In International Conference on Advanced Information Systems Engineering, pp. 131-142. Springer International Publishing, 2016.
   DOI: 10.1007/978-3-319-39564-7_14
   HAL: https://hal.archives-ouvertes.fr/hal-01334900/ (PDF available)

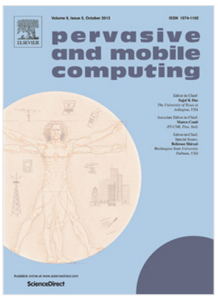
2. Ozgun Pinarer, Yann Gripay, Sylvie Servigne and Atay Ozgovde. "Real-time multi-application based sensor flux management." In 24th Signal Processing and Communication Application Conference (SIU'16),pp. 765-768. IEEE, 2016.
   DOI: 10.1109/SIU.2016.7495852 (turkish publication)

3. Sylvie Servigne, Yann Gripay, Ozgun Pinarer, John Samuel, Atay Ozgovde and Jacques Jay. "Heterogeneous Sensor Data Exploration and Sustainable Declarative Monitoring Architecture: Application to Smart Building." ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences (2016): 97-104.
   DOI: 10.5194/isprs-annals-IV-4-W1-97-2016
   HAL: https://hal.archives-ouvertes.fr/hal-01386110 (PDF available)

Appendix B

Journal Article: Dynamic Energy-aware Sensor Configuration in multi-Application Monitoring Systems

Accepted Manuscript



Dynamic energy-aware sensor configuration in multi-application monitoring systems

Ozgun Pinarer, Yann Gripay, Sylvie Servigne, Atay Ozgovde, Atilla Baskurt

PII: S1574-1192(17)30081-0
DOI: http://dx.doi.org/10.1016/j.pmcj.2017.08.005
Reference: PMCJ 885

To appear in: *Pervasive and Mobile Computing*

Received date : 13 February 2017
Revised date : 3 August 2017
Accepted date : 22 August 2017

Please cite this article as: O. Pinarer, Y. Gripay, S. Servigne, A. Ozgovde, A. Baskurt, Dynamic energy-aware sensor configuration in multi-application monitoring systems, *Pervasive and Mobile Computing* (2017), http://dx.doi.org/10.1016/j.pmcj.2017.08.005

Dynamic Energy-Aware Sensor Configuration in Multi-Application Monitoring Systems

Ozgun Pinarer[a,b], Yann Gripay[a], Sylvie Servigne[a], Atay Ozgovde[b], Atilla Baskurt[a]

[a] *Univ Lyon, CNRS, France*
INSA-Lyon, LIRIS, UMR5205, F-69621
Email: name.surname@insa-lyon.fr
[b] *Galatasaray University*
Ciragan Cad. No:36 34349 Istanbul, Turkey
Email: {opinarer, aozgovde}@gsu.edu.tr

Abstract

A typical pervasive monitoring system like a smart building depends on an infrastructure composed of hundreds of heterogeneous wireless sensor devices. Managing the energy consumption of these devices poses a challenging problem that affects the overall efficiency and usability. Existing approaches for sensor energy consumption typically assume a single monitoring application to consume sensor data and a static configuration for sensor devices. In this paper, we focus on a multi-application context with dynamic requirements and multi-modal sensor devices. We present 3SoSM, an approach to optimize interactions between application requirements and wireless sensor environment in real-time. It relies on an energy-aware dynamic configuration of sensor devices to lower energy consumption while fulfilling application requirements. To bind together sensor configuration and dynamic management of data streams, we design a sustainable multi-application monitoring system architecture for pervasive environments that collects application requirements for sensor data streams and optimizes them into sensor configurations. To demonstrate the effectiveness of our approach, a set of experiments are designed in the context of smart buildings. We comparatively evaluate our approach to show how dynamic sensor configuration for multiple monitoring applications indeed outperforms the mainstream duty-cycling method.

Keywords: pervasive environment, sensor data management, continuous query processing, wireless sensor network

1. Introduction

Smart building applications have long gained attention in the scientific community and nowadays in the industry. Buildings, whether smart or not, are among the primary consumers of the available energy sources today. Therefore, considerable effort is poured into the design of smart buildings where energy consumption is tried to get decreased without sacrificing the satisfaction of the occupants. To accomplish this aim, the components of the building itself, as well as the context of its users, should be continuously monitored by means of streams of data from sensory inputs.

A typical pervasive monitoring system like a smart building consists of multi-modal wireless sensor devices that are equipped with sensing, processing and communication facilities. Sensing part can measure physical quantities about the environment with some given sampling rate (sampling frequency), processing part is able to do some computation on the measured values and communication part is able to listen and send data packets to other sensor devices. These deployed wireless devices are autonomous in terms of energy: they have limited energy and battery lifetime. Smart building technology mainly relies on such wireless sensing infrastructure. In this field, initial commercial solutions are `Building Automation Systems` where a specific set of functionalities such as heating and ventilation control and lighting management is implemented by a single vendor. However, as the sensor infrastructure is becoming a standard component incorporated into the design and construction of the modern buildings, flexible application development by a third party is becoming an issue rather than the infrastructure itself.

In this context, one of the main contributions of this paper is to allow multiple applications to exploit the same smart building infrastructure while considering energy consumption of the sensor devices in this infrastructure. Our energy-aware monitoring system continuously adapt to various application requirements, to building actual context and to user configuration. These applications operate on a network of multi-modal wireless sensor devices and use declarative continuous queries to exploit sensor data streams. Application requirements as declared by the application developer can be fulfilled in a multitude of ways: our method translates these requirements as energy efficient acquisition and transmission schedules for the wireless sensor

network. We propose ***S**mart-**S**ervice **S**tream-**o**riented **S**ensor **M**anagement* (3SoSM), an approach to optimize interactions between application requirements and the wireless sensor environment in real-time. The core of our approach is the definition of a Schedule Time Pattern with acquisition, transmission and reception actions for each device configuration. The dynamic reconfiguration of devices is performed through the real-time update and optimization of Schedule Time Patterns according to application requirements that may change over time. With this approach, we expect to avoid unnecessary data measurements that may occur with static configuration and to promote grouped or even compressed data transmission when possible.

In this article, Section 2 presents an overview of our multi-application monitoring system architecture. Formalization of our approach is explained in Section 3 and the energy-aware optimization process is detailed in Section 4. Section 5 gives a brief description of our experimental platform to implement our approach and Section 6 describes the experiments we conducted. Experiment results are discussed in Section 7. Related works are given in Section 8 and finally, conclusions are given in Section 9.

2. Overview of 3SoSM

Our 3SoSM approach focuses on a multi-application monitoring system with multi-modal sensor devices (i.e. devices that can measure different physical quantities) and provides finer sensor configuration than duty-cycle and similar techniques. Our proposition of dynamic sensor management based on real-time application requirements is performed at the gateway level, in order to optimize energy consumption of sensor devices independently from the application layer and/or the query engine.

2.1. Monitoring Architecture for Smart Building Applications

In this study, we adopt a "declarative monitoring architecture", built upon a `Pervasive Environment Management System (PEMS)` as presented in [1, 2, 3]. Using declarative (*SQL-like*) continuous queries, an application can easily interact with distributed devices like sensors. Figure 1 presents our declarative monitoring application that has 4 main layers: Application, PEMS Query Engine, PEMS Gateway and WSN (Wireless Sensor Network). `Application` layer provides end-user access to the monitoring system. Application requirements are defined at this layer and are declaratively expressed as a set of continuous queries over distributed services [1]. `PEMS`

`Query Engine` is responsible for managing query executions of queries coming from the Application layer. This layer integrates sensor devices as non-conventional, dynamic and heterogeneous data sources. To manage that, it includes a continuous query engine that interacts with services provided by the environment, in particular sensor services. `PEMS Gateway` stands for managing bidirectional communication and interactions between PEMS Query Engine and WSN. `WSN` layer represents wireless sensor devices that acquire physical quantity measures and can communicate with other sensor devices and physical gateways.

We use the `Service-Oriented Continuous Query (SoCQ)` framework [1] for the `PEMS Query Engine`. The `SoCQ` engine handles a multi-application mechanism and supports multiple parameterized stream subscriptions to the same device. Moreover, it supports real-time user configuration of applications and context-aware applications through queries that can dynamically combine data, streams and services. The user may insert complex continuous queries with conditions over data streams to trigger new interactions with services during their execution. Interactions with services (discovery, invocations of methods, subscriptions to data streams) are handled by `PEMS Gateway`. However, the `SoCQ` Engine itself does not overcome the reconfiguration of sensor devices, and thus assumes a pre-existing static configuration. Like for other approaches presented in the Related Works section, it is an issue for the energy enhancement of the system.

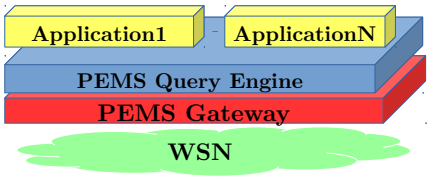


Figure 1: Declarative Monitoring Architecture.

2.2. From Dynamic Application Requirements to Dynamic Sensor Configurations

We consider that applications define their requirements as a set of continuous queries over parameterized data streams produced by sensor devices. For instance: "*Application A1 computes the average temperature over the last 10 minutes, with an update every 5 minutes, with an accuracy of 1 second and a maximum latency of 1 minute*". In this example, the application requests temperature measure data streams from all devices with temperature sensors. To express the application requirements concerning sensor data management, we consider the following parameters targeting sensor measures: • `temporal`

`window` introduces the time interval for calculating the result (for the given example: 10 min); • `update periodicity` stands for the refreshing rate of the result (5 min); • `acquisition periodicity` represents the temporal accuracy of measures (1 sec); • `maximum latency` presents the maximum acceptable delay between the acquisition of data and its transmission to the `PEMS` layer for result calculation (1 min). `temporal window` and `update periodicity` concern the computing of the result, whereas `acquisition periodicity` and `maximum latency` are related to acquisition/transmission of data by sensor devices.

Here, we propose a novel approach: management and optimization of the real-time application requirements to enhance the energy consumption of the whole system. To fulfill the application requirements, we define a Schedule Time Pattern for each sensor device and configure devices accordingly. Our approach assumes that sensor devices are dynamically configurable [4].

3. 3SoSM Formalization

We consider that application requirements are parameterized stream subscription requests and the network topology is a tree topology. A solution to our optimization problem is individual Schedule Time Patterns, namely SCO-PATTERNs.

3.1. Application Requirements

Let D be the set of wireless sensor devices in the `WSN` environment. Each sensor device $d_i \in D$ may have multiple modalities to acquire different physical quantity measures $m_i \in M$, e.g., temperature, humidity. Application requirements are defined in terms of data source requirements: targeted sensors d and requested measures m; and of temporal requirements: temporal window β, update periodicity p^{upd}, acquisition periodicity p^{acq} and maximum latency λ. The unit is the second (or millisecond, if required).

For clarity, we do not present β, p^{upd} in this paper. We then represent application requirements on sensors at a given time instant by a set of parameterized subscription requests $S = \{s_1, s_2, \ldots, s_n\}$, where:
$s_i = (d_i, m_i, p_i^{acq}, \lambda_i) \in D \times M \times \mathbb{N}^+ \times \mathbb{N}^+$

Example 1. *Suppose two applications with the following requirements on 6 sensors:*
***`Application 1`**: Temperature (m_T) of sensors d_1, d_3, d_5 with an acquisition*

periodicity of 2 sec and a latency of 4 sec;
Application 2*: Humidity (m_H) of sensors d_3, d_4, d_5, d_6 with an acquisition periodicity of 5 sec and a latency of 3 sec.*
Based on the previous notation, all application requirements are represented by a set of parameterized subscription requests:
$S = \{(d_1, m_T, 2, 4), (d_3, m_T, 2, 4), (d_5, m_T, 2, 4), (d_3, m_H, 5, 3), (d_4, m_H, 5, 3), (d_5, m_H, 5, 3), (d_6, m_H, 5, 3)\}$

3.2. Network Topology

To transmit data to the central base station (where results are computed), we suppose that deployed wireless sensor devices construct a logical tree (tree topology) where each sensor has only one neighbor to deliver data packets. This tree structure provides a uni-path for each sensor to reach the base station (sink device). In fact, our optimization algorithm relies on known and unique paths, as it does not consider a choice of network paths between two devices. Integrating more complex network topologies, such as mesh networks, would be a future work. However, the tree topology may be updated each time application requirements change, as it triggers a new optimization. For instance [5] proposes a new technique to organize the sensor devices that is likely preferable for outdoor monitoring systems: P-SEP (a prolong stable election routing algorithm). The authors consider two-level sensor device heterogeneities: advanced and normal devices. They propose a clustering mechanism and present new cluster head selecting policy to extend the lifetime of the system (lifetime of the system is evaluated in terms of FND (First Node Dies)). P-SEP puts forth efficient simulation results and network lifetime. Although we do not tackle routing issues, applying our dynamic network behavior due to the dynamicity of the application requirements, with the P-SEP mechanism could produce concrete improvements on the lifetime of the system. In our case, with our assumptions about the network, each sensor device has its own role in the topology based on the current applications and its location: `Source` (responsible for the data acquisition and transmission) and/or `Relay` (responsible for the reception and re-transmission), or `Sink` (responsible for only the reception).

Example 2. *A tree topology with 6 multi-modal sensor devices and a base station is illustrated in Figure 2. A tree topology has a layered form based on the distance to the base station.*

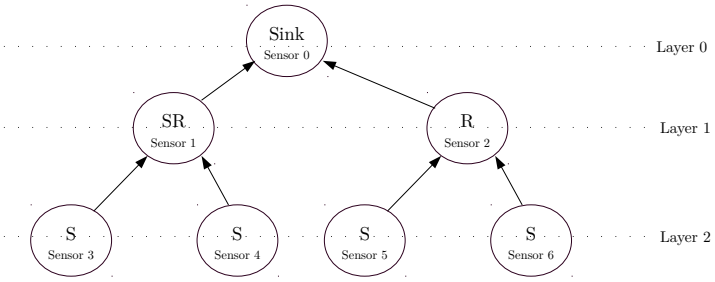


Figure 2: A tree-structured topology and different sensor device roles: Source (S), Relay (R), Source+Relay (SR), Sink.

3.3. SCO-PATTERN

The idea behind the `3SoSM` approach is to create a `Schedule Time Pattern` for each sensor device. This pattern, described in [6] is a schedule composed of sensor actions, here with three types of actions: `acquisition A, reception R, transmission T`. Based on their roles, only certain actions are pertinent to sensor devices. Sensor devices can configure themselves with a Schedule Time Pattern, and then execute periodically the scheduled actions: acquire a measure and store it in a memory buffer, transmit all buffered measures one hop towards the sink (and empty the buffer), receive measures from a lower-layer device and store it in a memory buffer.

We call this pattern **sensor configuration oriented pattern** or SCO-PATTERN. This pattern consists of timestamped events ($< timestamp, action >$ couples) and a length ℓ (or periodicity) of that pattern. These actions are enclosed by the length of the pattern: event timestamps are in time interval $[\ 0;\ \ell\ [$. A SCO-PATTERN is denoted by:
$\mathrm{P} = (\{(t_i, a_i)\}, \ell)$ with $\ell \in \mathbb{N}^+$, $t_i \in [0; \ell[$, $a_i \subset \{A_m, T, R\}$ where A_m indicates the data acquisition of the physical measure m such as A_T for temperature, A_H for humidity, etc.

4. 3SoSM: Energy-Aware Optimization Process

To optimize the energy consumption of the devices, since the communication costs are the most significant part of the energy budget [7], we want to minimize the number of transmission and reception actions of each device: less communication means less consumed energy. We first construct "preliminary" SCO-PATTERNs that have a common length with only acquisition actions (Acq) for all the devices. It is then necessary to insert the transmission (Tx) and reception (Rx) actions in these SCO-PATTERNs to obtain a solution, composed of the set of complete SCO-PATTERNs. A solution is

valid if all acquired data traverse the network topology from their source device to the base station before the "latency" associated with that data expires. The optimal solution is a valid solution that minimizes the number of Transmission / Reception actions of each device.

The objective of the optimization process is to transform the application requirements into a global schedule and then to build individual SCO-PATTERNs to configure each device. Application requirements are represented by a set of subscription requests $S = \{s_1, s_2, s_3, ...\}$. Hence the energy-aware optimization process can be expressed as a transformation function that gets a set of subscription requests (all subscription requests from all applications) and a network topology $\mathcal{N}$ and produces a SCO-PATTERN for each device: $f(\{s_1, s_2, ..., s_n\}, \mathcal{N}) = \{\mathrm{P}_{d_1}, \mathrm{P}_{d_2}, ..., \mathrm{P}_{d_m}\}$.

4.1. Preliminary SCO-PATTERN

The preliminary SCO-PATTERN is specific for each sensor device. It is periodic, and its length indicates the periodicity. Each sensor device may have different acquisition periodicities from different applications. However, in order to build a global view of the network, a common length is required: it is the lowest common multiple of acquisition periodicities from all subscription requests. Moreover, the length of the schedule should be greater than any latency of the subscription requests to properly handle data expiration time: The final length is the lowest multiple of this initial common length greater than any data expiration time.

Example 3. *The common length of the patterns (for the given applications in Example 1) is calculated as:*
Initial common length $\ell_{min} = LCM(2, 2, 2, 5, 5, 5, 5) = 10$
Maximum latency $\lambda max = MAX(4, 4, 4, 3, 3, 3, 3) = 4$
Final length $\ell_{schedule} = 10\ sec > \lambda_{max}$

Then, based on the periodicity of subscription requests, acquisition actions are defined. Furthermore, each acquisition action is tagged with its maximum latency (from the subscription request).

Example 4. *Preliminary* SCO-PATTERN *for* d_3 *with subscriptions* $\{(d_3, m_T, 2, 4), (d_3, m_H, 5, 3)\}$ *:*
$P = (\{(0, \{A^4_T, A^3_H\}), (2, A^4_T), (4, A^4_T), (5, A^3_H), (6, A^4_T), (8, A^4_T)\}, 10)$

4.2. Transmission Constraint and Optimization Goal

To form a global schedule, we adopt a similar approach to Galpin et al. [8]. A predefined granularity determines the slots in the time space: the schedule is divided into time-slots and each slot may be filled with sensor actions. For the following examples, granularity is set to 0.5 *sec* (e.g. 20 slots for 10 seconds).

The major constraint of the system is that all the acquired data must pass through the network topology up to the base station before the expiration of their "latency" in order to fullfill the application requirements. A transmission action on a sensor device during a single time slot stands for a transmission of all the data (acquired by itself and received from lower-layer neighbors) not yet sent, including acquisitions on the same time slot. This transmission action requires a reception action at the same time slot on the sensor device that receives the data. Besides there are other constraints due to the radio protocol such as a device can only receive data from a single device at a time on the same time slot and a device can not receive and send on the same time slot.

4.3. Searching Optimal Communication Slots

We propose an optimization algorithm to choose the optimal communication slots from the preliminary SCO-PATTERN. The optimization process starts by propagating transmission and reception constraints due to the acquisition events in a bottom-up process, and then analyzes the possible reception actions of the base station and tries to find the most energy efficient communication slots, and finally propagates those choices to lower layers and continues the analyze in a top-down process.

4.3.1. Schedule Table Initialization

We use a schedule table to represent the sensor action slots for Tx/Rx. Each sensor device has its own schedule and each schedule is composed of two parts: transmission and reception part. The length of the schedule is defined by the common length of preliminary SCO-PATTERNs. Each line of the schedule table corresponds to an acquired measure: its transmission action for the transmission part and its reception action for the reception part.

The transmission part indicates the possible transmission slots for the sensor device for each acquired measure. According to the latency information λ, every sensor device has the initiative to keep the data (instead of

sending it immediately) and send it at an optimal later moment under the condition that this acquired data arrives at the base station before the expiration of its latency. The reception part represents the possible reception slots for each data coming from lower-layer neighbors (zero, one or several depending on the network topology). Schedule Table for Sensor 3, Sensor 1 and Sensor 0 (sink device) are illustrated in Figure 3.

We propagate bottom-up constraints for data transmission and reception based on latencies (from preliminary SCO-PATTERNs) and on the network topology.

Transmission Part. Information from the preliminary SCO-PATTERN is used to fill the transmission part. For a given sensor device, each acquired data and each received data is represented as a line in the transmission part. A line is filled with the residual latency information starting from the time slot where the device acquires the data or from the time slot following the time slot where the device received the data, and until residual latency reaches 0 (in fact, 1 time slot before at 1 hop from the sink, 2 time slots at 2 hops, etc.).

For each device d, $T_d = [e_{i,j}]$ is a 2D matrix with dimensions m_dxn. m_d is the number of data to transmit and n stands for the number of time slots. $e_{i,j}$ represents a potential transmission event of data i at time slot j: $e_{i,j} = \lambda_{i,j}$ is the residual latency, $e_{i,j} = 0$ means that data i can not be transmitted at that time slot.

Example 5. *Transmission parts of the schedules of Sensors 3 and 1 are presented respectively in Figure 3a, 3b. For instance, Sensor 3 measures temperature and humidity values. For A2#1 (first acquisition for humidity), the latency is 3sec. Hence, to send that data, Sensor 3 has 5 slots (t=0,0.5, 1, 1.5, 2 sec) before the latency is expired as it is 2 hops from sink. Transmission part of the schedule of Sensor 1 covers transmission of its own data and also received data from Sensor 3 and 4 (see Fig 3b).*

Reception Part. For a given sensor device, the reception part is based on the transmission part of the schedules of lower-layer neighbors. Each line represents a data to receive and is linked to one line from the transmission part of the source device. Possible reception slots are filled with the data residual latency from the corresponding transmission time slot.

For each device d, $R_d = [e_{i,j}]$ is a 2D matrix with dimensions m_dxn where m_d is the number of data to receive and n stands for the number of time slots.

		Time(sec)																			
Acquisition	Measure	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5
Acq of S3:A1	temperature	A1#1				A1#2				A1#3				A1#4				A1#5			
Acq of S3:A2	humidity	A2#1										A2#2									

		Time (sec)																			
Transmission	To	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5
TX of S3:A1#1	S1	4	3.5	3	2.5	2	1.5	1													
TX of S3:A1#2	S1					4	3.5	3	2.5	2	1.5	1									
TX of S3:A1#3	S1									4	3.5	3	2.5	2	1.5	1					
TX of S3:A1#4	S1													4	3.5	3	2.5	2	1.5	1	
TX of S3:A1#5	S1	2	1.5	1														4	3.5	3	2.5
TX of S3:A2#1	S1	3	2.5	2	1.5	1															
TX of S3:A2#2	S1											3	2.5	2	1.5	1					

(a) Schedule of Sensor 3 (Acquisition + Transmission).

		Time(sec)																			
Acquisition	Measure	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5
Acq of S1:A1	temperature	A1#1				A1#2				A1#3				A1#4				A1#5			

		Time (sec)																			
Reception	From	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5
RX of S3:A1#1	S3	4	3.5	3	2.5	2	1.5	1													
RX of S3:A1#2	S3					4	3.5	3	2.5	2	1.5	1									
RX of S3:A1#3	S3									4	3.5	3	2.5	2	1.5	1					
RX of S3:A1#4	S3													4	3.5	3	2.5	2	1.5	1	
RX of S3:A1#5	S3	2	1.5	1														4	3.5	3	2.5
RX of S3:A2#1	S3	3	2.5	2	1.5	1															
RX of S3:A2#2	S3											3	2.5	2	1.5	1					
RX of S4:A1#1	S4	3	2.5	2	1.5	1															
RX of S4:A1#2	S4											3	2.5	2	1.5	1					

		Time (sec)																			
Transmission	To	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5
TX of S1:A1#1	S0	4	3.5	3	2.5	2	1.5	1	0.5												
TX of S1:A1#2	S0					4	3.5	3	2.5	2	1.5	1	0.5								
TX of S1:A1#3	S0									4	3.5	3	2.5	2	1.5	1	0.5				
TX of S1:A1#4	S0													4	3.5	3	2.5	2	1.5	1	0.5
TX of S1:A1#5	S0	2	1.5	1	0.5													4	3.5	3	2.5
TX of S3:A1#1	S0		3.5	3	2.5	2	1.5	1	0.5												
TX of S3:A1#2	S0						3.5	3	2.5	2	1.5	1	0.5								
TX of S3:A1#3	S0										3.5	3	2.5	2	1.5	1	0.5				
TX of S3:A1#4	S0														3.5	3	2.5	2	1.5	1	0.5
TX of S3:A1#5	S0	2	1.5	1	0.5														3.5	3	2.5
TX of S3:A2#1	S0		2.5	2	1.5	1	0.5														
TX of S3:A2#2	S0												2.5	2	1.5	1	0.5				
TX of S4:A1#1	S0		2.5	2	1.5	1	0.5														
TX of S4:A1#2	S0												2.5	2	1.5	1	0.5				

(b) Schedule of Sensor 1 (Acquisition + Reception + Transmission).

		Time (sec)																			
Reception	From	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5
RX of S1:A1#1	S1	4	3.5	3	2.5	2	1.5	1	0.5												
RX of S1:A1#2	S1					4	3.5	3	2.5	2	1.5	1	0.5								
RX of S1:A1#3	S1									4	3.5	3	2.5	2	1.5	1	0.5				
RX of S1:A1#4	S1													4	3.5	3	2.5	2	1.5	1	0.5
RX of S1:A1#5	S1	2	1.5	1	0.5													4	3.5	3	2.5
RX of S3:A1#1	S1		3.5	3	2.5	2	1.5	1	0.5												
RX of S3:A1#2	S1						3.5	3	2.5	2	1.5	1	0.5								
RX of S3:A1#3	S1										3.5	3	2.5	2	1.5	1	0.5				
RX of S3:A1#4	S1														3.5	3	2.5	2	1.5	1	0.5
RX of S3:A1#5	S1	2	1.5	1	0.5														3.5	3	2.5
RX of S3:A2#1	S1		2.5	2	1.5	1	0.5														
RX of S3:A2#2	S1												2.5	2	1.5	1	0.5				
RX of S4:A1#1	S1		2.5	2	1.5	1	0.5														
RX of S4:A1#2	S1												2.5	2	1.5	1	0.5				
RX of S5:A1#1	S2		3.5	3	2.5	2	1.5	1	0.5												
RX of S5:A1#2	S2						3.5	3	2.5	2	1.5	1	0.5								
RX of S5:A1#3	S2										3.5	3	2.5	2	1.5	1	0.5				
RX of S5:A1#4	S2														3.5	3	2.5	2	1.5	1	0.5
RX of S5:A1#5	S2	2	1.5	1	0.5														3.5	3	2.5
RX of S5:A2#1	S2		2.5	2	1.5	1	0.5														
RX of S5:A2#2	S2												2.5	2	1.5	1	0.5				
RX of S6:A1#1	S2		2.5	2	1.5	1	0.5														
RX of S6:A1#2	S2												2.5	2	1.5	1	0.5				

(c) Schedule of Sensor 0 (Sink device) (Reception).

Figure 3: Schedules (highlighted slots are the optimal communication slots based on our process).

$e_{i,j}$ represents a potential reception event of data i at time slot j: $e_{i,j} = \lambda_{i,j}$ is the residual latency, $e_{i,j} = 0$ means that data i can not be received at that time slot.

Example 6. *For the given example, reception parts of the schedule for Sensor 1 and Sink are represented respectively in Figure 3b and 3c. For instance, reception part of Sensor 1 represents the possible reception slots for each acquired data of Sensor 3 and 4 with their residual latency.*

4.3.2. Optimal Choice of Tx/Rx Slots

For each device d_i, potential transmission and reception slots are now identified in their schedule table (matrices T_{d_i} and R_{d_i}). The optimization

process should now choose the most energy-efficient slots, in order to minimize the number of Tx/Rx slots for each device. From the reception part of the schedule, the algorithm tries to group the receptions from the same sensor device. For each time slot, it calculates the total number of possible receptions for each neighbor. The algorithm searches the latest slot in which maximum number of reception can occur. The latest slot indeed creates more opportunities for the lower-layer devices. Otherwise, it would force the lower-layer devices to send the data earlier.

In our example, the algorithm analyzes reception part of Sensor 0 (Figure 3c). It starts by deciding the best communication slot for Sensor 1 and then for Sensor 2. For Sensor 1, the algorithm chooses t=7.5 since the maximum of possible receptions is 6 and the latest occurrence of 6 is at t=7.5. Slot occupancy is updated: the slot for t=7.5 becomes occupied. Then the algorithm repeats the same decision process for Sensor 2 and chooses t=7, as the latest possible occurrence of value 4 is at t=7. Although t=7.5 is the actual latest maximum, this slot is already occupied, so the communication slot for Sensor 2 is set to t=7. The status of the process after the first iteration is given in Figure 4. The algorithm updates the reception part of the device by defining the unique Rx slot for each concerned data, then it repeats the same process to decide communication slots for remaining data. This iteration continues until all data have a defined Rx slot.

	Time (sec)																		
From	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5
S1	3	6	6	6	5	6	4	4	3	4	4	6	5	6	6	6	3	4	4	4
S2	1	4	4	4	3	4	2	2	1	2	2	4	3	4	4	4	1	2	2	2
slot occupancy	-	-	-	-	-	-	-	-	-	-	-	-	-	-	X	X	-	-	-	-

Figure 4: Searching for Optimal Rx Slot on Sensor 0
Slot occupancy: Free -, Occupied X.

Our optimization algorithm is a greedy algorithm [9]. Greedy algorithms are often used in ad hoc mobile networking to efficiently route packets with the fewest number of hops and the shortest delay possible. Here, the best solution for Rx slots is found locally on the sink device. Chosen Rx slots are propagated to the Tx part of lower-layer neighbors in the network topology. Tx/Rx constraints are then locally updated and the process iterates for those devices. This top-down process continues until all Tx/Rx slots are defined for all devices. The finalized reception part of Sensor 0 is illustrated in Figure 3c. The green slots are the optimal slots found by our algorithm. Finalized schedules of Sensor 1 and Sensor 3 are also presented in Figure 3.

4.4. Building Complete SCO-PATTERN

Once schedules are finalized after searching optimal communication slots for each device, the last step is to extract Tx/Rx events to obtain the complete SCO-PATTERNs.

Example 7. *Here are the* SCO-PATTERN*s for the sensor devices of our example:*

$P_1 = (\{(0, \{A_T\}), (1.5, \{R\}), (2, \{A_T, R\}), (2.5, \{T\}), (4, \{A_T\}), (6, \{A_T\}), (6.5, \{R\}), (7, \{R\}), (7.5, \{T\}), (8, \{A_T\}), (9, \{R\}), (9.5, \{T\})\}, 10)$

$P_2 = (\{(0.5, \{R\}), (1, \{R\}), (1.5, \{T\}), (5, \{R\}), (5.5, \{T\}), (6, \{R\}), (6.5, \{R\}), (7, \{T\})\}, 10)$

$P_3 = (\{(0, \{A_T, A_H\}), (2, \{A_T, T\}), (4, \{A_T\}), (5, \{A_H\}), (6, \{A_T\}), (7, \{T\}), (8, \{A_T\}), (9, \{T\})\}, 10)$

$P_4 = (\{(0, \{A_H\}), (1.5, \{T\}), (5, \{A_H\}), (6.5, \{T\})\}, 10)$

$P_5 = (\{(0, \{A_T, A_H\}), (1, \{T\}), (2, \{A_T\}), (4, \{A_T\}), (5, \{A_H, T\}), (6, \{A_T, T\}), (8, \{A_T\})\}, 10)$

$P_6 = (\{(0, \{A_H\}), (0.5, \{T\}), (5, \{A_H\}), (6.5, \{T\})\}, 10)$

Once SCO-PATTERNs are completed, they are sent by the Gateway to sensor devices to configure them. Sensor devices then execute the given actions periodically, and data flow to the base station where continuous queries are computed. If an application changes its requirements, the configuration process starts again from the beginning.

5. Experimental Platform

In this section, we present the tools used in this study: `SoCQ` Engine as a continuous query engine and `WSNet` simulator for the wireless sensor network environment.

5.1. Continuous Query Engine: SoCQ Engine

As a framework for *PEMS*, we use the `SoCQ` (**S**ervice-**o**riented **C**ontinuous **Q**uery) framework [1]. It takes a data-oriented perspective on the pervasive environment: it provides a unified view and access to the various and heterogeneous resources available in the environment. Pervasive applications can then be created in a declarative way using service-oriented continuous queries over such an environment.

Within the `SoCQ` framework, *XD-relations* (eXtended Dynamic Relations) represent standard relations, that may be updated, or data streams, that continuously produce data. The definition of *XD-relations* can also include virtual attributes and binding patterns that together enable queries to interact with distributed services: service discovery, method invocation, stream

subscription. Queries may be one-shot queries (like standard SQL queries) or continuous queries (with a dynamic result, like a stream). Furthermore, invocations of service methods and subscriptions to service streams can be parameterized.

We illustrate `SoCQ` in the context of Smart Building. Table 1 shows a discovery query, the resulting XD-Relation, then a one-shot query and a continuous query over this XD-Relation. The discovery query `(DISCOVER Services)` searches for sensor services that provide a location, a method to get the current temperature, and a continuous stream of temperature measures. The resulting XD-Relation `TemperatureServices` has one ServiceID attribute (here, the URI of a sensor), a Location, and a virtual attribute for Temperature. When executed, the one-shot query `(SELECT ONCE)` selects services located in a given room and retrieve the current temperature by invoking `getTemperature` method. While executing, the continuous query `(SELECT STREAMING)` subscribes to the `temperature` stream of every discovered service to build a resulting data stream. If new services are discovered and/or some services become unavailable, the continuous query automatically adapts the corresponding stream subscriptions.

Table 1: Example of SoCQ queries for a Smart Building.

CREATE RELATION TemperatureServices (ServiceID SERVICE, Location STRING, Temperature NUMBER VIRTUAL) USING (getTemperature[ServiceId]():(Temperature), temperature[ServiceId]():(Temperature) STREAMING) AS DISCOVER SERVICES PROVIDING PROPERTY Location STRING, METHOD getTemperature ( ) : (NUMBER), STREAM temperature ( ) : (NUMBER)	SELECT * ONCE FROM TemperatureServices WHERE Location = "501.340" USING getTemperature;	SELECT * STREAMING UPON insertion FROM TemperatureServices USING temperature[1];

5.2. WSN Simulator: Modified `WSNet`

In our platform, we integrated the `WSN` simulator `WSNet` [10]. `WSNet` is a modular event-driven simulator, more precisely a discrete event simulator (`DES`). `WSNet` adopts basic functionality of *DES*: in order to avoid simulating every time splice, the time line is split into events and no change is presumed to occur in the system between consecutive events; thus the simulation can directly jump in time from one event to the next. However, as the `PEMS` works on real-time, **we modified the time scheduler** of the `WSNet` simulator to avoid time scheduling difference: we introduced a time factor to run experiments in real-time or n-times faster ($\times 10$, $\times 20 \ldots$).

5.3. Gateway: 3SoSM Gateway

In a declarative monitoring architecture (see Figure 1), the `Gateway` is a technical bridge between two environments: it manages interactions and bidirectional communication between the `PEMS Query Engine` and the `WSN`. We implemented the `3SoSM` principles in a ***3SoSM Gateway***. It is implemented in Java, and interacts with the `SoCQ` engine and `WSNet`. `3SoSM Gateway` has two primary modules: the `Service Manager`, that manages `SoCQ` services representing available sensor devices (real devices or simulated devices); and the `Subscription Manager`, that continuously analyses application requirements to generate new SCO-PATTERNs for sensor devices when required. In a typical scenario, multiple applications launch continuous queries concerning sensors to the *PEMS*. The gateway manages required parameterized stream requests and, according to the `3SoSM` optimization process, generates optimal sensor configurations. With our integrated prototype, we can execute multiple `SoCQ` queries that dynamically subscribe to streams provided by `WSNet` sensor devices. The `Gateway` transparently optimizes the subscriptions and configures the `WSNet` sensor devices with SCO-PATTERNs.

6. Experiments

6.1. Experimental Setup

The simulations are performed using the `modified WSNet`. We simulate one part of the topology of our physical platform `SoCQ4Home` deployed in our LIRIS laboratory: 70 sensor devices are located at specific positions over a floor of the building (10m $\times$ 60m $\times$ 4m). The topology is illustrated in Figure 5. The deployed sensor devices have fixed positions during the simulation and we consider that they have enough energy until the end of the simulation. During the simulation process, the initial energy level of sensor devices is set on purpose to emphasize the lifetime difference between both cases.

We adopt known pervasive environment communication protocol `Zigbee IEEE 802.15.4`, simulated with a `UDG` propagation model (which is a strong simplification for building environments, as it does not take into account environmental effects), 35m transmission range, and basic radio module states for devices (idle, active, sleep, transmission, reception). Calculation of energy consumption is based on CPU and radio components, adopted from [7]: $\mathcal{E}_{device} = \mathcal{E}_{CPU} + \mathcal{E}_{radio} = \mathcal{E}_{CPU} + (\mathcal{E}_{sleep} + \mathcal{E}_{TX} + \mathcal{E}_{RX}), with\ \mathcal{E}_{component/mode} =$

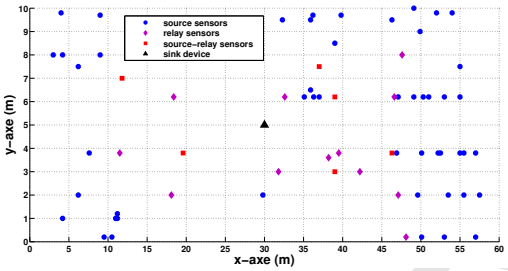


Figure 5: Topology of the network
(Sensor roles are set based on the Application 1 of the scenario given below).

$I_{mode}.V_{drain}. \triangle t_{mode}$. This model is a well-known energy consumption calculation model preferred by many researchers. For instance the paper [11] proposes a temperature-aware task mapping approach for the mapping of multi-application to NoC-based many-core systems by balancing workloads among cores. In our study, we adopt a similar approach to calculate the energy consumption of all devices. We implemented sensor devices for `WSNet` so that they can reconfigure themselves when they receive a new SCO-Pattern packet, and so that `WSNet` can monitor their simulated energy consumption while executing sensor actions.

6.2. Experimental Scenario

To evaluate our approach, we compare 2 types of architecture:
1) an architecture with basic duty-cycle `WSN` devices, requiring a static configuration predefined according to scenario requirements;
2) an architecture with `3SoSM` approach where `WSN` devices can be dynamically configured with SCO-PATTERNs generated by the `3SoSM Gateway`.

We design four scenarios: "Temperature of Occupied Rooms", "Comfort Temperature Range", "Room occupancy based on CO_2 emission" and "Luminosity of rooms". Each scenario is performed during one day (1440min). Here, we present only one of the performed scenarios due to space limitations.

Scenario "Temperature of Occupied Rooms": One of the features of a typical smart building is monitoring temperatures and occupancy of the environment. Here we suppose that occupancy sensor devices are not always active but has their own duty cycle for detecting the motion.
In this scenario, there exists two applications:
Application 1: Monitor the temperature of each room with a data acquisition every 15 sec and a latency of 60 sec.

Application 2: Monitor temperature of occupied rooms with a data acquisition every 1 sec and a latency of 4 sec.

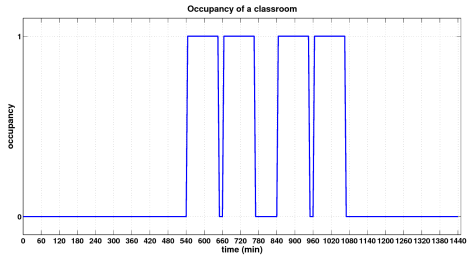
Here, *Application 1* requests the temperature from all temperature sensor devices with given application requirements. *Application 2* also requests a temperature with a special constraint: it intends to track the temperature only for occupied rooms. For this application requirement, occupancy and temperature sensor devices are needed.

For the duty-cycle approach, we adopt a static configuration. In this case, we consider the worst case situation: occupation may be true at any location during the simulation. Based on this constraint, sensor devices are configured with: $p^{acq} = 1sec, p^{tx} = 4sec$ (periodic transmission)

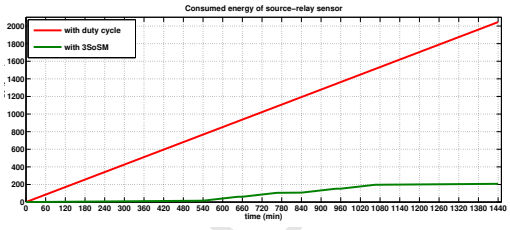
In our approach, we propose a dynamic sensor configuration based on the real-time context. To respond to *Application 1* and *Application 2*, sets of `SoCQ` queries are implemented. For *Application 2*, from the occupancy stream, we extract the location attribute of the occupied rooms. Then, we list the temperature sensor devices that are located in these rooms and implement a new stream query to subscribe to those temperature services with given application requirements. Since these are continuous queries, once a room is not occupied any more or an unoccupied room become occupied, lists are refreshed and new subscriptions or unsubscriptions are handled in real-time. Then, the `3SoSM Gateway` creates schedules for each sensor device and generates SCO-PATTERNs to configure sensor devices accordingly.

Briefly, *Application 1* receives temperature data from relevant devices under every condition. However, *Application 2* has a remarkable condition: at least one room should be occupied so that it starts to receive temperature data. Here, there are two subscriptions to the same set of temperature sensor devices located in occupied rooms with different application requirements. During the simulation (one day), energy consumption (in terms of joules) of every sensor are monitored periodically and logged.

Besides, three more scenarios are performed. Scenario 2 consists of tracking temperature of each room and requesting more frequently temperatures for rooms that are out of the current comfort temperature range. Scenario 3 is based on the occupancy information retrieved from CO_2 emission detected in the room and scenario 4 is related to the luminosity of a classroom. For these scenarios, two applications with different application requirements are executed during a day.



(a) Occupancy of a room (24h).



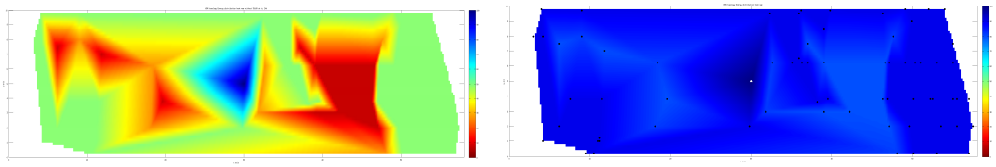
(b) Average consumed energy (in joule) of a source-relay device (24h). Upper line for duty cycling, lower line for `3SoSM`.

Figure 6: Occupancy information and Energy consumption of the most significant device type: source-relay device with duty cycling and with our approach `3SoSM`.

7. Results and Discussion

During the experiments, energy consumption of each sensor device is monitored. Figure 6 shows the average energy consumption of source-relay devices. As a remark, the sink device is not considered as a part of this energy consumption calculation. Here, we present the results of the first scenario in detail for one day (1440min). Firstly, the room is not occupied (see Figure 6a), hence the `3SoSM Gateway` creates a schedule and configures the relevant sensor devices according to that situation. Once the room is detected as occupied, the `3SoSM Gateway` re-creates a schedule and re-configures the relevant sensor devices. In fact, the `3SoSM Gateway` creates a new schedule and configures sensor devices to adapt to each contextual change in the environment. Thus, adapting the system to the dynamic context avoids sensor devices from unnecessary data acquisitions and transmissions. Hence, the economy of energy can be achieved with this context-awareness, as shown in Figure 6b.

Evolution of the energy consumption for the `WSN` can be observed with heat maps as well. Figure 7 represents the heat map of the network based on the energy level of the sensor devices at the end of the simulation (t=24h), with duty-cycle method (Figure 7a) and with our 3SoSM approach (Figure 7b), according to the given topology (see Figure 5). With the settings of the scenario "Temperature of Occupied Rooms", we observe that our approach `3SoSM` enhances the lifetime of the sensor devices significantly. For source-relay sensor devices, with given application requirements, `3SoSM` extends their lifetime by 8.9 times on average.



(a) Heatmap of remaining energy at t=24h with duty cycle.

(b) Heatmap of remaining energy at t=24h with 3SoSM.

Figure 7: Heatmap of remaining energy with duty-cycle and with 3SoSM.

Table 2: Summary of the Experiments.

Scenarios	Application Parameters				Lifetime (day)	
	Application 1		Application 2		Duty cycling	3SoSM Approach
	p^{acq} (sec)	λ (sec)	p^{acq} (sec)	λ (sec)		
Temperature of Occupied Rooms	15	60	1	4	15	146
Comfort Temperature Range	20	60	10	20	9	90
Room Occupancy based on CO2	30	120	15	60	11	95
Luminosity of Rooms	30	100	20	80	10	34

By applying our approach to given application requirements, we optimize the energy consumption and reduce the unnecessary communication cost. Obtained results on energy saving and lifetime extension depend on the application requirements and the context of the application such as temperature and presence. For the current settings, the most unfavorable case is the situation where every room is occupied during the entire simulation time. Even in that case, `3SoSM` achieves a concrete energy enhancement since the regular duty-cycle approach does not benefit from latency requirement which provides a tangible energy saving by allowing grouped transmission of multiple acquired data.

Besides, the details of the other performed scenarios are summarized in Table 2 for each scenario, application requirements are given and the average lifetime of source-relay sensor devices are presented. We also obtain a significant energy enhancement and a lifetime extension by avoiding unnecessary communication.

However, balancing the workload of the network over the sensor devices like [11] introduces, advanced clustering mechanism like [5] presents, may further enhance the network lifetime and the obtained results. [5] presents that with a clustering mechanism and selecting cluster headers in an optimal way may extend the lifetime of the network better than any other cluster

based networks. With more populated topologies and with massive subscription requests, a discussion about the optimization process appears: in which conditions our approach may fail to find the optimal slots on the schedules (since a slot is reserved for a single device)? In these cases, granularity should be decreased to have enough slots for the sensor devices.

8. Related Works

WSN are related to several different research domains such as pervasive environment, and smart building systems. In this paper, we give a brief summary of the following research areas that are concerned by our study: energy issues in WSN, sensor network query processing (SNQP), sensor-based Smart Building Management Systems (SBMS).

Energy Issue in WSN. Wireless devices bring important constraints such as limited battery lifetime. [12] classifies the existing studies on energy consumption of wireless sensor devices into 3 subgroups: duty-cycle, data-driven and mobility-based approaches. Our sensor configuration based subscription management approach benefits from scheduled rendezvous of duty-cycling and adaptive sampling/transmission of data-driven approaches.

Sensor Network Query Processing. The main functionality of Sensor Network Query Processors (SNQP) is to handle continuous queries and sensor data streams [13]. [14, 15] propose an adaptive in-network aggregation operator ADAGA for query processing on sensor devices in order to filter and reduce the volume of sensor data. The main functionality of ADAGA is to regulate/adjust sensor activities based on energy levels and memory usage of sensor devices. As we propose in our own study, ADAGA is also capable of processing query parameters but the study does not tackle multi-application context and multi-modality of wireless devices as proposed in our own work.

We use the Service-oriented Continuous Query (SoCQ) Engine [1] to manage sensor data streams. SoCQ has a strong advantage from its rivals (especially SNEE [8]) as it supports multi-application contexts. However, it supports neither energy-awareness nor real-time sensor configuration. Our approach provides these features and improvement to the SoCQ Engine.

Among available technologies for SNQP, [8] is the closest study to our approach. This study deal with the conflict between the quality of service and sensor acquisition/transmission settings. Authors propose an optimization of application requests and the generation of a query execution plan. Generated

execution plan provides a global view of the network and a schedule that shows when to execute a sensor action (sleep, listen, receive/send a packet). In terms of sensor scheduling and parametrization of a query, this approach is highly close to ours. Still, query execution plan covers a single query with a single expectation.

Sensor-based Building Systems. Some other studies focus on the design and data management aspects as well. Most of those approaches are proposed for a specific application and adopt a static configuration for sensor devices. Sensor-actuator based environment management systems are also studied in this area. [16] mentions that sensor-actuator interactions has a significant effect on efficient building monitoring. Moreover, the study points out that estimation of our daily activities has a crucial role on managing heterogeneous sensor devices in the environment and provides energy saving and longer lifespans. [5] indicates that clustering and energy efficiency have been considered in wide area of `WSN` applications and propose a mechanism in which clusters are dynamically built up by neighbor nodes, to save energy and prolong the network lifetime.

[17, 18] are the closest studies to ours. [17] presents intelligent building architecture based on a self-adapting intelligent gateway. [18] presents self-adapting algorithms for context-aware systems. These studies propose dynamic system management while processing user preferences. However, these studies are bounded by predefined building applications and the relation between application requirements and sensor configuration is not established. Besides, these approaches do not benefit from the potential reconfiguration of acquisition and transmission frequencies: sensor configuration stays static during the system lifetime. Moreover, energy consumption within the `WSN` is not considered as a major issue.

[19] is also close to our approach. Author implicates event-triggered dynamic sensor configuration. The study proposes a ZigBee-based intelligent self-adjusting sensor platform (`ZiSAS`) that can autonomously reconfigure middleware, network topology, sensor density, and sensing rate based on the environmental situation. However, unlike our proposition, user preferences, application requirements, query mechanism or data stream processing are not handled.

Overview of most known existing studies on Smart Building applications is summarized in Table 3. For the categorization of these studies, we benefit from the key functionalities of a Smart Building Management System

Table 3: Overview of most known existing studies on Smart Building applications.

Study	Multi-App	Dynamic User Config.	Context-Awareness	Real-Time Sensor Config.	Energy-Aware Monitoring	Query Engine	Experimentation
Doukas et al. [21]	–	–	√	–	–	–	Real Testbed
Chen et al. [22]	–	–	√	–	–	–	Simulation
Byun et al. [19]	–	–	√	√	√	–	Real Testbed
Xiang et al. [23]	–	–	√	–	–	√	Real Testbed
Rutishauser et al. [24]	–	–	√	–	–	√	Real Testbed
Servigne et al. [25]	–	–	–	–	–	–	Real Testbed
Mamidi et al. [26]	–	–	√	–	–	–	Real Testbed
Kailas et al. [27]	–	–	√	–	–	–	Real Testbed
Agarwal et al. [28]	–	–	√	–	–	–	Real Testbed
Preisel et al. [29]	–	–	–	–	√	–	Real Testbed
Schor et al. [30]	–	–	√	–	√	–	Real Testbed
Li et al. [31]	–	–	√	–	√	–	Real testbed
3SoSM	√	√	√	√	√	√	Real Testbed and Simulation

(SBMS) [20]. The last line of the table is allocated for our approach 3SoSM. Our research motivation is that the existing approaches are mostly WSN-level techniques that do not tackle real-time dynamic interactions between application continuous queries and the physical environment, in a multi-application context where sensor devices are multi-modal and requirements are dynamic. Hence, this lack creates a gap between the computing environment and the physical environment, that can be both managed by pervasive applications.

To address this gap and propose a clear solution for multi-application monitoring systems, we design a sustainable multi-application monitoring system architecture for pervasive environments that collects application requirements for sensor data streams and optimizes them into sensor configurations. We propose energy-aware dynamic sensor device re-configuration (as a result of an optimization process) to lower energy consumption while fulfilling real-time application requirements. Compared to existing studies, our approach 3SoSM provides a multi-application mechanism and allows dynamicity for user configurations and network-aware sensor management while optimizing the sensor communication for enhancing energy consumption by real-time sensor configuration.

9. Conclusion

In this paper, we focus on one of the major challenges of pervasive monitoring systems like smart buildings: how to optimize/reduce the energy consumption of the monitoring architecture itself while managing sensor data streams. Existing studies do not tackle the energy consumption of the monitoring system and commonly adopt static configurations for sensor devices.

Since application requirements are dynamic with the context, a dynamic sensor configuration can be a suitable option to solve this problem. We introduce a sustainable declarative monitoring architecture where we adopt the PEMS principles to separate application development and optimization of device interactions. We present our approach 3SoSM that is based on a WSN scheduling mechanism for sensor actions. We propose an optimization algorithm to find the optimal communication time slots. As an outcome, an optimized schedule generates SCO-PATTERNs to configure each device. We introduce our implementation 3SoSM Gateway that supports the optimization process for multiple parameterized subscriptions from dynamic application requirements. Our approach is validated by the experiments using the SoCQ Engine and a modified WSNet simulator. Moreover, our approach gives opportunities to use real testbed data and simulation data which is not so common in the pervasive environment research domain. Smart building applications are our target application, however, this approach can be applicable to other data-centric pervasive environments as well [2].

Conflict of Interest

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] Y. Gripay, F. Laforest, J.-M. Petit, Socq: A framework for pervasive environments, in: ISPAN'09, IEEE, 2009, pp. 154–159.

[2] S. Surdu, Y. Gripay, V.-M. Scuturici, J.-M. Petit, P-bench: benchmarking in data-centric pervasive application development, in: Transactions on Large-Scale Data-and Knowledge-Centered Systems XI, Springer, 2013, pp. 51–75.

[3] S. Servigne, Y. Gripay, O. Pinarer, J. Samuel, A. Ozgovde, J. Jay, Heterogeneous sensor data exploration and sustainable declarative monitoring architecture: Application to smart building, ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci IV-4/W1 (2016) 97–104.

[4] C. Perera, A. Zaslavsky, M. Compton, P. Christen, D. Georgakopoulos, Semantic-driven configuration of internet of things middleware, in: SKG'13, IEEE, 2013, pp. 66–73.

[5] P. G. V. Naranjo, M. Shojafar, H. Mostafaei, Z. Pooranian, E. Baccarelli, P-sep: a prolong stable election routing algorithm for energy-limited heterogeneous fog-supported wireless sensor networks, The Journal of Supercomputing 73 (2) (2017) 733–755.

[6] O. Pinarer, Y. Gripay, S. Servigne, A. Ozgovde, Energy enhancement of multi-application monitoring systems for smart buildings, in: International Conference on Advanced Information Systems Engineering, Springer, 2016, pp. 131–142.

[7] O. Pinarer, A. Ozgovde, Improving the energy efficiency of wearable computing units using on sensor fifo memory, International Journal of e-Education, e-Business, e-Management and e-Learning 5 (2) (2015) 105.

[8] I. Galpin, A. A. Fernandes, N. W. Paton, Qos-aware optimization of sensor network queries, The VLDB Journal 22 (4) (2013) 495–517.

[9] I. Cardei, M. Cardei, Energy-efficient connected-coverage in wireless sensor networks, International Journal of Sensor Networks 3 (3) (2008) 201–210.

[10] Wsnet / worldsens simulator, `http://wsnet.gforge.inria.fr/`.

[11] S. Cao, Z. Salcic, Z. Li, S. Wei, Y. Ding, Temperature-aware multi-application mapping on network-on-chip based many-core systems, Microprocessors and Microsystems 46 (2016) 149–160.

[12] G. Anastasi, M. Conti, M. Di Francesco, A. Passarella, Energy conservation in wireless sensor networks: A survey, Ad hoc networks 7 (3) (2009) 537–568.

[13] F. Jabeen, S. Nawaz, In-network wireless sensor network query processors: State of the art, challenges and future directions, Information Fusion 25 (2015) 1–15.

[14] A. Brayner, A. L. Coelho, K. Marinho, R. Holanda, W. Castro, On query processing in wireless sensor networks using classes of quality of queries, Information Fusion 15 (2014) 44–55.

[15] A. Brayner, A. Lopes, D. Meira, R. Vasconcelos, R. Menezes, An adaptive in-network aggregation operator for query processing in wsn, Journal of Systems and Software 81 (3) (2008) 328–342.

[16] F. Viani, F. Robol, A. Polo, P. Rocca, G. Oliveri, A. Massa, Wireless architectures for heterogeneous sensing in smart home applications: Concepts and real implementation, Proceedings of the IEEE 101 (11) (2013) 2381–2396.

[17] J. Byun, S. Park, Development of a self-adapting intelligent system for building energy saving and context-aware smart services, IEEE Transactions on Consumer Electronics 57 (1) (2011) 90–98.

[18] T. Cioara, I. Anghel, I. Salomie, M. Dinsoreanu, G. Copil, D. Moldovan, A self-adapting algorithm for context aware systems, in: RoEduNet'10, IEEE, 2010, pp. 374–379.

[19] J. Byun, B. Jeon, J. Noh, Y. Kim, S. Park, An intelligent self-adjusting sensor for smart home services based on zigbee communications, IEEE Transactions on Consumer Electronics 58 (3) (2012) 794–802.

[20] T. A. Nguyen, M. Aiello, Energy intelligent buildings based on user activity: A survey, Energy and buildings 56 (2013) 244–257.

[21] H. Doukas, K. D. Patlitzianas, K. Iatropoulos, J. Psarras, Intelligent building energy management system using rule sets, Journal of Building and Environment 42 (10) (2007) 3562–3569.

[22] H. Chen, P. Chou, S. Duri, H. Lei, J. Reason, The design and implementation of a smart building control system, in: ICEBE'09, Citeseer, 2009.

[23] S. Xiang, W. Wu, K.-L. Tan, Optimizing multiple data acquisition queries in sparse mobile sensor networks, in: MDM'12, IEEE, 2012, pp. 137–146.

[24] U. Rutishauser, J. Joller, R. Douglas, Control and learning of ambience by an intelligent building, IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans 35 (1) (2005) 121–132.

[25] S. Servigne, Y. Gripay, J.-M. Deleuil, C. Nguyen, J. Jay, O. Cavadenti, R. Mebrouk, Data science approach for a cross-disciplinary understanding of urban phenomena: Application to energy efficiency of buildings, Procedia Engineering 115 (2015) 45–52.

[26] S. Mamidi, Y.-H. Chang, R. Maheswaran, Improving building energy efficiency with a network of sensing, learning and prediction agents, in: Proceedings of the 11th AAMAS, Vol. 1, IFAAMAS'12, 2012, pp. 45–52.

[27] A. Kailas, V. Cecchi, A. Mukherjee, A survey of communications and networking technologies for energy management in buildings and home automation, Journal of Computer Networks and Communications 2012.

[28] Y. Agarwal, B. Balaji, R. Gupta, J. Lyles, M. Wei, T. Weng, Occupancy-driven energy management for smart building automation, Journal of Power 50 (100) (2010) 150.

[29] M. Preisel, A. Diaz, W. Wimmer, Energy consumption of smart meters, Journal of Information and Communication Technologies (2013) 37.

[30] L. Schor, P. Sommer, R. Wattenhofer, Towards a zero-configuration wireless sensor network architecture for smart buildings, in: BuildSys'09, ACM, 2009, pp. 31–36.

[31] C. Li, F. Meggers, M. Li, J. Sundaravaradan, F. Xue, H. Lim, A. Schlueter, Bubblesense: wireless sensor network based intelligent building monitoring, Proceedings of the ICT4S'13 (2013) 159–166.

FOLIO ADMINISTRATIF

THESE DE L'UNIVERSITE DE LYON OPEREE AU SEIN DE L'INSA LYON

NOM : **PINARER** DATE de SOUTENANCE : **15.12.2017**

Prénoms : **OZGUN**

TITRE : **Sustainable Declarative Monitoring Architecture: Energy optimization of interactions between application service oriented queries and wireless sensor devices - Application to Smart Buildings**

NATURE : Doctorat Numéro d'ordre : 2017LYSEI126

Ecole doctorale : **Ecole Doctorale ED 512**
Informatique et Mathématiques de Lyon (INFOMATHS)

Spécialité : **Informatique**

RESUME : La dernière décennie a montré un intérêt croissant pour les bâtiments intelligents. Les bâtiments traditionnels sont les principaux consommateurs d'une partie importante des ressources énergétiques, d'où le besoin de bâtiments intelligents a alors émergé. Ces nouveaux bâtiments doivent être conçus selon des normes de construction durables pour consommer moins. Ces bâtiments intelligents sont devenus l'un des principaux domaines d'application des environnements pervasifs. En effet, une infrastructure basique de construction de bâtiment intelligent se compose notamment d'un ensemble de capteurs sans fil. Les capteurs basiques permettent l'acquisition, la transmission et la réception de données. La consommation d'énergie élevée de l'ensemble de ces appareils est un des problèmes les plus difficiles et fait donc l'objet d'études dans ce domaine de la recherche. Les capteurs sont autonomes en termes d'énergie. Etant donné que la consommation d'énergie a un fort impact sur la durée de vie du service, il existe plusieurs approches dans la littérature. Cependant, les approches existantes sont souvent adaptées à une seule application de surveillance et reposent sur des configurations statiques pour les capteurs. Dans cette thèse, nous contribuons à la définition d'une architecture de surveillance déclaratif durable par l'optimisation énergétique des interactions entre requêtes applicative orientées service et réseau de capteurs sans fil. Nous avons choisi le bâtiment intelligent comme cas d'application et nous étudions donc un système de surveillance d'un bâtiment intelligent. Du point de vue logiciel, un système de surveillance peut être défini comme un ensemble d'applications qui exploitent les mesures des capteurs en temps réel. Ces applications sont exprimées dans un langage déclaratif sous la forme de requêtes continues sur les flux de données des capteurs. Par conséquent, un système de multi-applications nécessite la gestion de plusieurs demandes de flux de données suivant différentes fréquences d'acq/tx de données pour le même capteur sans fil, avec des exigences dynamiques requises par les applications. Comme une configuration statique ne peut pas optimiser la consommation d'énergie du système, nous proposons une approche intitulée Smart-Service Stream-oriented Sensor Management (3SoSM) afin d'optimiser les interactions entre les exigences des applications et l'environnement des capteurs sans fil, en temps réel. 3SoSM offre une configuration dynamique des capteurs pour réduire la consommation d'énergie tout en satisfaisant les exigences des applications en temps réel. Nous avons conduit un ensemble d'expérimentations effectuées avec un simulateur de réseau de capteurs sans fil qui ont permis de valider notre approche quant à l'optimisation de la consommation d'énergie des capteurs, et donc l'augmentation de la durée de vie de ces capteurs, en réduisant notamment les communications non nécessaires.

MOTS-CLÉS : environnement pervasif, réseau de capteurs sans fil, gestion de flux de données, traitement de requête continue

Laboratoire de recherche : INSA Lyon, LIRIS

Directeur de thèse : BASKURT, Atilla Professeur INSA Lyon

Présidente de jury : LIBOUREL, Thérèse Professeur Emérite, Université de Montpellier

Composition du jury :

BABAU, Jean-Philippe	Professeur, Université de Brest	Rapporteur
BASKURT, Atilla	Professeur, INSA Lyon	Directeur de thèse
BOUJU, Alain	Maître de Conférences HDR, Université de La Rochelle	Examinateur
GRIPAY, Yann	Maître de Conférences, INSA Lyon	Examinateur
PINET, François	Directeur de Recherche, IRSTEA Clermont-Ferrand	Rapporteur
SERVIGNE, Sylvie	Maître de Conférences, INSA Lyon	Examinatrice