Intelligent flood adaptative contex-aware system

Jie Sun

N° d'ordre :
EDSPIC :

Université Clermont Auvergne

École doctorale

Sciences Pour l'Ingénieur de Clermont-Ferrand

Thèse présentée par

Jie SUN

Pour obtenir le grade de

Docteur d'Université

Discipline

Informatique

Titre de la thèse

Intelligent Flood Adaptive Context-aware System

Soutenue publiquement le 23/10/2017

Membres du jury

DR Jérôme EUZENAT (Inria Grenoble Rhône-Alpes)	Président
PR Nadine PIAT (ENSMM Besançon)	Rapporteur
PR Robert LAURINI (KSI Chicago)	Rapporteur
PR Kun-Mean HOU (LIMOS)	Directeur de thèse
CR Catherine ROUSSEY (Irstea Clermont-Ferrand)	Encadrant
IR Gil DE SOUSA (Irstea Clermont-Ferrand)	Encadrant
IR HDR Jean-Pierre CHANET (Irstea Clermont-Ferrand)	Encadrant (invité)

This dissertation is dedicated to

To my parents, Bingqin SUN and Xiuying MA for their love and support,

And to my thesis advisors, Prof. Kun-Mean HOU, Dr. Gil DE SOUSA, Dr. Catherine ROUSSEY and Dr. Jean-Pierre CHANET for their vision, encouragement, support and guidance.

Acknowledgements

My education at Université Clermont Auvergne has been a journey full of memories, and I cannot thank more to the people who helped me during the five-year study in France.

Firstly, I would like to thank my four advisors, Prof. Kun-Mean HOU, Dr. Gil DE SOUSA, Dr. Catherine ROUSSEY and Dr. Jean-Pierre CHANET, who gave me a lot of help in this foreign country. Thank you for sharing with me your knowledge, vision and passion for research. Thank you for believing in my ability and self-motivation all the time. I learned so much from you and I believe this experience will be great fortune for me and my future career.

I also would like to thank Prof. Jérôme EUZENAT, Prof. Nadine PIAT and Prof. Robert LAURINI for accepting to be part of my jury, for their valuable time and feedback.

Many thanks for the grant of the French Auvergne-Rhône-Alpes region and, the grant of the European Regional Development Fund (ERDF), which have financial supports for this thesis. My advisors and I would like to thank the team of the Orgeval observatory for their help. I feel lucky that Irstea and LIMOS has provided me with the opportunity to become a researcher.

Big thanks to all my colleagues from Irstea and LIMOS. It was a great pleasure to share this five-year time with you. I am very thankful to Dr. Hongling SHI from LIMOS for his suggestions and help. I am very grateful to Dr. Jian-jin LI and Dr. Christophe De Vaulx for their kind help and precious advices. I also would like to express many thanks to present and former team members of SMIR group for their cooperation, help, countless discussion and idea exchange. Thanks to Yibo CHEN, Bin TIAN, Peng ZHOU, Peter DIAO, Jean CONNIER and Qing XU. Without your help, sharing knowledge and experiences, I cannot achieve any results alone, and the time with you let me believe only team work can overcome the challenges in the real-world applied research.

I would like to thank my parents for their love and support. Thank you for encouraging me to pursue my dreams.

Résumé

A l'avenir, les domaines de l'agriculture et de l'environnement vont pouvoir bénéficier d'informations en masse de qualité croissante, obtenues à partir de mesures hétérogènes acquises par des capteurs (ou sondes). Ces capteurs utilisent de plus en plus des technologies sans fil pour transmettre les données collectées. Ces capteurs sans fil disposent de ressources limitées dont la principale est l'énergie qui est consommée majoritairement par la communication. La gestion de ces ressources dans le cadre des réseaux de capteurs sans fil (RCSF) restent encore aujourd'hui une problématique importante. Les données collectées par les capteurs sans fil alimentent des outils d'aide à la décision (OAD). Dans cette thèse, nous nous intéressons spécifiquement aux systèmes sensibles et adaptatifs au contexte basés sur un RCSF et un OAD. Ces systèmes doivent intégrer plusieurs flux de données issus de capteurs hétérogènes pour déterminer l'état du phénomène naturel observé. L'ensemble des données collectées compose le contexte. Des techniques de raisonnement sont appliquées sur le contexte pour l'enrichir et déduire l'état du phénomène naturel. Ensuite, Ces systèmes proposent des actions d'adaptation basées sur l'état du phénomène. Pour créer un service complet d'adaptation au contexte, des informations sur les capteurs sans fil eux-mêmes, comme leur niveau d'énergie, sont également intégrer au contexte. Par conséquent, la combinaison des décisions et actions communes basées sur le contexte avec une meilleure utilisation des ressources limitées du RCSF est la problématique principale de cette thèse. Ainsi, nous proposons une formalisation pour la conception et la mise en place de systèmes sensibles et adaptatifs au contexte pour l'agriculture et l'environnement. Cette formalisation se base sur des ontologies pour intégrer les différents flux de données issus de capteurs et modéliser le contexte. Un moteur d'inférence à base de règles permet d'enrichir le contexte. Les systèmes obtenus permettent à la fois de préserver le fonctionnement des capteurs et de maintenir le niveau de qualité de service (QdS) des données collectées en accord avec le phénomène étudié. Pour illustrer notre proposition, un cas d'utilisation environnementale complexe, l'étude des inondations dans un bassin hydrographique, en s'appuyant sur un RCSF pour la collecte de données, est présentée. Cette thèse a produit un logiciel de simulation des systèmes sensibles au contexte. Ce système intègre un système de simulation multi-agents (JADE) avec un moteur d'inférence à base de règles (Jess). La modélisation du contexte a été réalisée en réutilisant des ontologies existantes. Les données d'observations d'un bassin versant, l'Orgeval, ont permis d'alimenter le système de simulation. Un modèle énergétique de capteur sans fil a aussi été développé pour alimenter le système de simulation. Plusieurs systèmes sensibles au contexte ont été simulés pour montrer l'impact de l'adaptation sur la durée de vie du RCSF.

Mots-clefs : ontologies, inférences à base de règles, formalisation, mesures et données hétérogènes, intégration de données issues de capteurs, RCSF, ressources limitées, OAD, systèmes sensibles et adaptatifs au contexte, QdS, agriculture, environnement.

Abstract

In the future, agriculture and environment will rely on huge quantities of information with increasing quality obtained through sensors heterogeneous measurements. These sensors use more and more the wireless communication technologies to transmit the collected data. These sensors have limited resources, mainly the energy which is mostly consumed by the communication activity. For the wireless sensor networks (WSN), the management of these resources is still today an important issue. The data collected by these sensors are used in decision support systems (DSS). In this dissertation, we focus on a daptive context-aware systems based on a WSN and a DSS. These systems have to integrate many data streams, generated by heterogeneous sensors, in order to establish the state of a natural phenomenon. All the collected data compose the context. Reasoning techniques are applied to the context in order to enrich it and to infer the state of the natural phenomenon. Then, these systems provide adaptation actions based on t his phenomenon state. To generate a full adaptive context-aware service, information about the sensors themselves, such as their energy level, are also taken into account in the context. Therefore, defining common decisions and actions based on the context, with a better use of the limited resources of the WSN, is the main issue addressed on t his dissertation. Thus, a formalization for the design and the deployment of adaptive context-aware systems for agriculture and environment is proposed. This formalization relies on ontologies in order to integrate different data streams from sensors and to model the context. A rule-based inference engine is used to provide new information to enrich this context. The obtained systems allow both to preserve the resources of the wireless sensors and to maintain the required quality of service (QoS) level of the collected data according to the studied phenomenon. To illustrate our proposal, a complex environmental use case, the study of floods in a watershed, relying on a WSN, is described. A simulator for context-aware systems was developed during this PhD thesis. This simulator integrates a multi-agent system (JADE) and a rule engine (Jess). The context modeling is based on existing ontologies. Real data from a watershed, l'Orgeval, have been used as input data of the simulator. An energy model of wireless sensor has also been designed to provide input data for the simulator. Different context-aware systems have been simulated to evaluate the impact of the adaptation on the WSN lifetime.

Keywords: ontologies, rule-based inferences, formalization, heterogeneous measurements and data, sensors data streams integration, WSN, limited resources, DSS, adaptive context-aware systems, QoS, agriculture, environment.

Table of contents

List of Figures

List of Tables

Glossary

The items in this glossary are listed in alphabetical order, with any symbol and numeric appearing at the end.

Agent action in JADE: An element contained in message exchanged between agents. (See the definition at page 109)

Average slope of water flow rate per communication interval: The average of all the slope values acquired by a w atercourse node during a communication interval. This aggregation value is send to DSS by water flow nodes. (See the first appearance at page 72)

Environmental context: Any information that can be used to characterize the situation of an environmental entity. (See the first time appearance at page 20)

Flood context: Any information that can be used to characterize the situation of a flood entity. (See the first time appearance at page 58)

Concept in JADE: Concepts are entities with a complex structure that can be defined in terms of slots in JADE. (See the definition at page 109)

Context: "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." (Abowd et al., 1999) (See page 26)

Our definition of context: A set of entities characterized by their states, plus all information that can help to derive any state changes of these entities. (See page 46).

Context acquisition process: A process that acquires the context from various sources. (See the definition at page 30)

Context adaptation process: A process that triggers actions based on context changes. (See the definition at page 33)

Context distribution process: A process that distributes the high-level context to the consumers who are interested in context. (See the definition at page 33)

Context modeling process: A process that models the collected data. (See the definition at page 33)

Context reasoning process: A process that derives high-level context from low-level context. (See the definition at page 33)

Entity of interest (EI): Entity whose characterization is obtained from one or many other entities and required by the application. (See the definition at page 46)

Environmental Entity of Interest (EEI): entities of interest whose characterization is obtained from one or many other EOE. (See the definition at page 49)

Environmental Observable Entity (EOE): observable entities dedicated to the observation of an environmental phenomenon. (See the definition at page 49)

Flood entity (F): the EI of our environmental application. **(**See the definition at page 60.)

High-level context: "Context which represents the aggregated state of a d evice, is inferred by multiple low-level local contexts, for example, the status of a WSN." (Liu et al., 2013) (See page 27)

Our definition of High-level context: Context which only contains the qualitative data (e.g. protocol states in WSN, the status of a WSN, etc.). (See page 28)

High-level environmental context: The high-level context of our environmental application. **(**See the first time appearance at page 20)

High-level WSN context: The high-level context of our network application. (See the first time appearance at page 51)

Class: A data type in object-oriented programming that consists of a group of objects with the same properties and behaviors and that is arranged in a hierarchy with other such data types. (See the first time appearance at page 38)

Instance: A concrete occurrence of any object. (See the first time appearance at page 36)

Last water flow rate: Last measurement of water flow rate acquired by of watercourse node during a communication interval. This measurement is send to DSS. (See the first appearance at page 73)

Low-level context: **"**context about subcomponents of a device, such as protocol states and sensor readings in WSN." (Liu et al., 2013) (See page 27);

Our definition of context: Context which only contains the quantitative data (e.g. sensor readings in WSN). (See page 28)

Low-level environmental context: The low-level context of our environmental application. **(**See the first time appearance at page 20)

Low-level WSN context: The low-level context of our network application. **(**See the first time appearance at page 51)

Max slope of water flow rate per communication interval: The maximum of all the slope values acquired by a watercourse node during a communication interval. This aggregation value is send to DSS by water flow nodes. (See the first appearance at page 73)

Max slope of water flow rate per fixed time interval: The maximum of all the slope values acquired by a watercourse node during a fixed time interval. This aggregation value is send to DSS by water flow nodes. (See the first appearance at page 74)

Max slope of water flow rate per 6h: The maximum of all the slope values acquired by a watercourse node during the last 6 hours. This aggregation value is send to DSS by water flow nodes. (See the first appearance at page 90)

Max water flow rate: the maximum of all water flow rate measurements acquired by of watercourse node during a communication interval. This measurement is send to DSS. (See the first appearance at page 73)

Observable entity (OE): Entity that is directly observed by sensors. (See the definition at page 47)

Ontology: Complex schema in Semantic Web technologies. (W3C, 2017) (See the definition at page 111)

Ontology in JADE: a vocabulary composed only of "concepts", "predicates" and "agent actions". JADE ontology is a set of element schemas. (See the definition at page 110)

Ontology Web Language (OWL): language used to describe complex schema in Semantic Web technologies. (OWL, 2017) (See the definition at page 111)

Outlet: The place that the waters of the tributary stream join another body of water, such as a river, a lake or an ocean. (See the definition at page 60)

Outlet entity (O): One OE of our environmental application. This OE is observed by outlet node. (See the definition at page 61.)

Outlet node: Node which observes outlet phenomenon. (See the first appearance at page 61.)

Physical sensor: "A device that converts real world data (Analog) into data that a computer can understand by using ADC (Analog to Digital converter)." (Sensor, 2017) (See page 47)

Precipitation entity (P): One OE of our environmental application. This OE is observed by precipitation node. (See the definition at page 61)

Precipitation node: Node which measures the rainfall amount. (See the definition at page 61)

Predicate in JADE: The expressions that say something about the status of the world. (See the definition at page 110.)

Rainfall amount per communication interval: The sum of all rainfall amount measurements acquired by a precipitation node during a communication interval. This aggregation value is sent to DSS. (See the first appearance at page 66)

Rainfall amount per day: The sum of all rainfall amount measurements acquired by a precipitation node during a whole day. This aggregation value is sent to DSS. (See the first appearance at page 66)

Rainfall amount per fixed time interval: The sum of all rainfall amount measurements acquired by a precipitation node during a a fixed time interval day. This aggregation value is sent to DSS. (See the first appearance at page 67.)

Rainfall amount per 24h: The sum of all rainfall amount measurements acquired by a precipitation node during the last 24 hours. This aggregation value is sent to DSS. (See the first appearance at page 90)

Raw data: "unprocessed data retrieved directly from the data source such as sensors." (Sanchez et al., 2006) (See page 27)

Sensor: "A sensor is any entity that can follow a sensing method and thus observe some Property of a FeatureOfInterest. Sensors may be physical devices, computational methods, a laboratory setup with a person following a method, or any other thing that can follow a Sensing Method to observe a Property." (Compton et al., 2012) (see page 47)

Sensor Network Entity of Interest (SNEI): Entity of interest whose characterization is obtained from one or many other SNOEs. (See the definition at page 53)

Sensor Network Observable Entity (SNOE): observable entities dedicated to the observation of sensor network. (See the definition at page 53)

State: "A qualitative data which changes over time, summarizing a set of information." (Bendadouche et al., 2012) (See the definition at page 47)

Term in JADE: The expressions identifying entities that "exist" in the world. (See the definition at page 110)

Watercourse entity (W): One OE of our environmental application. This OE is observed by Watercourse node. (See the definition at page 61)

Watercourse node: Node which observes watercourse phenomenon. (See the first appearance at page 62)

Watershed: "A watershed is an area of land where all surface water from rain, melting snow, and tributary stream converges to a common outlet which is a single point at a lower elevation. Watershed is also named drainage basin or catchment basin. The outlet is where the waters of the tributary stream join

another body of water, such as a river, a lake or an ocean". (Perlman, 2017) (See the definition at page 59)

WSN context: Any information that can be used to characterize the situation of a network entity. (See the first time appearance at page 21)

Internet of Things IoT: "A world-wide network of interconnected objects uniquely addressable, based on s tandard communication protocols" (INFSO, 2008) (See more definitions at page 24)

JADE: A software environment to build agent systems for the management of networked information resources in compliance with the FIPA specifications for interoperable multi-agent systems. (See the first appearance at page 89)

Jess: A rule engine for the Java platform that was developed by Sandia National Labs. (See the first appearance at page 91)

Virtual sensor: "A virtual sensor is a software sensor as opposed to a physical or hardware sensor. Virtual sensors provide indirect measurements of abstract conditions (that, by themselves, are not physically measurable) by combining sensed data from a group of heterogeneous physical sensors" (Kabadayi et al., 2006). (See page 47)

Wireless Sensor Network WSN: A wireless network consisting of spatially distributed autonomous devices using sensors to monitor physical or environmental conditions. (See the first appearance at page 20)

1 Introduction

The increasingly use of sensors envisioned at the beginning of the 2000's (Akyildiz *et al*., 2002) is now a reality as shown, for example, in environmental (Othman *et al*., 2012) and agricultural domains. Nowadays, agricultural and environmental domains need automatically observations of the complex natural phenomena (e.g., flood risk and event, disease risk, etc.). Sensors can acquire measurements which are raw observation data. These measurements are basic natural phenomenon observations (*e.g*., humidity measurement, temperature measurement, etc.). These basic observations should be interpreted to acquire more informative data. That means a state of complex phenomenon (*e.g*., plant hydric stress or flood) is deduced from the basic observations (*e.g*., rainfall amount). So, a Decision Support System (DSS) deduces complex phenomenon observation from these basic observations.

A DSS will evaluate the complex natural phenomenon observation and send recommendations or alerts if necessary. In sensor network, one sensor can observe one kind of basic phenomenon observation. Therefore, the measurements of the whole network are heterogeneous according to different types of sensors.

Dealing with environmental phenomena implies that the network is deployed in outdoor environment and communicates by means of wireless technology. Thus we consider that each node has to save its energy to improve its lifetime. For example, agriculture outdoor deployment is for middle or long term (in average 1 to 5 years). Environment outdoor deployment usually takes place in areas, difficult to access such as desert, swamp and forest. Thus providing energy plugins for each wireless sensor manually is not possible.

Sensors will cost energy when acquiring and sending measurements. Energy is a critical resource for a wireless sensor and, by extension, for a Wireless Sensor Network (WSN). So improving the lifetime of a WSN is an essential issue. This dissertation is dedicated to WSN, which is now viewed as part of IoT (Atzori *et al*., 2010). IoT paradigm implies a huge number of (heterogeneous) objects connected a worldwide network, with uniquely addressable, based on standard communication protocols (INFSO, 2008).

So the challenge we are facing is how to use the collected data from WSN at their best to provide complex natural phenomenon observations, while preserving wireless sensor resources such as energy. This dissertation is focused on the observation of a watershed to determine flood risk alert. But our proposition is generic and may be adapted to other type of complex phenomenon observations.

According to the challenge, three issues must be solved:

1. How to analyze, interpret and understand the data collected by WSN?
2. How to solve heterogeneity of collected data?
3. How to preserve wireless sensor resources (such as energy)?

Context awareness principle can play an important role in tackling data analysis, interpretable and understandable issues. Semantic Web technologies are also one possible solution to solve the data heterogeneity. It is a new trend that some of sensor measurement data can be available on the Linked Open Data (LOD) based on the development of Semantic Web

technologies and Open Data. However, it may cost much energy to publish this kind of data generated by a WSN in the LOD cloud.

To better use these limited resources, all the context-aware system components involved in data acquisition process have to work together in a cooperative way, from the component that collects raw data to the one that provides indicators to end users. Generally, these main components are the wireless sensors, the gateway(s) and the remote DSS.

The acquisition and transmission frequencies required by the DSS, through the gateway, have to be consistent with the energy available at the level of the wireless sensors. For some alert applications such as fire prevention, data transmission is sometimes more important than the "survival" of a node of the network. Thus, all the components implied in the data acquisition process have to adapt their behaviors to the context in order to achieve the best performances. A WSN is also subject to unpredictable events that, without fast interventions, can threaten the stability of the whole system.

However, the context of WSN is only one factor that we concern in the context-aware system. The environmental context is another one that we must take into account. The environmental context can be a very complex one due to the complex basic natural phenomena. So our main proposition is to merge two contexts: WSN context and environmental. Our context-aware system deals with a more complex context than the ones described in the literature. Thus, we can specify different policies for the adaptation based on the WSN context and environmental context.

Moreover, the adaptation policies we propose should not be at the expense of the quality of the final application decision, called Quality of Service (QoS) of the application. So, studying potential impact on the QoS and proposing adaptation approach are also important for our adaptation.

So we can conclude that the combination of the common decisions and actions based on the context with better using the limited resources of the WSN is the issue addressed in this dissertation. More precisely, we propose a formalization to define high-level context, which is integrated into an adaptive context-aware system, will be used to reduce the number of exchanged communication packets. Meanwhile, we propose the adaptation approach to limit the impact on the Quality of Service (QoS) of the context-aware system.

The report is organized as follows:
The section "State of the Art" presents the main existing concepts related to context-aware systems. Moreover, we focus on the main challenges of WSN while applying it into IoT paradigm and context-aware system. Context-aware Computing is a possible solution. We compare with existed context-aware systems in terms of three functionalities: Technology used, Data Information and Behaviors (for Context Adaptation) during a context life cycle. We address our contributions based on evaluation of these existed context-aware systems. The following section, untitled "Main Proposition", explains our proposition of context formalization in order to build any context system based on WSN. As mentioned, environmental context deals with many basic natural phenomenon observations, it is complex. WSN context deals with the data from all the sensor node components. It is also complex. So the context in our context-aware system is complex one. To illustrate this formalization, we

propose three kinds of context-aware systems: Environmental context-aware system, WSN context-aware system and an adaptive context-aware system integrated into the two previous context-aware systems. The context in this system is the fusion of the environmental and WSN contexts. The section “Use Case” shows the application of our formalization with the design of four context-aware systems. The first use case, called WSN, is dedicated to node energy monitoring and uses simulated data. The second use case is an environmental one and is dedicated to flood monitoring. This use case uses real historical data provided by Irstea institute. The third use case integrates the two previous one. The last use case is an adaptive context-aware system based on WSN context and environmental one.
The “Modeling” section presents our implementation of this formalization in order to develop context-aware systems based on the previous environmental use cases. This section presents in detail all the models designing, parameters and configurations in terms of the context life cycle processes. “Simulations” section presents feasibility of our formalization by simulating different context-aware systems. Moreover, we want to demonstrate that the adaptations approach of sensor nodes will improve the lifetime of any context-aware system, and, in the meantime, have limited impact on the Quality of Service (QoS) of the system. The last section concludes this dissertation.

2 State of the Art

2.1 Wireless Sensor Network and Internet of Things

Wireless Sensor Networks (WSN) is one of the key technologies that enable capturing and communicating the observation and measurement of data collected from the physical environments. WSN collects large amounts of heterogeneous data (rainfalls, temperatures, water levels, etc.) (Bendadouche *et al*., 2012).

The WSN is built with "nodes" – from a few numbers to several hundreds or even thousands. Each node in a WSN is equipped from one to several sensors. Sensors can be homogeneous or heterogeneous. A sensor node might vary in size from that of a shoebox down to the size of a grain of dust. The cost of sensor nodes is variable, depending on the complexity of the individual sensor nodes. The topology of the WSNs can vary from a simple star network to an advanced multi-hop wireless mesh network ("Wireless Sensor Network", 2017).

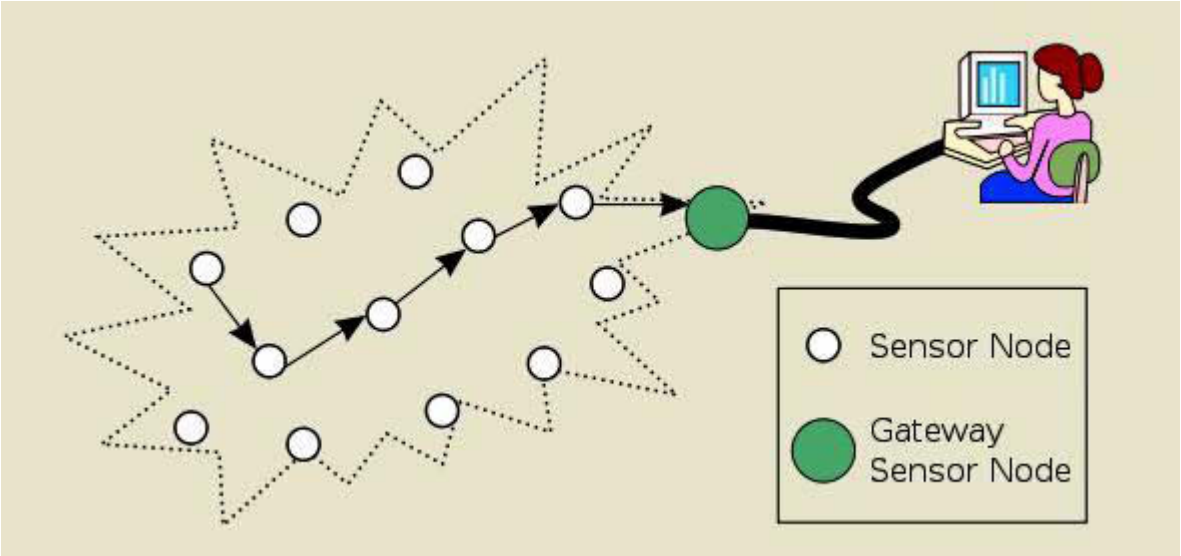


Figure 2.1.1: Typical multi-hop wireless sensor network architecture (Reddy et al., 2009)

Figure 2.1.1 presents typical multi-hop wireless sensor network architecture. A sensor network is composed of a large number of sensor nodes, which are densely deployed either inside the phenomenon or very close to it.

However, one of the most important constraints on sensor nodes is the resources such as energy, memory, computational speed and communications bandwidth. Sensor nodes carry limited, generally irreplaceable, power sources (Akyildiz *et al*., 2002).
Besides, agriculture outdoor deployment is for middle or long term. Environment outdoor deployment usually takes place in areas difficult to access. Moreover, providing energy plugins for each wireless sensor node manually is not recommended. So improving the lifetime of a WSN is an essential issue especially in agriculture and environment applications.
During the past decade, the Internet of Things (IoT), as an emerging paradigm, has gained significant attention.

(Guillemin *et al*., 2009) provides a definition of IoT as presented in Figure 2.1.2:

"*The Internet of Things allows people and things to be connected Anytime, Anyplace, with Anything and Anyone, ideally using Any path/network and Any service*". This definition includes a broad vision of IoT.

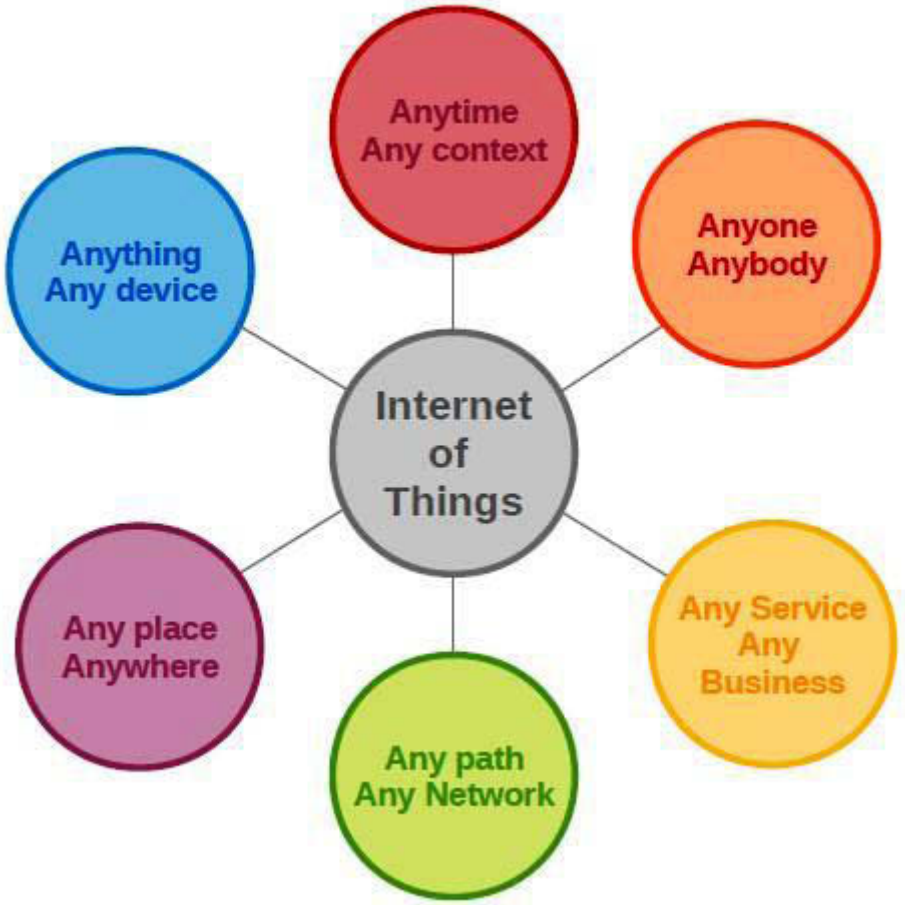


Figure 2.1.2: Definition of the Internet of Things (Guillemin et al., 2009)

(Atzori *et al*., 2010) gives another definition of IoT more specifically. The name ''Internet of Things" syntactically is composed of two terms. ''Internet" is a network-oriented vision of IoT, while "Things" focuses on generic ''objects" to be integrated into a common framework. ''Internet of Things" semantically means ''*a world-wide network of interconnected objects uniquely addressable, based on standard communication protocols*" (INFSO, 2008). This implies a huge number of (heterogeneous) objects involved in the process. So the most challenging issues are about the object unique addressing, the representation and storing of the exchanged information (Atzori *et al*., 2010). "Semantic oriented", with the perspective of IoT, is the possible solution.

Semantic technologies play a key role in how to represent, store, interconnect, search, and organize information generated by IoT. Semantic technologies can exploit appropriate modeling solutions for thing description, reasoning with data generated by IoT, semantic execution environments and architectures that accommodate IoT requirements and scalable storing and communication infrastructure (Toma *et al*., 2009).

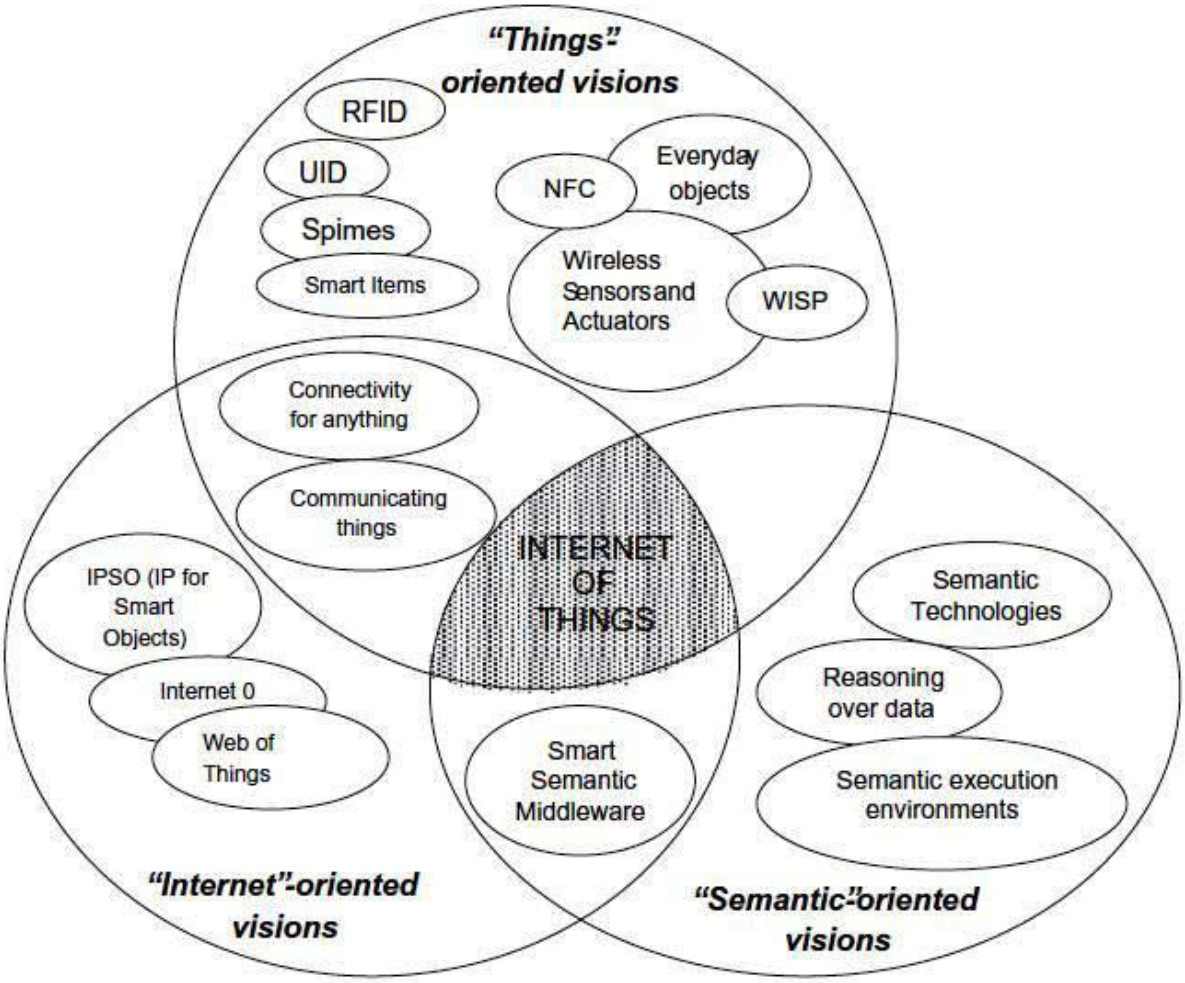


Figure 2.1.3: ''Internet of Things" paradigm as a result of the convergence of different visions (Atzori et al., 2010)

In Figure 2.1.3, the concept "Spime", which is a contraction of "space" and "time", is a neologism for a futuristic object, characteristic to the Internet of Things (Thomas, 2006), that can be tracked through space and time throughout its lifetime (Sterling, 2005). They are essentially virtual master objects that can, at various times, have physical incarnations of themselves (Thomas, 2006) (Maciag *et al*., 2010). An object can be considered a spime when all of its essential information is stored in the cloud (Sterling, 2004).

In Figure 2.1.3, the main concepts, technologies and standards are highlighted and classified with reference to their contribution to IoT visions. From such an illustration, it clearly appears that IoT paradigm shall be the result of the convergence of the three main visions: ''Internet oriented", "Things oriented" and "Semantic oriented" (Atzori *et al*., 2010).

Remember that the ultimate goal of IoT is to create "*a better world for human beings*". The objects around us could know what we like, what we want, and what we need, and act accordingly without explicit instructions (Perera *et al*., 2014).

Based on the work of (Perera *et al*., 2014), sensor networks are an essential component of IoT. IoT comprises everything that sensor networks (SN) include. Furthermore, it comprises a

thick layer of software such as middleware systems, frameworks, APIs and many more software components. The software layer is installed across computational devices (both low and high-end) and the cloud as presented in Figure 2.4. As the most of the sensors deployed today are wireless, thus WSN is essential to IoT.

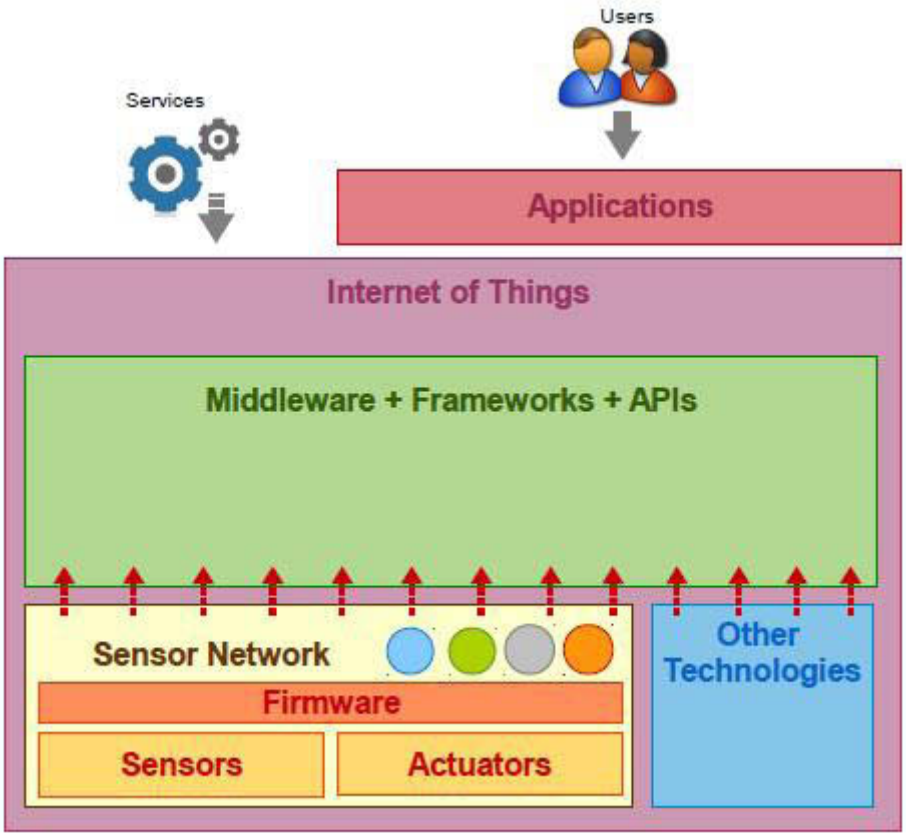


Figure 2.1.4: Relationship between sensor networks and IoT (Perera et al., 2014)

However, due to the advances in sensor technology, sensors are getting more powerful, cheaper and smaller in size, which have stimulated large scale deployments. That means billions of sensors will be deployed and these sensors will generate what we call big data (Zaslavsky *et al*., 2012). These collected data may not have any value unless we analyze, interpret, and understand them. Besides, it is also a challenge to process all the data collected by those sensors considering billions of sensors which are connected to the Internet, in IoT paradigm. Context-awareness, which is a core feature of ubiquitous and pervasive computing systems, will play a critical role in solving the issues we mentioned previously.

We will now describe in detailed context-awareness in the following section.

2.2 Context-aware System

2.2.1 Definition of Context

Many definitions of "Context" were provided by different researchers, but we adopt the definition provided by (Abowd *et al*., 1999) as presented in the following paragraph. This definition can be used to identify context from data in general (Perera *et al*., 2014). If we

consider a data item, we can easily identify whether the data item is context or not by using this definition.

"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".

2.2.1.1 Difference Between Raw Data and Context

The work in (Sanchez *et al*., 2006) explained the distinction between raw data and context information:

- Raw data: are unprocessed and are retrieved directly from the data source such as sensors,
- Context information is generated by processing raw data. Context is also 'add-on information'. Context information is combined with raw data during processing step to extract relevant data.

Therefore in general, the data captured by sensors are raw (sensor) data. It is not the context information. If these data can be used to generate context information, we identify these data as context.

2.2.1.2 Categories of Sensors According to Raw Data and Context

There are different types of sensors that can be used to acquire context. In general, "sensor" is referred to a hardware device. Based on the work in (Indulska *et al*., 2003), sensors can be divided into three categories: physical, virtual, and logical.

- Physical sensors are hardware devices equipped with a variety of sensor (*e.g*., temperature, humidity, microphone and touch). They generate sensor data by themselves,
- Virtual sensors do not necessarily generate sensor data by themselves. Virtual sensors also do not have a physical presence. For example: human also can be considered as a sensor. For instance, a man reads the date in a calendar then he can publish it as sensor data on the web,
- Logical sensors (also called software sensors) combine physical sensors and virtual sensors in order to produce more meaningful information. For example, weather station uses several physical sensors to collect weather information. They also collect information from virtual sensors *e.g*. maps, calendars, etc. Finally, the weather information is produced by combining both physical and virtual sensors.

However, based on the definition of "ssn:Sensor" in Semantic Sensor Network Ontology (SSNO) (Compton *et al*., 2012):

"*A sensor can do (implements) sensing: that is, a sensor is any entity that can follow a sensing method and thus observe some Property of a FeatureOfInterest. Sensors may be physical devices, computational methods, a laboratory setup with a person following a method, or any other thing that can follow a Sensing Method to observe a Property*."

We divide sensors into two categories: physical sensor and virtual sensor in opposition to physical one.

We adapt the definition of physical sensors presented in the work of (Sensor, 2017):
"*A device that converts real world data (Analog) into data that a computer can understand by using ADC (Analog to Digital converter)*".

Therefore, data acquired from physical sensors are raw (sensor) data.

On the other hand, a virtual sensor is:
"*A virtual sensor is a software sensor as opposed to a physical or hardware sensor. Virtual sensors provide indirect measurements of abstract conditions (that, by themselves, are not physically measurable) by combining sensed data from a group of heterogeneous physical sensors*" (Kabadayi *et al*., 2006).

The context data are acquired from virtual sensors.

2.2.1.3 Categories of Context

Context can be identified as different types based on different perspectives. The work in (Perera *et al*., 2014) defined a context categorization scheme: primary context and secondary context in terms of an operational perspective.

- Primary context is any information retrieved without using existing context and without performing any kind of sensory data fusion operations, e.g. temperature sensor readings as temperature in a forest,
- Secondary context is any information that can be computed using primary context. For example: the average temperature in one month in the forest.

However, the definition of secondary context is a broad definition. Some secondary context can also be used to characterize the status of an entity by basing on the several primary contexts: temperature, air humidity or smoke etc. Secondary context can be inferred to represent whether the forest is in fire or not. This secondary context is higher level than the one only describes the situations as mentioned above.

The work in (Liu *et al*., 2013) defined another context categorization scheme: low-level context and high-level context.

- Low-level context refers to the context about subcomponents of a device, such as protocol states and sensor readings in WSN,
- High-level context represents the aggregated state of a device, is inferred by multiple low-level local contexts, for example, the status of a WSN.

However, this definition of low-level context contains not only the quantitative data (e.g. sensor readings in WSN) but also the qualitative data (e.g. protocol states in WSN, etc.).

Therefore, we propose our context categorization scheme: low-level context and high-level context.

- Low-level context is equal to primary context, it only contains the quantitative data (e.g. sensor readings in WSN),
- High-level context contains the qualitative data (e.g. protocol states in WSN, the status of a WSN, etc.).

2.2.2 Definition of Context-aware

In computer science, context awareness refers to the idea that computers can both sense, and react based on their environment (Schilit *et al*., 1994). Devices may have information about the circumstances. Under these circumstances, devices are able to operate and react accordingly based on context reasoning, or an intelligent stimulus.

The definition of Context Awareness, which we adopted, is "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" (Abowd *et al*., 1999). The reason is that we can easily identify whether a system is a context-aware system or not by using this definition.

The definition of context is traditionally associated with the design of context-aware applications. In these applications, contexts are the information to describe the situation of any entity relevant to the applications. The entities can be people, places, or things. The applications are context consumers that receive information from context producers. These applications may also produce their own contexts to other context consumers (*e.g.* other applications). By exchanging contexts, the applications can adapt their behaviors according to shared contexts.

In WSN, this concept of context can be extended to describe the situation of any entity relevant to a network when designing context-aware networks. The entities can be nodes, links, or protocols. It may also include human users and the physical environment of nodes. Similarly, context can be shared between context producers and consumers in the network (Liu *et al*., 2013).

2.2.3 Definition of Adaptive Context-aware System

There is a definition for adaptive context-aware applications (Efstratiou *et al*., 2004): "*Adaptive Context-aware applications are applications that modify their behavior (adapt) according to changes in the application's context.*" That means an adaptive context-aware application should be expected to be able to adapt to a variety of contextual triggers. We adopt this definition because heterogeneous measurements, which are sampled from different

sensors in WSN, can deduce heterogeneous contexts. This definition is appropriate for our dissertation.

2.2.3.1 Context Life Cycle

In IoT diagram, context-awareness is no longer limited to desktop, web or mobile applications. The context management has become an essential functionality in software systems, e.g. application or middleware.

A data life cycle (Figure 2.2.1) shows how data move from phase to phase in software systems. Specifically, it explains where data are generated and where data are consumed (Perera *et al*., 2014). The work in (Hynes *et al*., 2009) has classified data life cycles into two categories: Enterprise Lifecycle Approaches (ELA) and Context Lifecycle Approaches (CLA). CLA focuses on context management.

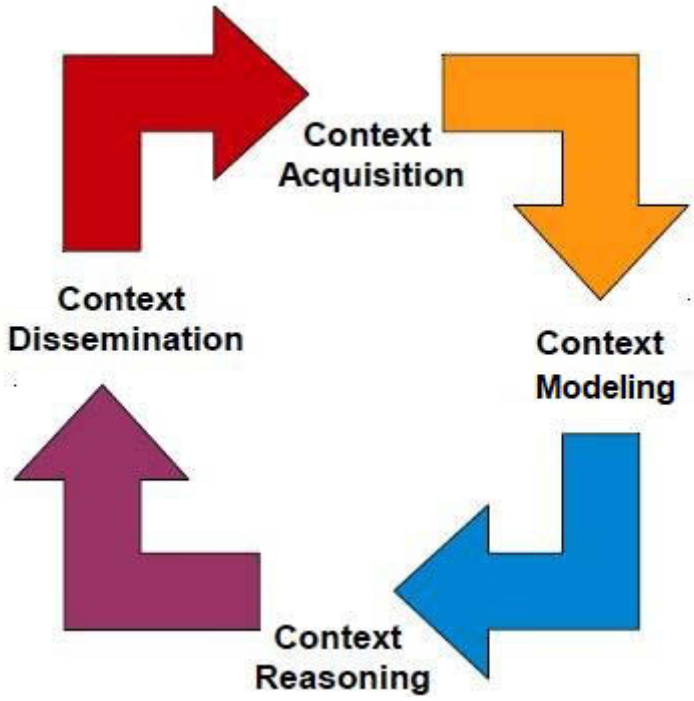


Figure 2.2.1: A form of a context life cycle (Perera et al., 2014)

2.2.3.2 Proposed a Context Cycle of an Adaptive Context-aware System Based on WSN

Based on the work in (Perera *et al*., 2014), Figure 2.2.1 presents a form of a context life cycle composed of four phases: Context Acquisition, Context Modeling, Context Reasoning and Context Dissemination.

Based on the four phases of a context life cycle, we propose a context cycle of an adaptive context-aware system based on WSN in Figure 2.2.2.

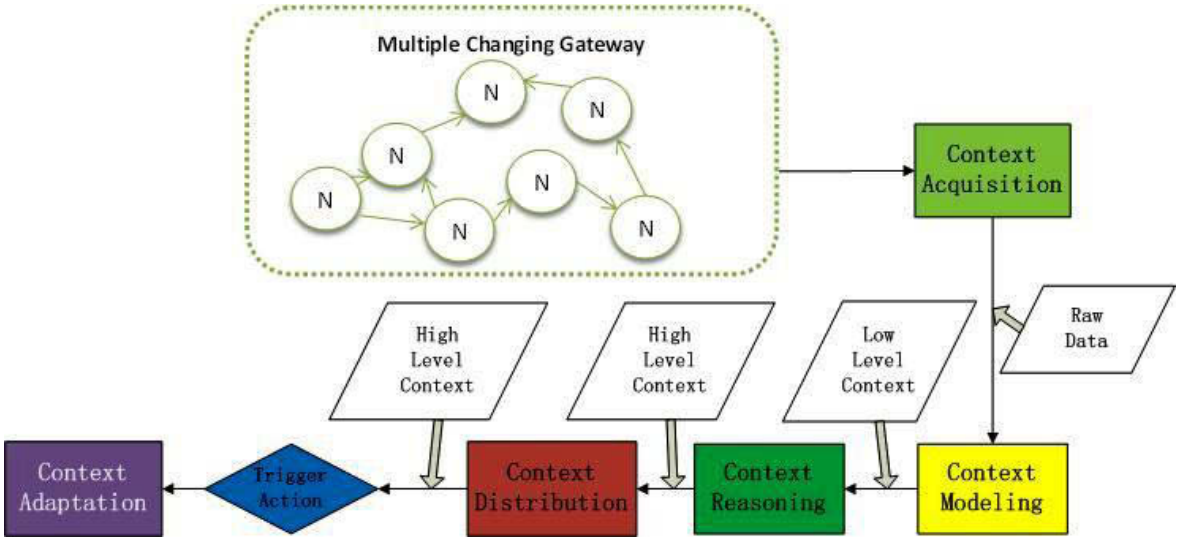


Figure 2.2.2: Context cycle of an adaptive context-aware system based on WSN

An adaptive context-aware system is concerned with Context Acquisition, Context Modeling, Context Reasoning, Context Distribution and Context Adaptation based on the recognized context.

In our adaptive context-aware system, in general, WSN acquire measurements, which are raw data, then send these measurements to gateways. Gateways can do pre-processing, e.g. data filtering to get clean raw data.

- Context Acquisition: Context needs to be acquired from various sources. The sources can be physical sensors or virtual sensors. The techniques used to acquire context can be varied based on different types of contexts. The techniques used to acquire context can be varied based on responsibility, frequency, context source, and sensor types, etc. (Perera *et al.*, 2014). Table 2.2.1 presents the comparison of context acquisition methods based on these different techniques. Context acquisition is using sensors to perceive a situation, e.g. temperature in the forest.

Table 2.2.1: Comparison of context acquisition methods based on different techniques (adaptation from the work in (Perera et al., 2014))

Techniques	Methods	Advantages	Disadvantages	Applications
Based on Responsibility	**Push**	**Sensing and Communication:** **- Sensor hardware makes the major decisions;** **- Can be both instant or interval**	**- Decision made based on reasoning based on less amount of data**	**- Sensors have enough processing power** **- Event detection without software level complex data processing and reasoning**
	Pull	**Sensing and Communication:** **- Software of the sensor data**	**- More communication bandwidth is required**	**- Event detection where large amount of data need**

		consumer makes the major decisions; **- Can be both instant or interval**		**to be dealt with**
Based on Frequency	**Instant events**	**- Saving energy** **- Get more accurate data**	**- Difficult to detect event when there are different types of sensors**	**- Event detect with small types of sensors**
	Interval events	**- Either sensors sense and communicate data in a predefined frequency** **- Study the behavior by collecting sensor data over time**	**- May waste energy** **- Less accurate as the sensor communication frequency changes** **- Miss some occurrence of events due to inaccuracy**	**- Event detect with massive types of sensors**
Context Source	**Sensor hardware**	**- Efficient by direct communication with the sensors** **- Have more control over sensor configuration and data retrieval process**	**- Significant hardware level embedded device programming and configuring**	**- Applications where limited number of sensors are involved**
	Middleware infrastructure	**- Easy to manage and retrieve context**	**- Require more resources (e.g. processing, memory, storage)**	**- IoT application**
	Context servers	**- Less resources required** **- Can retrieve data faster with less effort and technical knowledge**	**- Less efficient as the context need to be pulled from server over the network**	**- Applications where significant amount of context are required but have only limited resources**
Sensor Types	**Physical sensors**	**- Error detection is possible and relatively easy**	**- Provide raw sensor data**	**- Applications which collect physically observable phenomenon**
	Virtual sensors	**- Provide high-level context information**	**- Difficult to find error in data** **- Filling missing values is not easy as they are mostly non-numerical**	**- Applications which collect information that are costly and impossible to collect directly through single physical sensor**

- Context Modeling: which is a process that the collected data need to be modeled and represented according to a meaningful manner, e.g., by adding semantics and metadata (NISO Press, 2004) to the values of temperature, air humidity or smoke. There are six most popular context modeling techniques: key-value, markup schemes, graphical, object-based, logic-based, and ontology-based modeling (Perera *et al*., 2014). Table 2.2 presents the comparison of these six context modeling and representation techniques. There is no single modeling technique which is ideal to be used in context modeling. So incorporating multiple modeling techniques is the best way to produce efficient and effective results, which will mitigate each other's weaknesses.

Table 2.2.2: Comparison of context modeling and representation techniques (adaptation from the work in (Perera et al., 2014))

Techniques	Advantages	Disadvantages	Applications
Key-Value	- Simple - Flexible - Easy to manage when small in size	- Strongly coupled with applications - No structure or schema - Hard to retrieve information - No way to represent relationships	- Applications with limited data transferring and any other less complex temporary modeling requirements
Markup Scheme Tagged Encoding (e.g. xml)	- Flexible - More structured - Validation possible through schemas - Processing tools are available	- No standards for structures - Can be complex when many levels of information are involved	- Applications which store data temporarily, transfer data among applications, and transfer data among application components.
Graphical or relational (e.g. database)	- Allows relationships modeling - Easier information retrieval - Different standards and implementations - Validation possible through constraints	- Complex querying - Configuration may be required - Interoperability among different implementation is difficult - No standards but governed by design principles	- Applications which are for long term and large volume of permanent data archival or store historic context
Object Based	- Allows relationships modeling - Can be well integrated using programming languages - Processing tools are available	- Hard to retrieve information - No standards but govern by design principles - Lack of validation	- Applications which store very short term, temporary, and mostly stored in computer memory.
Logic-Based	- Generate high-level context using low-level context - Simple to model and use- Support logical reasoning	- No standards - Lack of validation - Strongly coupled with applications	- Applications which generate high-level context using low-level context (e.g., generate new knowledge)
Ontology-Based	- Support semantic reasoning - More expressive representation of context - Strong validation - Application independent and allows sharing - Strong support by standardizations - Fairly sophisticated tools available	- Representation can be complex - Information retrieval can be complex and resource intensive	- Applications which model domain knowledge and structure context based on the relationships defined by the ontology

- Context Reasoning: Modeled data need to be processed to derive high-level context information from low-level context. Table 2.2.3 presents the comparison of these three context reasoning decision modeling techniques. Context reasoning could deduce new knowledge, which could be understood better, based on the available context. This part is to deduce high-level context, e.g. get the result of forest is in fire or not.

Table 2.2.3: Comparison of context reasoning decision modeling techniques (adaptation from the work in (Perera et al., 2014))

Techniques	Advantages	Disadvantages	Applications
Rules	- Simple to define - Easy to extend - Less resource	- Defined manually - Can be error prone due to manual work - No validation or quality checking	- Applications where low-level context need to be converted into high-level context
Fuzzy Logic	- Allow more natural representation - Simple to define - Easy to extend - Less resource - Can handle uncertainty	- Defined manually - Can be error prone due to manual work - No validation or quality checking - May reduce the quality (e.g., precision) of the results due to natural representation	- Applications where low-level context need to be converted into high-level context - Applications where allow approximate reasoning instead of fixed and crisp reasoning.
Ontology-Based	- Allow complex reasoning - Allow complex representation - More meaningful results - Validation and quality checking is possible - Can reason both numerical and textual data	- Data need to be modeled in a compatible format (e.g., OWL, RDF) - Limited numerical reasoning - Low performance (e.g., require more computation power and time)	- Applications where knowledge is critical. - Applications allow the context information to be store according to the ontology structure and automatically reason later when required

- Context Distribution: High-level context needs to be distributed to the consumers who are interested in context, e.g. the context distribution process sends an alert message to the alarm and sprayer systems, which are context consumers. The distribution is based on real-time processing.

- Context Adaptation: This phase is used to trigger action based on the recognized high-level context. Based on the work in (Efstratiou *et al.*, 2004), adaptation can be defined

as a combination of three conceptual entities as presented in Table 2.2.4. Therefore, the monitoring entity is high-level context, e.g. the forest in on fire. The adaptation policy is to send an alert and change the temperature in the forest. Then, the adaptive mechanism will perform the adaptations: the alarm system triggers the alarm and the water system sprays water, which will cool the temperature and prevent the fire burning.

Table 2.2.4: Comparison of context reasoning decision modeling techniques (adaptation from the work in (Perera et al., 2014))

Entities	Aim	Characteristic
A monitoring entity	- Monitor a number of contextual attributes that may trigger the application to adapt	- Can either be part of an application or the system itself.
An adaptation policy	- Responsible for deciding when the application should adapt based on the information gathered by the monitoring entity	- An application is designed with a set of policies that implement the application's default behavior.
The adaptive mechanism	- Performing the necessary changes when triggered by the adaptation policy.	- The adaptive mechanism is tightly coupled with the semantics of the application.

2.3 Ontology for WSN

As we mentioned above, IoT can provide our physical world with anytime anywhere accessing to information about any interconnected 'things'. That means that IoT gives rise to the possibility of an omniscient awareness to our physical world. Therefore, IoT is potentially a large-scale context-aware system, in which such awareness is used for making intelligent decisions and autonomous responses to situations and events that have occurred (Liu *et al*., 2013).

WSN is the essential foundation of IoT; WSN node may be equipped with multiple types of sensors, which can collect large amounts of heterogeneous data as presented in section 2.1. How to make a formal and structured description of a set of concepts and relationships between sensory data for a WSN is essential to solve heterogeneity issue of sensors, which is called context modeling in Figure 2.2.2.

Moreover, in context modeling, sensory data can be generated as modeled context. The high-level context is deduced by inferences from the modeled context in the context reasoning phase of a context-aware system. It is possible to make interaction easier by sharing the high-level context and exchanging it between heterogeneous devices. Thus, the interoperability issues of IoT are solved. In addition, this approach can reduce the dimensions of information

(e.g. communication frequency) to be communicated, which preserves the energy of sensor nodes.

Ontologies are a possible solution to describe the sensors, their data and their context. They also define metadata vocabularies.

Compared with other solutions, ontology has two important advantages:

- Ontology could form an effective knowledge representation system when given a domain and vocabulary could perform an effective ontological analysis of the field or domain which clarifies the structure of knowledge,
- Ontologies enable data integration and knowledge sharing. Data integration involves combining data residing in different sources and provides users with a unified view of these data (Lenzerini *et al*., 2002). Shared ontologies let us build specific knowledge bases, which describe specific situations. For example, different sensor devices manufacturers can use a common vocabulary and syntax to build catalogs describing their products. Then the manufacturers could share the catalogs and use them in automated design systems. This kind of sharing vastly increases the potential for knowledge reuse (Chandrasekaran *et al*., 1999).

2.3.1 Definition of Ontology

The work in (Gruber, 1993) proposes the definition of ontology:

"*An ontology is an explicit specification of a conceptualization*". This definition did not mention that the conceptualization should be expressed in a formal machine readable format.

The work in (Borst, 1997) defined an ontology as a "*formal specification of a shared conceptualization*". This definition not only defines that the conceptualization should be expressed in a formal format, but also presents that the conceptualization should be expressed in a shared view.

The work in (Studer *et al*., 1998) merged these two definitions above. We adopt this following definition:

"*An ontology is a formal, explicit specification of a shared conceptualization. A conceptualization refers to an abstract model of some phenomenon in the* (physical) *world by having identified the relevant concepts of that phenomenon. Explicit means that the type of concepts used, and the constraints on their use are explicitly defined. For example, in medical domains, the concepts are diseases and symptoms. The relations between them are causal and a constraint is that a disease cannot cause itself. Formal refers to the fact that the ontology should be machine readable, which excludes natural language. Shared reflects the notion that an ontology captures consensual knowledge, that is, it is not private to some individual, but accepted by a group*."

2.3.2 Categories of Ontology

In agriculture and environment domain, the authors in (Roussey *et al*., 2010) define ontologies as new computer science tools used to design phenomenon model and entities of a domain. Ontologies are used to improve the communication process between different agents (human or computer system) by sharing and reusing information, data or knowledge.

In general, ontologies are composed of different components:

- Concepts, for example, the concept "Sensor",
- Instances, for example, the precipitation sensor included in the weather station located at Montoldre is an instance of the concept "Sensor",
- Properties, for example, the weather station is located at Montoldre. The location of a weather station is a property ,
- Logical formula, for example, the concept "Sensor" is defined by the formula "*System* *Device* *SensingDevice*".

All these ontology components are linked together thanks to relations. Semantic relations link only concepts together (for example the "part-of" relationship indicates that a concept is composed of another one), instance relations connect only instances (Roussey *et al*., 2010).

According to the usage of these components, the work in (Roussey *et al*., 2010) present three kinds of ontologies:

- Terminological Ontologies mainly focus on terms and their relationships. A concept can be referenced by several terms, which are ambiguous. In terminological ontology, which is a result of terminology agreement between a user's community, a term is used to represent a concept to avoid ambiguity.
- Data Ontologies model consensual and standard data model to facilitate data exchange between information systems. Data ontologies are built upon terminological ontologies describing the terminological agreement between data users.
- Logical Ontologies require clear semantics for the language used to define the concept, clear motivations for the adopted distinctions between concepts as well as strict rules about how to specify concepts and relationships. The meaning of concept in logical ontology is guaranteed by formal semantics by using formal logic, e.g. First Order Logic or Description Logic.

2.3.3 Semantic Sensor Network Ontology (SSNO)

In our case, ontologies have three objectives:

- Defining metadata about WSN or observation of phenomenon in context modeling phase,
- Defining the knowledge needed for reasoning purposes in context reasoning phase,

- Publish data on the Web using Semantic Web technologies, that is to say enrich Linked Open Data. Linked Data is a method of publishing structured data so that it can be interlinked and become more useful through semantic queries (Bizer *et al*., 2009). Linked Data may also be open data, in which case it is usually described as linked open data (LOD).

The ontology should describe the WSN and its components, e.g., sensor. The ontology should also describe the measurement process of WSN, e.g. observation and measurements.

Based on the work in (Compton *et al*., 2009), the Semantic Sensor Network Ontology (SSNO) is able to describe most of the spectrum of sensor and observation concepts and it covers a wider range of concepts than the other ontologies.

In SSNO, the measurements generated by sensors are associated to "ssn:SensorOutPut" class and their values to the "ssn:observationValue" one.

CSIRO (Compton *et al*., 2009) has been used as the initial version of the SSNO. So SSNO is the main ontology used for our dissertation. SSNO is a data ontology. SSNO can store the observations made by sensors. SSNO is also a logical ontology. SSNO has clear semantics and some reasoning techniques may be applied on the stored data in order to infer new data.

2.3.3.1 Characteristic of SSNO

Based on the work in (Compton *et al*., 2012), SSNO can describe sensors in terms of capabilities, measurement processes, observations and deployments.

The W3C Semantic Sensor Network Incubator Group developed four use cases of SSNO, which were focused into four categories:

- Data discovery and linking, which focus on finding and linking data, given accuracy, spatial or temporal bounds,
- Device discovery and selection, which require the ontology to represent sensor types, models, methods of operation and common meteorological definitions like accuracy, precision, measurement range, and allowing sensor capabilities to be defined relative to prevailing conditions,
- Provenance and diagnosis, which need context information to deduce trust levels or to analyze previous measurements. The context information is from sensor and observation data, deployment information, custodian descriptions, maintenance schedules and data linkage,
- Device operation, tasking and programming, which require sufficient information to reprogram a device or understand the consequences, in terms of energy usage or network cost, of a reprogramming.

SSNO can be seen from four main perspectives:

- A sensor perspective, which focuses on what senses, how it senses, and what is sensed,
- An observation perspective, which focuses on observation data and related metadata,
- A system perspective, which focuses on systems of sensors and deployments,
- A phenomenon description, focusing on which natural feature is observed and which property of this natural phenomenon is sensed.

2.3.3.2 Stimulus-Sensor-Observation (SSO) Pattern

The SSNO is built around a central Ontology Design Pattern (ODP) describing the relationships between sensors, stimulus, and observations: the Stimulus-Sensor-Observation (SSO) pattern (Janowicz *et al*., 2010). SSO pattern is presented in Figure 2.3.1.

We are interested in the classes in SSO pattern: "ssn:Stimuli", "ssn:Sensor", "ssn:Observation", "ssn:ObservationValue", "ssn:FeatureOfInterest" and "ssn:Property". The former three concepts are central concepts as presented in bold format in Figure 2.3.1.

- "ssn:FeatureOfInterest". "*A feature is an abstraction of real world phenomena (thing, person, event, etc.)*."
- "ssn:Property". "*An observable Quality of an Event or Object. That is, not a quality of an abstract entity as is also allowed by DUL's Quality (Any Entity that cannot be located in space-time, e.g. mathematical entities: formal semantics elements, regions within dimensional spaces, etc.), but rather an aspect of an entity that is intrinsic to and cannot exist without the entity and* ***is observable by a sensor***."
- "ssn:Sensor". See the definition of "ssn:Sensor" in section 2.2.1.
- "ssn:Observation". "*An Observation is a Situation in which a Sensing method has been used to estimate or calculate a value of a Property of a FeatureOfInterest. Links to Sensing and Sensor describe what made the Observation and how; links to Property and Feature detail what was sensed; the result is the output of a Sensor; other metadata details times etc*."
- "ssn:ObservationValue". "*The value of the result of an Observation. An Observation has a result which is the output of some sensors. The result is an information object that encodes some values for a Feature*."
- "ssn:Stimuli". "*An Event in the real* (physical) *world that 'triggers' the sensor. The properties associated to the stimulus may be different to eventual observed property. It is the event, not the object that triggers the sensor*".

The SSO pattern encompasses three of the four perspectives mentioned above except system perspective. System perspective is more about system organization and deployments than sensing, but clearly links to the pattern.

The SSO pattern has been developed as a minimal, common ground for heavy-weight ontologies for the Semantic Sensor Web, as well as to explicitly address the need for lightweight semantics in the Linked Data cloud (Compton *et al*., 2012).

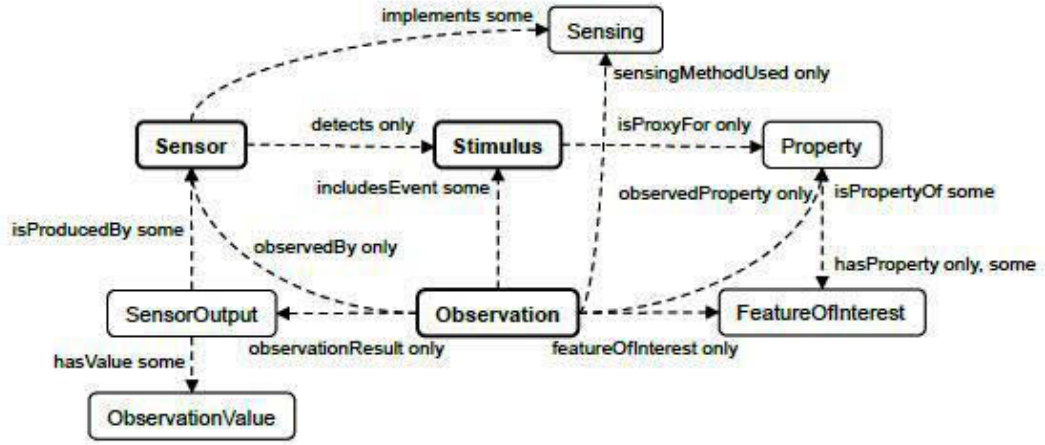


Figure 2.3.1: Stimulus-Sensor-Observation pattern of SSNO (Compton et al., 2012)

2.3.3.3 Uses of the SSNO

In general, SSNO has been used as part of an architecture for the Web of Things (Moraru *et al*., 2011), in sensing for manufacturing (Wenzel *et al*., 2011), for representing humans and personal devices as sensors (Corsar *et al*., 2011) and as part of a Linked Data infrastructure for Sensor Web Enablement (SWE) (Sensor Web Enablement, 2017) (Yu *et al*., 2011). SWE is a suite of standards developed and maintained by the Open Geospatial Consortium. SWE standards enable developers to make all types of sensors, transducers and sensor data repositories discoverable, accessible and usable via the Web.

2.4 Agricultural and Environmental Context-aware System

As previously mentioned, a context-aware system is composed of several phases: context acquisition, context modeling, context reasoning, and context distribution. Context adaptation phase is additionally concerned in an adaptive context-aware system.

Therefore, in this section, we will make a comparison of agricultural and environmental context-aware systems in terms of functionalities in these five phases. The functionalities are:

- Used technology ,
- Data information,
- Behaviors (for context adaptation).

Besides, energy is also taken into account for the agricultural and environmental context-aware system.

Table 2.4.1: Comparison of agricultural and environmental context-aware systems

Systems Phases and Functionalities		(Goumopoulos *et al*., 2014)	(Gray *et al*., 2011)	(Hwang *et al*., 2011)	(Gutierrez *et al*., 2013)	(Huang *et al*., 2015)
Context Acquisition	Technology used	WSAN	Channel Coastal Observatory (cco), WaveNet	WSN and CCTVs	WSUs and WIU	WSN
	Data Information	**Raw data:** Strawberry's Ambient Temperature, etc.	**Raw data:** Coastal flooding raw data	**Raw data:** Environmental data relevant to the green house's crop growth; greenhouse and crop images	**Raw data:** Soil moisture and temperature Data and node energy	**Raw data:** Water quality data and node energy
Context Modeling	Technology Used	PLANTS Ontology	SemSorGrid4Env ontology network	Ontology	Database	WSN-based lake-water monitoring ontology
	Data Information	**Low-level context:** Plant context of Strawberry	**Low-level context:** Environmental context of coastal flooding	**Low-level context:** Environmental context of green house's crop growth	**Low-level context:** Environmental context of crops and energy context	**Low-level context:** Environmental context of Water quality and energy context
Context Reasoning	Technology Used	Rule-based Reasoning	Rule-based Reasoning	Ontology-based Reasoning	Rule-based Reasoning	SWRL-based rules reasoning
	Data Information	**High-level context:** State of Strawberry's stress	**High-level context:** State of coastal flooding	**High-level context:** State of green house's crop growth	**High-level context:** State of crop growth	**High-level context:** State of Water quality, observation-demand, device and data
Context Distribution	Technology Used	Subscribe/Publish	Subscribe/Publish	Subscribe/Publish	Subscribe/Publish	Subscribe/Publish
	Data Information	**High-level context:** State of	**High-level context:** Coastal	**High-level context:** Emergency	**Low-level context:** Data in	**High-level context:** State of Water

		Strawberry's stress	flooding alert	calls and alarms	graphics in web application	quality,observation-demand, device and data
Context Adaptation	Technology Used	Software Controller	None	Software Controller	Software Controller	Software Controller
	Data Information	**High-level context:** State of Strawberry's stress	None	**High-level context:** State of green house's crop growth	**High-level context:** State of crop growth	**High-level context:** State of Water quality,observation-demand, device and data
	Behaviors	Irrigate Strawberry when it is in drought stress	None	Control environment facilities	Microcontroller-based gateway to control water quantity	WSN Self-configuration

In the work of (Goumopoulos *et al*., 2014), the authors present an irrigation system for strawberries. The irrigation decision support system diagnoses the stress of strawberries to make decision and adapt the irrigation method to the strawberry growing context. There are two drawbacks:

- In context modeling phase, PLANTS Ontology is not described in detail,
- The resource of sensor nodes is not taken into account as strawberries grown in greenhouse.

Different from the work of (Goumopoulos *et al*., 2014), the work in (Gray *et al*., 2011) describes the SemSorGrid4Env ontology network in detail in designing and organization. The SemSorGrid4Env ontology network is composed of different ontologies that can be classified into four layers according to the different aspects that the ontology represents:

- To represent sensor networks and their observed information about properties of certain features of interest,
- To represent the services provided by the SemSorGrid4Env infrastructure and the datasets which can be accessed to,
- To represent schema metadata about relations and relational streams,
- To represent the geographic and administrative regions of the south coast of England.

SemSorGrid4Env ontology is based on SSNO.

However, the context-aware system in the work of (Gray *et al*., 2011) presents two drawbacks:

- There is no context adaptation phase. The system just sends a coastal flooding alert, by email or SMS text message, when a flooding event is forecast to occur, or has been detected to have occurred, for their area of responsibility,
- The resource of sensor nodes is not taken into account.

The work in (Hwang *et al*., 2011) proposes a context-aware middleware to efficiently process data collected from ubiquitous greenhouses by applying WSN technology. The context-aware middleware is also adopted to integrate different forms of data from heterogeneous sensor nodes. The context-aware middleware also provides filtering functions to filter the collected data:

- The context-aware service deduces the state of crop growth from the environmental data collected (e.g. soil humidity, illuminance and temperature). It can also control environment facilities (fan, heater, etc.) to adapt to the state changes of crop growth,
- The event processing service sends emergency calls and alarms, by email or SMS text message, based on real-time image information of the greenhouse provided by CCTVs. The purpose is preventing dangers such as burglary and fire.

However, the work in (Hwang *et al*., 2011) has three limitations:

- The process, which deduces high-level context from low-level context, is not described in detail in context reasoning phase,
- The ontology design is not explained in detail,
- The resource of WSN is not taken into account.

On the contrary, the work of (Gutierrez *et al*., 2013) proposes an automated irrigation system to optimize water use for agricultural crops. The system has a distributed wireless network of soil moisture and temperature sensors (Wireless Sensor Units (WSUs) and a Wireless Information Unit (WIU)) placed in the root zone of the plants. In addition, a gateway unit handles sensor information, triggers actuators, and transmits data to a web server application. In this system, an algorithm was developed with threshold values of temperature and soil moisture that was programmed into a microcontroller-based gateway to control water quantity. The energy level of sensor nodes is also taken into account. The system was powered by appropriate solar cell panel, which provides sufficient energy to keep the WSU working for the whole crop season.

However, the work in (Gutierrez *et al*., 2013) has three drawbacks:

- The context modeling phase is not described in detail,
- The energy consumption model of sensor nodes and the energy provided model of photovoltaic panels are not described in detail,
- The sensor nodes configuration adaptation process of the context adaptation phase is not provided.

The work in (Huang *et al*., 2015) presents a WSN self-configuration approach for lake water quality monitoring. The WSN self-configuration approach is to change WSN resource configuration depending on t he contextual information, for example, changing the communication frequency and sensing frequency of WSN. WSN-based lake-water monitoring ontology is proposed which is based on SSNO and the work in (Bendadouche *et al*., 2012). This ontology focuses on describing methods for monitoring and managing WSN nodes from the energy consumption perspective. A rules-based reasoning process is proposed to conduct the decision support system.

However, the work in (Huang *et al*., 2015) has three drawbacks:

- The energy consumption model of sensor nodes is not presented,
- The reasoning rules that dynamically control the energy consumption is quite simple,
- The proposed approach is focused on the adaptation of WSN configuration in harsh or dangerous environment. The approach does not deals with end-user feedbacks, e.g. emergency calls and alarms.

2.5 Conclusion

In this section, we present the issues of applying WSN into IoT paradigm. So the research questions are:

- How to understand, analyze and interpret the data generated by IoT?
- How to process the sensory data collected by WSN where billions of sensors are connected to the Internet in IoT paradigm?

Context-awareness (context-aware computing) is a possible solution to overcome the issues above.

Then we present context-aware system in term of:

- Definition of context. Difference between raw data and context, categories of sensors according to the difference between raw data and context and categories of context are also presented,
- Definition of context-aware system,
- Definition of adaptive context-aware system.

We also propose a context cycle of an adaptive context-aware system based on WSN. A comparison of methods based on different techniques in context acquisition, context modeling, context reasoning, context distribution and context adaptation phases are also presented.

As IoT is potentially a large-scale context-aware system, the main challenges of WSN in context-aware system in agricultural and environmental in detail are:

- How to solve heterogeneity issue of sensors?
- How to solve the interoperability issues in context-aware system?

- How to preserve the energy of nodes to extend the lifetime of WSN?

Therefore, the main work of this dissertation is focused on " context-aware computing for WSN in environmental applications", and more specifically it is the manner to improve the use of WSN in decision making process for environmental applications?

The environmental applications imply:

- Limited available resources, e.g. energy of sensor nodes,
- Good quality of data,
- Real-time data.

The solutions can be:

- Ontology is a possible approach to solve heterogeneity issue of sensors in context modeling phase,
- High-level context deduced from modeled context can solve the interoperability issues of context-aware system in context reasoning phase,
- Reducing the dimensions of information (e.g. communication frequency) to be communicated can preserve the energy of sensor nodes by the high-level context being shared or exchanged between heterogeneous devices in context adaptation phase.

Consequently, we more focus on these phases in terms of functionalities:

- Technology used,
- Data Information,
- Behaviors (for context adaptation phase).

At last, we made a comparison of agricultural and environmental context-aware systems in these three functionalities.

Based on all observations above, our contributions must be:

- Design of an ontology network model based on SSNO in context modeling phase,
- Design of two steps rules based reasoning process: deduce low-level context from raw data and deduce high-level context from low-level context in context reasoning phase,
- Design of energy model of sensor nodes,
- Proposition of WSN configuration adaptation policy in context adaptation phase.

3 Main Proposition

3.1 Context Formalization

The work in (Bendadouche *et al*., 2012) defines the concept of entity "state" as:

"*A qualitative data which changes over time, summarizing a set of information*"

In (Sun *et al*., 2016), we propose our definition of "context" as:

"*A set of entities characterized by their states, plus all information that can help to derive any state changes of these entities*"

In addition to this definition, we propose to consider the two following classes of entities:

- ***Observable entity***: entity that is directly observed by sensors,
- ***Entity of interest***: entity whose characterization is obtained from one or many other entities and required by the application.

Our formalization reuses the definition of ***sensors*** proposed in the SSN Ontology report (Compton *et al*., 2012):

"*A sensor is any entity that can follow a sensing method and thus observe some Property of a FeatureOfInterest. Sensors may be physical devices, computational methods, a laboratory setup with a person following a method, or any other thing that can follow a Sensing Method to observe a Property*"

This definition is large and gathers different types of sensors. Some of them can be physical sensors and virtual sensors but other type of sensors are also possible like human sensor. A physical sensor definition we reused is:

"*A device that converts real world data (Analog) into data that a computer can understand by using ADC (Analog to Digital converter)*" (Sensor, 2017).

On the other hand, a definition of virtual sensor we will reuse is:

"A virtual sensor is a software sensor as opposed to a physical or hardware sensor. Virtual sensors provide indirect measurements of abstract conditions (that, by themselves, are not physically measurable) by combining sensed data from a group of heterogeneous physical sensors" (Kabadayi *et al*., 2006).

If we consider an application of fire forest prevention, an observable entity would be the temperature, air humidity or smoke. The values of the observable entities e.g., temperature and humidity are measurements from physical sensors, for example, physical temperature and physical humidity sensors respectively. The smoke can be observed by a human sentry. A

sentry detects the presence of a smoke in the forest with a telescope or a binocular. The sentry is a kind of human sensor. The entity of interest would be the fire (a starting fire).

The purpose of these definitions of entities is to divide the reasoning process into several reasoning steps to create the high-level context as illustrated in Figure 3.1.1. Rule-based reasoning is often used to deduce high-level context (Perera *et al*., 2014). Our work presented in (Sun *et al*., 2016) is the first to promote the division of rules into several reasoning steps. This proposition makes the management of rules easier and let envision the dissemination of rules in separate network components like the gateway or the node. We also expect that improving the reasoning capability of a node will reduce data exchanges inside the network. Indeed, the state of an entity of interest cannot be established directly from the low-level context. So we can see in Figure 3.1.1 that several reasoning steps are necessary to build the high-level context.

- The inputs of the reasoning steps are the outputs of sensors, called observation values. "***Observation values***" are defined according to SSN Ontology report (Compton *et al*., 2012).

An observation value is:

"*The value of the result of an Observation. An Observation has a result which is the output of some sensors, the result is an information object that encodes some values for a Feature.*"

For example, observation values may be the measurements provided by physical sensors or the output value computed by virtual sensors.

- The low-level context contains observation values which are the sensor measurements or, computed values combined from sensor measurements, stored in the context data model. The state of observable entities is inferred from the low-level context as indicated by the dotted arc in Figure 3.1.1. A t this stage, the high-level context contains the states of the observable entities,
- The state of an entity of interest is inferred from the states of other entities. The high-level context is enriched by the state of the entity of interest.

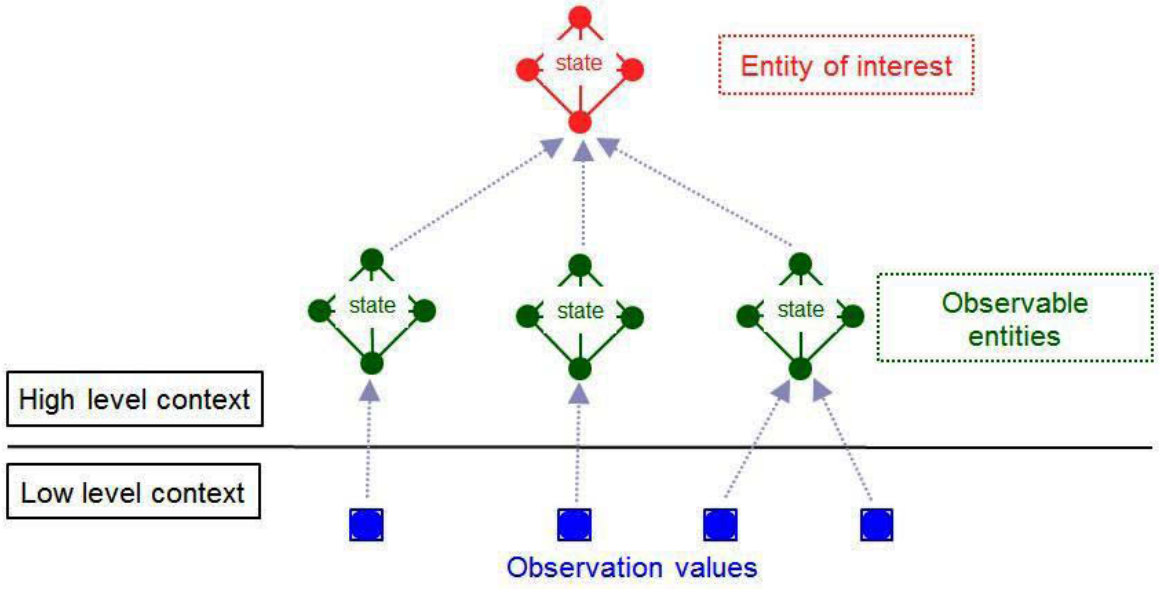


Figure 3.1.1: Classes of entities

To illustrate this formalization, we propose, in the following sections, three kinds of context-aware systems:

- Environmental context-aware system,
- WSN context-aware system, and
- Adaptive context-aware system.

3.2 Environmental Context-aware System

At the beginning of WSN, most of the systems deployed for the observation of environmental phenomena focused on the data acquisition process. With advances in hardware and processing capabilities of the wireless sensors, systems have improved and proposed context-aware functionalities as mentioned in Figure 2.2.1. The sequence diagram of Figure 3.2.1 has 3 agents:

- "User" agent represents the final user of the application,
- "DSS" agent identifies the Decision Support System part of the environmental context-aware system,
- "Node" agent represents the wireless sensor nodes in the system.

Figure 3.2.1 presents a generic scenario:

- Sensors acquire environmental measurements and nodes collect metadata that are useful to build the environmental context,
- Nodes send their environmental measurements and metadata to a DSS,

- The DSS receives the data sent by the nodes. These first three steps are “the context acquisition” process colored in green in Figure 3.2.1,
- The DSS interprets and organizes the acquired data through a specific context data model to build the low-level environmental context. This is the “context modeling” colored in yellow in Figure 3.2.1,
- The DSS infers high-level environmental context from the low-level environmental one based on, for example, machine learning or rule engine. The “context reasoning” process is colored in dark green in Figure 3.2.1,
- The DSS distributes the high-level environmental context to the different consumers. This is the “context distribution” process which is colored in red color in Figure 3.2.1.

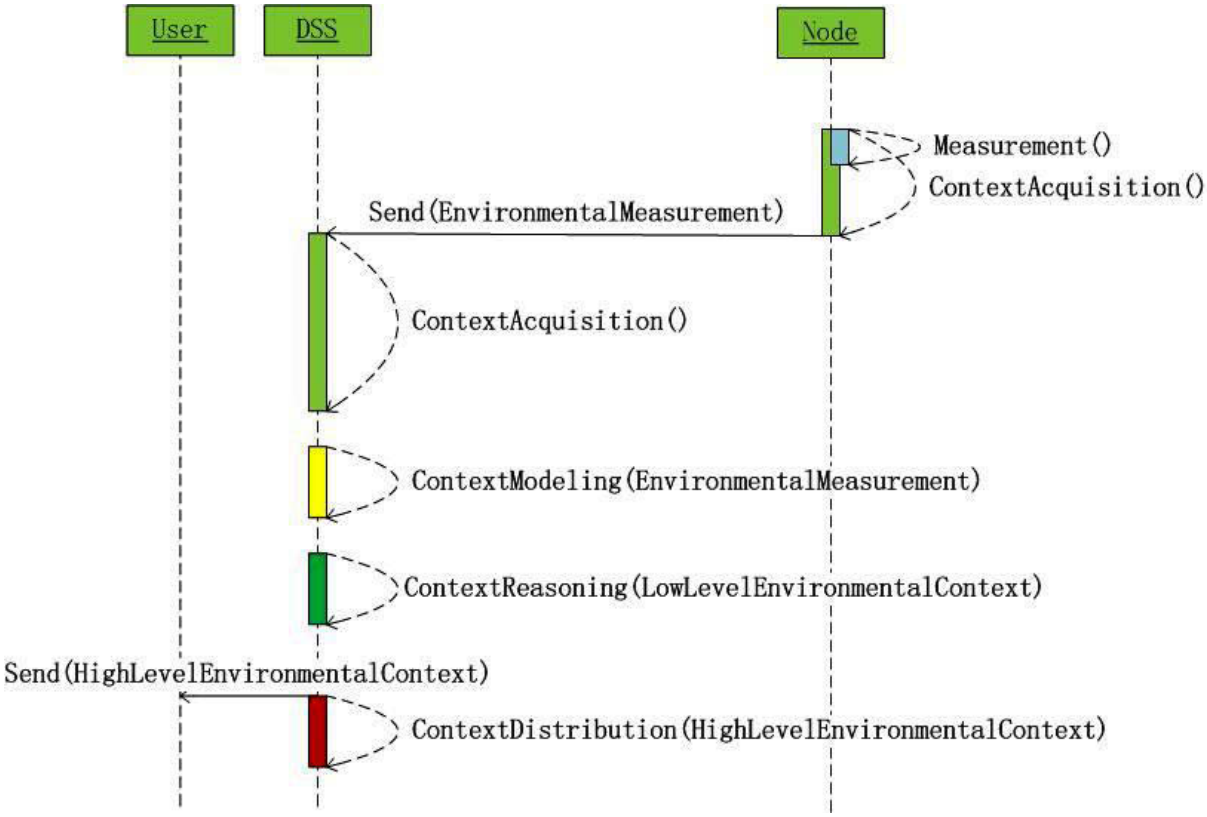


Figure 3.2.1: Sequence diagram of a generic environmental context-aware system

From our formalization, an environmental context-aware system is composed of several entities:

- Environmental Observable Entities (EOE) are observable entities dedicated to the observation of an environmental phenomenon,
- Environmental Entities of Interest (EEI) are entities of interest whose characterization is obtained from one or many other EOE.

Generally, an EOE has two states “Normal” or “Risky”. EEI can have several states depending on the observed environmental phenomenon and the supported application, which should satisfy end users’ needs.

Figure 3.2.2 illustrates our final reasoning step. The state of an EEI is inferred from the states of two EOEs: EOE1 and EOE2. In this example, EEI has three states: “State 1”, “State 2” and “State 3”. Figure 3.2.2 depicts an environmental finite-state machine associated to this EEI.

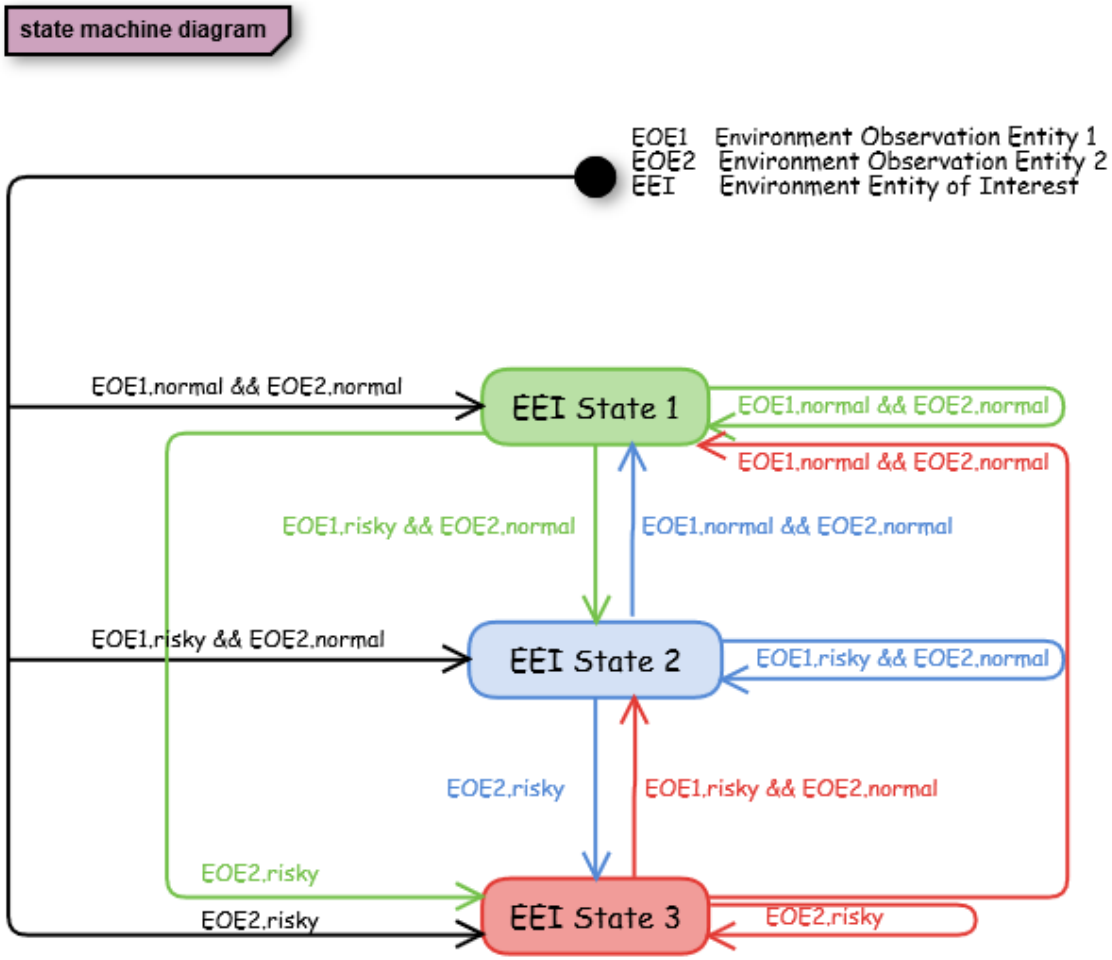


Figure 3.2.2: Generic environmental phenomenon finite-state machine

We assume that the EOEs of this environmental phenomenon have different priority levels. In Figure 3.2.2, t he priority levels are defined as followed: EOE2 > EOE1. It means that whatever the EOE1 state is, if the EOE2 state is “Risky”, we switch to EEI “State 3”. The cause is that EOE2 is more important than EOE1 according to the observed environmental phenomenon.

Most of the time, some environmental phenomena can also follow a “chronological” order that will impact the priority level. If we take the example of an application of fire forest

prevention, we generally detect a sharp increase of the temperature before getting dense smoke. The opposite is also possible as dense smoke announces high temperature.

3.3 WSN Context-aware System

As shown in Figure 2.2.1, a complete context-aware system must also focus on the WSN management in order to better use the limited resources of the wireless sensors that compose it. In our case, we are concerned with the generic limited resources of the wireless sensors, so the WSN context in this system is a generic one, not dedicated to each corresponding node. Figure 3.3.1 presents a generic scenario of a context-aware WSN management system:

- Sensors acquire WSN measurements and nodes collect metadata that are useful to build the WSN context,
- Nodes send those data to the DSS,
- The DSS receives the data sent by the nodes. All these steps compose the context acquisition process colored in green in Figure 3.3.1,
- The DSS interprets and organizes the acquired data through a specific context data model to build the low-level WSN context. This is the context modeling colored in yellow in Figure 3.3.1,
- The DSS infers high-level WSN context from the low-level WSN one. The context reasoning process is colored in dark green in Figure 3.3.1,
- The DSS distributes the high-level WSN context to the different consumers. The context distribution process is colored in red in Figure 3.3.1.

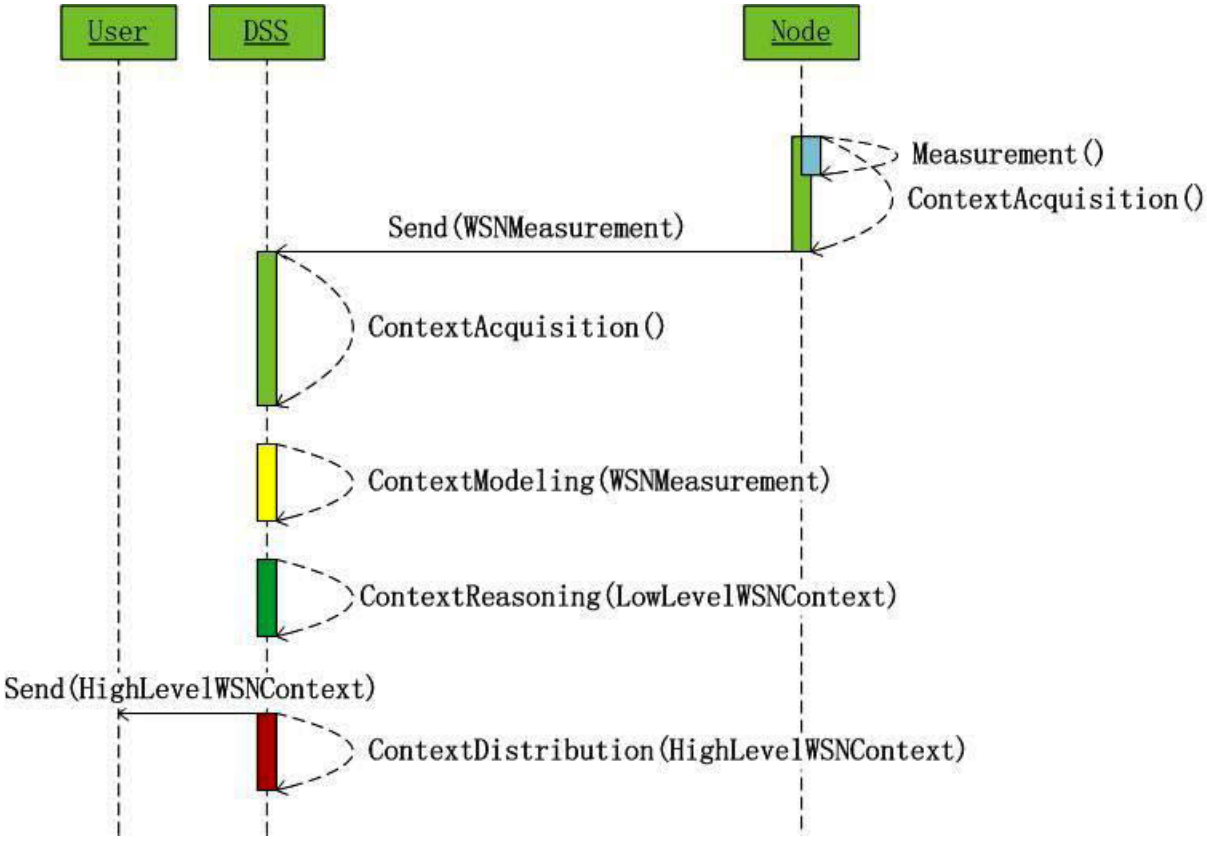


Figure 3.3.1: Sequence diagram of a generic WSN context-aware System

In our formalization, a WSN Context-aware System is composed of the following entities:

- Sensor Network Observable Entities (SNOE) which, such as indicated in the observable entity definition, are directly observed by sensors but dedicated to the observation of a sensor network,
- Sensor Network Entity of Interest (SNEI) which are entities of interest whose characterization is obtained from one or many other SNOEs.

Generally, a SNOE has two states "Stable" or "Critical". A SNEI has several states depending on the observed Sensor Network.

Figure 3.3.2 illustrates our final reasoning step for a WSN Context-aware management system. The state of an Entity of Interest (SNEI) is inferred from the state of three observable entities (SNOE1, SNOE2 and SNOE3). SNEI has two states: "State A" and "State B". Thus, Figure 3.3.2 presents the finite-state machine dedicated to SNEI.

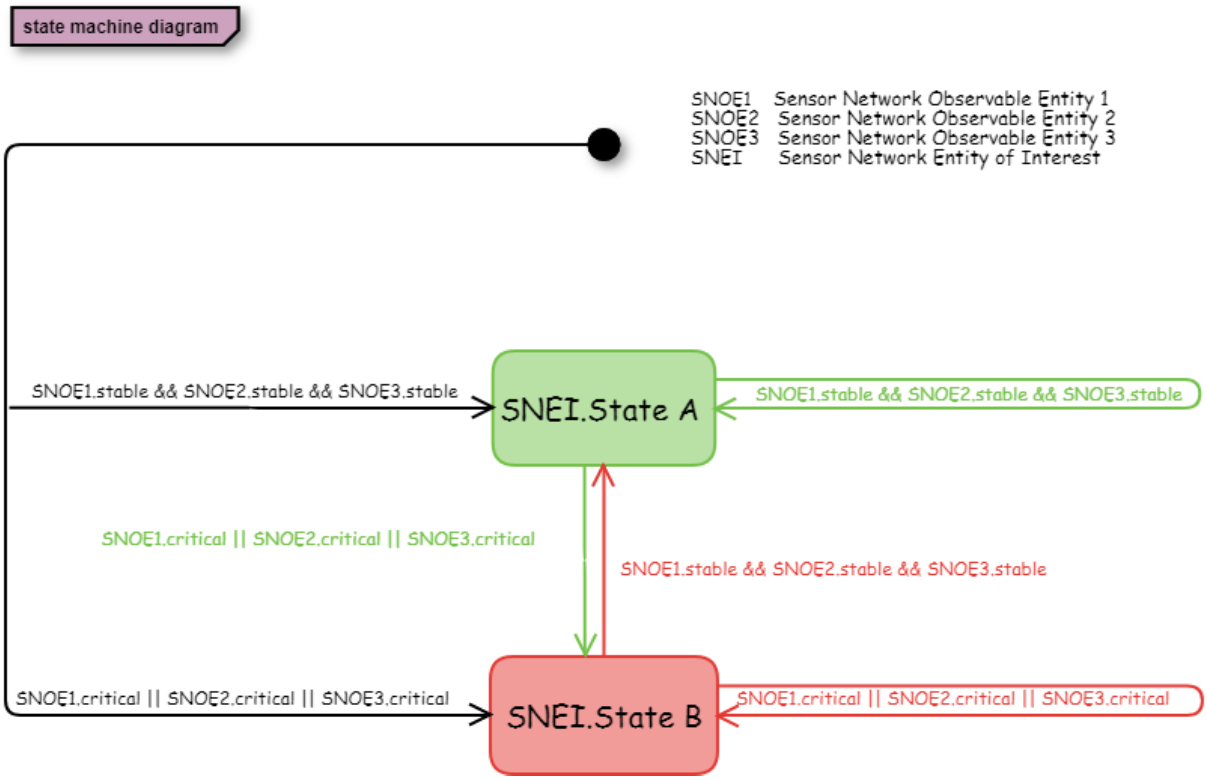


Figure 3.3.2: Generic WSN finite-state machine

The state of SNEI is deduced from the states of the three Sensor Network Observable Entities (SNOE1, SNOE2 and SNOE3). There is no assumption of priority level of SNOEs here. If one SNOE is "Critical", the state of SNEI is also "State B". If all the SNOEs are "Stable", the state of SNEI is "State A".

3.4 Adaptive Context-aware Systems

In this section, we will describe three types of adaptive context-aware systems:

- Environmental adaptive context-aware system,
- WSN adaptive context-aware system, and
- Global adaptive context-aware system.

The first adaptive context-aware system focuses on the studied environmental phenomenon. The goal of this system is to automatically modify several behaviors of the components of the monitoring system as soon as possible to avoid or prevent an environmental disaster. The adaptive context system will ask to some of its components to adapt their behaviors based on the state of EEI. In the fire forest prevention example, we can imagine that the system is composed of temperature and smoke sensors, an alarm system and a water sprayer around the forest. EOE 1 is Temperature, EOE 2 is Smoke and EEI is Fire. The alarm and sprayer systems are context consumers. If EEI is in state "EEI State 2" (Temperature entity is "Risky" and Smoke entity is "Normal"), then there exist some conditions for the starting of a forest fire. Therefore, the context distribution process sends an alert message to the alarm and the sprayer systems. The alarm system automatically switches on the alarm. The water system sprays water in its area. The evaporation of water will cool the temperature and prevent the fire burning. As soon as the system will observe a decrease of temperature, the state of EEI becomes "EEI State1". The context distribution process sends a normal message to the context consumers to stop the alarm and the sprayer.

Thus the goal of the WSN adaptive context-aware system is to better use the limited resources of the WSN. The system asks the WSN context consumers to do context adaptation based on the state of SNEI. We also take fire forest prevention as example. But there, we focus on the energy level of the battery and on the quantity of free memory of the storage device in each temperature sensor node. The SNOE1 is Battery, the SNOE2 is Storage and the SNEI is Node. The nodes are also context consumers. If Node is "SNEI State B" (Battery is "Critical" or Storage is "Critical"), the node will automatically reduce their frequency of sending messages to the DSS. Consequently, here, the context adaptation is the reduction of the message sending frequency.

The global adaptive context-aware system integrates the two previous adaptive context-aware systems. The context in this system is thus the fusion of the environmental and WSN contexts. The new context has several entities of interest: EEI (fire) and SNEIs (temperature and smoke nodes). To do so, several observable entities are considered such as: EOE1 (temperature), EOE2 (smoke), SNOE1 (node battery) and SNOE2 (node storage).

The goal of this system is to automatically modify several behaviors of the monitoring system components as soon as possible to prevent or avoid an environmental disaster and taking care of the lifetime of temperature nodes. For example, the system can switch on the fire alarm and spray water and can automatically increase or reduce the communication frequencies of the

nodes. There is a priority issue of adaptation actions (prevent node death or prevent fire). The priority is determined by the expert systems according to the end user's requirements. Our work focuses on the adaptation of the WSN in order to improve node lifetime if there is no environmental risk. To summarize, sending collected sensory data is more important than the survival of a specific node.

Here again, our adaptive context-aware system adapts the communication frequency of the nodes to the states of SNEI (node) and of EEI (fire). This message sending frequency will depend on these both states. The increasing message sending frequency can lead to an out-of-energy situation for a node. Our adaptation process also implies that two nodes belonging to the same network can have different communication frequencies based on the EEI and SNEI states. Indeed, depending on communication conditions and required actions, each node can consume a different amount of energy. Operating policies, actions to be done (depending on the EEI and SNEI states), can thus be different according to each element of the WSN (nodes, gateway or even DSS).

The Figure 3.4.1 presents a generic scenario for an adaptive context-aware system. As we mentioned, the context is the fusion of the environmental and WSN contexts. In order to highlight clearly the processes of each context, we present the two contexts separately in a "chronological" order:

- Nodes collect environmental and WSN measurements and metadata that are useful to build the context. Notice that, in such adaptive system, the context is the fusion of the environmental and WSN contexts. Then, they send these measurements to the DSS. This is the context acquisition process colored in green in Figure 3.4.1,
- The DSS receives the data sent by the nodes. Then, it interprets and organizes the acquired data through the context data model to build the low-level context. This is the context modeling colored in yellow in Figure 3.4.1. The Figure 3.4.1 presents two context modeling processes in order to highlight that now the context is richer and composed of the environmental and WSN context elements,
- The DSS infers the high-level context from the low-level one. Two context reasoning processes are colored in dark green in Figure 3.4.1. The first process deduces the high-level environmental context elements. The second process deduces the high-level WSN context elements,
- The DSS distributes the high-level environmental context to the different consumers. The context distribution process is colored in red in Figure 3.4.1,
- The DSS automatically modify several behaviors (e.g., the water system sprays water) and establishes the new node configurations (e.g., modification of the sending frequency) based on the states of the two EIs (EEI and SNEI). It sends these new actions and configurations to the nodes. The nodes take into account these orders and they adapt the actions and their operating mode. The "context adaptation" process is colored in purple in Figure 3.4.1.

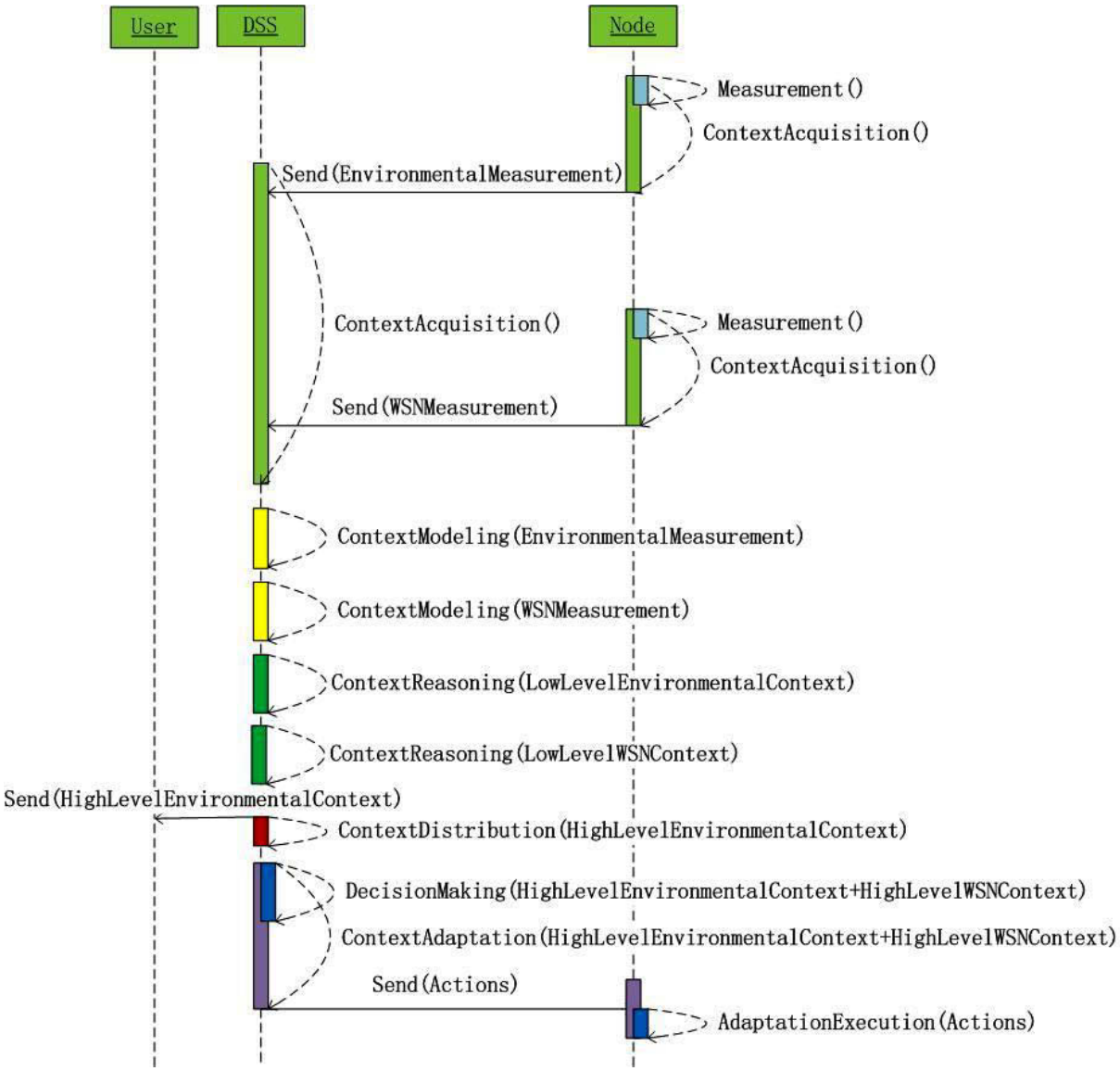


Figure 3.4.1: Sequence diagram of a generic adaptive context-aware system

In the following sections, we provide the use cases for each of the defined context-aware systems. One is for the environmental system. The other is for the WSN system with focus on the wireless sensor entity. Afterwards, we experiment our formalization.

4 Use Case

This section presents in detail the use cases of the context-aware system and one use case of the adaptive context-aware system. The first use case is dedicated to node energy monitoring. This use case uses simulated data. The second one is dedicated to flood monitoring: this use case uses historical data of the French Orgeval watershed provided by Irstea institute (Garnier *et al*., 2014). The third context-aware system integrates the two previous systems. At last, we present an adaptive context-aware system based on t he third context-aware system. We present these systems by increasing complexity order.

4.1 Node Context-aware System

For the context-aware system dedicated to network management, we focus on the lifetime of a WSN. Based on the work in (Akyildiz *et al*., 2002), the lifetime of a sensor network depends on the lifetime of the power resources of the nodes. In agricultural and environmental applications, a wireless sensor node has generally to operate for long period of time without any human intervention. Therefore, the power resources of nodes are critical for the lifetime of a sensor network. In this context-aware system, we focus on the energy level of the battery of the wireless sensor node.

Initial real battery level of a node enables to estimate theoretical lifetime of a wireless sensor for a defined application. However, in real world, two sensor nodes of the same type, deployed in equivalent conditions, may have different battery levels evolution due to message sending errors. In fact, the node energy consumption is influenced by many factors e.g., weather condition, interferences, etc. They could have different energy consumption values during the same time interval. This use case is dedicated to the observation of generic consumption of node. Our study is not to estimate all the factors that influence the node energy consumption but to evaluate the contribution of our context-aware approach. Thus, we use simulated data from the ideal configurations of a node.

A node context-aware system focuses on the operating mode of a node. Thus, there is as many node context-aware systems as wireless sensor nodes.

As explained in previous section, to establish the node context, we define only one entity: the Node entity (*N*) which is both a SNOE and a SNEI. The Node entity is an SNOE because the energy level of its battery is directly acquired by the node. This Node entity is also a SNEI because it represents the context system as we defined before. The Node entity has two states: “stable” and “critical”.

Figure 4.1.1 pr esents the composition of the node context and the reasoning step. The observation values are energy levels of the node battery. These measurements are acquired

from a virtual sensor that simulates the energy level of the battery. The measurements are stored in the context data model and compose the low-level node context.

The deduction of the Node entity state is inferred from the low-level node context as indicated by the dotted arc in Figure 4.1.1. At that point, the high-level node context contains the states of the *N* entity.

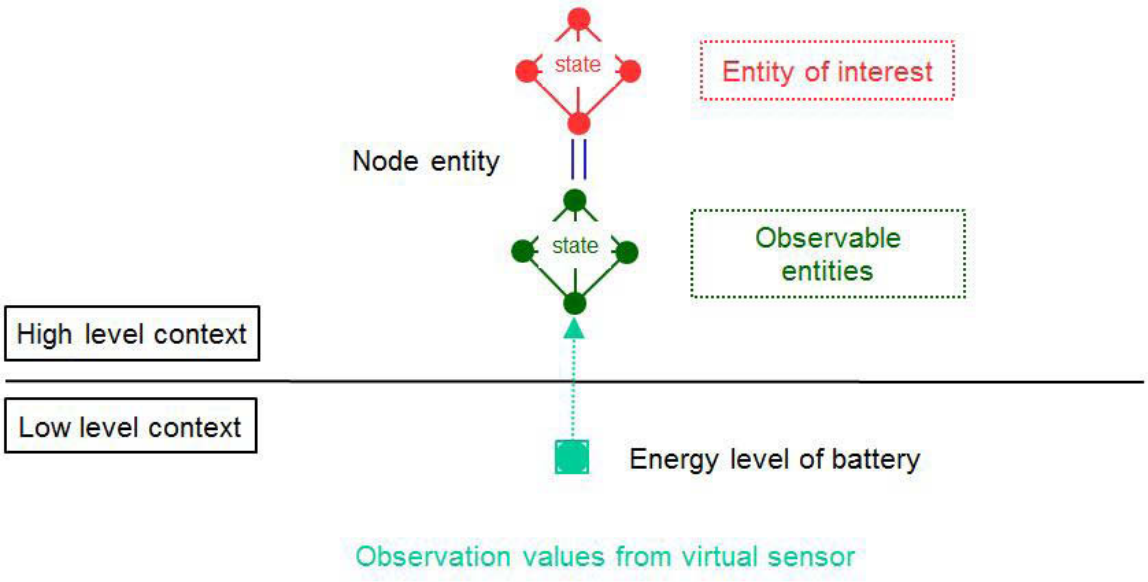


Figure 4.1.1 Node context

4.1.1 Node Entity "*N*"

The observation value is the energy level of the node battery. This value is called $Energy_{Node}$ and is produced by an energy model. The system computes the Node entity *N* state by comparing the $Energy_{Node}$ with a threshold called $Th_{NodeEnergy}$. This threshold is fixed by experimentation.

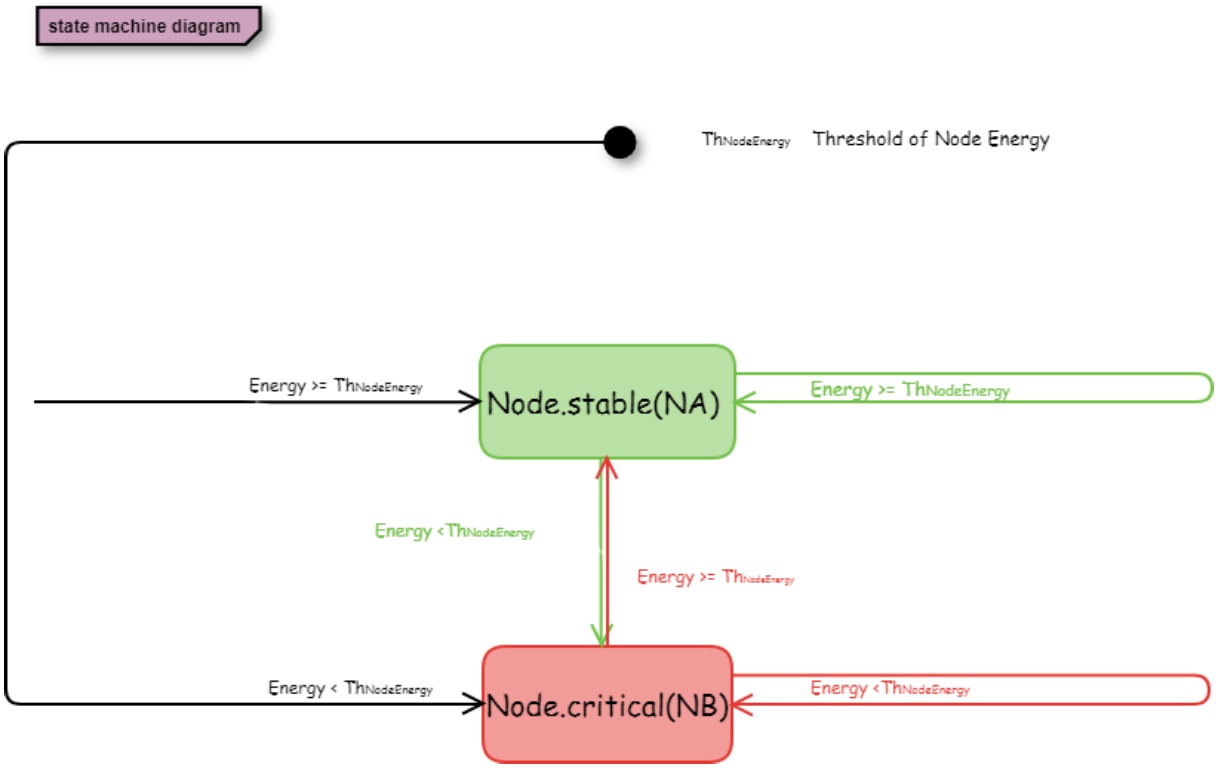


Figure 4.1.2: Example of energy finite-state machine

Figure 4.1.2 presents an example of energy finite-state machine that deduces the Node state from the observation value:

- When $Energy_{Node}$ is bigger than or equivalent to the threshold $Th_{NodeEnergy}$, the Node entity "*N*" is in the "normal" state (*NA*),
- When $Energy_{Node}$ is smaller than the $Th_{NodeEnergy}$, the Node entity "N" is in the "critical" state (*NB*).

4.2 Flood Context-aware System

The considered environmental application is a watershed monitoring system which is able to detect flood events. The goal of this system is not only to detect these flood events but also to evaluate flood risks.

4.2.1 Watershed Description and Sensor Network Description

Watershed is a polysemic word and has two definitions according to the Merriam-Webster dictionary (Merriam-Webster, 2017):

- "*A region or area bounded peripherally by a divide and draining ultimately to a particular watercourse or body of water*",
- *"A crucial dividing point, line, or factor"*.

In our work, a watershed is defined in accordance with the first definition. We will adapt the explanation of what is a watershed by the work of (Perlman, 2017):

"A watershed is an area of land where all surface water from rain, melting snow, and tributary stream converges to a common outlet which is a single point at a lower elevation. Watershed is also named drainage basin or catchment basin. The outlet is where the waters of the tributary stream join another body of water, such as a river, a lake or an ocean". The watershed is usually monitored in order to detect or forecast flood events.

The work on (Jackson, 2015) explains why a flood event happens in a watershed: *"there's a sudden "burst" of heavy rain, the rainwater won't be able to infiltrate fast enough and the water will instead enter the river via surface runoff. This leads to a sudden and large increase of river's discharge, which can result in a flash flood"*.

In order to evaluate flood risks, we consider that the watershed is equipped with a WSN in charge of data acquisition. Acquired sensory data are of different types. The system observes the precipitations that occur in some places of the watershed. It also monitors the water flow rate of the watercourses that compose the tributary stream and the water flow rate of the watershed outlet. To complete this monitoring system, a DSS is in charge of the detection of the flood events or forecast flood risks based on the different observations.

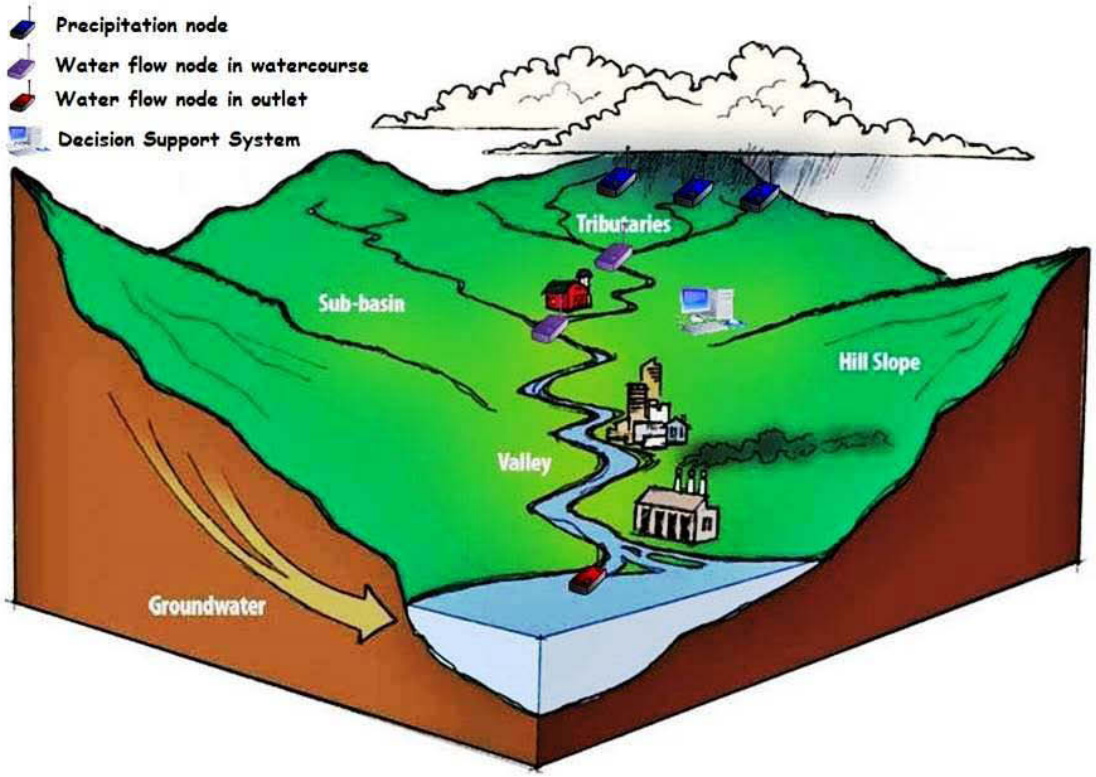


Figure 4.2.1: Example of watershed monitoring (Heathcote, 1998)

Figure 4.2.1 is an example of wireless sensors deployment in order to monitor a watershed. The network contains "precipitation nodes" measuring the precipitation quantity. Precipitation nodes are located at the higher places of the watershed. The network is also composed of wireless sensors called "water flow nodes". Each water flow node is equipped with a stream gauge measuring the water flow rate. One of these wireless sensors is located on the outlet of the watershed. This specific sensor node is called "outlet node". One of our assumptions is that the WSN has a star topology: each node communicates directly with the DSS. Thus, we do not introduce routing protocol constraints at this step.

4.2.2 Flood Context Description

Our environmental application is a flood context-aware system. The flood context is composed of four entities:

- The "Precipitation entity (*P*)" which is an EOE. Its state is inferred from the precipitation measurements collected by the "precipitation nodes" located at different points of the watershed. This "Precipitation entity" has two states: "normal" and "risky",
- The "Watercourse entity (*W*)" which is an EOE. Its state is inferred from the water flow rate measurements collected by the "water flow nodes" located at different points of the tributary stream. This "Watercourse entity" has two states: "normal" and "risky",
- The "Outlet entity (*O*)" which is also an EOE. Its state is inferred from the water flow rate measurements collected by the "outlet node" located on the outlet of the watershed. This "Outlet entity" has two states: "normal" and "risky",
- The "Flood entity (*F*)" is the EEI of our environmental application. The Flood entity is not an EOE but its state depends on the states of all the EOEs. The Flood entity has four states: "Normal (*F1*)", "Rainy (*F2*)", "Risky (*F3*)", and "Flood (*F4*)". "Normal" state means that nothing happens in the watershed. "Rainy" state means that the watershed has received a lot of precipitations. "Risky" state means that some tributaries are overflowing. The flood is thus coming. "Flood" state means that the flood is there, the main river is overflowing. Application users want to know as soon as possible when a flood risky state is reached.

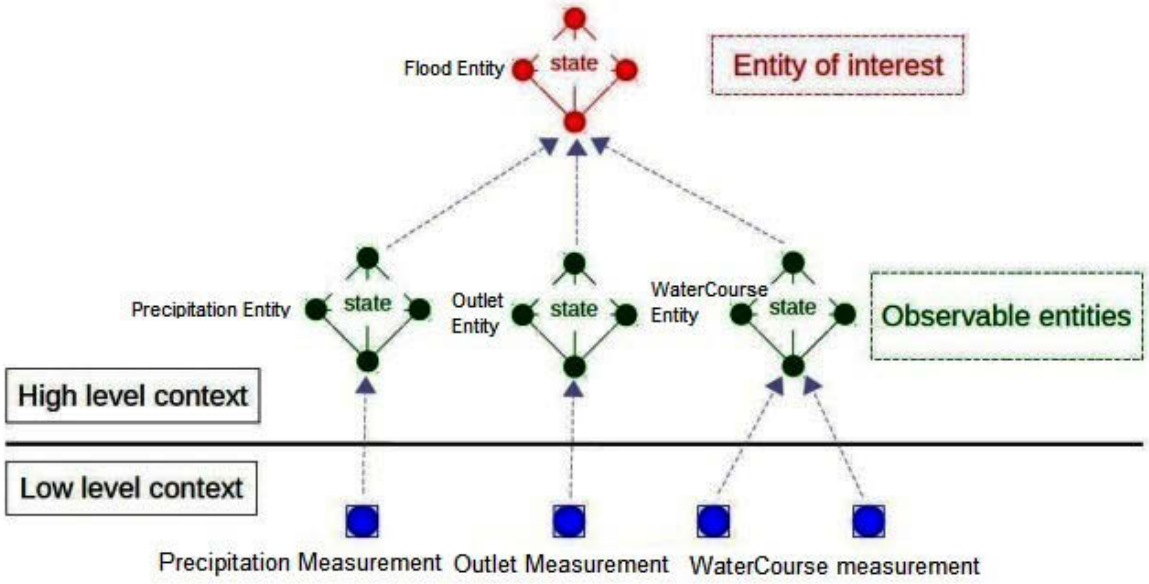


Figure 4.2.2: Flood context

Figure 4.2.2 presents the composition of the flood context and the different reasoning steps. The measurements of the precipitation nodes, watercourse nodes and outlet node, stored in the context data model, compose the low-level flood context.

The deduction of the Flood entity state is composed of several steps:

- EOE state deduction (e.g., Precipitation, Watercourse and Outlet entities). The state of EOE is inferred from the low-level flood context as indicated by the dotted arc in Figure 4.2.2. A t that point, high-level flood context contains the states of the Precipitation, Watercourse and Outlet entities,
- The state of the Flood entity (EEI) is inferred from the states of the EOEs. The high-level flood context is enriched by the state of the Flood entity.

Figure 4.2.3 presents the finite-state machine of our second reasoning step: the state of the Flood entity (EEI) is inferred from the states of the EOEs. Due to the chronological order of environmental phenomenon as explain in (Jackson, 2015), the EOEs have different priority level:

$$P < W < O$$

Where

P is Precipitation

W is Watercourse

O is Outlet

This diagram follows every step of the emergence of a flood. Usually, when a flood event occurs, the Flood entity will move from the "Normal (*F1*)" state to the "Rainy (*F2*)" one, proceed to the "Risky (*F3*)" one and finish with the "Flood (*F4*)" one. We will present the finite-state machine of the first reasoning step in the section 4.2.3.

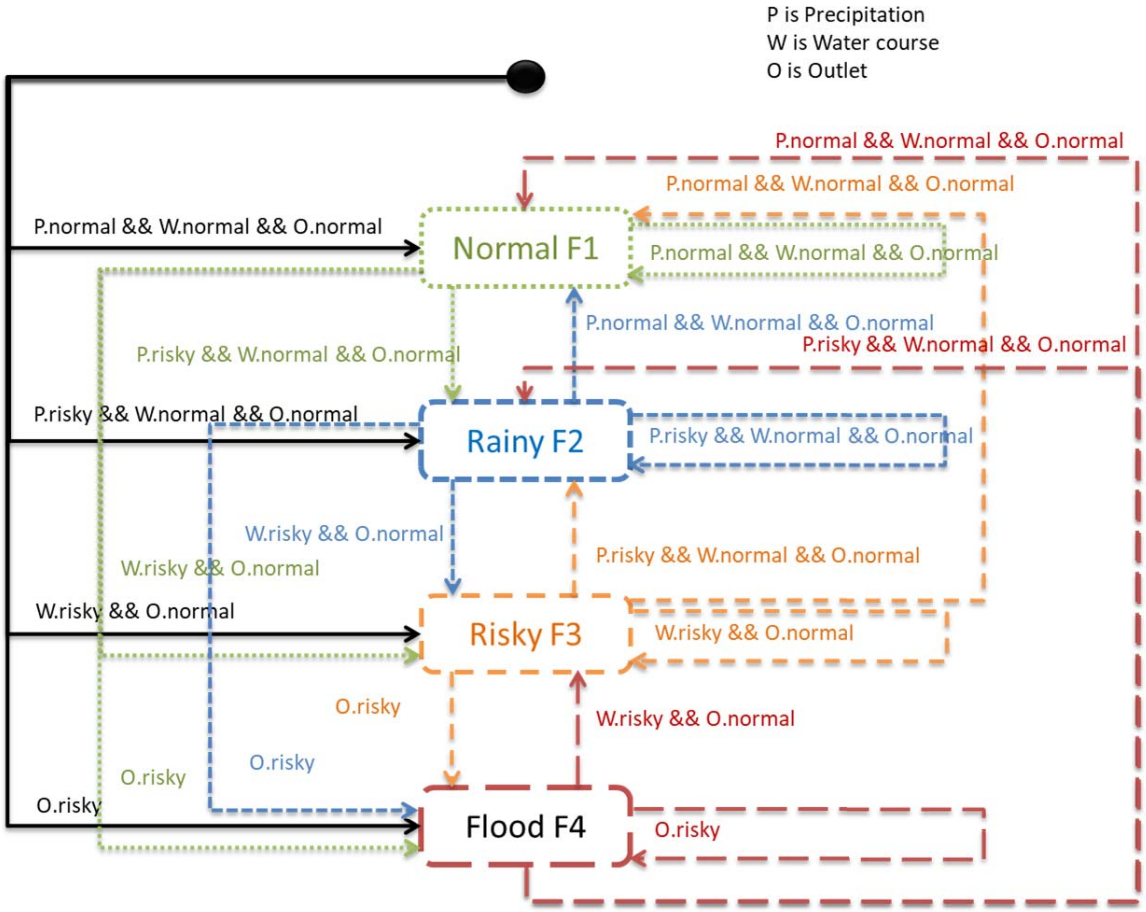


Figure 4.2.3: Example of flood finite-state machine

In order to implement our reasoning steps in a flood context-aware system, we will present the processes of this system. The flood context-aware system is composed of a WSN in charge of the data acquisition and a DSS is involved in all the processes. Figure 4.2.4 presents a sequence diagram of all the processes involved in our flood context-aware system.

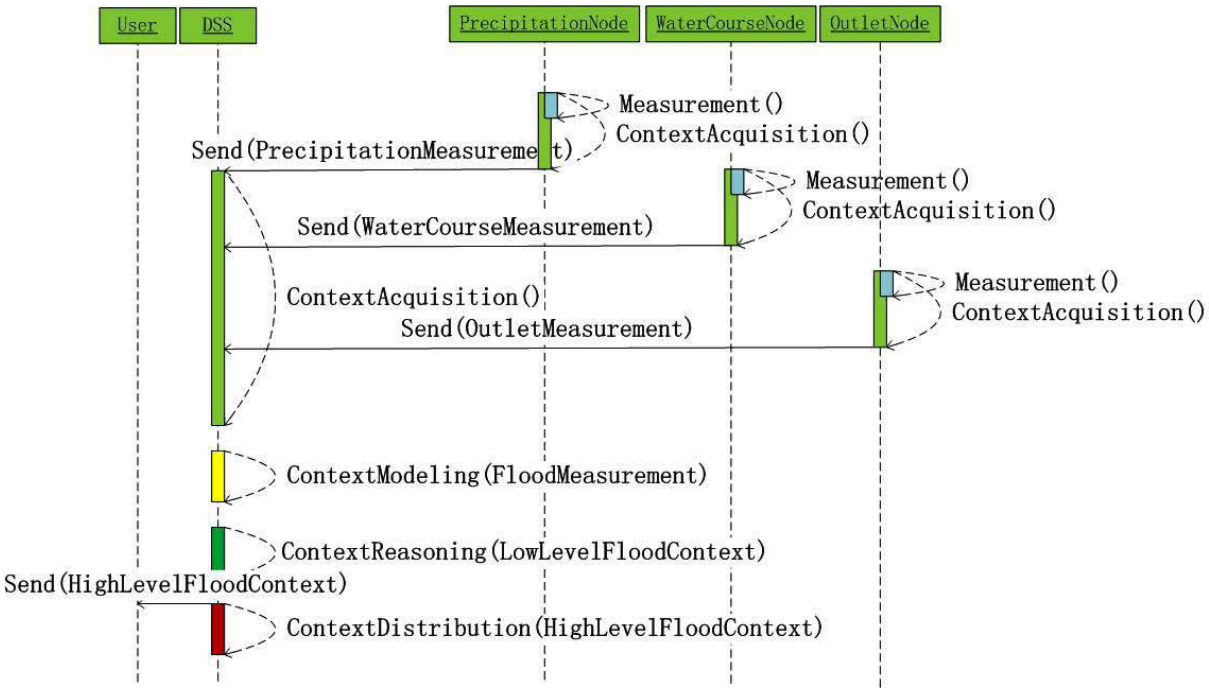


Figure 4.2.4: Sequence diagram of the flood context-aware system

The sequence diagram of Figure 4.2.4 has 5 agents. User" agent and "DSS" agent have been already mentioned in Figure 3.2.1. " PrecipitationNode" agent identifies any of the precipitation nodes used in the flood context-aware system. "WaterCourseNode" agent represents any of the watercourse nodes involved in the flood system. "OutletNode" agent corresponds to the outlet node. All the processes already defined in the environmental context-aware system (see section 3.2) are mentioned in Figure 4.2.4 and summarized as follows:

- Precipitation nodes, watercourse nodes and the outlet node collect their environmental measurements. Precipitation nodes, watercourse nodes and the outlet node send their environmental measurements to a DSS. The DSS receives the measurements sent by the nodes. This first three steps are the context acquisition process colored in green in Figure 4.2.4,
- The DSS interprets and organizes the acquired sensory data through a specific context data model to build the low-level flood context. This is the context modeling colored in yellow in Figure 4.2.4,
- The DSS infers the high-level flood context from the low-level one. The context reasoning process is colored in dark green in Figure 4.2.4,
- The DSS distributes the high-level flood context to the different consumers. This is the context distribution process which is colored in red in Figure 4.2.4.

In the following section, we propose different configurations of context-aware system components in order to deduce the EOEs states which include:

- Node may acquire different type of measurements,

- Node may aggregate its measurements in order to build the content of its messages sent to the DSS,
- DSS receives messages from several nodes and has to aggregate their contents in order to deduce the EOEs states.

4.2.3 Configuration of Flood Context-aware Components

The WSN of the flood context-aware system is composed of several type of nodes:

- Many precipitation nodes,
- Many watercourse nodes,
- One outlet node.

Each type of node may be able to acquire different type of measurements. For example, a watercourse node can acquire the water flow rate or the water level of a stream. The system configuration should specify which measurement each type of node acquires. Moreover, node acquires a measurement at a given sample/acquisition frequency. Node sends a message to the DSS at a given communication frequency. This communication frequency may not be equal to the acquisition frequency. Thus, the node has to aggregate its measurements in order to build the message content sent to the DSS. The system configuration specifies the sample frequency, the communication frequency and the node aggregation function applied to the acquired data. The WSN is composed also of several precipitation nodes and several watercourse nodes. Thus, the DSS receives several messages from different nodes of the same type (messages from precipitation nodes, messages from watercourse nodes, messages from outlet node). The DSS has to aggregate the messages contents of the same type of nodes in order to deduce the state of the observable entity associated to each node type. For each type of message, the DSS has to apply a specific aggregation function.

In the next sections, we focus first on the precipitation nodes and all the treatments applied on their measurements in order to deduce the *P* entity state. Then, we will treat the case of the watercourse nodes and the associated *W* entity. Finally, we will consider the *O* entity with its single outlet node.

4.2.3.1 Precipitation Node Configuration and Precipitation Entity (*P*)

To evaluate the flood risk, we need to evaluate the total amount of precipitations that falls on the watershed during a certain time period. Thus, all the precipitation nodes acquire the rainfall amount that fall during the sample frequency. As mentioned above, the sample frequency can be different from the communication one. Thus, several aggregation functions may be used by precipitation nodes.

4.2.3.1.1 Precipitation Node Aggregation Function

The first aggregation function may be the sum of the measurements acquired during the time interval of the communication frequency. This value is called the "**rainfall amount per communication interval**". The main drawback of this solution is that, in low wireless

communication quality where packets do not reach the DSS, the loss of a packet will imply the loss of some precipitation measurements.

In order to be able to recover some precipitation measurements when a packet is lost, precipitation node may send the rainfall amount that fall down from a fixed time instant. With this proposition, the node sends an aggregated historical value that contains the data acquired during the communication interval plus some previous measurements. A beginning time instant t_0 is fixed for every day. The historical measurement is the sum of all the acquired measurements of the node from t_0 up to new communication instant t_i. That is equal to the content of its last message sent plus the new acquired rainfall amount per communication interval. We call this new type of value the "**rainfall amount per day**". This proposition has the advantage to recover precipitation measurements when communication may lose packets. This proposition has the drawback to hide some evolution of the rainfall amount per day. Let imagine that we fix the beginning time instant t_0 at 00:00 for every day. The value sent at 01/xx/20xx 23:55 can be very high because it is the sum of the precipitations for nearly the whole day. The value sent at 02/xx/20xx 00:05 may be very small, compared to the previous one, because it is just the sum of few measurements from 00:00 to 00:05 of the day 02/xx/20xx. That means that the precipitation entity has more chances to change its state at the end of the day and few chances to change its state at the beginning of the day. However, a flood event can occur anytime during day or night. In order to avoid this drawback, the historical value may be computed from a sliding window instead of a fixed time interval.

We will explain the sliding window. First, let us describe the time series: Time series $T=t_1, t_2,\ldots, t_m$ which are a set of scalar observations ordered by time. Since we focus on the aggregation values of short subsections of time series T, we call these short subsections of time series: subsequences. Subsequence C of time series T is a contiguous sampling $t_p, t_{p+1},\ldots,t_{p+n-1}$, where p is an arbitrary position in the time series, such as that: 1 p $m-n+1$. The number n m. Typically, subsequences are extracted from a time series with the use of a sliding window:

Sliding window subsequences, for a time series T of length m and a user-defined subsequence length n, are all possible subsequences of T that can be found by sliding a window of size n across T.

In our dissertation, we only focus on two aggregation values: Sum and Max. So, the sum formula for the sliding window is:

Equation 4.1: Sum of sliding window function

$$SumObs(T,p) = \sum_{i=0}^{n-1} Obs(t_{p+i}) \quad such\ as\ p = 1,2,\ldots,(m-n+1)$$

Where *Obs(t)* correspond to the observation of a phenomenon at a time *t*. The phenomenon could be related to the entity Precipitation or Watercourse or Outlet, etc.

SumObs(T,p) is the sum of the observations for a specific subsequence of the time series *T*. The length of the subsequence is *n* and it starts at the time t_p.

So the Max formula for the sliding window is:

Equation 4.2: max of sliding window function

$$MaxObs(T,p) = MAX_{i=0}^{n-1}\ Obs(t_{p+i})\ such\ as\ p = 1, 2, \dots, (m - n + 1)$$

Where *Obs(t)* correspond to the observation of a phenomenon at a time *t*. The phenomenon could be related to the entity Precipitation or Watercourse or Outlet, etc.

MaxObs(T,p) is the maximum of the observations for a specific subsequence of the time series *T*. The length of the subsequence is *n* and it starts at the time t_p.

Our last proposition is also based on an historical value. We use a sliding window (or a fixed time interval) instead of a fixed time instant. The historical measurement is the sum of all the acquired measurements from the fixed time interval. For example, we fix the time interval at 24 hours. We assume that the next communication time instant will be 02/xx/20xx 00:00. So the precipitation node aggregate its measurements from 01/xx/20xx 00:00 to 02/xx/20xx 00:00. If the next communication time instant is 02/xx/20xx 00:01, the node aggregates its measurements from 01/xx/20xx 00:01 to 02/xx/20xx 00:01. We call this new type of value "**rainfall amount per fixed time interval**". This time interval is composed of several hours. This proposition has the advantage to detect a flood event anytime of a day. However, if two precipitation nodes, configured with the time interval of 2 hour s, have two different communication frequencies, then DSS will receive rainfall amount values started from different time instants. For example, node *Xa* sends its "rainfall amount per fixed time interval" at 10:01 and node *Xb* sends its "rainfall amount per fixed time interval" at 10:03. The DSS has two types of historical values. Node *Xa* aggregates rainfall amount values started from 08:01 and node *Xb* aggregates rainfall amount values from 08:03. We assume that if the time interval is very large compare to the communication frequency, our final evaluation of Precipitation state change will suffers little impact.

Then, in the following, we will present the aggregation functions applied by the DSS on the message sent by the precipitation nodes.

4.2.3.1.2 DSS Aggregation Function

As mentioned previously, there are several precipitation nodes deployed in the watershed. Thus, the DSS has to aggregate all the last values received by the different precipitation nodes. To evaluate the flood risk, we need to evaluate the total amount of precipitations that falls on the watershed during a certain time period. We call this value "**precipitation watershed**". To establish this value, the DSS will apply a sum function to aggregate all the last received

values. The equation 4.3 presents the DSS aggregation function applied on the precipitation node messages.

Equation 4.3: DSS aggregation function applied on the precipitation node messages

$$Precipitation_{watershed} = \sum_{i=1}^{nbPrecipitationNode} PrecipitationNode_i$$

Where

nbPrecipitationNode is the number of precipitation nodes deployed in the watershed.

$PrecipitationNode_i$ is the last value sent by the precipitation node *i* to the DSS. It can be "rainfall amount per communication interval" or "rainfall amount per day" or "rainfall amount per fixed time interval".

$Precipitation_{watershed}$ is the value used by the DSS to deduce the Precipitation entity state change. According to the configuration of the precipitation nodes, it may be the total amount of precipitation that falls on the watershed during the communication interval ("rainfall amount per communication interval") or the total amount of precipitation that falls on the watershed from a fixed time instant ("rainfall amount per day") or the total amount of precipitation that falls on the watershed during a fixed time interval ("rainfall amount per fixed time interval")

4.2.3.1.3 Configuration Examples

We will illustrate some different configurations using a sample of real history data of the Orgeval watershed (c.f. Figure 5.1). We use three precipitation nodes that correspond to three weather stations located on the Orgeval watershed:

- Station Melarchez-P35,
- Station Boissy-P28,
- Station Boissy-Meteo.

Table 4.2.1 presents examples of various computed values when precipitation nodes compute their "**rainfall amount per day**". The beginning time instant t_0 is fixed at 00:00 for all days. The communication frequency of the nodes is fixed to one communication per hour. The acquisition frequency of the node is fixed to one acquisition per minute. The Table 4.2.1 is organized as follows:

- The first column, entitled "Time Interval", presents the communication time interval.
- The second column, entitled "Nodes", contains the names of the weather stations,
- The third column, entitled "Acquired value", presents the sum of the acquired measurements made by the node during the communication time interval. This value is indeed equal to the "rainfall amount per communication interval",
- The fourth column, entitled "Previous value", presents the "rainfall amount per day" previously sent by the node,

- The fifth column, entitled “PrecipitationNode”, is the value sent by the node to the DSS. It represents an updated value of “rainfall amount per day”,
- The sixth column, entitled “$Precipitation_{watershed}$”, is the value computed by the DSS based on the last received messages.

The unit of all the measurements values is the millimeter (mm).

As an example, the first line of Table 4.2.1 presents the value computed during the time interval from 01/02/2008 22:00 to 01/02/2008 23:00. During this time interval, the “Melarchez-P35” station/node has acquired some measurements. The sum of all these measurements is equal to 0. No rain falls down during this time interval. The last value sent by this node is equal to 27.584. This is the sum of rainfall amount acquired by the node from 01/02/2008 00:00 to 01/02/2008 22:00. Thus this node will send the value 27.584 to the DSS at 01/02/2008 23:00. This value presented in column entitled “PrecipitationNode” is indeed the sum of the two previous column values. The DSS during the time interval has received three values from the three different precipitation nodes. Then, it sums all these values and obtains the value 91.139.

As you may notice, there is a huge difference between the resulting values computed by the DSS when the communication interval is before or after the beginning time instant 00:00 (see lines 2 and 3). The precipitation watershed value decreases from 91.579 to 0.19. Moreover, during the day, the precipitation watershed value computed by the DSS follows always the same evolution. Starting from the beginning time of the day, this value is always increasing and never decreasing.

Table 4.2.1: Example of “rainfall amount per day” computation using Orgeval watershed data

Time Interval	Nodes	Acquired Value	Previous Value	PrecipitationNode	$Precipitation_{watershed}$
01/02/2008 22:00 - 01/02/2008 23:00	Melarchez-P35	0	27.584	27.584 (0 + 27.584)	91.139
	Boissy-P28	0	29.155	29.155 (0 + 29.155)	
	Boissy-Meteo	0	34.4	34.4 (0 + 34.4)	

01/02/2008 23:00 - 02/02/2008 00:00	Melarchez-P35	0.167	27.584	27.751 (0.167 + 27.584)	91.579
	Boissy-P28	0.073	29.155	29.228 (0.073+29.155)	
	Boissy-Meteo	0.2	34.4	34.6 (0.2+34.4)	
02/02/2008 00:00 - 02/02/2008 01:00	Melarchez-P35	0.19	0	0.19 (0.19 + 0)	0.19
	Boissy-P28	0	0	0 (0 + 0)	
	Boissy-Meteo	0	0	0 (0 + 0)	
02/02/2008 01:00 - 02/02/2008 02:00	Melarchez-P35	0	0.19	0.19 (0 + 0.19)	0.39
	Boissy-P28	0	0	0 (0 + 0)	
	Boissy-Meteo	0.2	0	0.2 (0.2 + 0)	

Table 4.2.2 presents an example of various generated values when precipitation nodes use the "**rainfall amount per fixed time interval**" computation. The time interval is fixed to 24h. The communication frequency of the nodes is fixed to one communication per hour. The sample frequency of the node is fixed to one sample per minute. The Table 4.2.2 is organized as follows:

- The first column, entitled "Time Interval", presents the communication time interval.
- The second column, entitled "Nodes", is the names of the weather stations,
- The third column, entitled "Acquired value", presents the sum of the acquired measurements made by the node during the communication time interval,
- The fourth column, entitled "Rainfall amount last 23h", presents the sum of the total rainfall amount fall during the last 23h,
- The fifth column, entitled "PrecipitationNode", is value sent by the node to the DSS. It represents the update value of "rainfall amount per fixed time interval",

- The sixth column, entitled "$Precipitation_{watershed}$", is the value computed by the DSS based on the last messages received.

The unit of the last four columns values above is millimeter (mm).

Table 4.2.2: Example of "rainfall amount per fixed time interval" computation using Orgeval watershed data

Time Interval	Nodes	Acquired Value	Rainfall Amount Last 23 Hours	PrecipitationNode	$Precipitation_{watershed}$
01/02/2008 22:00 - 01/02/2008 23:00	Melarchez-P35	0	27.584	27.584 (0 + 27.584)	91.139
	Boissy-P28	0	29.155	29.155 (0 + 29.155)	
	Boissy-Meteo	0	34.4	34.4 (0 + 34.4)	
01/02/2008 23:00 - 02/02/2008 00:00	Melarchez-P35	0.167	27.584	27.751 (0.167 + 27.584)	91.579
	Boissy-P28	0.073	29.155	29.228 (0.073+29.155)	
	Boissy-Meteo	0.2	34.4	34.6 (0.2+34.4)	
02/02/2008 00:00 - 02/02/2008 01:00	Melarchez-P35	0.19	25.921	26.111 (0.19 + 25.921)	87.172
	Boissy-P28	0	28.261	28.261 (0 +28.261)	
	Boissy-Meteo	0	32.8	32.8 (0 + 32.8)	
02/02/2008 01:00 - 02/02/200	Melarchez-P35	0	25.238	25.238 (0 + 25.238)	84.416
	Boissy-P28	0	27.378	27.378 (0 + 27.378)	

8 02:00	Boissy-Meteo	0.2	31.6	31.8 (0.2 + 31.6)	

The first line of Table 4.2.2 presents the value computed during the time interval from 01/02/2008 22:00 to 01/02/2008 23:00. During this time interval the "Melarchez-P35" node has acquired some measurements. The sum of all these measurements is equal to 0. No rain falls down during this time interval. This node computes the rainfall amount that falls down during the last 23h is equal to 27.584 (from 31/01/2008 23:00 to 01/02/2008 22:00). Thus, this node will send the value 27.584 to the DSS at 01/02/2008 23:00. This value presented in column entitled "PrecipitationNode" is indeed the sum of the two previous column values. The DSS during the time interval has received three values from the three different precipitation nodes. Then, it sums all these values and obtains the value 91.139.

The precipitation watershed values of Table 4.2.2 can increase or decrease in the same day and do not follows the same evolution.

4.2.3.1.4 Deduction of Precipitation Entity State (P)

The DSS computes the state of the Precipitation entity by comparing the "$Precipitation_{watershed}$" value with a threshold. The threshold is depending on the watershed. It is computed from the observation data of the watershed archive and is fixed by experimentation. We call the threshold "$Th_{Precipitation}$". So, we can implement our first step to deduce the P entity state based on the comparison between $Precipitation_{watershed}$ and $Th_{Precipitation}$.

4.2.3.2 Watercourse Node Configuration and Watercourse Entity (W)

To evaluate the flood risk, we need to evaluate the increase of flow rate values for all the tributaries of the watershed. The watercourse node is, for example, a stream gauge and acquires a water flow rate value at a specific time instant. To evaluate risk, we need to compute the slope between two flow rate values acquired by the same node. The equation 4.2 provides the computation of the slope.

Equation 4.2: Slope of water flow rate

$$WatercourseSlope_i(t_k) = \frac{Waterflow_i(t_k) - Waterflow_i(t_j)}{t_k - t_j}$$

such as $t_j < t_k$

$Waterflow_i(t_k)$ represents the flow rate value measured by this node i at the time instant t_k. $WatercourseSlope_i(t_k)$ represents the slope computed between two measurements of the node i. The first and the last measurements were made respectively at the time instants t_j and t_k.

If the slope is above 0 then it means that the flow rate increase. If the slope is under 0 then it means that the flow rate decreases. We want to detect a sharp increase of the flow rate as soon as possible. That is to say we want to detect a high positive value of a water flow rate slope.

If the slope is above 0 then it means that the flow rate increase. If the slope is under 0 then it means that the flow rate decreases. We want to detect a sharp increase of the flow rate as soon as possible. That is to say, we want to detect a high positive value of a water flow rate slope.

We will present different configuration of the system depending on which component (node or DSS) will compute the slope.

4.2.3.2.1 Watercourse Node Aggregation Function

Our first proposition is that a watercourse node sends a water flow rate value and the DSS computes the slope related to these node values. If the communication frequency is different from the acquisition frequency, the node has to select one value from all the water flow rate values it has acquired. The first configuration of the watercourse node is to send its last water flow rate value to the DSS. We call this value: the "**last water flow rate**". One of the drawbacks of this proposition is that if an increase, followed by a decrease of flow rate, happens during the communication interval, this evolution of flow rate will not be taken into account by the system.

Thus, another possibility is that the node sends the maximum of the values acquired during the communication interval. We call this value; the "**max water flow rate**". The drawback of this proposition that it is not possible to detect all the increase of flow rate when there is more than one increase that happen during a communication interval.

Another solution is that a watercourse node computes the slope between two successive acquired flow rate values and sends to the DSS a slope value. If the communication frequency is different from the sample frequency, the node has to select one slope value from all the slopes values it has computed during the communication interval. The node can send the average slope value and send it to the DSS. We call this value: the "**average slope of water flow rate per communication interval**". Note that the average function will be impacted if the water flow rate decreases a little bit during the communication interval.

Another proposition is that the watercourse node computes the max slope value during the communication interval and sends it to the DSS. We call this value: the "**max slope of water flow rate per communication interval".** Note that the maximum function will not be impacted by a flow rate decrease during a communication interval. Due to the fact that the flow rate can have a quick evolution, the maximum function applied on the slope will have various results during a day interval: for example, from -10 to + 10. The main drawback of this solution is that, the evolution of water flow can be delayed to detect when there is more than one sharp increase of flow rate that happen during a communication interval. For

example, during the communication interval [0h00, 0h02], the evolution of water flow during time interval [0h00, 0h01] is a sharp increase with this slope is called $slope_1$. The evolution of water flow during time interval [0h01, 0h02] is also a sharp increase with this slope is called $slope_2$. We assume that the value of $slope_2$ is bigger than the value of $slope_1$. Thus, watercourse node will send $slope_2$ to the DSS at time instant 0h02. DSS will detect the evolution of water flow at time instant 0h02. However, the fact is that the beginning of the evolution of the water flow happens at time instant 0h01. So, using this solution can be less reactive to detect the evolution of water flow.

The main drawback of all these solutions is that the loss of packets will imply the loss of water flow rate measurements or slope computations. A solution, as proposed in the configuration of Precipitation node, is that the watercourse node sends an historical value of the slope. The node computes the max of slope from a sliding window. We fix a time interval that is longer than the communication interval. We call this value: the "**max slope of water flow rate per fixed time interval".** We expect that this value will have a more homogeneous evolution during a day interval.

Then, we will present the aggregation functions applied by the DSS on the message sent by the watercourse nodes.

4.2.3.2.2 DSS Aggregation Function

As mentioned previously, there are several watercourse nodes deployed in the tributaries of the watershed. To evaluate the flood risk, we need to evaluate the water flow rate increase of all the tributaries of the watershed. This increase is estimated by the slope between two measurements of water flow rate made by a watercourse node. In fact, there are several watercourse nodes. The DSS has to manage as many slope values as the number of watercourse nodes. Thus, the DSS has to aggregate all the last computed slope values of each node. The value resulting from the aggregation is called "**watercourse watershed**". The first possibility is to use an average function. The equation 4.3 presents the computation of the average between the slopes of each watercourse node.

Equation 4.3: Computation of the watercourse watershed based on the average function

$$Watercourse_{watershed}(t_l) = \frac{\sum_{i=1}^{nbWatercourseNode} WatercourseNode_i(t_k)}{nbWatercourseNode}$$

such as $t_k \leq t_l$

$WatercourseNode_i(t_k)$ is the last computed slope value of the watercourse node i. The value was computed at time instant t_k. Depending of the node configuration, the slope value related to each node can be:

- The slope between two last "**last water flow rate**" values sent by the node i to the DSS,

- The slope between two last "**max water flow rate**" values sent by the node *i* to the DSS,
- The last "**average slope of water flow rate per communication interval**" value sent by the node *i* to the DSS,
- The last "**max slope of water flow rate per communication interval"** value sent by the node *i* to the DSS,
- The last "**max slope of water flow rate per fixed time interval"** value sent by the node *i* to the DSS.

nbWatercourseNode is the number of watercourse nodes that send messages to the DSS.

$Watercourse_{watershed}(t_l)$ is the average between all the last slope values related to the watercourse nodes. t_l is the max between all the time instant where a watercourse slope *i* was computed.

The second possibility is to use a maximum function. The equation 4.4 pr esents the computation of the max slope between the slopes values related to each watercourse node.

Equation 4.4: Computation of the watercourse watershed based on the maximum function

$$Watercourse_{watershed}(t_l) = \max_{i=1}^{nbWatercourseNode} WatercourseNode_i(t_k)$$

such as $t_k \leq t_l$

$WatercourseNode_i(t_k)$ is the last slope value related to the watercourse node *i*. The value was computed at time instant t_k. The value depends on the node configuration like in equation 4.3.

nbWatercourseNode is the number of watercourse nodes that send messages to the DSS.

$Watercourse_{watershed}(t_l)$ is the maximum between all the last slope values related to the watercourse nodes. t_l is the max between all the time instants where a watercourse slope *i* was computed.

4.2.3.2.3 Configuration Examples

We will illustrate some different configurations using a sample of real history data of the Orgeval watershed. We use two watercourse nodes that correspond to two stream gauges located on two tributaries of the watercourse:

- Les Avenelles station,
- Melarchez station.

The sample and communication frequencies are fixed to one measurement or communication per minute.

Tables 4.2.3, 4.2.4 and 4.2.5 present examples of various computed values depending on the watercourse node configuration. These tables illustrate how slope values are computed based on the nodes messages. In addition, they help to verify whether average and max functions applied by the DSS are impacted if the water flow rate decreases. As indicated previously, for simplification reason, the communication frequency is equal to the acquisition one. All the tables have the same structure:

- The first column, entitled "Time", is the communication or the acquisition time interval. The acquisition and the communication are made at the ending point of the interval,
- The second column, entitled "Nodes", indicates the name of the stream gauge,
- The third column, entitled "Acquired value", is the water flow rate value acquired by the node. The unit of the measurement is liter per second,
- The fourth column, entitled "Message", is the value, in liter per second, sent by the node,
- The fifth column, entitled "WatercourseNode", is the slope related to the node measurement,
- The sixth column, entitled "$Watercourse_{watershed}$", is the aggregation value computed by the DSS.

The purpose of Table 4.2.3 is to highlight one configuration where watercourse nodes send their "last waterflow rate" values to the DSS. In case of the communication frequency is equal to the acquisition one; the nodes send all their acquired measurements to the DSS. The DSS computes the slope based on the last messages received. Then, the DSS computes the average between the two slopes of Les Avenelles and Melarchez stations.

Table 4.2.3: Example of watercourse node configuration ("last water flow rate" + average)

Time	Nodes	Acquired Value	Message	WatercourseNode	$Watercourse_{watershed}$
05/02/2008 21:59	Avenelles	1209.5	1209.5		
	Melarchez	356.490	356.490		
05/02/2008 22:00	Avenelles	1210	1210	0.5	0.505 $\frac{0.5+0.510}{2}$
	Melarchez	357	357	0.510 $\frac{357-356.490}{1}$	
05/02/2008 22:01	Avenelles	1210.5	1210.5	0,5	1.95
	Melarchez	360.4	360.4	3.4	
05/02/2008 22:02	Avenelles	1211	1211	0.5	1.95
	Melarchez	363.8	363.8	3.4	

For example, in row 5 of the "Nodes" column of Table 4.2.3, the Les Avenelles station acquires the water flow rate value 1210.5 at 05/02/2008 22:01. It sends this value to the DSS at 05/02/2008 22:01. The DSS computes the slope of the Les Avenelles station based on the two last values received: = 0.5. The slope is presented in the column "WatercourseNode". The Melarchez station acquires the water flow rate value 360.4 at 05/02/2008 22:01. It sends this value to the DSS at 05/02/2008 22:01. The DSS computes the slope of the Melarchez station based on the two last received messages: = 3.4.

The DSS computes the average between the "$Watercourse_{watershed}$" value of the two stations:

$= 1.95$.

Table 4.2.4 presents a second configuration where the watercourse nodes send the "last waterflow rate" values to the DSS. The DSS computes the slope based on the last messages received. Then, the DSS computes the maximum between the two slopes of Les Avenelles and Melarchez stations.

Table 4.2.4: Example of watercourse node configuration ("last water flow rate" + max)

Time	Nodes	Acquired Value	Message	WatercourseNode	$Watercourse_{watershed}$
05/02/2008 21:59	Avenelles	1209.5	1209.5		
	Melarchez	356.490	356.490		
05/02/2008 22:00	Avenelles	1210	1210	0.5	$0.510 \quad \frac{357 - 356.490}{1}$
	Melarchez	357	357	$0.510 \quad \frac{357 - 356.490}{1}$	
05/02/2008 22:01	Avenelles	1210.5	1210.5	0.5	3.4
	Melarchez	360.4	360.4	3.4	
05/02/2008 22:02	Avenelles	1211	1211	0.5	3.4
	Melarchez	363.8	363.8	3.4	

For example, in row 5 of the "Nodes" column of Table 4.2.4, the Les Avenelles station acquires the water flow rate value 1210.5 at 05/02/2008 22:01. It sends this value to the DSS at 05/02/2008 22:01. The DSS computes the slope of the Les Avenelles station based on the two last values received: = 0.5. The slope is presented in the column "WatercourseNode". The Melarchez station acquires the water flow rate value 360.4 at 05/02/2008 22:01. It sends this value to the DSS at 05/02/2008 22:01. The DSS computes the slope of the Melarchez station based on the two last messages received: = 3.4. The DSS computes the maximum between the "$\text{Watercourse}_{\text{watershed}}$" values of the two stations: 3.4.

Comparing Tables 4.2.3 and 4.2.4, we can see that the value of "$\text{Watercourse}_{\text{watershed}}$" at 05/02/2008 22:01, is 1.95 in Table 4.2.3. This value is smaller than the value (3.4) sent by the Melarchez station. But, in Table 4.2.4, the two values are equal. That means when there is a quick evolution of the water flow rate, the slope of the water flow rate in watercourse computed by the average function applied in the DSS has increased more smoothly than the one computed by the max function applied in the DSS.

The Table 4.2.5 presents a configuration where the nodes compute the slope value. Notice that the communication frequency is equal to the acquisition one. So, the node does not apply any aggregation function. Thus, the node computes the slope based on the two last acquired water flow rate values and sends this slope value to the DSS. Thus, the "watercourse node" value corresponds either to the "average slope of water flow rate per communication interval" or the "max slope of water flow rate per communication interval". Due to the fact that the node computes and sends the slope to the DSS, the columns entitled "watercourse node" and "message" are merged. The DSS receives the slope (Watercourse node value) from the node and then computes the maximum between the Watercourse node values of each node.

Table 4.2.5: Example of watercourse node configuration ("max slope of water flow rate per communication interval" + max)

Time	Nodes	Acquired	WatercourseNode/	$\text{Watercourse}_{\text{watershed}}$

		value	Message	
05/02/2008 21:59	Avenelles	1209.5		
	Melarchez	356.490		
05/02/2008 22:00	Avenelles	1210	0.5	0.510 $\frac{357 - 356.490}{1}$
	Melarchez	357	0.510 $\frac{357 - 356.490}{1}$	
05/02/2008 22:01	Avenelles	1210.5	0.5	3.4
	Melarchez	360.4	3.4	
05/02/2008 22:02	Avenelles	1211	0.5	3.4
	Melarchez	363.8	3.4	

For example, in row 5 of "Node" column of Table 4.2.5, the Les Avenelles station acquires the water flow rate value 1210.5 at 05/02/2008 22:01. The node computes the slope between its two last measurements: = 0.5. Then, it sends this slope value to the DSS. The Melarchez station acquires the water flow rate value 360.4 at 05/02/2008 22:01. It computes the slope between its two last measurements: = 3.4. Then, it sends this slope value to the DSS. The DSS receives a slope value from each node. The slope is presented in column "WatercourseNode / Message". The DSS computes the maximum between the *WatercourseNode* values of the two stations: 3.4.

4.2.3.2.4 Deduction of Watercourse Entity State (W)

The DSS computes the state of the Watercourse entity by comparing the $Watercourse_{watershed}$ value with a threshold. The threshold is depending on the watershed. It is computed from the observation data of the watershed archive and thus, is fixed by experimentation. We call this threshold: $Th_{Watercourse}$. So, we can implement our first step to deduce the W entity state based on a comparison between the value of $Watercourse_{watershed}$ and the threshold $Th_{Watercourse}$.

4.2.3.3 Outlet Node Configuration and Outlet Entity (O)

Like watercourse node, outlet node samples water flow rate values of the outlet at a specific time instant using, for example, a stream gauge. To evaluate flood, we need to compute the slope between two water flow rate values acquired by the outlet node. The equation of the computation of the slope is similar to equation 4.2 in section 4.2.3.2.

Equation 4.4: Slope of the Outlet in the watershed

$$OutletSlope(t_l) = \frac{Waterflow(t_l) - Waterflow(t_k)}{t_l - t_k}$$

such as $t_k < t_l$

Waterflow(t_l) represents the flow rate measured by the outlet node at the time instant t_l.

Waterflow(t_k) represents the flow rate measured by the outlet node at the time instant t_k.

OutletSlope(t_l) represents the slope computed between two measurements of the node. The first and the last measurements were made respectively at the time instants t_k and t_l.

We also want to detect a sharp increase of the water flow rate as soon as possible, which corresponds to a high positive value of a water flow rate slope.

Outlet node and Watercourse nodes are both water flow nodes. So, they have the same node aggregation function. The advantages and drawbacks of each possibility are also the same as Watercourse node. However, here, there is only one Outlet node, no DSS aggregation function has to be considered. The name of this variable is called: the "**Outlet watershed**".

4.2.3.3.1 Outlet Node Aggregation Function

There are five possibilities. The first configuration for the outlet node is to send its last water flow rate value to the DSS, named the "**last water flow rate**" (per communication interval).

Another possibility is that the outlet node sends the maximum of the values acquired during the communication interval, named the "**max water flow rate**" (per communication interval).

The third solution is that the outlet node computes the average slope value and sends it to the DSS, named the "**average slope of water flow rate per communication interval**".

Another proposition is that the outlet node computes the max slope value during the communication interval and sends it to the DSS. We call this value: the "**max slope of water flow rate per communication interval".**

The last one is to send an historical value of the slope. We call this value: the "**max slope of water flow rate per fixed time interval**".

4.2.3.3.2 Deduction of the Outlet Entity State (O)

The DSS computes the state of the Outlet entity by comparing the $Outlet_{watershed}$ value with a threshold. The threshold is depending on the watershed. It is computed from the observation data of the watershed archive and is thus fixed by experimentation. We call this threshold: "Th_{Outlet}". Consequently, we can implement our first step to deduce the O entity state by comparing the value of $Outlet_{watershed}$ and the threshold "Th_{Outlet}".

The Figure 4.2.5 is an improvement of the previous formalization of the Flood context.

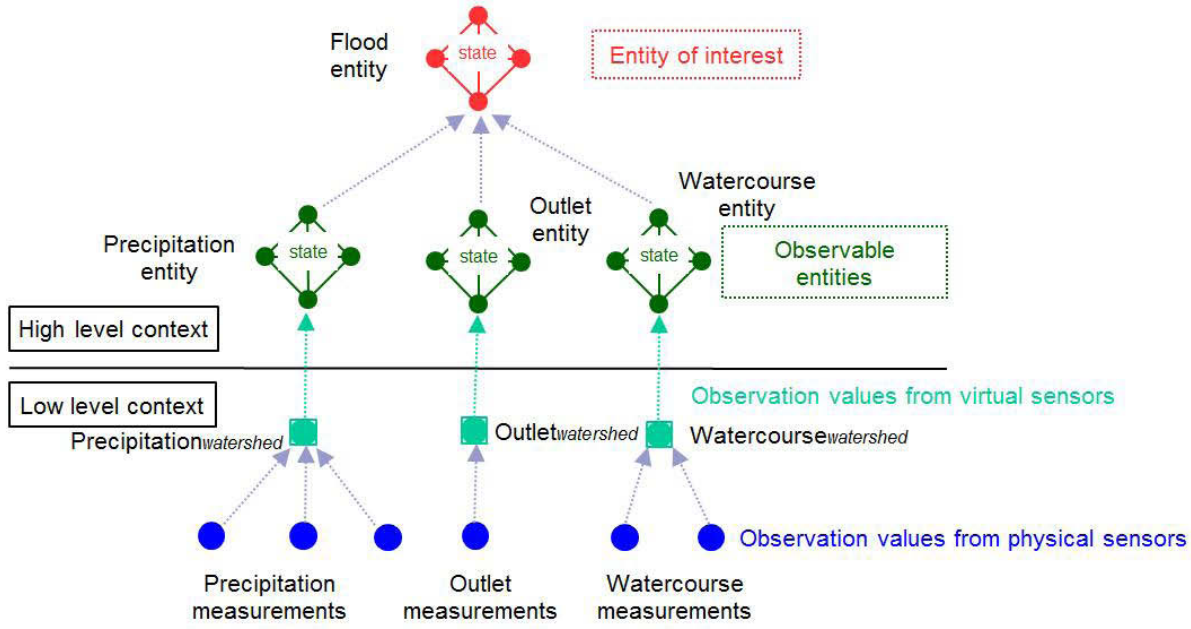


Figure 4.2.5: Flood context

Figure 4.2.5 identifies the different observation values provided by sensors. Blue dots are measurements acquired by the physical sensors: weather (pluviometer) stations and stream gauges. These measurements are aggregated and treated through several processes to calculate the three observation values: $Precipitation_{Watershed}$, $Watercourse_{Watershed}$ and $Outlet_{Watershed}$. They are outputs of virtual sensors and they are represented by green squares.

4.2.3.4 All the Possible Configurations for the Sensor Nodes

Based on the different possible node and DSS aggregation functions presented in sections 4.2.3.1 to 4.2.3.3, Table 4.2.6 resumes some possible configurations of flood context-aware system and is organized as follows:

- The first column is the identifier of the system configuration,
- The second column is the configuration of the Precipitation nodes. The aggregation function used by the Precipitation node is always the "sum function",
- The third column is to represent the Watercourse node aggregation functions to aggregate the acquired measurements,
- The fourth column is the configuration of the Watercourse nodes,
- The fifth column is the aggregation function used by the DSS to compute the slope of Watercourse of watershed from the measurements sent by Watercourse nodes,
- The sixth column is the configuration of Outlet node. We can see in the Table 4.2.6, we use different colors to represent different configurations of Precipitation node, Watercourse node, Outlet node and the aggregation function used by DSS for the three nodes.

Table 4.2.6: Possible configurations for the sensor nodes

	Configuration of "Precipitation nodes"	Aggregation function used by "Watercourse nodes"	Configuration of "Watercourse nodes"	Aggregation function used by DSS for "Watercourse nodes"	Configuration of "Outlet node"
Config 1	Rainfall amount per communication interval		Last water flow rate	Average Function	Last water flow rate
Config 2	Rainfall amount per day		Last water flow rate	Average Function	Last water flow rate
Config 3	Rainfall amount per fixed time interval		Last water flow rate	Average Function	Last water flow rate
Config 4	Rainfall amount per fixed time interval		Last water flow rate	Max Function	Last water flow rate
Config 5	Rainfall amount per fixed time interval	Max Function	Max water flow rate	Average Function	Max Water flow rate

Config 6	Rainfall amount per fixed time interval	Max Function	Max water flow rate	Max Function	Max Water flow rate
Config 7	Rainfall amount per fixed time interval	Average Function	average slope of water flow rate per communication interval	Average Function	average slope of water flow rate per communication interval
Config 8	Rainfall amount per fixed time interval	Average Function	average slope of water flow rate per communication interval	Max Function	average slope of water flow rate per communication interval
Config 9	Rainfall amount per fixed time interval	Max Function	max slope of water flow rate per communication interval	Average Function	max slope of water flow rate per communication interval
Config 10	Rainfall amount per fixed time interval	Max Function	max slope of water flow rate per communication interval	Max Function	max slope of water flow rate per communication interval
Config 11	Rainfall amount per fixed time interval	Max Function	max slope of water flow rate per fixed time interval	Average Function	max slope of water flow rate per fixed time interval
Config 12	Rainfall amount per fixed time interval	Max Function	max slope of water flow rate per fixed time interval	Max Function	max slope of water flow rate per fixed time interval

The configurations which are in bold format are planned to be implemented. The reasons are:

- Based on the introduction of section 4.2.3.1, for Precipitation Entity, using the configuration of Precipitation node: **rainfall amount per fixed time interval** is better than the other ones. Moreover, the aggregation function implied on DSS is the sum function,
- Based on the introduction of section 4.2.3.2, there are two aggregation functions involved on the DSS to compute the slope of Watercourse Entity at a specific time interval: average and max functions. Max function is better than the average one as we mentioned. Compared with max function, a quick evolution of the water flow rate will increase more smoothly when the DSS uses the average function during a communication interval. So, the **max function** is used.

In order to verify the advantages and drawbacks of the different chosen configurations, we evaluate them using the three following computed values, presented in section 4.2.3.2 and 4.2.3.3, for the Watercourse nodes and the Outlet node:

- Last water flow rate

- Max slope of water flow rate per communication interval,
- Max slope of water flow rate per fixed time interval.

4.3 Flood and Node Context-aware System

The flood context-aware system and the node context-aware system are merged into the flood and node context-aware system. In this new system, the context is the fusion of the environmental and the WSN contexts. The new context has two entities of interest:

- EEI (Flood entity),
- SNEI (Node entity).

Several observable entities are considered such as:

- Precipitation entity,
- Watercourse entity,
- Outlet entity,
- Node entities.

These entities will be described more precisely in the next sections.

Remember that the goal of this system is to provide a service for the context consumers to cope with flood monitoring in a watershed and energy state of wireless sensor nodes. As mentioned in section 3.4, there is a priority issue for these two contexts. The priority is determined by the end users. In our context-aware system, flood monitoring is sometimes more important than the "survival" of a sensor node. For example, when the *F* entity state is "Risky (*F3*)" or "Flood (*F4*)", and *N* entity state is "Critical (state *NB*)", context consumers e.g., users will let the WSN of flood monitoring system continue to work instead of reducing its communication frequency.

The context of this system is composed of several entities shown as below:

- The "Precipitation entity (*P*)" which is an EOE. The "Precipitation entity" has two states: "Normal" and "Risky" (shown in section 4.2.2),
- The "Watercourse entity (*W*)" which is an EOE. The "Watercourse entity" has two states: "Normal" and "Risky" (shown in section 4.2.2),
- The "Outlet entity (*O*)" which is also an EOE. The "Outlet entity" has two states: "Normal" and "Risky" (shown in section 4.2.2),
- The "Flood entity (*F*)" is the EEI of our environmental application. The "Flood entity" is not an EOE. Its state depends on the states of all the EOEs. The "Flood entity" has four states: "Normal", "Rainy", "Risky", and "Flood" (shown in section 4.2.2),

- The “Node entity (*N*)”. There are as many *N* entities as wireless sensor nodes. The “*N* entities” are both SNOE and SNEI. They have two states: “Stable” and “Critical” (shown in section 4.1).

Figure 4.3.1 is a formalization of the fusion of Flood and Node contexts.

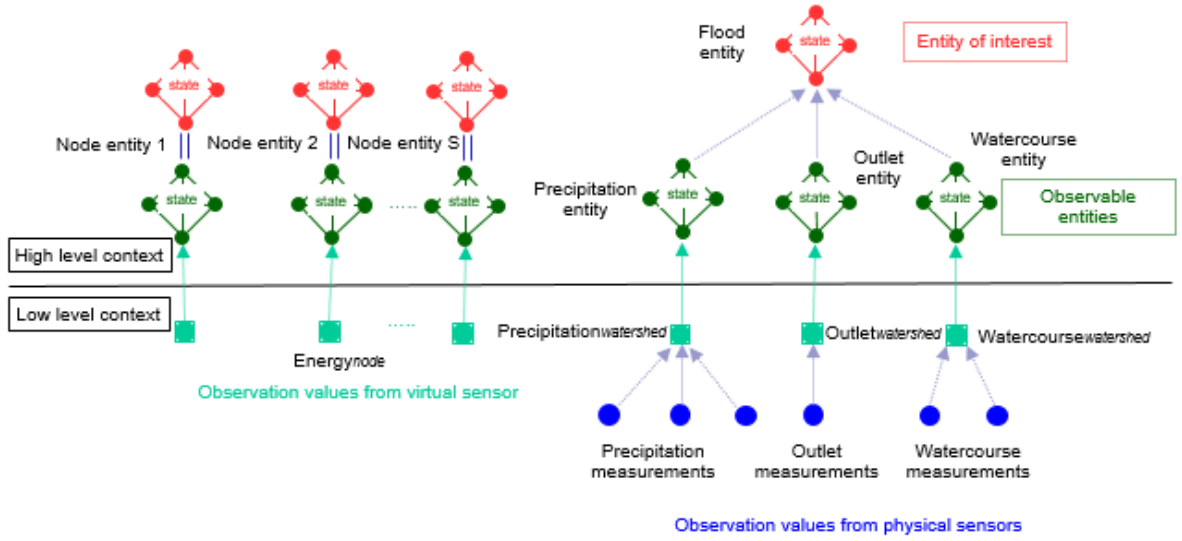


Figure 4.3.1: Flood and node contexts

Figure 4.3.1 presents the entities compositions and the reasoning steps in flood and node context-aware system. In Figure 4.3.1, *S* is the number of virtual sensor nodes considered in the system. $Energy_{node}$ is the measurement acquired from a virtual sensor that simulates the energy level of the battery as indicated by the green squares related to Node entities.

4.4 Adaptive Context-aware Systems

As mentioned in section 3.4, there are three types of adaptive context-aware systems. The first one is a flood adaptive context-aware system. The system can modify its behaviors when, for example, opening the gate of a dam when *F* entity state is “Risky (*F3*)” or “Flood (*F4*)”. However, our work focuses on the adaptation of the wireless sensor nodes in order to better use their limited resources. So, this adaptive context-aware system is not considered here.

The second one is a node adaptive context-aware system. The wireless sensor nodes can reduce their frequencies of sending messages to the DSS when the *N* entity state is “Critical” (state *NB*). Nodes can also increase their frequencies of sending messages to the DSS when the *N* entity state is “Stable” (state *NA*).

The third one is based on the flood and node context-aware system as mentioned in section 4.3. We propose an adaptive context-aware system that can integrate both the flood

phenomenon monitoring and the WSN management. The goal of this system is to be able to adapt the communication frequency of each wireless sensor node based on the Flood entity state and Node entity state. The difference with the two previous adaptive context-aware systems is that the context of this system is the fusion of two different contexts namely flood and node contexts. Moreover, the adaptation is made based on these two different contexts. As mentioned in section 4.3, there is a priority issue applied on these two contexts. Similarly as in the previous context-aware system, the flood context has more priority than the node one.

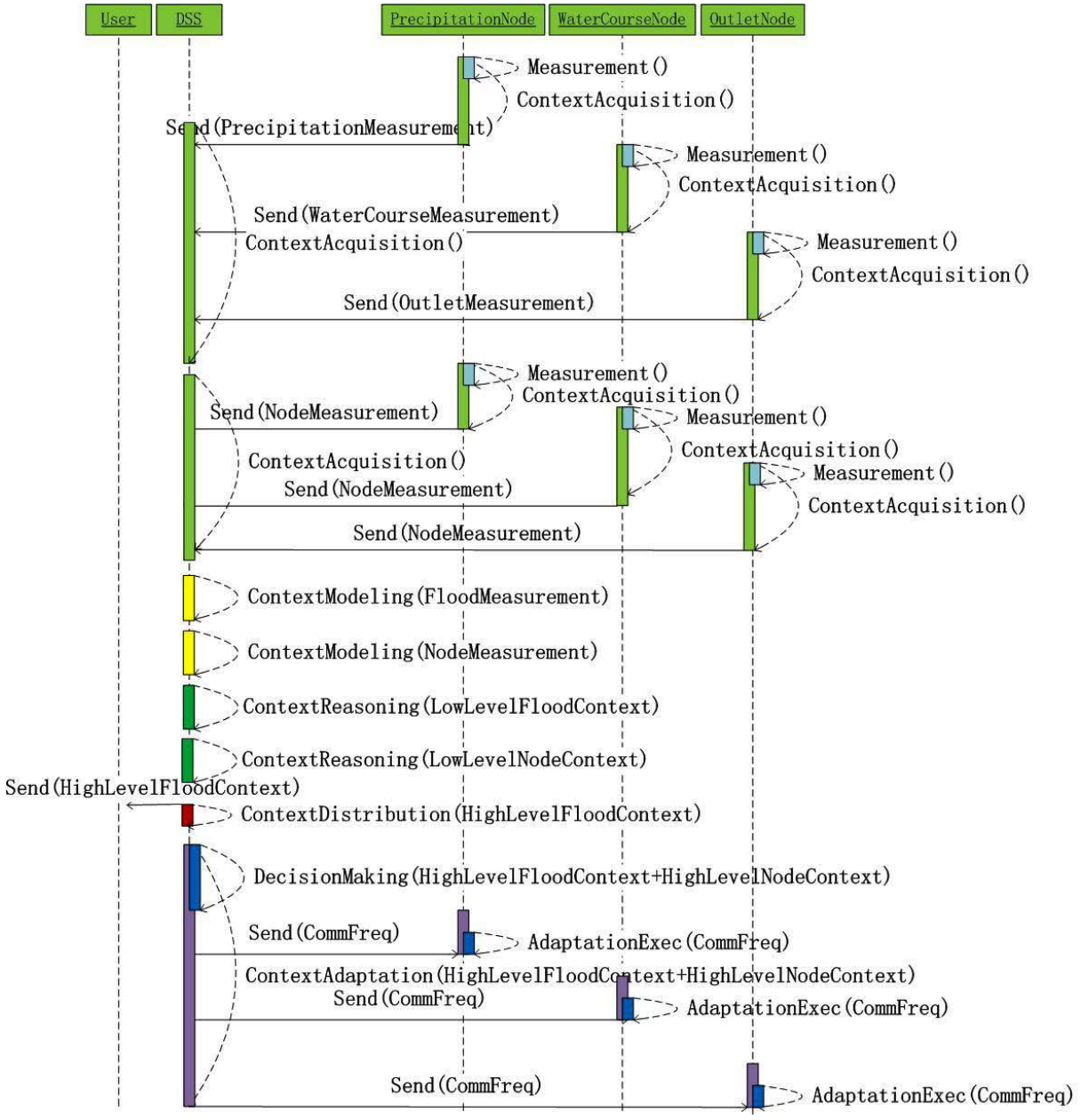


Figure 4.4.1: Sequence diagram of an adaptive context system integrating two different context-aware systems

Figure 4.4.1 identifies the context adaptation based on the flood and node contexts in this adaptation system. In order to show the processes of each context, we present the two

contexts separately in a "chronological" order. In the "context adaptation" process, which is colored in purple in Figure 4.4.1, the DSS establishes the new node configurations based on the high-level context, namely, the new communication frequency, which is deduced from the *F* entity state and the *N* entity state of each node. Then, the DSS correspondingly sends the new communication frequencies to each node respectively. Each node will change their communication frequencies to the new provided one.

5 Modeling

In order to evaluate our formalization of context-aware systems, we use a simulation tool to implement it. This chapter explains how we simulate the processes involved in an adaptive context-aware system. One section is dedicated to one process: acquisition, modeling, reasoning, distribution and adaptation. To begin, we present the configurations of our adaptive context-aware system.

5.1 System

Our simulation system takes as input some real hydrological data. Irstea research institute provides data collected from the French Orgeval watershed (Garnier *et al.*, 2014). First, we present the Orgeval basin and we fix some node configuration parameters in order to adapt the simulation to the Orgeval watershed. Then, we present main functionalities of the simulation tool and the simulation architecture based on JADE platform.

5.1.1 Orgeval Basin

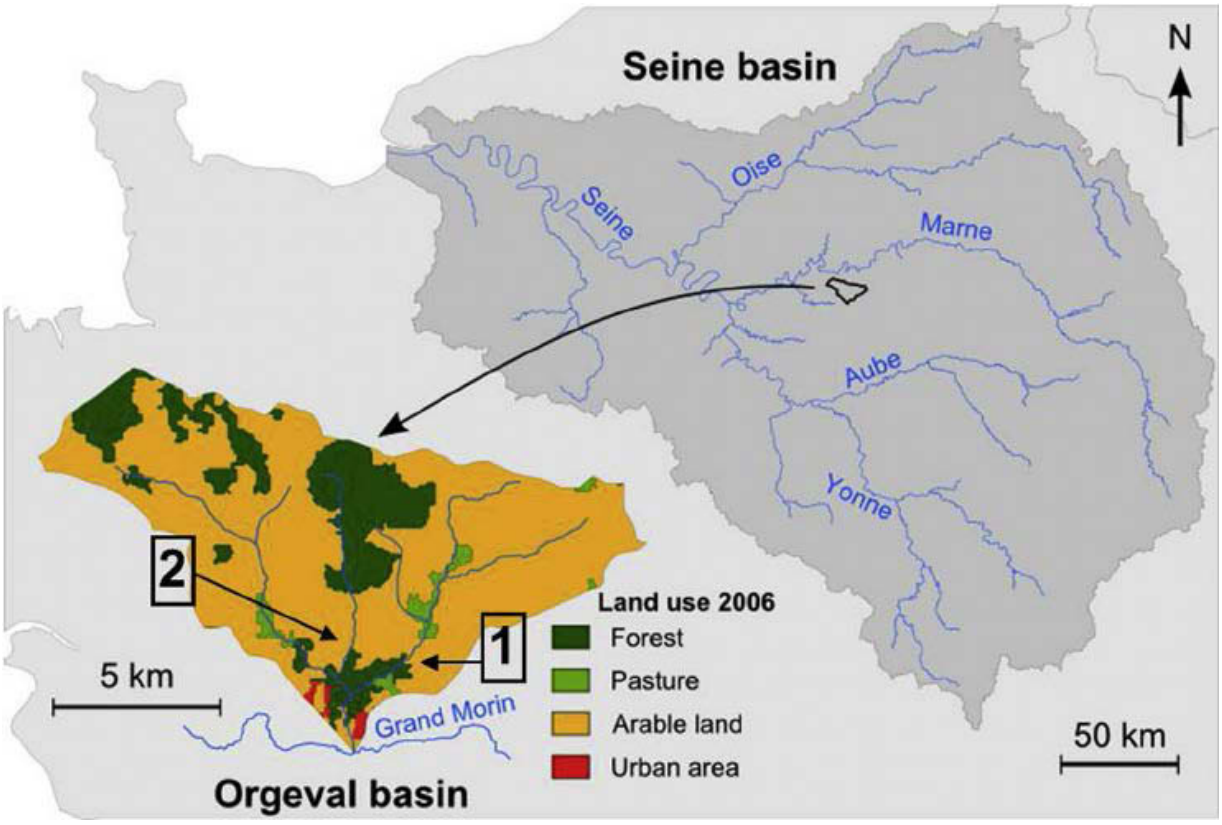


Figure 5.1.1: Location of the Orgeval watershed in the Seine Basin (Garnier *et al.*, 2014)

The Orgeval basin is located in the Brie Area (80 km East from Paris in the Seine-et-Marne department). Orgeval is a tributary of the Grand Morin river, with flood issues. Irstea has studied the hydrological behavior of the Orgeval river, because floods affected the town of

Coulommiers. Orgeval basin is followed by Irstea since 1962. Thus, Irstea has built up a long-term set of hydrological and climatic data by studying the catchment area.

Seven hydrological stations currently distributed throughout the Orgeval basin monitor flow on a continuous basis. Rainfall is currently monitored on a continuous basis at eight points throughout the watershed and around it. From the available data, we select the data of a sub-basin of Orgeval. This sub-basin is monitored by 3 weather stations:

- Melarchez-P35 station,
- Boissy-Meteo station,
- Boissy-P28 station.

The outlet of the sub-basin is Le Theil monitored by a stream gauge having the same name.

Two stream gauges monitored the tributary:

- Les Avenelles station,
- Melarchez station.

Based on the work in (Perlman, 2017), there are many factors that influence the appearance of flood events. These are, for example, precipitation, infiltration, evaporation and storage. These factors are universal in nature and not particular to a single stream. Moreover, we can underline that not all precipitations that fall in a watershed flow out. It needs time for the precipitation to accumulate and to become a surface runoff. The factors, except precipitation, are impacted by the characteristics of the watershed. Therefore, the accumulation time is different according to the considered watershed. Besides, there are many factors that determine the water flow rates of the watercourses and of the outlet of a watershed. For example, land cover and land slope have a great impact on infiltration and on rainfall runoff. Consequently, the time interval of the observation of the water flow rate is also different according to the considered watershed.

(Lilas *et al*., 2012) has modeled a flood prediction system for the sub-basin le Theil of the Orgeval watershed. Their model takes as input the precipitation that falls down in the watershed during 24h. The Orgeval watershed is known as a rapid watershed. When one tributary flows out, the outlet flows out 3 hours or 6 hours later. Based on the characteristics of the Orgeval watershed, we specialize the configuration of the node. For the precipitation node, the duration of the aggregated value "rainfall amount per fixed time" is thus fixed to 24 hours. We create a new aggregated value named **"rainfall amount per 24h".** For the watercourse node and the outlet node configuration, the duration of the aggregated value "max slope of water flow rate per fixed time interval" is fixed to 6 hours. Therefore, we create a new aggregated value named **"max slope of water flow rate per 6h".**

5.1.2 Simulation Architecture

Our simulation is based on t he multi-agent system JADE (Java Agent DEvelopment Framework) (Bellifemine *et al*., 1999) (Bellifemine *et al*., 2005). JADE (Java Agent DEvelopment Framework) is a software framework fully implemented in the Java language. It simplifies the implementation of multi-agent systems through a middleware that complies with the Foundation for Intelligent Physical Agents (FIPA) specifications and through a set of graphical tools that support the debugging and deployment phases (Bellifemine *et al*., 2001). In our simulation tool, some agents should integrate reasoning capability in order to deduce the high-level context.

In this section, we will present the functionalities of JADE tool and our simulation architecture.

First, the three main functionalities of JADE are:

- Agent communication: exchange of messages between agents,
- Message content modeling by one ontology: ontology models the contents of exchanged messages between agents. Thus the interoperability issues between sensor data will be solved by the JADE ontology. We will also use this ontology to model the context,
- Integration with other tools: possible use of tools like rule engine as a decision component of an agent. We decide to use the Jess rule engine to implement the reasoning capability of an agent (Friedman-Hill *et al*., 2003). Notice that there are a lot of systems using and implementing the same things. Like the works in (Subercaze *et al*., 2011) and (Bittencourt *et al*., 2009) integrate JADE with different rule engines like Jena.

In our simulation, we implement the functionalities from Figure 5.1.1. We develop an ontology to model the content of the messages exchanged between agents. This ontology takes its elements from others existing ontologies. Thanks to our ontology, we build the low-level context from collected and aggregated measurements (e.g., rainfall amount per 24h, max slope of water flow rate per 6h and node energy). Then, the DSS agent uses Jess engine to infer the high-level context. A first set of rules infer the states of observable entities (e.g., Precipitation, Watercourse, Outlet and Node). A second set of rules infer the state of the entity of interest (e.g., Flood entity) from the state of the observable entities.

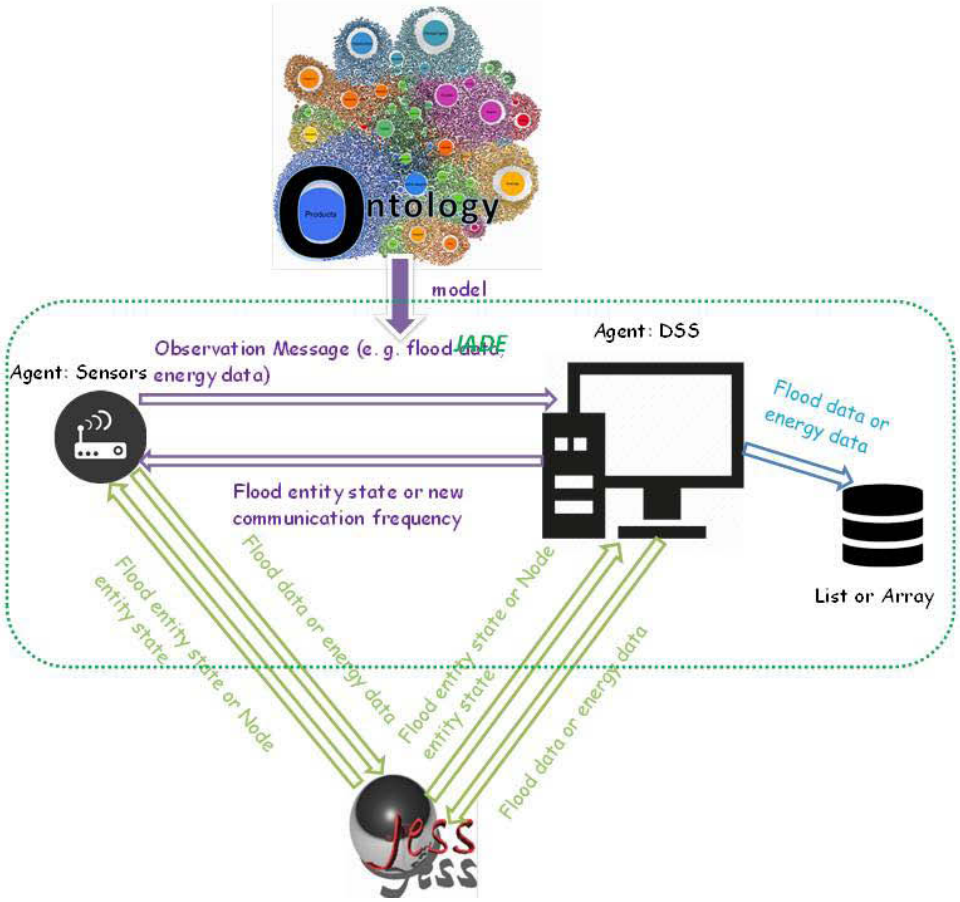


Figure 5.1.2 Simulation architecture

For our JADE simulation, an agent contains a physical sensor, which acquires the measurements, associated to virtual sensors which compute the aggregated values. We create seven agents:

- One agent per weather station. These agents correspond to Precipitation nodes. They extract the rainfall amount measurements from archive files. They compute rainfall amount per 24 hour s. They also provide the energy measurements at a given simulation time using our energy model presented in section 5.2.3,
- One agent per stream gauge. Two agents are Watercourse nodes. They extract the water flow rate of the tributaries from archive files. They compute last water flow rate or max slope of water flow rate per 6h. They also provide the energy measurements at a given simulation time using our energy model presented in section 5.2.3. One agent is the Outlet node. It extracts the water flow rate of the outlet from archive files. It computes last water flow rate or max slope of water flow rate per 6h. It also provides the energy measurements at a given simulation time using our energy model presented in section 5.2.3,
- One agent to represent DSS.

All the agents can integrate Jess to perform the reasoning.

5.2 Context Acquisition

In this section, we will present the data sources used to build the flood context and node context for our simulation.

About the flood context, we use the archive of the Orgeval watershed of the whole year 2007 to train our simulation tool. To test our simulation tool, we use the archive of the watershed from February 2008 because we know that there were very much precipitations. It generated a risky situation during this month.

For the node context, we decide to propose an energy model based on t he Libelium Waspmote PRO v1.2 solution (Libelium PRO v1.2, 2017). We decide to focus on this kind of wireless sensor nodes because Irstea of Clermont-Ferrand uses this commercial solution in their outdoor experiments and these nodes have been worldwide distributed.

We made the assumption that the sample frequency will be the same for the flood and node contexts (one measurement per minute). Each node carries out, during the same time interval, the acquisition of flood context measurements and the node energy measurements.

5.2.1 Flood Data Source: Orgeval Dataset

The flood context in our simulation is based on the Orgeval dataset archive. This latter is a long-term data sets in hydrology done by the Oracle observatory team of Irstea. Irstea develops the "Hydrology Observatory Database" (BDOH) website in order to exchange hydrological observatory datasets with other scientific teams (BDOH, 2017).

BDOH is designed to enable the management, saving and access of hydrological and biogeochemical data from long-term observatories. We use BDOH to select the weather stations and stream gauges and time intervals when the measurement stations are open. The data from BDOH is not precise enough for our simulation. Therefore, the Oracle observatory team of Irstea provides us with the precise data from the national database "banque Hydro". The "banque Hydro" stores water level measurements (at variable times) from about 5000 measurement stations (of which approximately 3200 are currently in use) located on French rivers and provides access to information about the stations e.g., the purpose of the station, the precise location, the quality of measurements, the history and the available data, etc. (Hydro, 2017).

For our simulation, the Oracle observatory team provided us with new measurements:

- Precipitation: rainfall amount per hour,
- Last hour average of water flow of the stream gauges measurements.

For our simulation we need the measurements from year 2007 to year 2008. However, the banque Hydro does not provide us with the data at the good level of granularity (e.g., time frequency). In our simulation, the sample frequency for the flood and the node contexts is both one measurement per minute. Consequently, we made a treatment for these two measurements:

- For the rainfall amount per hour, it is divided into 60 to compute the measurements of rainfall amount per minute from year 2007 to year 2008,
- For the last hour average of water flow of the stream gauges measurements, first we compute the slope between two successive water flow measurements e.g., slope between 01/01/2007 00:00 and 01/01/2007 01:00. For a fixed time instant 01/01/2007 00:00, we can get the value of water flow measurements at this time instant e.g. 282.000 from our archive files; Based on the slope between 01/01/2007 00:00 and 01/01/2007 01:00, we can generate the value of water flow measurement from 01/01/2007 00:01 to 01/01/2007 01:00 successively. Base on this treatment, the time period we generate is also from year 2007 to year 2008.

As mentioned above, we have six stations in our simulation. Therefore, we generate six files corresponding to these stations respectively. The size of these six files is almost 180 MB. We have about 1,054,080 measurements in each file.

5.2.2 Node Data Source

As presented in the section 4.1, an energy measurement archive is not totally useful because many factors influence the energy consumption of a wireless sensor node. Consequently, we decide to propose an energy data model to simulate node energy consumption and production. First, we present the sensor node components because component activities influence the node energy management. We made the hypothesis that the node is a Libelium Waspmote, such as the PRO v1.2 version, that includes a battery and a solar panel. Secondly, we present some relevant works in the topic of wireless sensor energy modeling. We also present a generic overview of solar panel production model. Then, we propose our energy consumption and production model. Finally, we parameterize our model to simulate the energy management of a Libelium Waspmote node equipped with a solar panel.

5.2.2.1 Node Description

In the Figure 5.2.1, taken from the articles (Akyildiz *et al*., 2002) (Akyildiz *et al*., 2002b), and recalled in (Wang *et al*., 2004), the authors consider that a wireless sensor node has four main components which are:

- The sensing unit,
- The processing unit,
- The communication unit, (in Figure 5.2.1 the authors just focused on the transceiver, but we integrate it into a more generic component: the communication unit), and
- The power supply unit.

One of the most important components of a sensor node is the power supply unit. Since a sensor node is often difficult to access, the lifetime of a sensor node depends on the capacity of its power supply resource. Power supply unit can be supported by an energy harvesting unit

("Power Generator") as shown in Figure 5.2.1. Energy harvesting means extracting energy from the environment. Solar cells is an example of techniques used for energy harvesting.

Now, with the presence of microSD card reader such as in the Libelium Waspmote solutions (Libelium PRO v1.2, 2017), we can consider another component by separating the processing unit in:

- A processing unit, and
- A storage unit.

We can also consider another component by separating the power unit in:

- A power unit, and
- A solar panel unit.

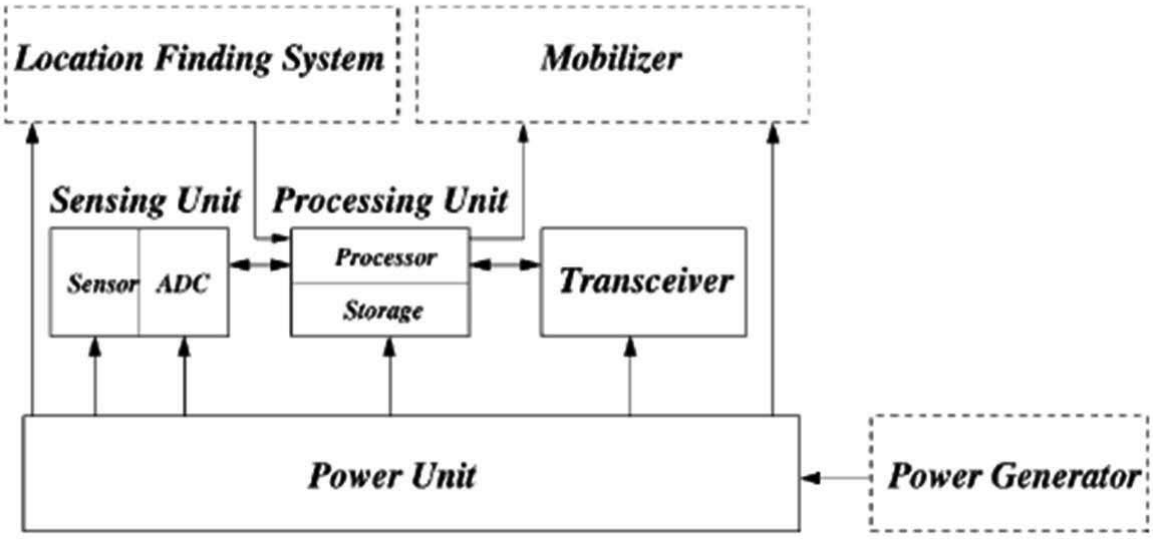


Figure 5.2.1 Sensor node components (Akyildiz et al., 2002) (Akyildiz et al., 2002b) (Wang et al., 2004)

The activity of a wireless sensor node can be defined from operating information related to these different units. For example, this could be the number of tasks in the processor queue, namely, the tasks that are waiting to be processed. For the storage unit, it could be the quantity of free memory.

In a detailed model of a wireless sensor node, the activity of a node can be divided into cycle of a fixed time interval e.g., one minute. However, not all the components are working during the fixed time interval. For example, the communication unit is designed not to be working in every time cycle. The purpose is to save energy of battery according to the work in (Akyildiz *et al*., 2002). It is to be noted that communication part consumes most part of the energy of a wireless sensor node.

Based on that, the energy level of a wireless sensor node depends on the aggregation of the initial value of the battery, the energy consumed by the activities of the wireless sensor node

components and the energy supplied by the power generator component such as a solar panel unit.

In order to model the evaluation of the energy consumed by a node and provided by a power supply, an energy model should be established. First, we present several existing works related to the design of energy models for embedded devices such as wireless sensor node.

5.2.2.2 Energy Consumption Model

In this section, we present some existing energy models. We study some of these existing models to provide our own energy model.

5.2.2.2.1 Terrasson Energy Consumption Model

The work in (Terrasson *et al*., 2009) presents a generic energy consumption model for micro sensor.

The authors provide a model with the following main components:

- Sensors,
- Analog to Digital Converter (ADC),
- Microcontroller (CPU and memory units), and
- RF transceiver (for transmission and reception).

The generic energy model is shown in equation 5.1.

Equation 5.1: Energy model of the article (Terrasson *et al*., 2009)

$$E_{node} = E_{sensor} + E_{ADC} + E_{\mu C} + E_{trans} + E_{rec}$$

Where

E_{sensor} is the energy cost consumed by the sensor for making measurements.

E_{ADC} is the energy cost consumed by the ADC in order to convert sensor measurements from analogical to digital data.

$E_{\mu C}$ is the energy cost for data processing, formatting and coding consumed by the microcontroller (µC).

E_{trans} is the energy cost for data communication (transmission) consumed by the RF transceiver.

E_{rec} is the energy cost for data reception consumed by the RF receiver.

In this model, the energy consumption of each node's component depends on its operating state: active or inactive (idle).

This energy model does not take into account energy costs of two activities:

- Data storage energy consumption,
- Communication energy consumption. When the transceiver is active, it w ill do sending, listening and receiving. However, in this model, the listening, which is the energy consumed when the communication module is waiting for data sending or receiving, is not taken into account.

5.2.2.2.2 Shi Energy Consumption Model

The work in (Shi *et al*., 2011) presents also a generic energy consumption model for a node. The authors consider that a node has four main activities:

- Sensing,
- Data Processing,
- Data Storage,
- Communication.

The node will be in "SLEEP" mode after these four main processes. This generic energy model is shown in equation 5.2:

Equation 5.2: Energy model of the article (Shi *et al*., 2011)

$$E_{node} = E_{sensing} + E_{processing} + E_{storage} + E_{comm} + E_{sleep}$$

Where

$E_{sensing}$ is the energy cost for making measurements,

$E_{processing}$ is the energy cost when the node is doing data processing operations such as conversion or "cleaning". For example, a node may check that its measurements belong to a value, of intervals

$E_{storage}$ is the energy cost when the node stores one or many measurements,

E_{comm} is the energy consumption of the communication module,

E_{sleep} is the energy cost when the node is in "SLEEP" mode.

This energy model involves data storage energy consumption part, but do not consider separately the three activities related to the communication activity which are: sending, listening and receiving.

Therefore, we propose our more general energy consumption model based on t hese two energy consumption models as presented in section 5.2.3.

5.2.2.3 Energy Production Model

As mentioned in section 5.2.2.1, the energy supply by the power generator component such as a solar panel unit influences the energy level of a sensor node. We make hypothesis that our

Libelium Waspmote node (PRO v1.2 version) has a solar panel. In this section, we present an existing energy production model: Aurora solar panel supply model.

5.2.2.3.1 Aurora Solar Panel Supply Model

The method of Aurora solar panel supply model is to consider the daily amount of solar radiation energy supply as presented in (Doshi *et al.*, 2011). The "average daily sun hours per day" (in kWh/m²) can be linked to the "peak sun hours per day" (in kW/m²). This latter metric refers to the solar insolation that a location receives for certain time duration. Thus, this duration can be deduced from the "average daily sun hours per day" metric as shown in Figure 5.2.2.

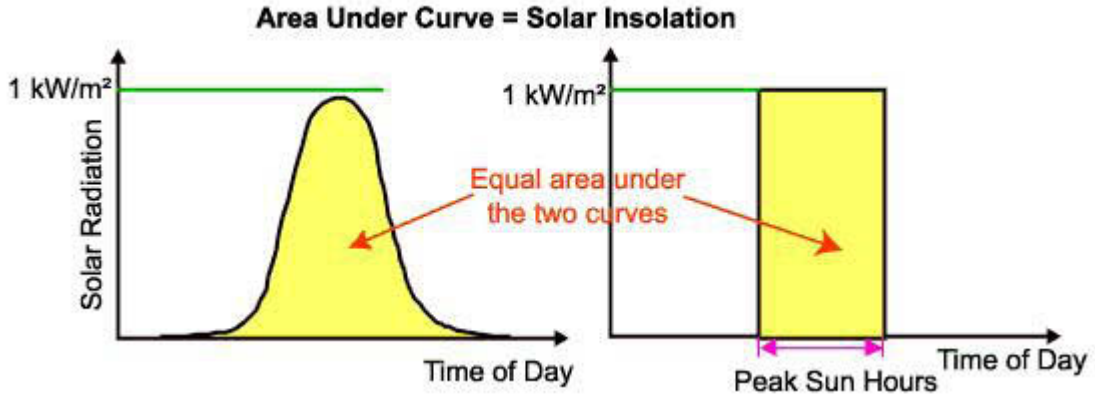


Figure 5.2.2: Calculating the solar radiation energy supply per day (Doshi et al., 2011)

For example, if the city of Baltimore receives a solar radiation of 4.7 kWh/m² per day, it is equivalent of the reception of a constant solar insolation of 1 kW/m² during 4.7 hours in a day. Thus, based on the characteristics of a solar panel, we can calculate the average energy provided by it per day.

5.2.3 Proposed Global Wireless Sensor Energy Model

Based on elements from (Libelium PRO v1.2, 2017) and the energy models mentioned above, we propose, in equation 5.3, our energy model for a wireless sensor node.

In our use case, the node energy will evolve during the day according to solar radiation level. A battery and a solar panel is used to supply energy.

Part of the consumed energy depends also on the number of communications and their packet size. Indeed, energy consumption of the communication is proportional to the size (bit number) transmitting message. For our simulation, our energy model is composed as followed:

- The energy of the battery at the starting time t_0. It is called $E_{battery}(t_0)$.
- A constant that indicates the battery capacity $Capacity_{battery}$ which cannot be exceeded.
- The energy produced by a solar panel during a time interval $[t_i, t_{i+1}]$ called $E_{solar}(t_i, t_{i+1})$.

- The energy consumed by activities (sensing, processing, communication, etc.) during a time interval $[t_i, t_{i+1}]$ called $E_{consum}(t_i, t_{i+1})$.

All these measurements are aggregated using the equation 5.3 to compute the wireless sensor node energy value at a time instant t_i: $E_{battery}(t_i)$.

Equation 5.3: Energy model of a node

$$E_{battery}(t_{i+1}) = \begin{cases} E_{battery}(t_i) + E_{solar}(t_i, t_{i+1}) - E_{consum}(t_i, t_{i+1}) & \text{if } E_{battery}(t_{i+1}) < Capacity_{battery} \\ Capacity_{battery} & \text{if } E_{battery}(t_{i+1}) \geq Capacity_{battery} \end{cases}$$

We will present our energy consumption model of a wireless sensor node and solar panel supply model in detail in the following 2 sections.

5.2.3.1 Proposed Energy Consumption Model

In our energy model, we consider a Libelium Waspmote node (PRO v1.2 version) which has a reference current. The activities of the node components consume extra currents based on the reference current.

There are four main activities associated to specific process to consider during a time cycle:

- Sensing,
- Data processing,
- Data storage,
- Communication: the communication activity is also composed of three sub-activities:
  - Sending,
  - Listening,
  - Receiving.

In the energy consumption model, the energy consumptions of a node and of its main processes are calculated separately. We consider that each node has its reference current. The value of this current varies according to the three operating states for a n ode: “ON” or “SLEEP” or “OFF”. When the node is in “ON” mode, there are four main activities that can be processing: sensing, data processing, data storage and communication. There are two operating states for each activity: “ON” or “OFF”. There are extra current consumed based on the reference current when each activity is in “ON” mode. The node activities will not consume energy when they are all in “OFF” mode. The node is in “SLEEP” after all the activities associated to the units are all in “OFF” mode, only the daemon activity is in “ON” mode. The purpose of this daemon is just to wait for an interruption triggered by the RTC (Real Time Clock). If all activities are in “OFF” mode, the node is in “OFF” mode.

So, we consider the energy consumptions of a wireless sensor node is separated in six main energy costs as shown in Equation 5.4.

Equation 5.4: Proposed energy consumption model of a node

$$E_{consum} = E_{on} + E_{sensing} + E_{processing} + E_{storage} + E_{comm} + E_{sleep}$$

Where

E_{on} is the energy cost of a node when it is in "ON" mode. The current during only the "ON" mode is equal to the reference current of the node.

$E_{sensing}$ is the energy cost for making measurements,

$E_{processing}$ is the energy cost when node is doing processes e.g., data "cleaning". For example, a node may check that its measurements belong to a value intervals,

$E_{storage}$ is the energy cost when node stores one or many measurements,

E_{comm} is the energy consumption of the communication unit,

E_{sleep} is the energy cost when node is in "SLEEP" mode.

Equation 5.5 presents our energy consumption model of the communication when its sub-activities are considered.

Equation 5.5: Proposed energy consumption model for the communication unit

$$E_{comm} = E_{comm_{on}} + E_{comm_{sending}} + E_{comm_{receiving}}$$

Where

$E_{comm_{on}}$ is the energy consumption when the communication module is in "ON" mode ("listening").

$E_{comm_{sending}}$ is the energy consumption when the node sends packets (containing one or many measurements) using its communication module.

$E_{comm_{receiving}}$ is the energy consumption when the node receives packets using its communication module.

In our energy consumption model, the activity of a node is divided into cycles of a fixed time interval e.g., one minute. Time used for data sensing, processing and storage is considered as constant. From one acquisition to another, the time for doing these activities are also considered similar for the sake of simplification. We use average powers of data sensing, processing and storage to calculate the energy consumptions. Therefore, parameters $E_{sensing}$, $E_{processing}$ and $E_{storage}$ are set as constant values. At the opposite the communication activity of the communication unit varies from one cycle to another.

Based on that, the values of the parameters E_{on} and E_{sleep} depends, for each cycle, on the duration of the sub-activities of the communication unit.

Therefore, our energy consumption model for a time interval $[t_i, t_{i+1}]$ can be expressed as explained in Equation 5.6.

Equation 5.6: Energy consumption model for a time interval $[t_i, t_{i+1}]$

$$E_{consum}(t_i, t_{i+1}) = E_{on}(t_i, t_i + d_{sensing} + d_{processing} + d_{storage} + d_{comm}) + E_{sensing} + E_{processing} + E_{storage} + E_{comm}(d_{comm}) + E_{sleep}(t_i + d_{sensing} + d_{processing} + d_{storage} + d_{comm}, t_{i+1})$$

such as $d_{sensing} + d_{processing} + d_{storage} + d_{comm} \leq t_{i+1} - t_i$

Where

$E_{sensing}$, $E_{processing}$ *and* $E_{storage}$ are the energy costs for sensing, processing and storage activities respectively.

$d_{sensing}$, $d_{processing}$ *and* $d_{storage}$ are the (constant) time durations for the activities of sensing, processing and storage.

d_{comm} is the variable time duration for the communication module activity. We have naturally:

- $d_{on} = d_{sensing} + d_{processing} + d_{storage} + d_{comm}$
- $d_{on} + d_{sleep} = t_{i+1} - t_i$

d_{on} is the duration when the node is in "ON" mode.

d_{sleep} is the duration when the node is in "SLEEP" mode.

Notes that: all these operations (sensing, processing, storage and communication) should be done during one activity cycle included in the interval $[t_i, t_{i+1}]$.

For the communication activity, we assume that the size of a sending packet is fixed (e.g., through padding mechanism). However, communication unit node will not always transmit and receive packets in an activity cycle. The parameter E_{comm} can be 0 in some cycle periods.

The communication unit can have three sub-activities when it is in "ON" mode. It can send packets, then listen for the incoming packets in a fixed time duration, and receive packets. After these three sub-activities, communication unit is in "OFF" mode, the communication unit does not consume energy. However, not all these three sub-activities exist in each activity cycle. For example, in our simulation, communication unit cannot send packet and cannot receive packet in some specific activity cycles. Thus, in this case, we consider the communication unit is in "OFF" mode, it does not consume energy. The duration of the communication activity varies according to the sub-activities of communication unit which are active during a given activity cycle.

Equation 5.7: Energy consumption of the communication unit for a time interval $[t_i, t_{i+1}]$

$$E_{comm}(t_i, t_{i+1}) = Ecomm_{on}(t_i, t_i + d_{listening} + d_{sending} + d_{receiving}) + E_{sending}(d_{sending}) + E_{receiving}(d_{receiving})$$

such as $d_{listening} + d_{sending} + d_{receiving} \leq t_{i+1} - t_i - d_{sensing} - d_{processing} - d_{storage}$

$$d_{sending} + d_{listening} + d_{receiving} = \begin{cases} 0, & \text{if communication is inactive} \\ d_{comm}, & \text{if communication is active} \end{cases}$$

Where

$d_{sending} \wedge d_{listening} \wedge d_{receiving} != 0$ if communication is active.

E_{common} is the energy consumed when the communication unit is in "ON" mode.

$d_{listening}$ is the time duration of the communication module when listening for incoming packets.

$d_{sending}$ is the time duration of the communication module when sending packets.

$d_{receiving}$ is the time duration of the communication module when receiving packets.

We have naturally $d_{comm} = d_{listening} + d_{sending} + d_{receiving}$.

Usually, the sending activity is dissociated to the listening and receiving activities. Of course, for these three activities, we need to activate the communication module. Differently, to receive a packet, we need to "listen" to the wireless medium. Thus, listening and receiving activities are linked. In our evaluation, presented in next section (Section 6), every sending activity is followed by a time interval listening which can be completed by a receiving activity.

5.2.3.2 Proposed Solar Panel Supply Model

Solar panel provides energy to each wireless sensor node. As previously said, our solar energy model is based on the Aurora Energy Model (Aurora Energy, 2012). In Aurora Energy Model, the total amount of energy provided during one day is proportional to a fixed number of hours per month where the solar radiation is maximized. This max value is called peak solar radiation and is represented by the constant *G*. For example, in February, one day's total energy provided by solar radiation is equivalent to the energy produced during 4.1 hours multiplies by the peak solar radiation. However, in reality, the value of peak solar radiation can be different when there is a rainy day. There, the energy generated by solar panel per day follows the equation 5.8:

Different models exist to represent energy generated by a solar panel such as ("Calculating the Kilowatt Hours Your Solar Panels Produce (Solar Panel Output)", 2016) (MESIA, 2014). For our simulation environment, we propose the equation 5.8 which is a simplified model, established from previous ones, and corresponds to what we need:

Equation 5.8: Energy generated by a solar panel per day

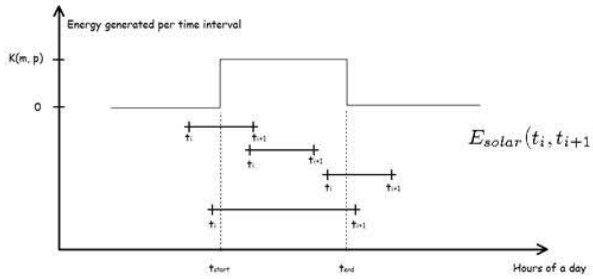
$$Esolar_{day}(m,p) = \begin{cases} G(m) * nbHour(m) * Area(p) * Efficiency(p) & \text{when it is a sunny day} \\ \frac{G(m)}{\alpha} * nbHour(m) * Area(p) * Efficiency(p) & \text{when it is a rainy day} \end{cases}$$

Where the parameters of this equation are defined as followed:

- *G(m)* is the peak solar radiation for the month *m* when it is a sunny day.
- *G(m)/* α is the peak solar radiation for the month *m* when it is a rainy day. We made assumption that α is equal to 2.
- *nbHour(m)* is the number of hours where the solar radiation is maximized. This value depends on the month *m*.
- *Efficiency(p)* represents the capacity of the solar panel *p* to transform radiation to energy (Solar Panel, 2017).
- *Area(p)* is the area of the solar panel *p*.

The solar energy E_{solar} for a time interval $[t_i, t_{i+1}]$ is estimated as indicated in equation 5.9.

Equation 5.9: Energy supply model of a solar panel for a time interval $[t_i, t_{i+1}]$:



$$E_{solar}(t_i, t_{i+1}) = \begin{cases} K(p,m) * (t_{i+1} - t_{start}) & \text{if } t_i < t_{start}, t_{start} <= t_{i+1} < t_{end} \\ K(p,m) * (t_{i+1} - t_i) & \text{if } t_{start} <= t_i, t_{i+1} <= t_{end} \\ K(p,m) * (t_{end} - t_i) & \text{if } t_{end} > t_i >= t_{start}, t_{i+1} > t_{end} \\ K(p,m) * (t_{end} - t_{start}) & \text{if } t_{start} > t_i, t_{end} < t_{i+1} \\ 0 & \text{if } t_i, t_{i+1} < t_{start} \text{ or } t_i, t_{i+1} > t_{end} \end{cases}$$

- *K(p,m)* is the value of energy generated by solar panel per time interval. It is equal to $E_{solar_{day}}(m,p)/nbHour(m) \times 60$,
- t_{start} is the timestamp where the *nbHour(m)* starts. The value is 12h- *nbHour(m)/2,*
- t_{end} is the timestamp where the *nbHour(m)* ends. The value is 12h- *nbHour(m)/2*.

5.2.4 Parameters

In this section, we will present the parameters of energy consumption model and solar panel model that we used for our simulation.

5.2.4.1 Parameters of the Energy Consumption Model

In our simulation, we use Libelium Waspmote (LW) PRO v1.2 node characteristics (Libelium PRO v1.2, 2017). Table 5.2.1 shows the parameters of energy consumption of different modes, components, tasks and resource required Libelium Waspmote PRO v1.2 node built around an ATmega1281 microcontroller.

Table 5.2.1: Parameters of energy consumption of different modes, components, tasks and resource required in libelium waspmote PRO v1.2 node built around an ATmega1281 microcontroller

Node Mode	Component	Task	Required Resources		Consumed Energy
			Current (mA)	Voltage (V)	
ON	Water Flow Sensor(YF-G1)	Sensing	15	5	
	ATmega1281 microcontroller	Active	15	3.7	
		Processing	0	3.7	
	EEPROM	Storage	7.3	3.7	
	XBee Zigbee Pro (ZB)	Active	45.56	3.7	
		Tx	105	3.7	
		Rx	50.46	3.7	
SLEEP	ATmega1281 microcontroller	Sleeping	0.055	3.7	

As we mentioned, the Libelium node has three modes: “ON”, “SLEEP” and “OFF”. Libelium node will not consume energy in “OFF” mode. So the Table 5.2.1 just presents “ON” and “SLEEP” modes of Libelium node.

Based on (Libelium PRO v1.2, 2017), the voltage of the Libelium node is 3.7 V when it is in “ON” and “SLEEP” modes. The value of reference current is 15 mA shown in row 2 (Current (mA)) and column 4 when the Libelium node is in “ON” mode (or ATmega1281 microcontroller is active). The variable of energy consumption is E_{on}. The value of the reference current is 0.055 mA as shown in row 10 and column 4 when the Libelium node is in “OFF” mode (or ATmega1281 microcontroller is sleep). The variable of energy consumption

associated is E_{sleep}. The Libelium node will cost more currents to execute some task processes e.g., sensing, storage or transmission based on this reference current.

Based on (Libelium Smart Metering, 2017), the water flow sensors we use are YF-G1. The values of current and voltage are presented in row 3 and column 4, row 3 and column 5 respectively when water flow sensors doing the task: sensing. The variable of energy consumption is $E_{sensing}$. The size of a transmitted packet is 74 bytes.

In our simulation, we will not perform data processing e.g., data cleaning. So, $E_{processing} = 0$.

The storage component is EEPROM. The values of current and voltage are presented in the column 4 (Current (mA)) and row 6, column 5 (Voltage (V)) and row 6 respectively. The variable of energy consumption is $E_{storage}$.

The communication module used is a XBee Zigbee Pro (ZB). XBee Zigbee Pro (ZB) has three tasks: *Active*, *Tx* and *Rx*. The values of current and voltage are presented in the column 4 (Current (mA)) and row 7, column 5 (Voltage (V)) and row 8 and column 5 (Voltage (V)) and row 9 respectively. The variables of energy consumption are $E_{sending}$ ("*Tx*") and $E_{receiving}$ ("*Rx*").

Based on the work in (Shi *et al*., 2011), we set the following values of duration of tasks: sensing, processing and storage presented in Table 5.2.2. So in Table 5.2.2, we can get the values of $E_{sensing}$, $E_{processing}$ and $E_{storage}$ based on corresponding values of current and voltage are presented in the Table 5.2.1 respectively.

Table 5.2.2: Values for the context acquisition

Energy Task	Task Duration (s)	Consumed Energy (J)
	= 1	0.075
	= 0	0
	= 0.010	0.000

For the communication part, based on the work in (Diao, 2011), $d_{sending}$ is set to 0.002 s and $d_{receiving}$ is set to 0.098 second as presented in Table 5.2.3. The only problem is to set the duration of the listening activity when the communication module is in "ON" mode.

As we mentioned in section 5.2, Irstea of Clermont-Ferrand uses an energy model based on the Libelium Waspmote PRO v1.2 in their outdoor experiments. Without energy provided by solar panel, it costs almost two weeks to run out the battery energy when the node's sample frequency (one measurement per minute) equals the communication one as presented in Figure 5.2.3.

So based on that, we can compute that the energy consumption of the communication equals approximately 3.77 J. Based on $d_{sending}$, $d_{receiving}$ and the values of current and voltage of XBee Zigbee Pro (ZB) tasks: *Active*, *Tx* and *Tr*, we can compute the value of d_{comm} equals 16.1 second.

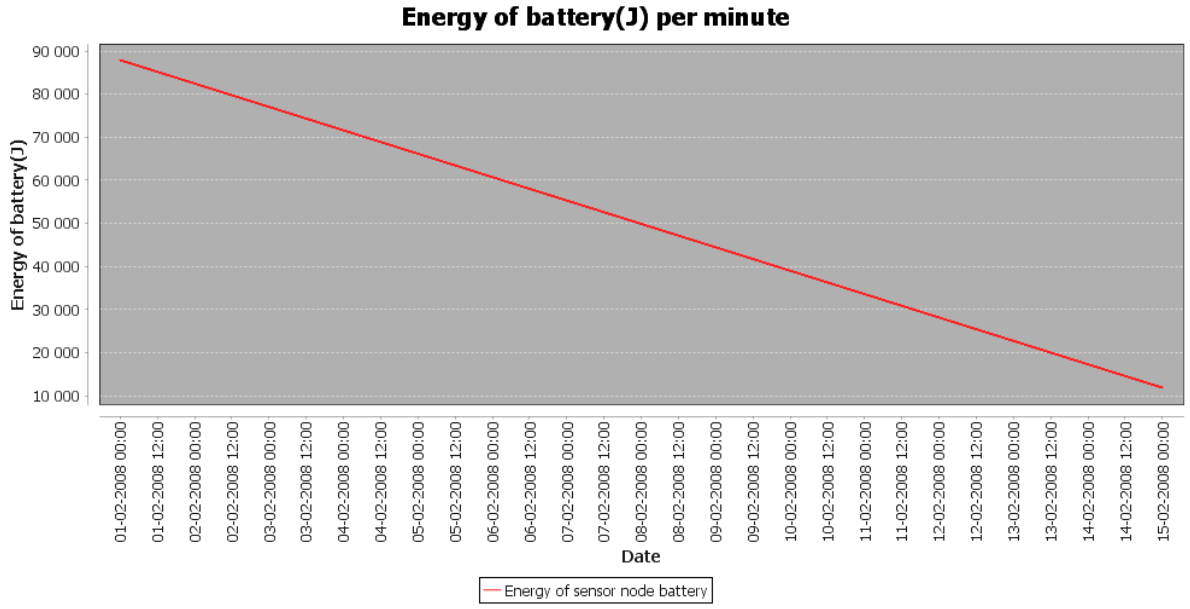


Figure 5.2.3: Battery energy of libelium node when the sample frequency equals the communication one

Thus, we can also get d_{on} and d_{sleep}, when XBee Zigbee Pro (ZB) is "ON" mode. E_{comm}, $E_{sending}$, $E_{receiving}$, E_{on} and E_{sleep} are presented in Table 5.2.3.

When XBee Zigbee Pro (ZB) is "OFF" mode, d_{comm}, $d_{sending}$, and $d_{receiving}$ are equal to 0. So d_{on} and d_{sleep} can be computed only based on $d_{sensing}$, $d_{processing}$ and $d_{storage}$. All These results are presented in Table 5.2.3.

Table 5.2.3: Values of context communication

ZB module mode	Energy Task	Task Duration (s)	Consumed Energy (J)
ON		= 16.1	2.714
		= 0.002	0.000
		= 0.098	0.018
		= 17.110	0.949

		= 42.890	0.009
OFF		= 0	0
		= 0	0
		= 0	0
		= 1.01	0.056
		= 58.990	0.012

Based on Tables 5.2.2 and 5.2.3, we get:

- When the communication unit (ZB module) is in "ON" mode, the rounding value of $E_{consum}(t_i, t_{i+1})$ is 3.77 J.
- When the communication unit is in "OFF" mode, the rounding value of $E_{consum}(t_i, t_{i+1})$ is 0.144 J.

5.2.4.2 Parameters of the Solar Panel Model

Based on the rigid (7V – 500mA) solar panel of Libelium (Libelium PRO v1.2, 2017), values of the parameters for the solar panel model are shown in Table 5.2.4:

Table 5.2.4: Parameters of solar panel energy model

Solar Panel Area	Solar Panel Voltage	Solar Panel Max Current	Efficiency
0.025×0.24	6.9 V	440 mA	0.14

Table 5.2.5 presents the relations between the values of the Peak Solar Radiation and the day weather. We assume that the value of the Peak Solar Radiation in sunny days is 2 times bigger than the one in rainy days.

Table 5.2.5: Relations between the peak solar radiation and the day weather

Day weather	Peak Solar Radiation
Sunny	1000 W/m^2

Rainy	500 W/m^2

We know that the energy level provided by solar radiation is fixed in a specific day and month as shown in equation 5.8. Therefore, we can compute the energy level provided by solar panel energy model for different weathers for a given day as shown in Table 5.2.6.

Table 5.2.6: Energy level provided by solar panel energy model for different weather conditions for a given day

	Theoretical $Esolar_{day}$ *(J)*	*Effective* $Esolar_{day}$ *(J)*
Sunny	12398.4	5461.2
Rainy	6199.2	3324.2

$Esolar_{day}$ theoretical column presents the energy level provided by solar panel energy model in sunny days and rainy days. However, in reality, Libelium Waspmote PRO v1.2 node built around an ATmega1281 microcontroller has $Current_{max}$ (100 mA) which is the max rechargeable current.

When the solar panel recharges Libelium node, if the output current of solar panel energy model is bigger than $Current_{max}$ (100 mA), the input current of Libelium node is $Current_{max}$ (100 mA).

When the solar panel recharges Libelium node, if the output current of solar panel energy model is smaller than $Current_{max}$ (100 mA), the input current of Libelium node is the output current of solar panel energy model.

The voltage of Libelium node is different (smaller) from the one of the solar panel. So, the real used "$Esolar_{day}$ " is "$Esolar_{day}$ effective" which is also shown in Table 5.2.6.

We also assign the values of *nbHour(m)* which is the peak solar radiation hours per day of each month based on (Aurora Energy, 2012) as shown in Table 5.2.7.

Table 5.2.7: The values of peak solar radiation hours per day of each month

Month	*nbHour(m)*
January	3.6 h
February	4.1 h

March	4.8 h
April	5.1 h
May	5.3 h
June	5.7 h
July	5.5 h
August	5.4 h
September	5.05 h
October	4.7 h
November	3.7 h
December	3.1 h

The presented energy model will be part of the context acquisition process when the Node (*N*) entity is considered in the decision process. It will be considered in some scenarios introduced in section 6.

5.3 Context Modeling

In our simulation, we build an ontology to model the exchanged message contents between agents and to reason over these data. The purpose of ontology in our use case is twofold:

1. Build the low-level context from messages sent by the agents,
2. Infer the high-level context from the low-level context.

In this section, we present how JADE uses one ontology for model the agent message content. We know there are some ontologies related to sensor description and sensor measurements. These ontologies are defined by Semantic Web technologies. Therefore, we present the most well-known called Semantic Sensor Network Ontology (SSNO) (Compton *et al*., 2012). Based on this ontology, we build a network of ontologies to describe our context schema. Finally we translate this Semantic Web context schema to build our JADE ontology.

5.3.1 Definition of Ontology in JADE

The message content in JADE is defined thanks to the Content Reference Model (Bellifemine *et al*., 2001). This model is presented in Figure 5.3.1. Content Reference Model defines all the

elements contained in message exchanged between agents. We present the definition of several elements provided by (Bellifemine *et al.*, 2001):

- "Terms" are expressions identifying entities that "exist" in the world and that agents may reason about,
- "Concept" is a subclass of "Term". Concepts are entities with a complex structure that can be defined in terms of slots (e.g., Sensor :id melarchez-p35_RainGauge),
- "Predicates" are expressions that say something about the status of the world. These expressions can be either true or false (e.g., Deploy(Sensor :id melarchez-p35_RainGauge) (Watershed :name "Orgeval")),
- "Agent Actions" are special concepts that indicate actions that can be performed by some agents (e.g., *FrequencyAdapt* is the action to modify the communication frequency of a node).

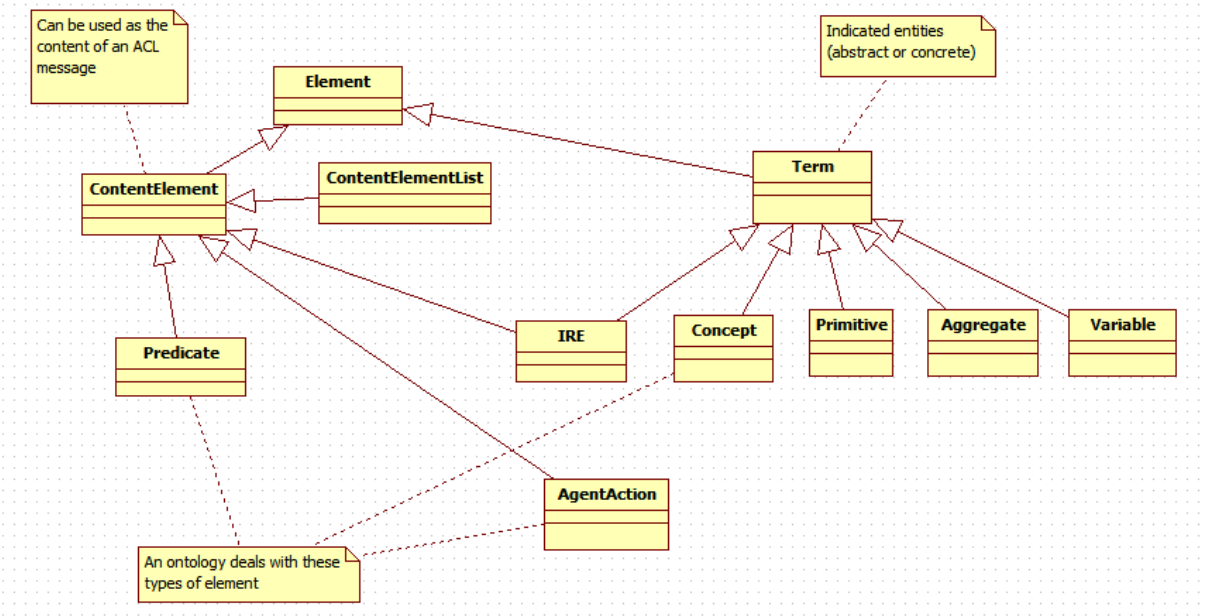


Figure 5.3.1: Class diagram of the content reference model (Bellifemine et al., 2001)

An ontology for a given domain in JADE is a vocabulary composed only of concepts, predicates and agent actions. JADE ontology is a set of element schemas. These element schemas define the structure of the "predicates", "agent actions" and "concepts" that are allowed in messages.

5.3.2 Ontology Network Based on SSN Ontology

Ontologies are also defined as complex schema in Semantic Web technologies (W3C, 2017). The language used to describe these schemas is Web Ontology Language (OWL) (OWL, 2017). In Semantic Web Ontology there already exist some ontologies dedicated to sensor description (Bendadouche *et al*., 2012), e.g., CSIRO (Compton *et al*., 2009), SSNO defined by W3C, etc. SSNO plays the role of a backbone where others ontologies can be plugged to

described a sensor network (measurements, sensor location, etc.). We decide to reuse these existing ontologies to build our JADE ontology.

5.3.2.1 OWL Language

There are two languages used to define schema in Semantic Web technologies:

- Resource Description Framework Schema, also named RDF Schema or RDFS, (RDF Schema 1.1, 2017),
- Web Ontology Language, also named OWL, (OWL, 2017).

OWL and RDF Schema defines classes, properties, and their hierarchies. OWL is more expressive than RDF Schema.

(Tomaiuolo *et al*., 2006) explains OWL as it relies on an object centered view of the world. It allows several types of components:

- Classes represent a set of instances. They are the type of things defined in the domain,
- Instances represent objects in the domain. An instance is typed by a class,
- Object properties define relations between two instances,
- Datatype properties define instance slots.

OWL can describe complex and rich relationships with greatly enhanced reasoning ability (Aldabagh *et al*., 2012). In particular, OWL proposes more operators than RDFS. For example, *owl:sameAs property* sets that two classes are equivalent or *owl:unionOf* operator defines a class as the union of several classes ("RDFS vs. OWL, 2017).

5.3.2.2 SSNO

The W3C Semantic Sensor Network Incubator group (the SSN-XG) produced the Semantic Sensor Network Ontology (SSNO) to describe sensors, sensor network and their measurements. The SSN ontology is defined in OWL.

The SSNO can describe sensors in terms of capabilities, measurement processes, observations and deployments (Compton *et al*., 2012). We reuse 4 classes of SSNO:

- "ssn:FeatureOfInterest". "*A feature is an abstraction of real world phenomena (thing, person, event, etc)*."
- "ssn:Property". "*An observable Quality of an Event or Object. That is, not a quality of an abstract entity as is also allowed by DUL's Quality, but rather an aspect of an entity that is intrinsic to and cannot exist without the entity and* ***is observable by a sensor***."
- "ssn:Sensor". "*A sensor can do (implements) sensing: that is, a sensor is any entity that can follow a sensing method and thus observe some Property of a FeatureOfInterest. Sensors may be physical devices, computational methods, a*

laboratory setup with a person following a method, or any other thing that can follow a Sensing Method to observe a Property."

- "ssn:Observation". "*An Observation is a Situation in which a Sensing method has been used to estimate or calculate a value of a Property of a FeatureOfInterest. Links to Sensing and Sensor describe what made the Observation and how; links to Property and Feature detail what was sensed; the result is the output of a Sensor; other metadata details times etc.*"

We take our use cases to illustrate the four classes of SSNO:

- All the observable entities and entities of interest will be defined as instance of "ssn:FeatureOfInterest", e.g., Watercourse, Outlet, Flood, etc.
- All types of values computed or measured by sensors will be defined as instance of the class "ssn:Property", e.g., "rainfall amount per 24h", "max slope of water flow rate per 6h", "last water flow rate", etc.
- Precipitation nodes, watercourse nodes and the outlet node will be defined as instances of class "ssn:Sensor".
- Based on the definition of "ssn:Observation", one sensor can produce many observations. An observation defines a situation of measurement. It is a core class of SSNO to describe sensor measurements.

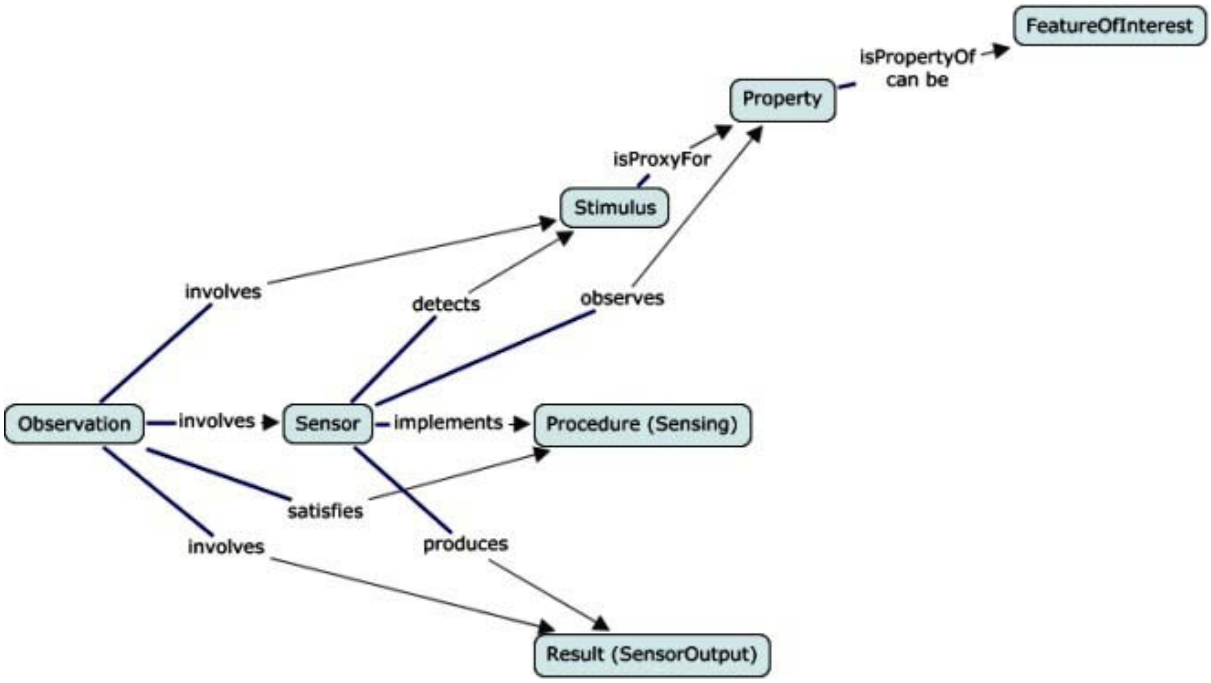


Figure 5.3.2: Stimulus-Sensor-Observation pattern of SSNO

The organization of these main classes is given by the Stimulus-Sensor-Observation design pattern of SSN ontology. As shown in Figure 5.3.2:

- An instance of the class “ssn:FeatureOfInterest” may be associated to several instances of the class “ssn:Property” by the object property “ssn:isPropertyOf”. Let take the example of “watercourse” instance of the “ssn:FeatureOfInterest” class. Two instances of “ssn:Property” class, max slope waterflow rate per 6h and last water flow rate, are linked to the “watercourse” instance,
- An instance of “ssn:Property” may be linked to several instances of “FeatureOfInterest” class. For example, max slope of water flow rate per 6h, the instance of the “ssn:Property” class, is linked to two instances of “ssn:FeatureOfInterest” class: “watercourse” and “outlet”,
- An instance of “ssn:Sensor” is linked to several instances of “ssn:Property” by the object property “ssn:observes”, e.g., one watercourse node, instance of “ssn:Sensor” class, is linked to two instances of “ssn:Property”: last water flow rate and max slope of water flow rate per 6h.

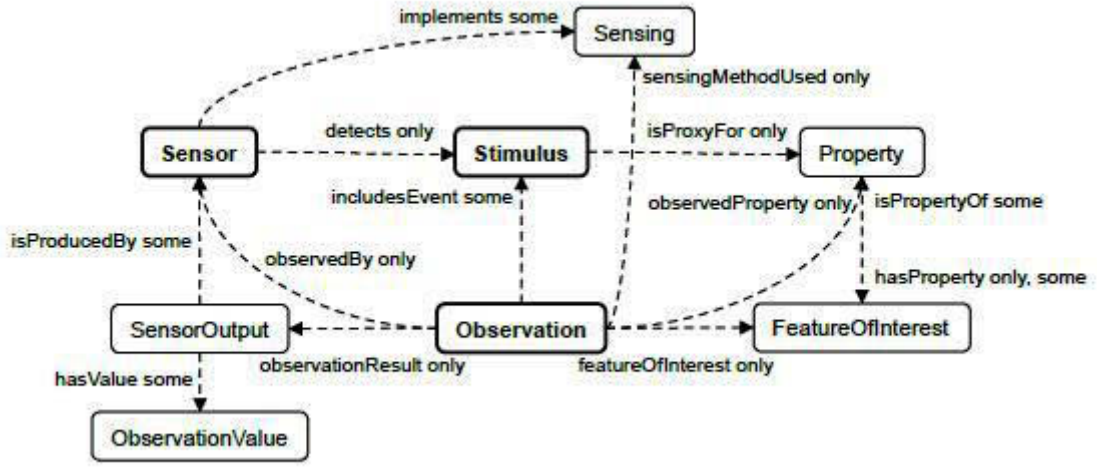


Figure 5.3.3: Stimulus-Sensor-Observation pattern of SSNO

An instance of “ssn:Observation” is linked to several instances as shown in Figure 5.3.3:

- An instance of “ssn:Sensor”, which does the sensing process, by the object property “ssn:observedBy”,
- An instance of “ssn:Property”, which explains the type of measurement performed, by the object property “ssn:observedProperty”,
- An instance of “ssn:FeatureOfInterest”, which describes the phenomenon observed, by the object property “ssn:featureOfInterest”,
- An instance of the class “ssn:SensorOutput”, which stores the measurement value, by the object property “ssn:observationResult”,
- An instance of the class “time:interval”, which stores the time interval of the measurement, by the object property “ssn:observationResultTime”.

We call our OWL ontology Irstea Hydro ontology. In Irstea Hydro ontology, we reuse only some parts of SSNO.

5.3.2.3 Other OWL Ontologies

Our observable entities and entity of interest are defined as instances of the class "ssn:FeatureOfInterest". To complete the ontology network based on SSNO, we reuse some ontology components which come from some dedicated ontologies. The Figure 5.3.4 presents these ontologies. The ontologies that we reuse are:

- The Wireless Semantic Sensor Network Ontology (WSSN) presented in (Bendadouche *et al*., 2012). We are interested in its definition of "state". In the work of (Bendadouche *et al*., 2012), "wssn:State" is a subclass of "ssn:Concept". A concept is defined by some descriptions and is used to classify entities. We reuse the object property: "wssn:hasState" that links any instance of "owl:Thing" class to an instance of "wssn:State" class,
- Climate and Forecast ontology (CF features, 2005). It represents the generic features defined by Climate and Forecast (CF) standard names vocabulary. We reuse the instance of class "ssn:FeatureOfInterest" called "cf-feature:rainfall",
- SSNO reuse two ontologies: DUL ontology and W3C Time ontology. Some DUL elements are used to describe the value measurements. W3C Time ontology is used to describe the time instant or interval.

5.3.2.4 Irstea Hydro Ontology

Irstea Hydro ontology is based on SSNO. Its goal is to describe the sensor network and the data used in our simulation. If needed, we can also extend Irstea Hydro ontology to publish our simulation data on t he Web of data using Semantic Web technologies. That means translating our simulation dataset into Linked Open Data (LOD) dataset.

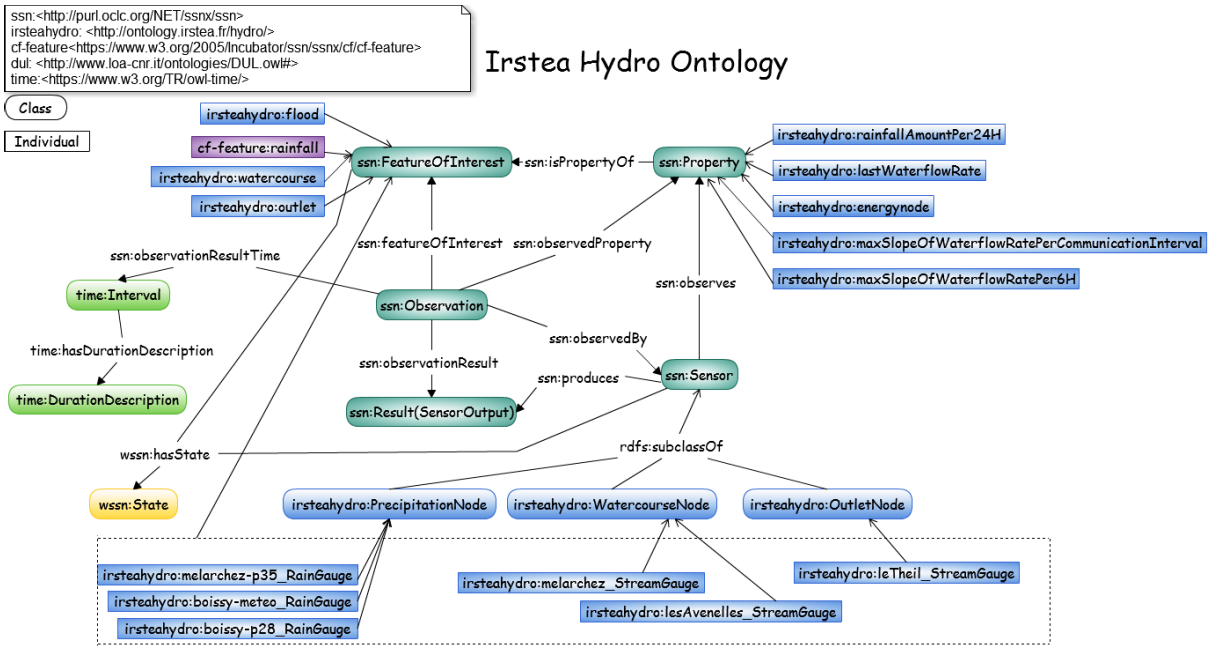


Figure 5.3.4: Irstea Hydro ontology based on an ontology network

Figure 5.3.4 presents the classes, object properties and the instances of classes used in Irstea Hydro Ontology. We reuse several instances from the ontologies network. We also create several instances defined by Irstea Hydro Ontology. We use prefixes and different colors to identify the classes, properties and the instances of classes we reuse and create.

- From SSNO, we reuse five classes: "ssn:Sensor", "ssn:Observation", "ssn:FeatureOfInterest", "ssn:Property" and "ssn:Result(SensorOutput)". We also reuse eight object properties: "ssn:observedProperty", "ssn:featureOfInterest", "ssn:observeBy", "ssn:observationResult", "ssn:observationResultTime", "ssn:isPropertyOf", "ssn:observes" and "ssn:produces". All the SSNO elements are colored in dark green in Figure 5.3.4,
- From W3C Time Ontology, we reuse four classes: "time:Instant", "time:Interval", "time:DateTimeDescription" and "time:DurationDescription". We also reuse several object property and datatype property. All the Time elements are represented in light green in Figure 5.3.4,
- From WSSN Ontology, "wssn:hasState" is reused as an object property that links an instance of class "ssn:FeatureOfInterest" (or an instance of the class "ssn:Sensor) to an instance of class "wssn:State". The class "wssn:State" is represented in yellow in Figure 5.3.4,
- From Climate and Forecast ontology, we reuse the instance "cf-feature:rainfall". This instance is represented in purple in Figure 5.3.4. "cf-feature:rainfall" is linked to several instances of "ssn:Observation" by the object property "ssn:featureOfInterest".

We also create several classes and instances of classes in our Irstea Hydro Ontology. They are represented in blue in Figure 5.3.4. We create subclasses of class "ssn:Sensor" and associated instances in order to define sensor instance based on the feature of interest it observes. As explain before, we create new "ssn:Property" instances and "ssn:FeatureOfInterest" instances. We will presents in detail Irstea Hydro ontology by providing several examples of observation made by different types of sensor nodes.

The base URI for Irstea Hydro ontology is http://ontology.irstea.fr/hydro/, prefixed as "irsteahydro". Hence all instances follow the same naming pattern "irsteahydro:resource/"+"class name of SSNO"+ "/" + "instance ID". Let us take the example of a watercourse node located at Les Avenelles. This node has a stream gauge sensor. The ID to identify this sensor is "lesAvenelles_StreamGauge". Thus, the URI associated to this node is irsteahydro:resource/sensor/lesAvenelles_StreamGauge.

First, we focus on the measurement of the energy of the node. Remember that we name a variable $Energy_{Node}$ in section 4 to store this measurement. Thus, we create an instance of "ssn:Property" with the URI "irsteahydro:resource/property/energyNode". The feature of interest is the node itself so we do not duplicate this information. But the object property "ssn:isPropertyOf" express the link between an "ssn:Property" instance and a "ssn:FeatureOfInterest" instance. Irstea Hydro ontology describes observations about energy

measurement for all the types of node: precipitation nodes, watercourse nodes and outlet node. Figure 5.3.5 presents an example of Irstea Hydro Ontology describing an observation about energy measurement of the outlet node.

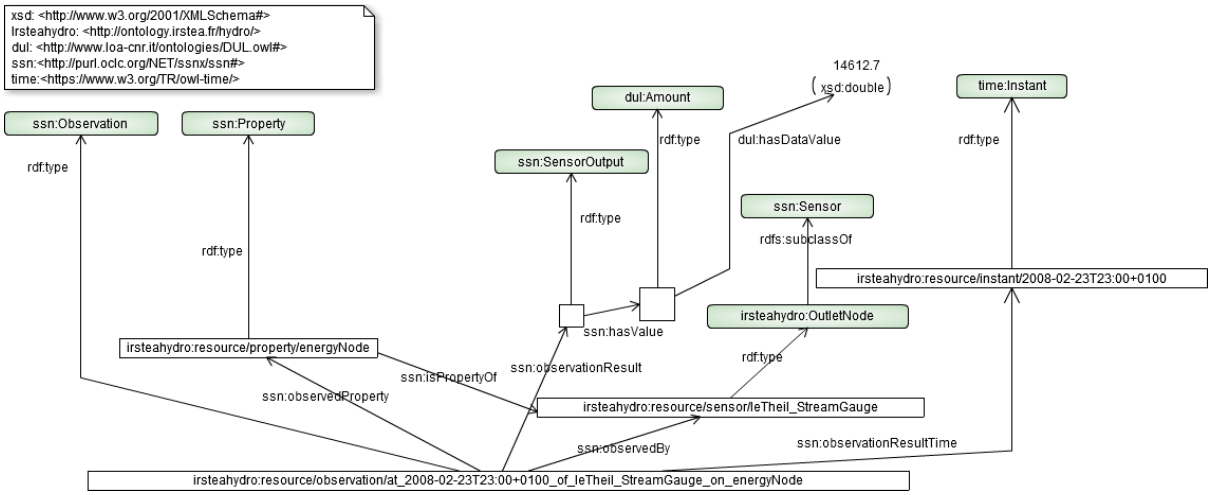


Figure 5.3.5: Description of an observation about node energy made by an outlet node

Figure 5.3.5 represents an observation done by the sensor named "leTheil_StreamGauge". This observation is about an energy measurement done at the given time instant "23/02/2008 23:00". For each measurement, we create an instance of "ssn:Observation" class. The ID of an observation is the concatenation of the time element ID + sensor ID + property ID. To describe this type of measurement using SSNO we need several instances, classes and properties. We create the class "irsteahydro:OutletNode", which is a subclass of class "ssn:Sensor". This class stores all the instances of sensor dedicated to the outlet feature of interest. We create the associated sensor instance "irsteahydro:resource/sensor/leTheil_StreamGauge". Concerning measurement value, SSNO reuse the DUL ontology. The xsd:double value (14612.7) is the measurement value. It is linked to an instance of "ssn:SensorOutput" by a path of properties: "ssn:hasValue" and "dul:hasDataValue". The node "irsteahydro:resource/instant/2008-02-23T23:00+0100" is an instance of "time:Instant" class that corresponds to the time value "23/02/2008 23:00".

Figure 5.3.6 presents an example of description of this time instant using the W3C Time ontology.

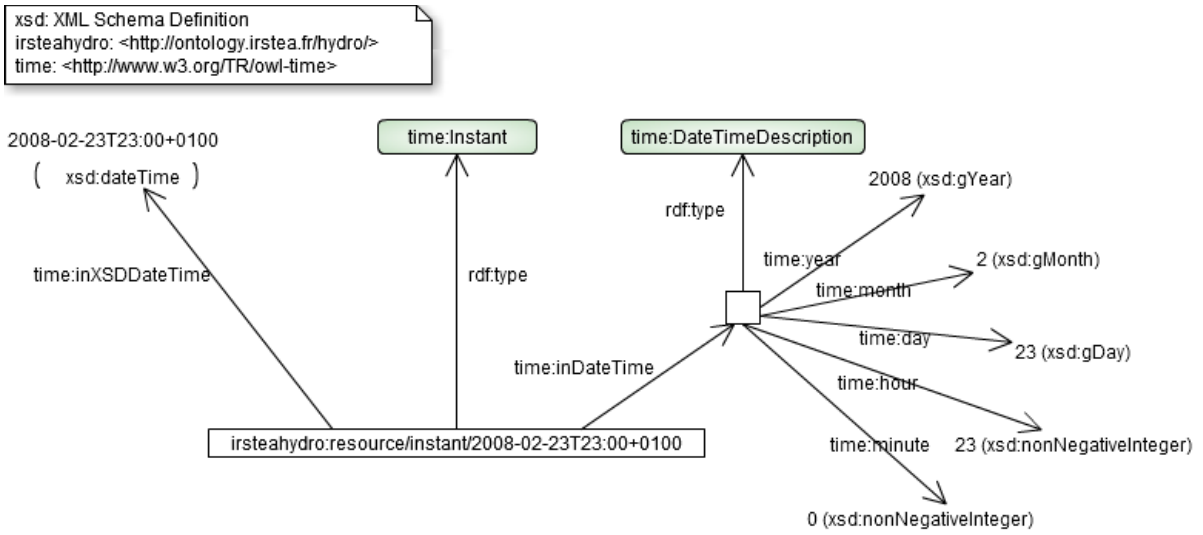


Figure 5.3.6: Description of an observation related to a time instant

The xsd:dateTime value "2008-02-23T23:00+0100" is linked to the instance of "time:Instant" by a d atatype property "time:inXSDDateTime". To describe more precisely all the components of a time instant the W3C Time ontology define the "time:DateTimeDescription" class with several associated datatype properties, such as "time:year", "time:month", "time:day", "time:hour", and "time:minute" illustrated in Figure 5.3.6.

Figure 5.3.7 pr esents an example of Irstea Hydro ontology describing an observation about rainfall amount per 24h measurement made by a precipitation node.

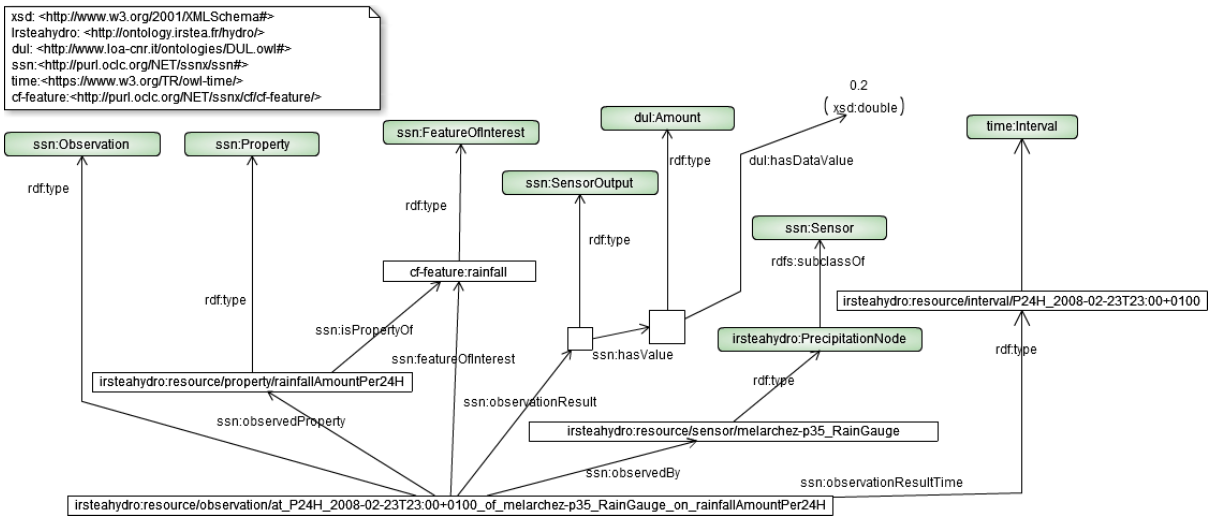


Figure 5.3.7: Description of an observation on rainfall amount per 24h made by a precipitation node

Figure 5.3.7 represents an observation done by the sensor named melarchez-p35_RainGauge. We reuse in this example the instance “cf-feature:rainfall” of the class “ssn:FeatureOfInterest” because the Climate and Forecast ontology does not define any “ssn:Property” instance that specified the duration of the rainfall, a new “ssn:Property” instance is created. This instance is identified by the URI “irsteahydro:resource/property/rainfallAmountPer24H”. In this example, we also create a new class “irsteahydro:PrecipitationNode”, which is a subclass of class “ssn:sensor”. This class gathers all sensor instances dedicated to observe precipitation phenomenon. Thus, we create the associated instance “irsteahydro:resource/sensor/melarchez-p35_RainGauge” to represent our specific precipitation node.

Like previous example, the xsd:double value (0.2) is linked to the instance of “ssn:SensorOutput” by a path of properties: “ssn:hasValue” and “dul:hasDataValue”.

The measurement of rainfall amount corresponds to a period of 24 hours ending at “23/02/2008 23:00” The object property “ssn:observationResultTime” links the instance of “ssn:Observation” to an instance of “time:Interval”.

Figure 5.3.8 presents an example of description of this time interval using the W3C Time ontology.

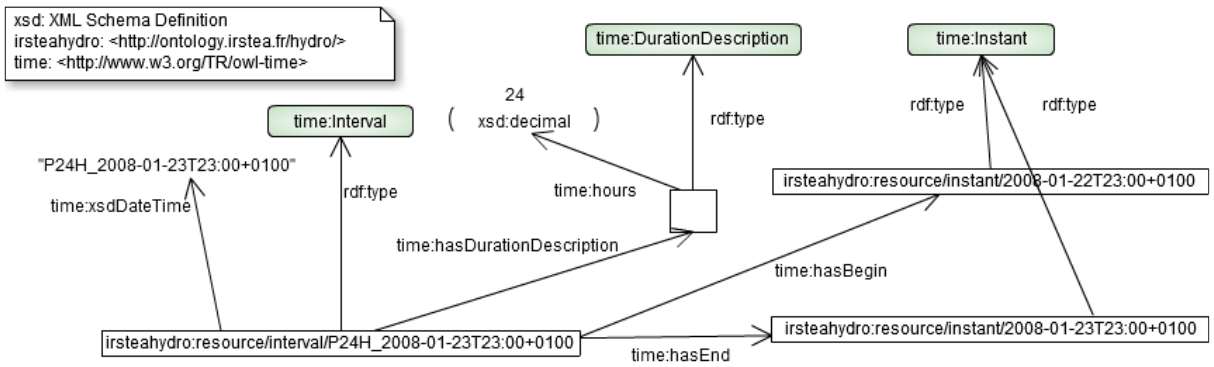


Figure 5.3.8: Description of a time interval using the Time ontology

Figure 5.3.8 presents the description of a time interval. The properties “time:hasBegin” and “time:hasEnd” link the “time:Interval” instance to the “time:Instant” instances that represent the begin and the end of the interval. The duration of the interval (24 hours) is specified by a property path: “time:hasDurationDescription” and “time:hours” from the “time:Instant” instance.

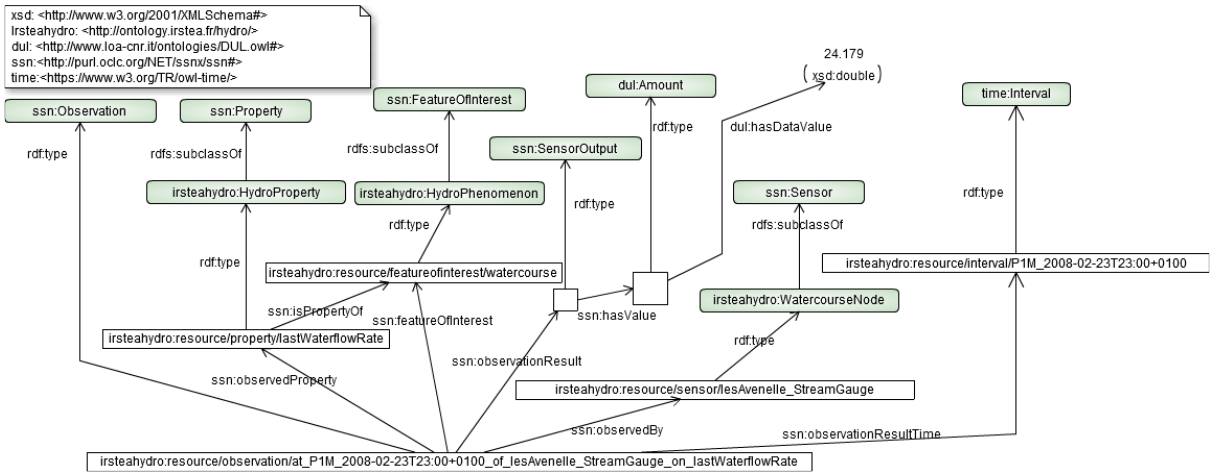


Figure 5.3.9: Description of an observation on last water flow rate of a watercourse node

Figure 5.3.9 represents an observation performed by the sensor lesAvenelles_StreamGauge. The measurement is about last water flow rate. For this purpose, we create a subclass of "ssn:FeatureOfInterest" to gather all instances dedicated to hydrographic phenomenon. This new class is named "irsteahydro:HydroPhenomenon". Then, an instance of this class is created which has the URI "irsteahydro:resource/featureofinterest/watercourse".

A subclass of "ssn:Property" is also created to gather the property instances dedicated to hydrographic phenomenon. This class is called "irsteahydro:HydroProperty". An instance of this new class is created with the URI "irsteahydro:resource/property/lastWaterflowRate".

We create a n ew class "irsteahydro:WatercourseNode", which is a subclass of class "ssn:sensor". An instance of this new class is created to represent the specific node :"irsteahydro:resource/sensor/lesAvenelles_StreamGauge".

The xsd:double value "24.179", which is the measurement value, is linked to the instance of "ssn:SensorOutput" by a path of properties: "ssn:hasValue" and "dul:hasDataValue".

The URI "irsteahydro:resource/interval/P1M_2008-02-23T23:00+0100" represents an instance of "time:Interval" class (see Figure 5.3.8 for more detail about interval description). The object property "ssn:observationResultTime" links the instance of "ssn:Observation" to the instance of "time:Interval".

Figure 5.3.10 shows an example of Irstea Hydro ontology describing an observation of a "max slope of waterflow rate per 6h" made by an outlet node.

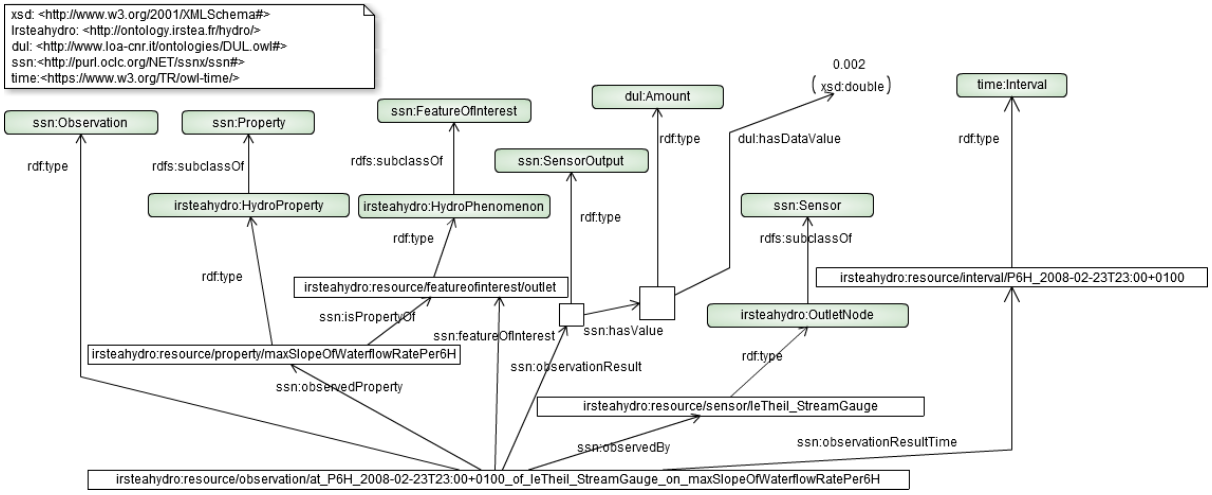


Figure 5.3.10: Description of an observation of a "max slope of waterflow rate per 6h" made by an outlet node

Figure 5.3.10 represents an observation done by the sensor leTheil_StreamGauge. This observation stores the aggregated value max slope of waterflow rate per 6h. This value corresponds to time interval from "23/02/2008 17:00" to "23/02/2008 23:00".

A new instance of "ssn:Observation" is created that will be linked to several instances to describe the situation of the measurement. A new instance of "irsteahydro:HydroPhenomenon" class is created. It is named "irsteahydro:resource/featureofinterest/outlet" and it represents the Outlet observable entity. We also create the associated instance of "irsteahydro:HydroProperty" class. This instance has the URI "irsteahydro:resource/property/maxSlopeOfWaterflowRatePer6H". We create a new class "irsteahydro:OutletNode", which is a subclass of class "ssn:Sensor". "irsteahydro:resource/sensor/leTheil_StreamGauge" is an instance of "irsteahydro:OutletNode" class. The xsd:double value "0.002" is linked to the instance of "ssn:SensorOutput" by a path of properties: "ssn:hasValue" and "dul:hasDataValue".

A new instance of "time:Interval" class is created to represent the interval from "23/02/2008 17:00" to "23/02/2008 23:00". This instance is linked to the "ssn:Observation" instance by the property "ssn:observationResultTime".

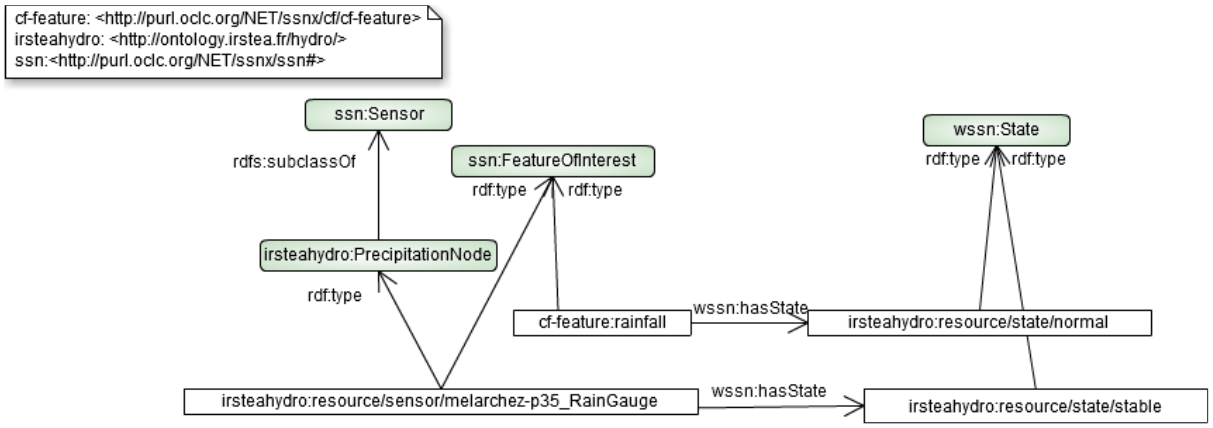


Figure 5.3.11: Description of entities states using WSSN ontology

Figure 5.3.11 represents the states of observed entities. Then "irsteahydro:resource/sensor/melarchez-p35_RainGauge" is an instance of "irsteahydro:PrecipitationNode" class. It is also an instance of "ssn:FeatureOfInterest". We create two instances of "wssn:State" class: "irsteahydro:resource/state/normal" and "irsteahydro:resource/state/stable". These two instances are linked to instances of "ssn:FeatureOfInterest" by the object property "wssn:hasState".

It can be noted that we use the same modeling focused on the "ssn:Observation" instance to describe a measurement made by a physical sensor or a computed value computed by a virtual sensor. For example a watercourse node aggregates some measurement of water flow rate to compute the max slope of water flow rate per 6h value. This is the same sensor node which does the measurement and the computing. A node has a sensing component and a processing component as presented in section 5.2. Thus, we decide to associate the sensor node ID to the observation instead of creating specific sensor ID to dissociate the physical sensor from the virtual one. The physical sensor or the virtual sensor may be identified by the "ssn:Property" instance associate to the observation.

In the following section, a method is presented in order to translate the Irstea Hydro ontology into a JADE ontology.

5.3.3 Creation of Our JADE Ontology

In this section, we will describe how we create the JADE ontology which models the agent message content from the Irstea Hydro ontology. Let us remind you that these two ontologies are not based on the same components (OWL components and JADE components). Thus, we will explain our translation method. The Figure 5.3.12 gives an overview of the creation of our JADE ontology.

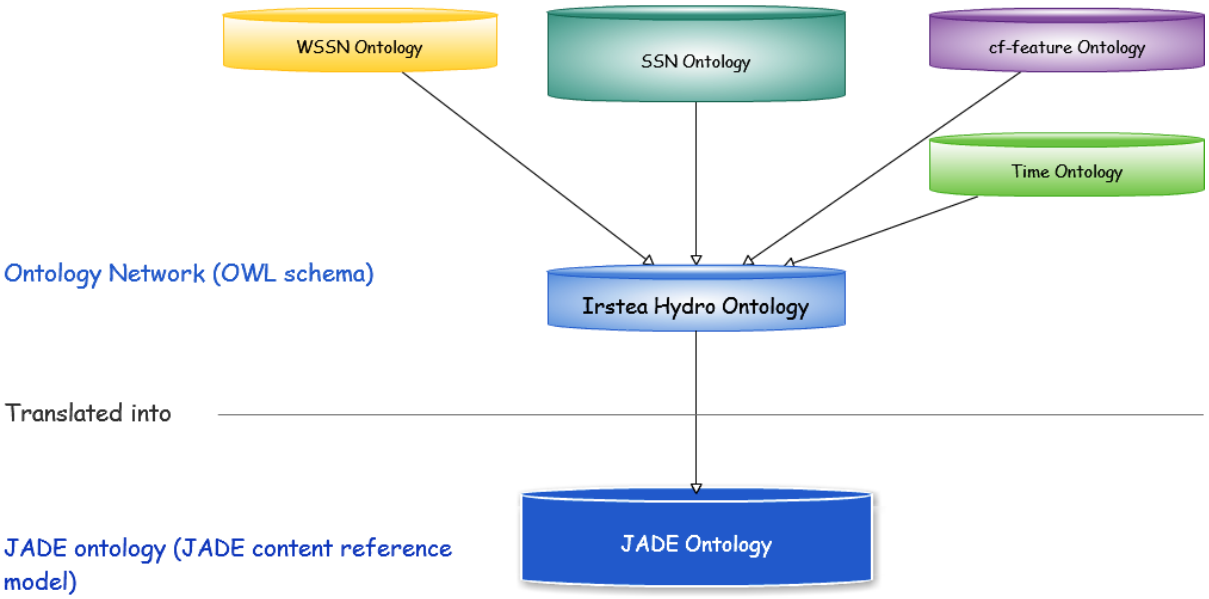


Figure 5.3.12: Architecture of JADE Ontology translated based on ontologies network

5.3.3.1 Translation Method of a Network of OWL Ontologies to One JADE Ontology

Based on the presentation of OWL language components and the JADE Content Reference Model, we propose several easy translation rules presented in Table 5.3.1. An OWL class is translated into a JADE concept. A data type property is translated into a JADE slot etc.

Table 5.3.1: Schema of OWL components translated into JADE ontology

OWL component	JADE content reference model component
Class	Concept
Datatype property	Slot
Object property	Predicate
Class Instance	Concept Instance

If we follow these translation rules we should translate each object property into a JADE predicate. In our Irstea Hydro ontology, there are lots of object properties that link two instances.

We notice that there exist different types of instances in our model. There are some of them which play the role of reference. For example, all “ssn:Observation” instances that describe the same type of measurement are linked to the same “ssn:Property” and

"ssn:FeatureOfInterest" instances. "ssn:Property" and "ssn:FeatureOfInterest" instances play a role of reference to retrieve all the measurements of the same type.

In Java programming language the reference role are played as string constant. Moreover in JADE Content Reference Model the role of a predicate is to be part of a QUERY-IF message. An agent asks by means of a QUERY-IF message whether a predicate is true or false to another agent. Then it waits for the answer. Thus it is time consuming. We decide not to use QUERY-IF message in our simulation because all the message between agents are INFORM messages or REQUEST messages as presented in Figure 5.3.13. INFORM messages are used by a sensor node agent to inform the DSS agent about their measurement value. REQUEST message are used by the DSS agent to ask a sensor node agent to change its communication frequency.

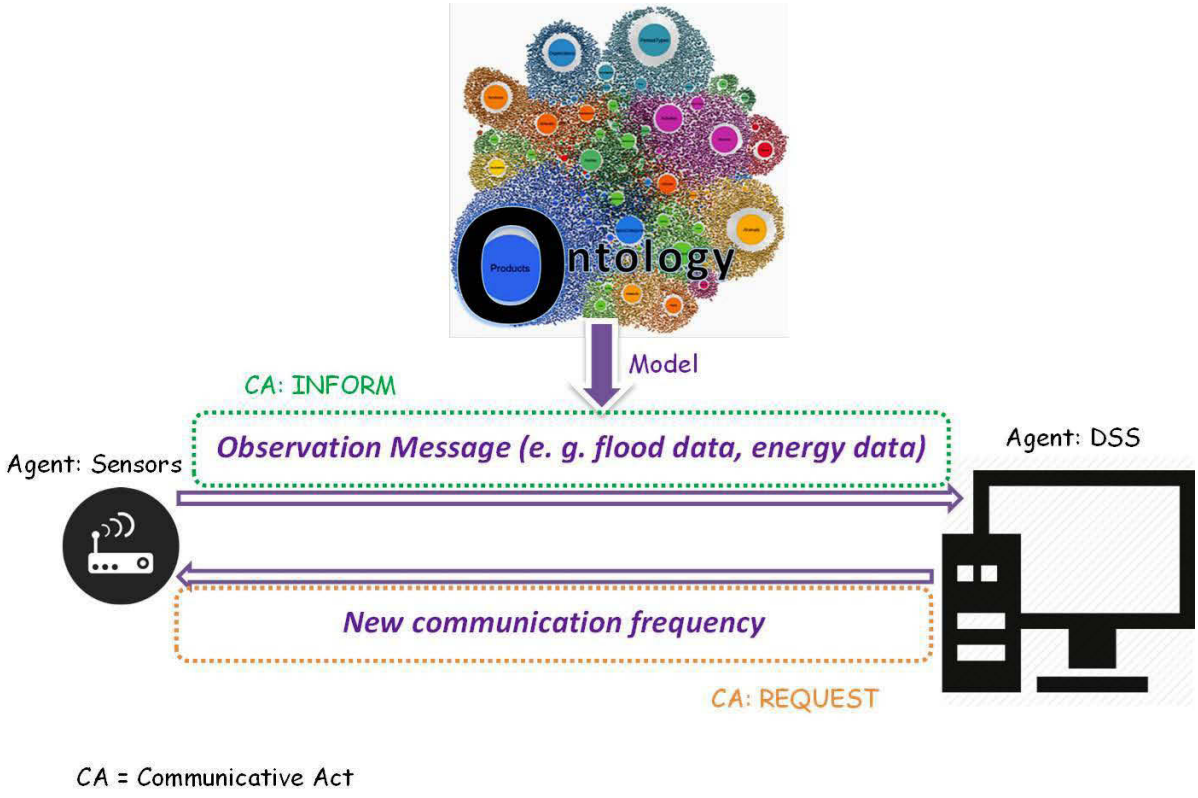


Figure 5.3.13: Communicative act of messages used among agents in our simulation

Thus, we decide to translate each OWL object property by a JADE slot as presented in Table 5.3.2. In our translation process, the names of the JADE components are borrowed from the name of the OWL components.

Table 5.3.2: Schema of OWL components translated into JADE ontology

OWL Component	JADE Content Reference Model Component
Class	Concept
Datatype Property	Slot
Object Property (link between a source instance and a target instance)	Slot in the target concept
Class Instance	Concept Instance

5.3.3.2 Our JADE Ontology

Figure 5.3.14 pr esents the components of JADE Ontology translated from Irstea Hydro Ontology. Figure 5.3.14 presents our JADE ontology by an UML class diagram.

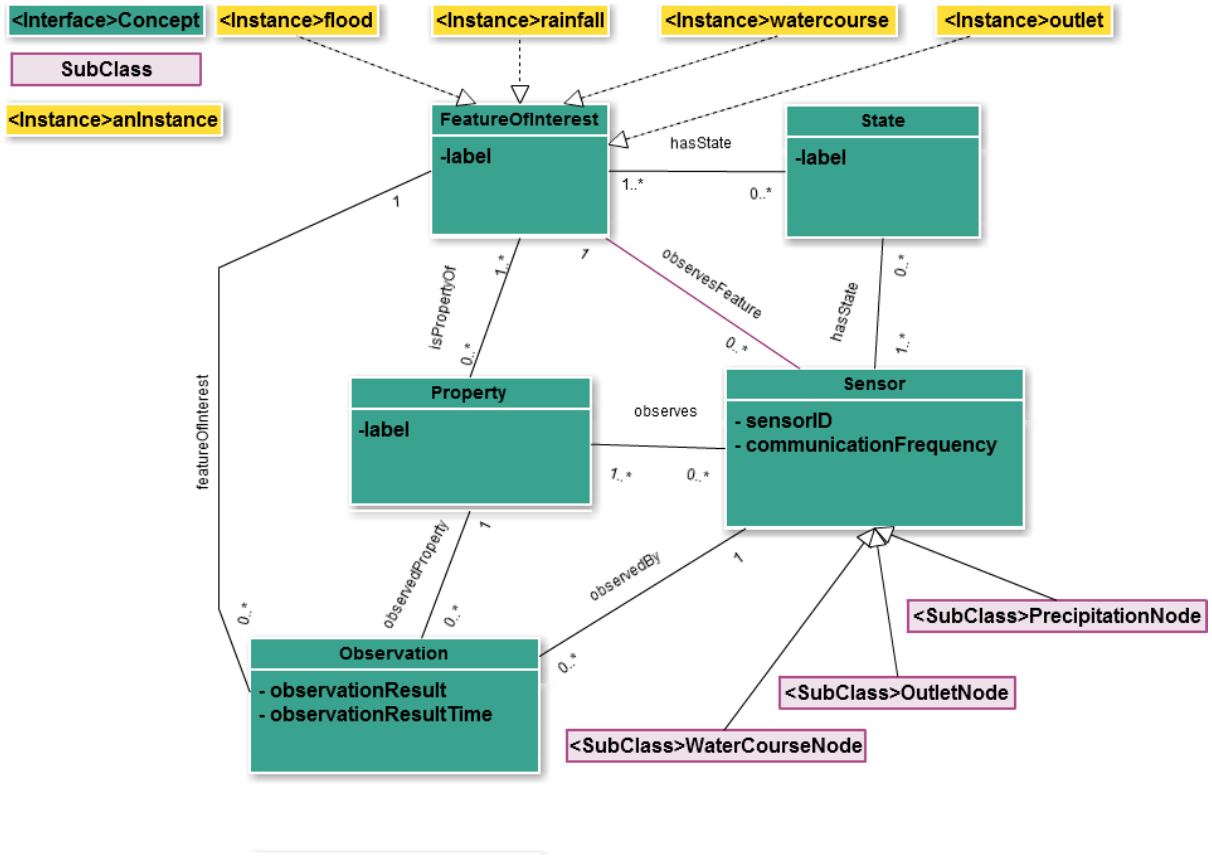


Figure 5.3.14: Conceptual model of JADE ontology

Our JADE ontology has four JADE concepts that come from SSNO classes: “Sensor”, “FeatureOfInterest”, “Property” and “Observation”. Note that the “Observation” concept has two slots: “observationResult” that stores the value and the “observationResultTime” that stores the date time of an instant or the date time of the ending instant of an interval.

The “Sensor” concept has two slots that do not come from Irstea Hydro: its *sensorId* and its *communicationFrequency*.

One JADE concept comes from the WSSN ontology: “State”.

Three concepts come from the translation of Irstea Hydro ontology: “PrecipitationNode”, “OutletNode” and “WaterCourseNode”. They all inherit from the “Sensor” concept.

We also create as many instances of “FeatureOfInterest”, “Property” and “State” concepts as necessary. All these concepts have a “label” slot to store their reference string constant. The Figure 5.3.14 presents only some instances of “FeatureOfInterest” concept namely: “flood”, “watercourse”, “outlet” and “rainfall”.

Notice that in SSNO, an instance of “ssn:Sensor” is not linked to an instance of “ssn:FeatureOfInterest” directly. However, we decide to add an association between “Sensor” concept and “FeatureOfInterest” concept. The connection link is presented as a pink straight line. The name of this new association is “observesFeature”. The cardinality between “Sensor” concept and “FeatureOfInterest” concept is 0..* to 1. In our JADE ontology, an instance of “Sensor” can observe only one instance of “FeatureOfInterest”. One instance of “FeatureOfInterest” can be observed by many instances of “Sensor”.

The Figure 5.3.15 presents the final class diagram of our JADE Ontology.

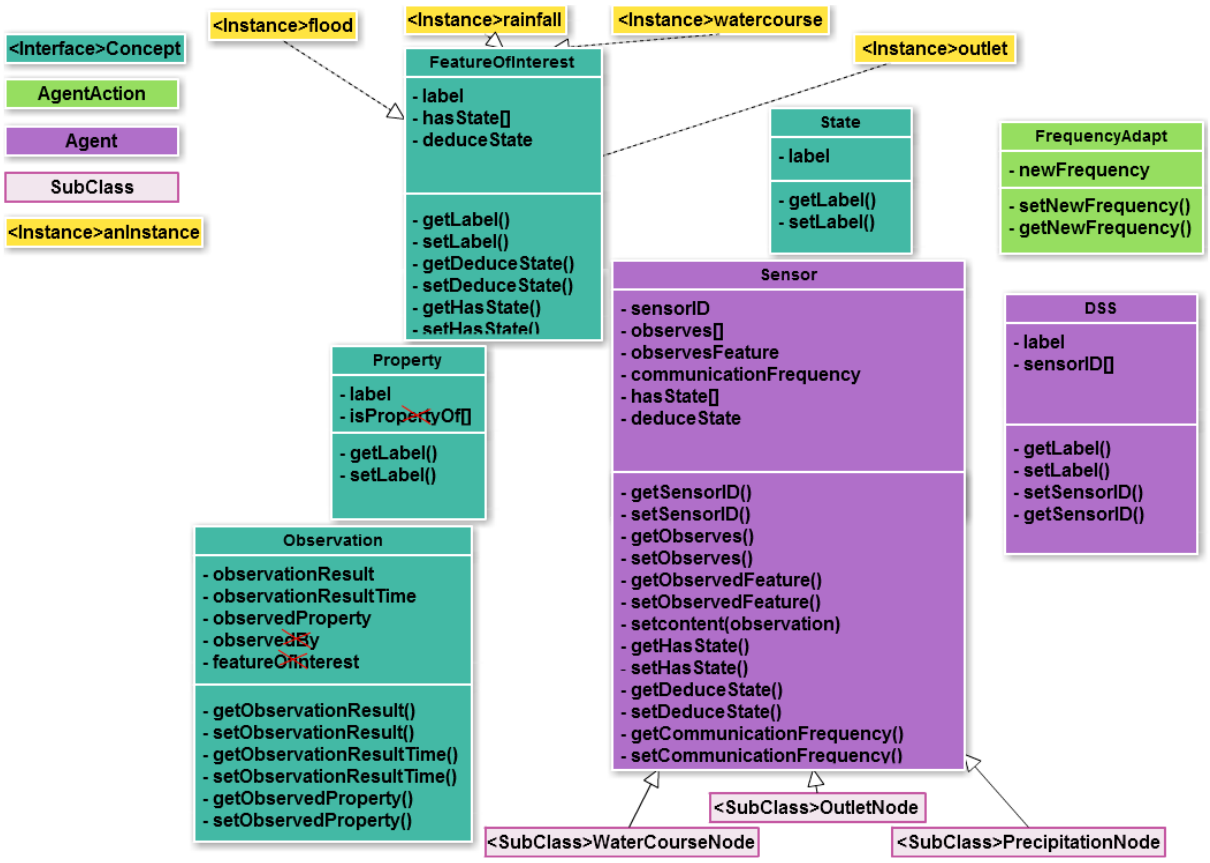


Figure 5.3.15: Class diagram of JADE ontology

Figure 5.3.15 presents the final JADE ontology. It has different elements than those presented in Figure 5.3.14. T able 5.3.3 pr esents all the entities of our JADE ontology. The lines presented in bold format mean that the entity is new compared with our Irstea Hydro ontology translation mechanism.

The “DSS” agent is added. It has the slot “sensorID[ ]” that contains the list of sensors. “Sensor” becomes JADE agent. The “DSS” agent will ask an action among the “Sensor” agents to adapt to the new communication frequency. Thus, the”FrequencyAdapt” agent action is added. It has the slot “newFrequency” to store the value of the frequency.

Table 5.3.3: Provenance of JADE ontology elements

JADE CRM Component	Type	OWL component
FeatureOfInterest	Concept	ssn:FeatureOfInterest
Property	Concept	ssn:Property
Sensor	Agent	ssn:Sensor
Observation	Concept	ssn:Observation
WatercourseNode	Agent	irsteahydro:WatercourseNode
OutletNode	Agent	irsteahydro:OutletNode
PrecipitationNode	Agent	irsteahydro:PrecipitationNode
State	Concept	wssn:State
DSS	Agent	
FrequencyAdapt	Agent Action	

Table 5.3.4 presents in detail the slots of the different JADE ontology elements:

- The first column indicates the owner of the slot.
- The second column presents the slot name.
- The third column indicates if the slot comes from an Irstea hydro ontology object property.
- The last column presents some comments about the slot.

Table 5.3.4: Provenance of the slots in our JADE ontology

JADE concept	JADE slot	OWL object property	Comment
Property	isPropertyOf[]: contains the label of some FeatureOfInterest instances associated to	ssn: isPropertyOf	Not use. Due to the fact that a Sensor instance is dedicated to one instance of FeatureOfInterest, this information is already stored

	the Property instance		in Sensor instance.
FeatureOfInterest Sensor	hasState[]: contains the label of the associated State instances.	wssn:hasState	
FeatureOfInterest Sensor	deduceState: contains the label of the current State instance.		This value is inferred by the DSS.
Sensor	observes[]: contains the label of some Property instance that the sensor instance can observe.	ssn:observes	
Sensor	observesFeature: contains the label of the FeatureOfInterest		We create it. A sensor is dedicated to the observation of one natural phenomenon, one instance of FeatureOfInterest.
Sensor	sensorID: the ID of the sensor		The URI of the ssn:Sensor instance reuses the sensorID.
Sensor	communicationFrequency: the communication frequency of the sensor		
Observation	observedBy: contains the sensorID	ssn:observedBy	Not use because it is part of the message header in JADE agent communication.
Observation	observedResult: contains the value	ssn:observedResult + ssn:hasValue+dul:hasDataValue	We decide to name the slot with the first object property name.
Observation	observedResultTime: contains the date time of a instant or the date time of the ending	ssn:observedResultTime+time:inXSDDateTime	

	Instant of an interval	or ssn:observedResultTime+time:hasEnd+time:inXSDDateTime	
Observation	observedProperty: contains the label of the Property	ssn:observedProperty	
Observation	featureOfInterest: contains the label of FeatureOfInterest	ssn:featureOfInterest	Not in use, it can be derived from the observedBy and the observedProperty. A sensor is dedicated to the observation of one environmental phenomenon, one instance of FeatureOfInterest.

As it may be noticed, if the slot is the translation of an object property, the slot name is a copy of the object property name. In our translation of Irstea Hydro ontology some slots may be discarded because their information are duplicated or may be derived from others. For example, this is the case for the slots "observedBy" and "featureOfInterest" of the "Observation" concept. All discarded slots are presented with a red cross in Figure 5.3.15. All new slot that do not come from irstea hydro ontology translation are presented in bold format in the Table 5.3.4.

5.4 Context Reasoning

As mentioned in our proposition (c.f. section 3.1), the main advantage of our context modeling is the ability to split the reasoning process into several steps which help the management of this process. The reasoning process is dedicated to the construction of the high-level context from the low-level one (c.f. section 4). All context information is stored in the JADE ontology (c.f. section 5.3).

Section 5.4.1 pr esents how to establish thresholds that will change the states of the Observable Entities: Node entity (*N*), Precipitation entity (*P*), Watercourse entity (*W*) and Outlet entity (*O*). This process builds the high-level context.

Section 5.4.2 introduces two sets of Jess rules and is illustrated with an example. Rules enrich the high-level context:

- The first example is about the state of Flood entity of interest (*F*) which is inferred from the state of *P*, *W* and *O* entities.

- The second example focuses on the state of *N* entity which is inferred from the energy level of battery.

5.4.1 Thresholds Setting

Due to the chronological order of the flood phenomenon, as mentioned in section 4.2.2, the EOEs have different priority levels:

$$P < W < O$$

Therefore, in our simulation, we want that the Precipitation entity should be at the risk state more often than the Watercourse entity. The Watercourse entity should be more often at the "Risky" state than the Outlet entity. Thus, we will fix the thresholds in order to have more state changes for the Precipitation entity than for the Watercourse entity. In the same way, the Watercourse entity will have more state changes than the Outlet entity.

To compute the threshold, we use scenario 1 configuration 4. That means that all nodes have the same configuration. Their communication frequency is equal to their sample frequency. They send their measurements to the DSS. Then the DSS computes some aggregated values based on these measurements:

- Precipitation nodes send the "rainfall amount per 24h" value to the DSS. The DSS computes the total amount of precipitations that fall on the watershed per 24 hours.
- Watercourse nodes send the "last water flow rate" to the DSS. The DSS computes the slope between the two last received values from the same node. Then, it computes the minimum and maximum of the slopes from the two watercourse nodes for a whole year.
- Outlet node sends the "last water flow rate" to the DSS. The DSS computes the slope between the two last received values.

To determine the thresholds dedicated to the observable entities e.g., Precipitation entity, we use real historical data from the French Orgeval watershed for one calendar year. We select the year 2007 when a risky event has appeared during this period. After, using these established thresholds, we will run simulations on the archive of year 2008 in order to test them.

To determine the threshold dedicated to. Precipitation entity, we use real historical data from the calendar year 2007. We select this year because a risky event has appeared. This 2007 archive will be used to compute the threshold. We will after run the simulation on archive from 2008. We compute the aggregated value associated to the entity, e.g.,$Precipitation_{watershed}$ or the Precipitation entity. At the time stamp of the risky event the aggregated value has reached its maximum. We also compute the minimum of the aggregated value for 2007. The threshold is between the min and the max of the aggregated value. We propose several functions to evaluate the threshold. The functions look like *[(max - min)/ x] + min* where *x* will be fixed experimentally Then we project the different value of threshold on

the evolution of the aggregated value during 2007. The goal is to select the value that best match our expectation. For example, the threshold has to fulfill the constraint relative to the priority order of observables entities. We also project the threshold value on 2008 to be sure that our simulation will run correctly.

First, we present the computation of the threshold for the Precipitation entity, called $Th_{Precipitation}$. We use real historical precipitation data from the Orgeval dataset during year 2007 and 2008.

5.4.1.1 Threshold of the Precipitation Entity (*P*)

We know that there was a real flood event in Orgeval watershed at the date "03/07/2007". Therefore, we compute the different values of the "rainfall amount per 24h" variable for the different considered Orgeval precipitation stations (Melarchez-p35, Boissy-meteo and Boissy-p28) from 01/01/2007 to 31/12/2007. The evolution of this variable is presented in Figure 5.4.1. We can see, in this Figure, where the time instant for the maximum value is "03/07/2007". This instant corresponds to the day when the flood event happened. The minimum sum of rainfall amount per 24 hours of Orgeval precipitation stations is 0. So, the value of $Th_{Precipitation}$ (not only for 2007 but for all the years) is supposed to be between the minimum and maximum values. Thus, we propose several functions to set $Th_{Precipitation}$ as shown below:

Equation 5.10: Maximum of total rainfall amount per 24h that falls on the watershed during one calendar year

$$MaxPrecipitation_{watershed} = \max_{i=01/01}^{31/12} Precipitation_{watershed}(i)$$

Equation 5.11: Minimum of total rainfall amount per 24h that falls on the watershed during one calendar year

$$MinPrecipitation_{watershed} = \min_{i=01/01}^{31/12} Precipitation_{watershed}(i)$$

where

$Precipitation_{watershed}$ is the total rainfall amount per 24h that falls on the watershed.

i represents a 24h interval that starts at midnight of a day.

$Precipitation_{watershed}$ is the maximum value of $Precipitation_{watershed}$ during a whole year

$Precipitation_{watershed}$ is the minimum value of $Precipitation_{watershed}$ during a whole year

We define $Th_{Precipitation}$ by the following Equation 5.12:

Equation 5.12: Threshold of the Precipitation Entity of the watershed

$$Th_{Precipitation} = \frac{MaxPrecipitation_{watershed} - MinPrecipitation_{watershed}}{\alpha} + MinPrecipitation_{watershed}$$

α is a variable that will be determined by experimentation.

We all know that the value of $Precipitation_{watershed}$ is 0. So the equation 5.12 becomes:

$$Th_{Precipitation} = \frac{MaxPrecipitation_{watershed}}{\alpha}$$

Figure 5.4.1 shows the sum of rainfall amount per 24 hours of the considered Orgeval precipitation stations (Melarchez-p35, Boissy-meteo and Boissy-p28) from 01/01/2007 to 31/12/2007.

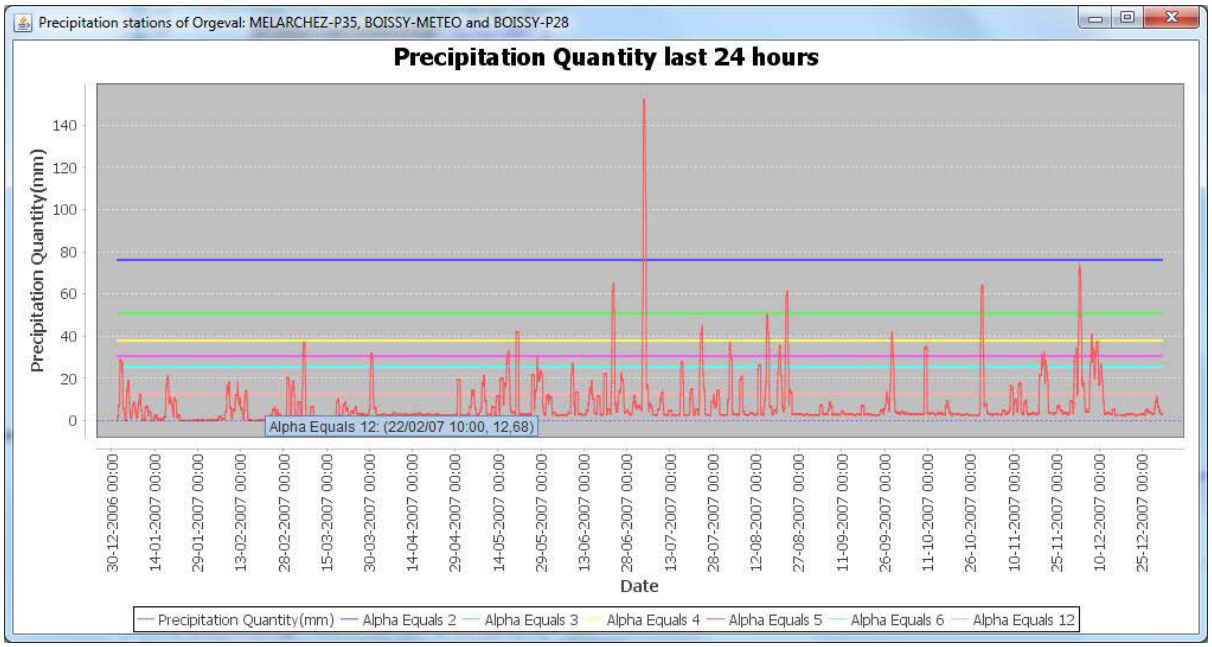


Figure 5.4.1: Sum of rainfall amount per 24 hours of the considered Orgeval precipitation stations for year 2007 with different applied α values

Figure 5.4.1 shows 6 straight lines:

- The purple line is the threshold computed from Equation 13 with $\alpha = 2$ and it detects 1 risky event.
- The green line is the threshold computed with $\alpha = 3$ and it detects 5 risky events.
- The yellow line is the threshold computed with $\alpha = 4$ and it detects 11 risky events.
- The pink line is the threshold computed with $\alpha = 5$ and it detects 20 risky events.

- The light blue line is the threshold computed with $\alpha = 6$ and it detects 24 risky events.
- The light red line is the threshold computed with $\alpha = 12$ and it detects 46 risky events.

These six threshold values are also projected on precipitation data of the Orgeval dataset during the year 2008. The Figure 5.4.2 s hows the sum of rainfall amount per 24h of considered Orgeval precipitation stations (Melarchez-p35, Boissy-meteo and Boissy-p28) from 01/01/2008 to 31/12/2008.

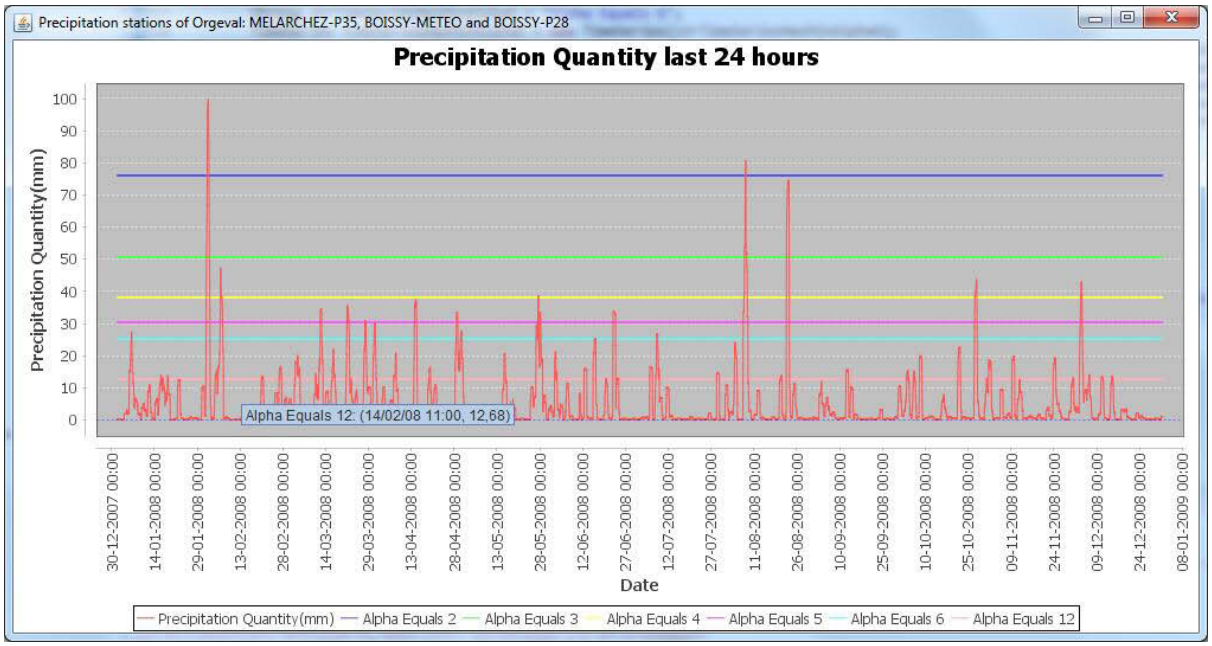


Figure 5.4.2: Sum of rainfall amount per 24 hours of considered Orgeval precipitation stations for year 2008 with different applied α values

For Figure 5.4.2, the number of risky events detected for each line is as follows:

- The purple line where $\alpha = 2$ detects 2 risky events.
- The green line where $\alpha = 3$ detects 3 risky events.
- The yellow line where $\alpha = 4$ detects 6 risky events.
- The pink line where $\alpha = 5$ detects 14 risky events.
- The light blue line where $\alpha = 6$ detects 18 risky events.
- The light red line where $\alpha = 12$ detects 50 risky events.

We can see, from Figure 5.4.2, that the light red straight line (where $\alpha = 12$) is as we expected. Thus, $Th_{Precipitation}$ value is set to 10 mm. This provides enough Precipitation entity state changes to run simulations.

5.4.1.2 Threshold of the Watercourse Entity (*W*)

The Watercourse entity (*W*) is observed by two watercourse nodes. Each node measures the water flow rate of a river (or tributary). We compute the slopes of water flow rate per minute for each node. Per minute, we have two slope values that come from two distinct watercourse nodes. For each minute interval we select the maximum of these two slope values. This maximum is stored in the $Watercourse_{watershed}$ variable (see section 4.2.3.2). The evolution of this variable is presented in Figure 5.4.3. From all the $Watercourse_{watershed}$ computed during this whole year, we establish the minimum and the maximum: $MinWatercourse_{watershed}$ and $MaxWatercourse_{watershed}$. As shown in Figure 5.4.3, the day when the maximum of the $Watercourse_{watershed}$ is reached at the date "03/07/2007". The minimum value of $Watercourse_{watershed}$ is smaller than 0. Thus, $Th_{Watercourse}$ from 01/01/2007 to 31/12/2007 is supposed to be between the maximum and minimum values. Therefore, we have established this $Th_{Watercourse}$ value by the formulas shown below:

Equation 5.13: Maximum of $Watercourse_{watershed}$ value during one calendar year

$$MaxWatercourse_{watershed} = \max_{i=01/01/2007\ \ 00:00:00}^{01/01/2008\ \ 00:00:00} Watercourse_{watershed}(t_l)$$

where

$Watercourse_{watershed}(t_l)$ is the maximum between the last two slopes values related to the distinct watercourse nodes. Remind that the slope is computed for one minute interval.

t_l represent one minute interval. t_l should belong to year 2007.

$MaxWatercourse_{watershed}$ is the maximum of all $Watercourse_{watershed}(t_l)$ computed from 01/01/2007 to 31/12/2007.

Equation 5.14: Minimum of $Watercourse_{watershed}$ value during one calendar year

$$MinWatercourse_{watershed} = \min_{i=01/01/2007\ \ 00:00:00}^{01/01/2008\ \ 00:00:00} Watercourse_{watershed}(t_l)$$

where

$Watercourse_{watershed}(t_l)$ is the maximum between the last two slopes values related to the distinct watercourse nodes.

t_l represent one minute interval. t_l should belong to year 2007.

$Precipitation_{watershed}$ is the minimum of all $Watercourse_{watershed}(t_l)$ computed from 01/01/2007 to 31/12/2007.

Equation 5.15: Threshold of Watercourse in watershed

$$Th_{Watercourse} = \frac{MaxWatercourse_{watershed} - MinWatercourse_{watershed}}{\alpha} + MinWatercourse_{watershed}$$

α is a variable that will be determined by experimentation.

We also propose a new equation to compute the threshold that does not take into account $MinWatercourse_{Watershed}$.

Equation 5.16: Threshold of Watercourse in watershed

$$Th_{Watercourse} = \frac{MaxWatercourse_{watershed}}{\alpha}$$

Figure 5.4.3 shows the maximum and minimum $Watercourse_{watershed}$ using the 2007 archive data of watercourse nodes: Les Avenelles and Melarchez.

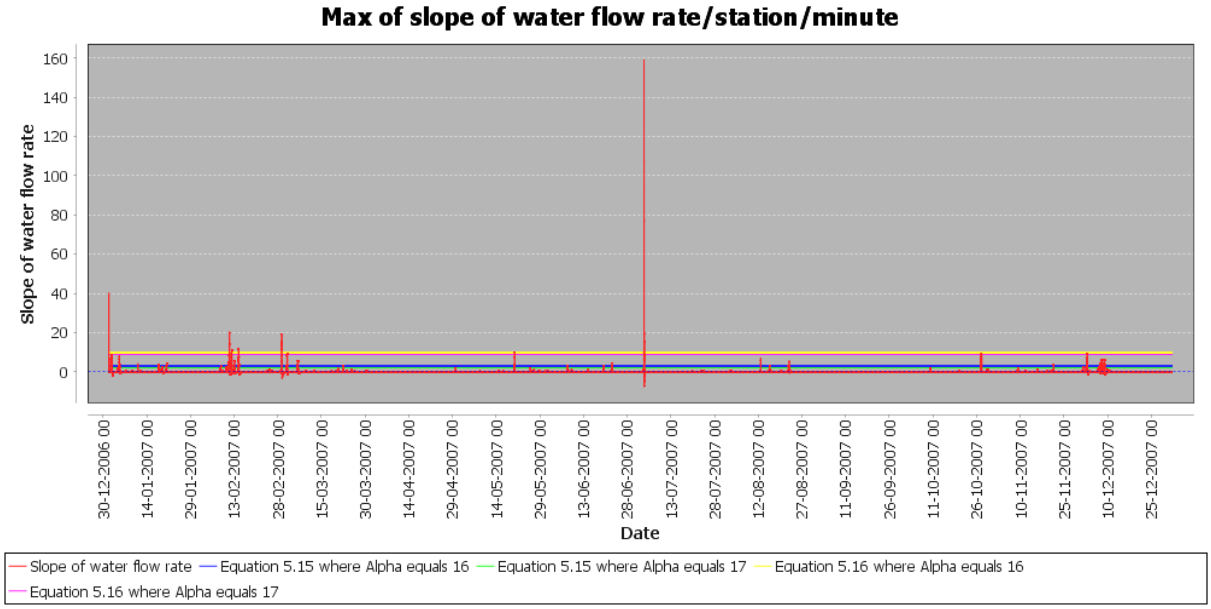


Figure 5.4.3: Maximum and minimum $Watercourse_{watershed}$ using 2007 archive of Les Avenelles and Melarchez watercourse nodes

Figure 5.4.3 shows 4 curves:

- The purple line is the threshold computed with equation 5.15 where $\alpha = 16$. It detects 17 risky events.
- The green line is the threshold computed with equation 5.15 where $\alpha = 17$. It detects 33 risky events.

- The yellow line is the threshold computed with equation 5.16 where $\alpha = 16$. It detects 4 risky events.
- The pink line is the threshold computed with equation 5.16 where $\alpha = 17$. It detects 8 risky events.

These 4 thresholds are also projected on watercourse data of the Orgeval Dataset for year 2008 in Figure 5.4.4.

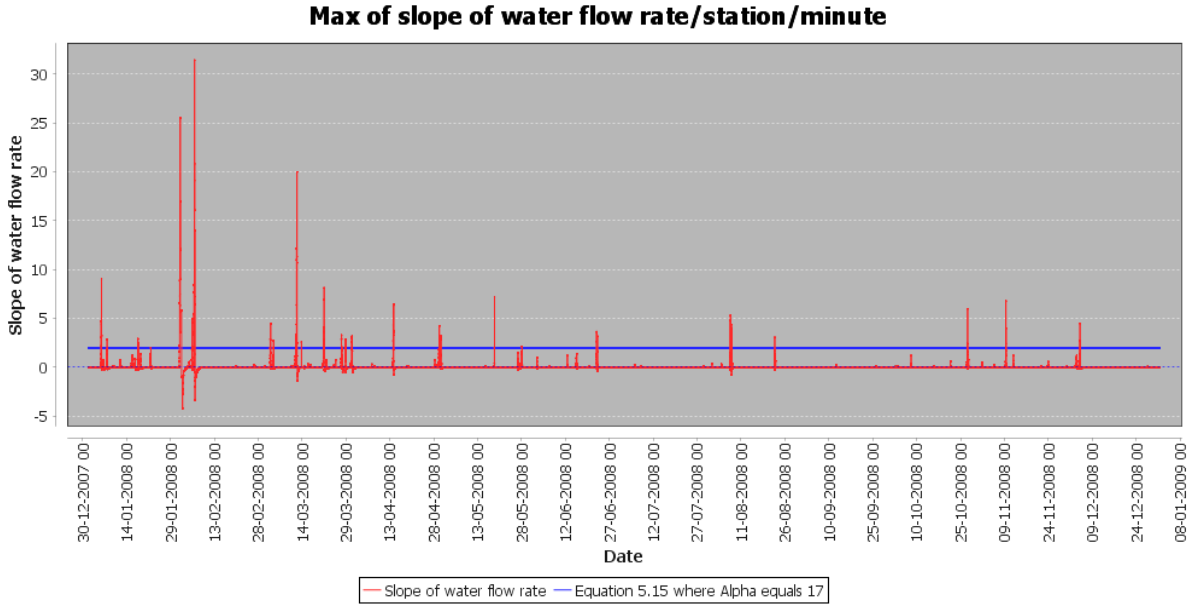


Figure 5.4.4: Maximum and minimum $Watercourse_{watershed}$ using 2008 archive of Les Avenelles and Melarchez watercourse nodes

Figure 5.4.4 only considers 1 straight line:

- The purple line is the threshold computed with equation 5.15 where $\alpha = 17$, it detects 21 risky events

This number is smaller than the one of Precipitation entity risky states (50) detected based on the $Th_{Precipitation}$.

We can see, in Figure 5.4.4, that there are enough Watercourse entity state changes to execute the simulation. $Th_{Watercourse}$ is fixed to "2".

5.4.1.3 Threshold of the Outlet Entity (*O*)

The Outlet entity is observed only by one node. $Outlet_{watershed}$ is the slope between two flow rate values from one minute interval. We compute the maximum ($MaxOutlet_{watershed}$) and the minimum $outlet_{watershed}$($MinOutlet_{watershed}$) values computed during a whole year. The calculation method of Th_{Outlet} is similar to the $Th_{Watercourse}$ as shown in Equation 5.17.

Equation 5.17: The threshold of Outlet in watershed

$$Th_{Outlet} = \frac{MaxOutlet_{watershed} - MinOutlet_{watershed}}{\alpha} + MinOutlet_{watershed}$$

where

α is a variable that will be determined by experimentation.

Equation 5.18 is another equation to compute the Th_{Outlet}.

$$Th_{Outlet} = \frac{MaxOutlet_{watershed}}{\alpha}$$

Figure 5.4.5 shows the evolution of the $Outlet_{watershed}$ values from the 2007 archive of the stream gauge station: le Theil. There are 6 straight lines where $\alpha = 3$, $\alpha = 4$, $\alpha = 5$ from up to down with equation 5.17 and equation 5.18 respectively from 01/01/2007 to 31/12/2007.

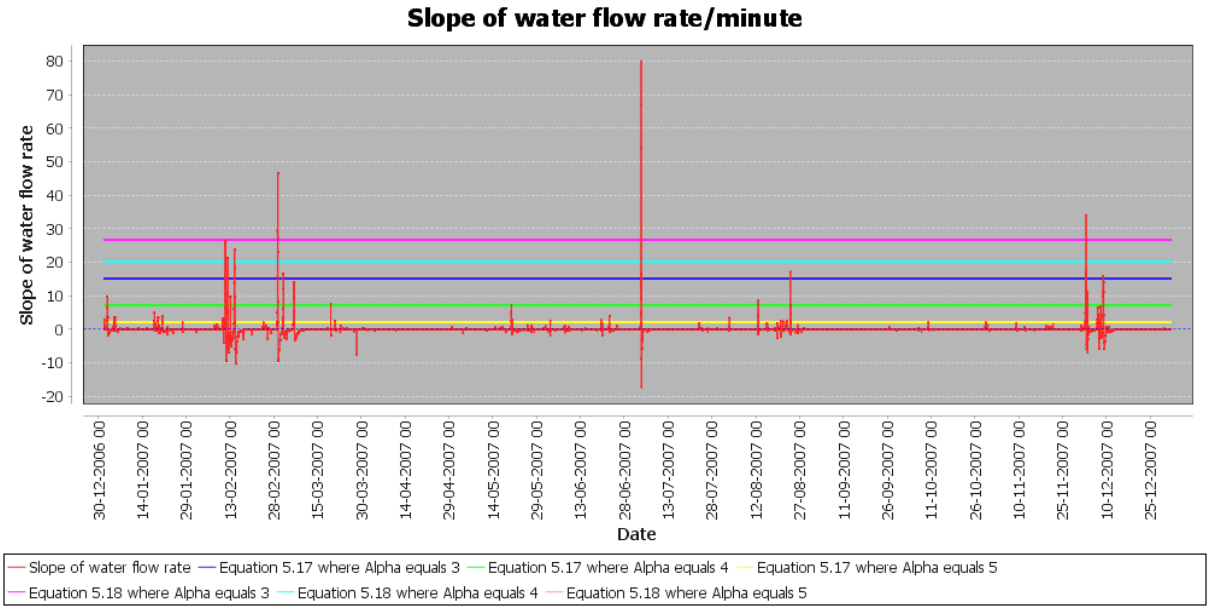


Figure 5.4.5: Maximum and minimum $Outlet_{watershed}$ values using 2007 archive of outlet node (le Theil)

Figure 5.4.5 shows six straight lines:

- The purple line is the threshold computed with equation 5.17 where $\alpha = 3$. It detects 10 risky events.
- The light green line is the threshold computed with equation 5.17 where $\alpha = 4$. It detects 13 risky events.

- The yellow line is the threshold computed with equation 5.17 where $\alpha = 5$. It detects 30 risky events.
- The pink line is the threshold computed with equation 5.18 where $\alpha = 3$. It detects 4 risky events.
- The light blue line is the threshold computed with equation 5.18 where $\alpha = 4$. It detects 6 risky events.
- The light red is the threshold computed with equation 5.18 where $\alpha = 5$. It detects 9 risky events.

These six thresholds are also projected on the outlet data of the Orgeval Dataset for year 2008 in Figure 5.4.5.

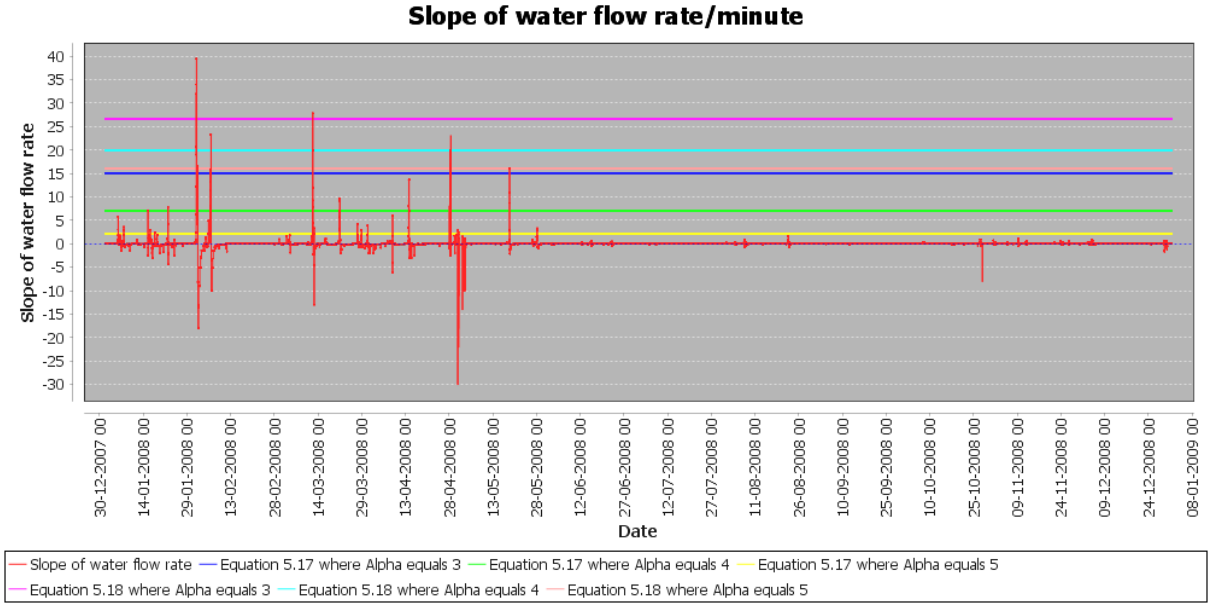


Figure 5.4.6: Maximum and minimum $Outlet_{watershed}$ value using 2008 archive of outlet node (le Theil)

We can see, from Figure 5.4.6, the yellow straight line is the threshold computed with equation 5.17 where $\alpha = 5$, it detects 18 risky events. The value of the threshold Th_{Outlet} is thus fixed to 2.

This number is smaller than the one of the Watercourse entity risky states (21) detected based on the $Th_{Watercourse}$. There are enough Outlet entity state change events to run the simulation.

Table 5.4.1 summarizes the thresholds for the three EOEs: $Th_{Precipitation}$, $Th_{Watercourse}$ and Th_{Outlet}.

Table 5.4.1: Table of EOE thresholds

Observable Entities	Precipitation	Watercourse	Outlet
Threshold	10 millimeters	2	2

5.4.1.4 Threshold of the Node Entity (*N*)

Based on our energy model proposed in section 5.2.2.2, we can draw a figure to show the energy level of a sensor node during one calendar year. Then, we try to define $Th_{NodeEnergy}$ based on this figure. First, we have to provide the parameters of our Energy Consumption Model.

We use a 6600 mA·h rechargeable battery based on the parameters from (Libelium PRO V1.2, 2017). The battery voltage is 3.7 V. Thus, the capacity of this battery is $6.6 \times 3.7 \times 60 \times 60 = 87912\ J$ (Joules). We use the rounding value 87900 J.

For our battery level consumption study during one calendar year, we start with a full battery (87900 J). The communication frequency equals to the sample ones: one measurement per minute. As mentioned in section 5.2.3, the value of peak solar radiation varies according to the different day weather: rainy or sunny days. Based on the parameters of the solar panel presented in section we use precipitation archive file of "melarchez-P35_RainGauge" weather station during year 2008 to draw the figure of energy level of the "melarchez-P35_RainGauge"sensor node as shown in Figure 5.4.7.

Before this simulation, first we want to fix the threshold of day weather based on Figure 5.4.1 during the calendar year 2007. In order to have enough Precipitation entity state changes to execute the simulation, we fix the value of threshold of day weather to 3 mm. Therefore, we make a program that generates a file to store the rainy days of the calendar year 2008 before the execution of simulation.

When the simulation begins, it first loads the file which stores the rainy days of the calendar year 2008 to judge the timestamp of measurements whether it is a rainy day or not. Thus, we can see in February, 2008 in Figure 5.4.8 at the beginning of February 2008, the energy level of battery is decreased. However, in the middle of February, 2008, the energy level of battery is increased. We know that the value of nbHour(m) is the same in a same month as presented in section 5.2.4. The reason is the energy supplied by solar panel varies according to the existence of rainy days or sunny days.

We fix the threshold for the day weather based on Figure 5.4.1 which concerns year 2007. This threshold is fixed to the value of "3 mm" in order to have enough weather conditions changes for our simulation. When the "rainfall amount per 24h" is higher than this "3 mm"

value, we consider that we are in a rainy day. Then, we make a program to generate a file to store the rainy days of the calendar year 2008 in order to execute our node energy simulation. When the simulation starts, it first loads the file which stores the rainy days of the calendar year 2008 to judge whether the timestamp of the measurements corresponds to a rainy day or not. This simulation also uses the parameters for the solar panel presented in section 5.2.4.2. To draw Figure 5.4.7, we use the precipitation archive file of "melarchez-P35_RainGauge" weather station during year 2008. Thus, this Figure represents the energy level of the "melarchez-P35_RainGauge" sensor node.

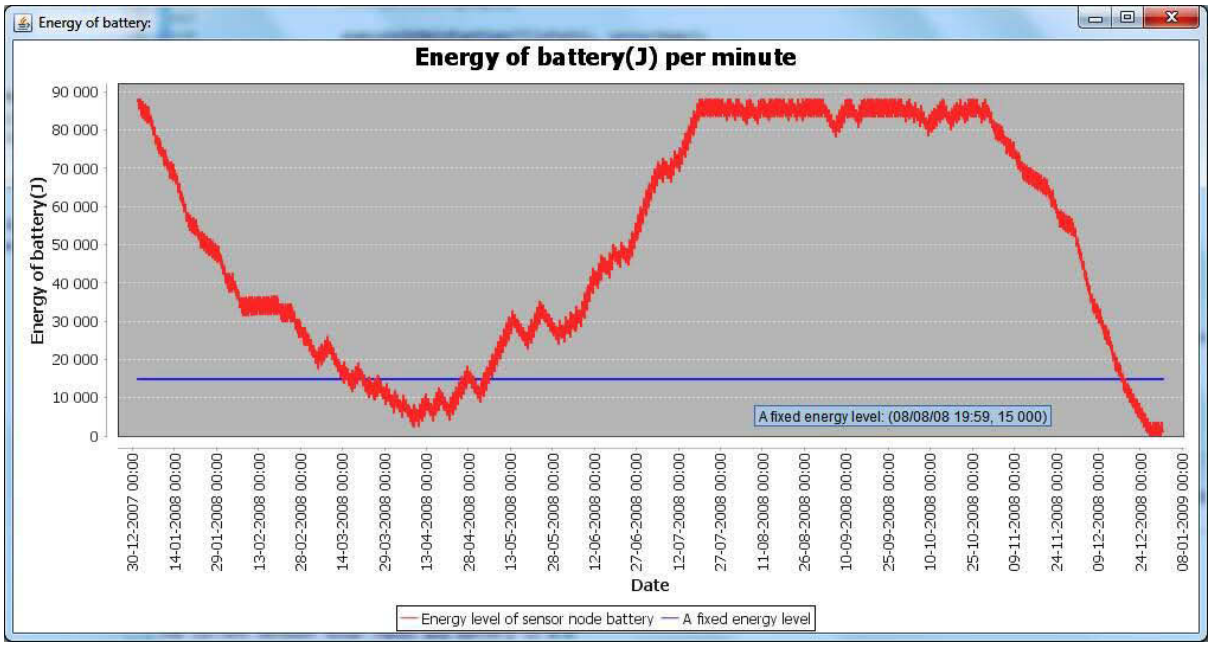


Figure 5.4.7: Energy level of a sensor node battery during the year 2008 (communication frequency equals to the sample one)

In Figure 5.4.7 we can see that the battery level of the sensor node is decreased from the 1st of January 2008 to 13th of February 2008 which is close to the lack of energy. That is because the energy consumption is higher than the energy supplied by the solar panel. Indeed, the peak solar radiation hours are small in these months as shown in Table 5.2.7 of section 5.2.4.2.

But, in the middle of February, 2008, the energy level of battery increases. We know that the value of *nbHour(m)* is the same for a given month as presented in section 5.2.4. So, the reason of this increase is related to the energy supplied by the solar panel according to rainy and sunny days.

The battery level of the sensor node is increasing from the end of April, 2008 to the end of July, 2008. That is because the energy consumption is smaller than the energy supplied by the solar panel because the peak solar radiation hours are big in May, June and July as shown again in Table 5.2.7.

The curve of the energy of battery of the sensor node from August, 2008 to October, 2008 is almost smooth and nearly constant. The reason is that we are near the value of the maximum energy capacity of the battery at the end of July, 2008 and that we slightly fluctuate around

this value from August to October due to a balance between the energy supplied by the solar panel and consumed energy.

The energy of the battery of the sensor node is sharply decreasing from November to December, 2008 due to small peak solar radiation hours during this period of the year. We can see that the energy of the battery of the sensor node is almost "0" at the end of December, 2008.

In order to test the feasibility of our formalization, we are interested in the curve from the beginning of February to the end of April, 2008 because there are a lot of energy state changes. Moreover, there are a big battery energy increase and decrease during this time period. This evolution of the energy of the battery is associated to different Node entity state changes and, potentially, different communication frequency adaptations which are important for our simulations.

Based on these results, we can fix the threshold of Node entity, named $Th_{NodeEnergy}$, to "15000 J" as shown by the blue line in Figure 5.4.7.

Depending on the wireless sensor energy value at a time instant t_i, we can establish the node state:

- If $NodeEnergy(t_i)$ is bigger than or equivalent to $Th_{NodeEnergy}$, the Node entity is in the "Stable" state.
- If $NodeEnergy(t_i)$ is smaller than $Th_{NodeEnergy}$, the Node entity is in the "Critical" state.

5.4.2 Jess Rules

Concerning the reasoning process, we use the Jess rule-based engine (Friedman-Hill *et al.*, 2003). Jess is also implemented in Java. According to our approach, we define two sets of rules.

The first reasoning step infers the state of the observable entity. The DSS computes the state of the observable entity (e.g. Node entity) by comparing the aggregated value of sensor measurements (e.g., *NodeEnergy*) with a given threshold (e.g., $Th_{NodeEnergy}$) as presented in Figure 5.4.8. The rule, presented in this figure, deduces the state "Stable (State A)" of the SNOE, namely the Node entity, from the aggregated value of sensor measurements *NodeEnergy*.

```
(defrule stateNodeEnergy_Stable

(declare (salience 20))

?n <- (node {energyValue >= ?*thresholdNodeEnergy*})

=> (modify ?n (state stable))

)
```

Figure 5.4.8: Rule deducing the state "Stable" of the node entity

The second set is dedicated to infer the state of the entity of interest from the states of observable entities. For example, the rule, presented in Figure 5.4.9, deduces the state "Risky (F3)" of the EEI, namely the Flood entity state, from the states of the defined EOEs (e.g., Precipitation, Watercourse and Outlet).

```
(defrule floodState_RiskF3

(declare (salience 10))

?p <- (precipitation {state     == normal})

?w <- (watercourse {state== risk})

?o <- (outlet {state     == normal})

?f <- (flood)

=> ( modify ?f (state f3))

)
```

Figure 5.4.9: Rule deducing the state "Risk" of the flood entity

5.5 Context Distribution

In our simulation, the context distribution is implemented as a task executed in an agent. A behavior represents this kind of task that an agent can carry out. A behavior is implemented as an instance of a class that extends the *jade.core.behaviours.Behaviour* interface. There are two kinds of behavior in JADE: primitive behaviors and composite behaviors.

Figure 5.5.1 presents the architecture of the primary behaviors available in JADE. In this figure, the interface of "Behaviour" class is *jade.core.behaviours.Behaviour*. There are two

methods: “addBehaviour” and “removeBehaviour”. Most of the time, the behaviors are added manually and disappeared automatically when their jobs are done. However, it is also possible to remove them explicitly.

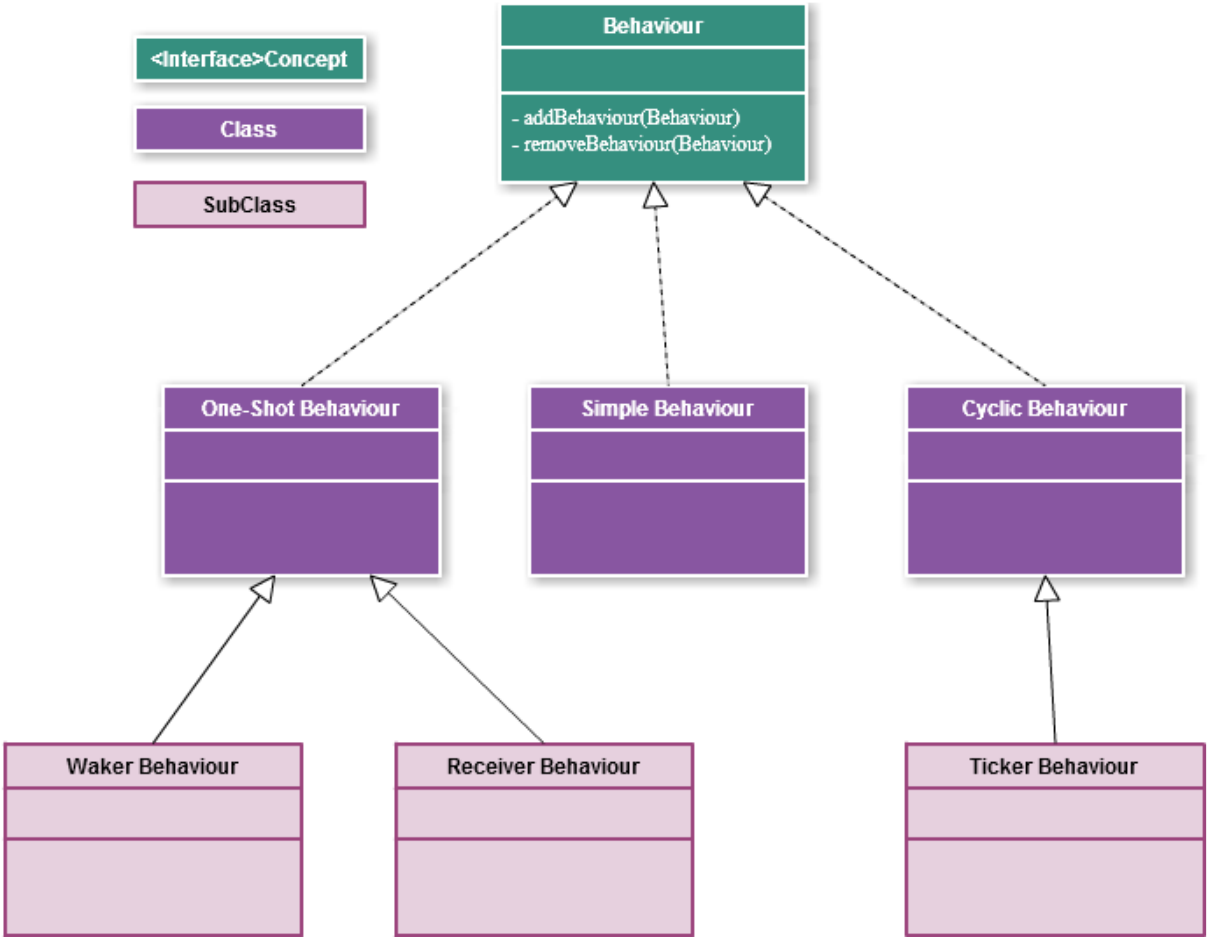


Figure 5.5.1: Architecture of the primary behaviors

Different classes implements the Behaviour interface. They are presented as follows:

- “CyclicBehaviour” is a class that stays active as long as its agent is alive and will be called repeatedly after every event. It is essential to handle message reception.
- “One-Shot Behaviour” is a class that executes only once. This class has 2 subclasses:
  - “Waker Behaviour” is one kind of “One-Shot Behaviour” that executes some user code once at a specified time.
  - “Receiver Behaviour” is one kind of “One-Shot Behaviour” that is triggered when a given type of message is received (or a timeout expires).

An agent can execute several behaviors in parallel or in sequence. These behaviors are the composite behaviors:

- “ParallelBehaviour” is a behavior that controls a set of children behaviors that execute in parallel.

- "SequentialBehaviour" is a behavior that executes its children behaviors one after the other and terminates when the last child has ended.

In our simulation, the execution of a "SequentialBehaviour" for context distribution is presented in Figure 5.5.2. This "SequentialBehaviour" relies on a "W akerBehavior" embedded in a "CyclicBehaviour". This "WakerBehaviour" executes context distribution (sending message) when a cer tain time interval is expired during the time cycle of a "CyclicBehaviour".

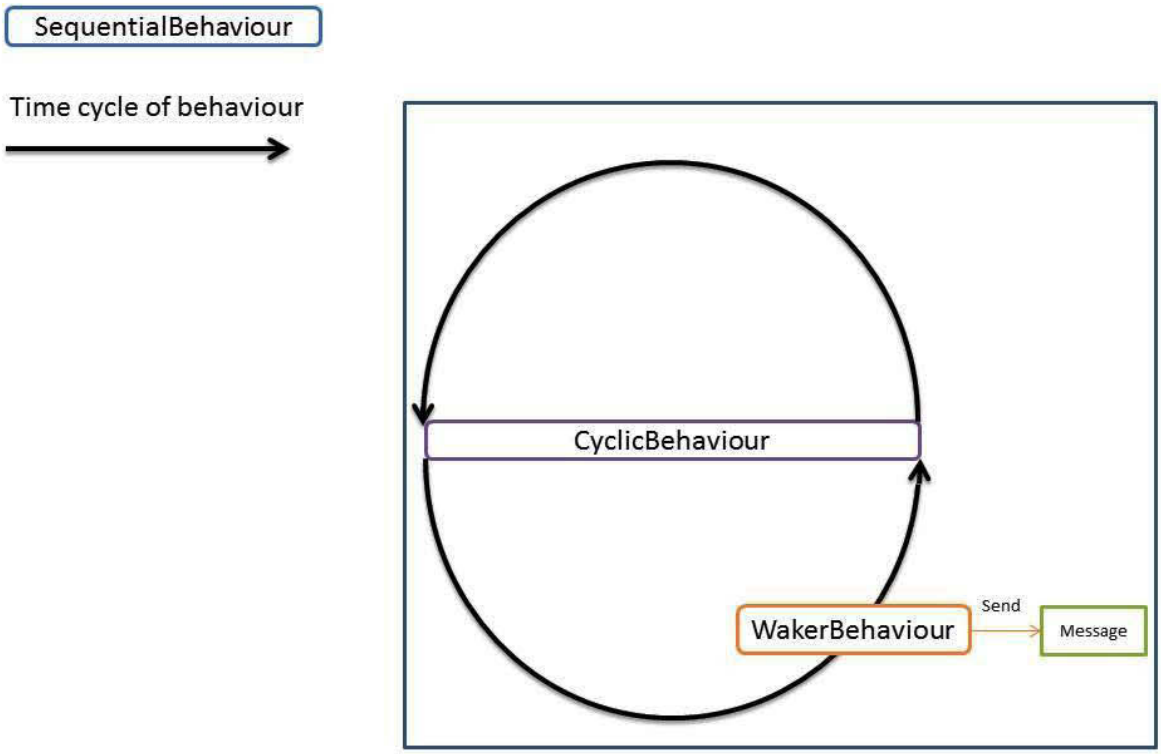


Figure 5.5.2: The execution of a "SequentialBehaviour" dedicated to the context distribution

In our simulation, there is also a synchronization problem to consider. There are one agent representing the DSS and six agents representing the sensor nodes. We create a n ode agent through inputting configuration parameters (e.g., agent's names or the directory of the simulation data file of each node agent) on the agent platform when the simulation runs. We create the DSS agent after creating all the six node agents. When one node agent is created, it searches for the DSS agent every fixed time duration of milliseconds. The agents start to send their measurements only when they find the DSS. However, there is no share timer in JADE, and also, there are time differences when we create each node agent by sequential order. Therefore, there are time differences for each agent to send packets when they find the DSS.

The time differences become bigger when the simulation runs during a long time period, especially the agents send their packets at different communication frequencies. So, sometimes, it happens that the DSS does not receive some packets in time from some agents. In order to avoid this kind of synchronization problem, we did a lot of experimentations:

- In our simulation, 1000 milliseconds represents 1 minute in reality. We increase the duration of milliseconds into 2000 or 3000 to represent 1 minute in reality. Thus, for each simulation cycle, the node agents have more time to reach the DSS. The main drawback is that the simulation lasts longer.
- We increase the duration of the receiving in the DSS in order not to miss the delayed packets.
- DSS checks the timestamp of each packet in order not to mix in the computation packets from different simulation cycles.

Some experimentation failed because the DSS has a limitation time to receive or judge the delayed packets. Some packets are delayed beyond the limitation time when the simulation executes for a long time. At the end, the proposed solution for this synchronization problem is:

- A "CyclicBehaviour" implemented in the DSS. This "CyclicBehaviour" will distribute a flag to each node per simulation cycle time (e.g., one minute). The node will acquire measurements of EOEs and execute others processes when they receive the flag.

The synchronization problem may not exist in some other simulation tools, such as NS-3 or OMNET++, dedicated to WSN. The reason of this problem may be due to the functionalities of JADE or the way we use JADE. In future works, it will be important to consider this problem adequately in a comparison with other simulators.

5.6 Context Adaptation

As mentioned in section 4.4, the adaptation of the wireless sensor nodes involves communication frequencies modulations. In our simulation, the sample frequency of the sensor nodes is fixed at one measurement per minute as indicated in section 5.2. At the beginning, the communication frequency is also set to one transmission per minute. Depending on the different types of context-aware system, the communication frequency will evolve.

5.6.1 Communication Frequencies of the Flood Context-aware System

The communication frequency of the Flood context-aware system depends on the state of the *F* entity. Table 5.6.1 presents the value of the communication frequency based on the F entity state. For this system, all the nodes will have the same communication frequency that may change over time. When the Flood entity reaches the "Risky (*F3*)" or "Flood (*F4*)" state, all the nodes will have to communicate with the maximum frequency which is equal to the sample frequency (one transmission per minute).

Table 5.6.1: Communication frequencies based on the flood entity states

F entity State	Normal (*F1*)	Rainy (*F2*)	Risky (*F3*)	Flood (*F4*)
Communication Frequency	Sample Frequency	Sample Frequency	Sample Frequency	Sample Frequency

5.6.2 Communication Frequencies of the Node Context-aware System

The communication frequency of the Node context-aware system depends on the state of the *N* entity. Table 5.6.2 presents the value of the communication frequency based on the *N* entity state. In this case, each node can have, at a given time, a different communication frequency. In addition, this communication frequency may change over time. When the *N* entity reaches the "Stable (*NA*)" state, the node will have to communicate with the maximum frequency which is equal to the sample frequency (one transmission per minute). When the *N* entity reaches the "Critical (*NB*)" state, the node will reduce the communication frequency tohalf sample frequency.

Table 5.6.2: Communication frequencies based on the node entity states

N entity State	Stable (*NA*)	Critical (*NB*)
Communication Frequency	Sample Frequency	Sample Frequency

Thus, when the energy level of the sensor node is smaller than $Th_{NodeEnergy}$, the communication frequency is set to half sample frequency as shown in Table 5.6.2. T he communication frequency equals the sample frequency when the energy level of the sensor node is bigger than $Th_{NodeEnergy}$. This is the adaption part.

We present, in Figure 5.6.1, the energy level of the sensor node simulated during the year 2008 when the node adapts its frequency according to its node entity state.

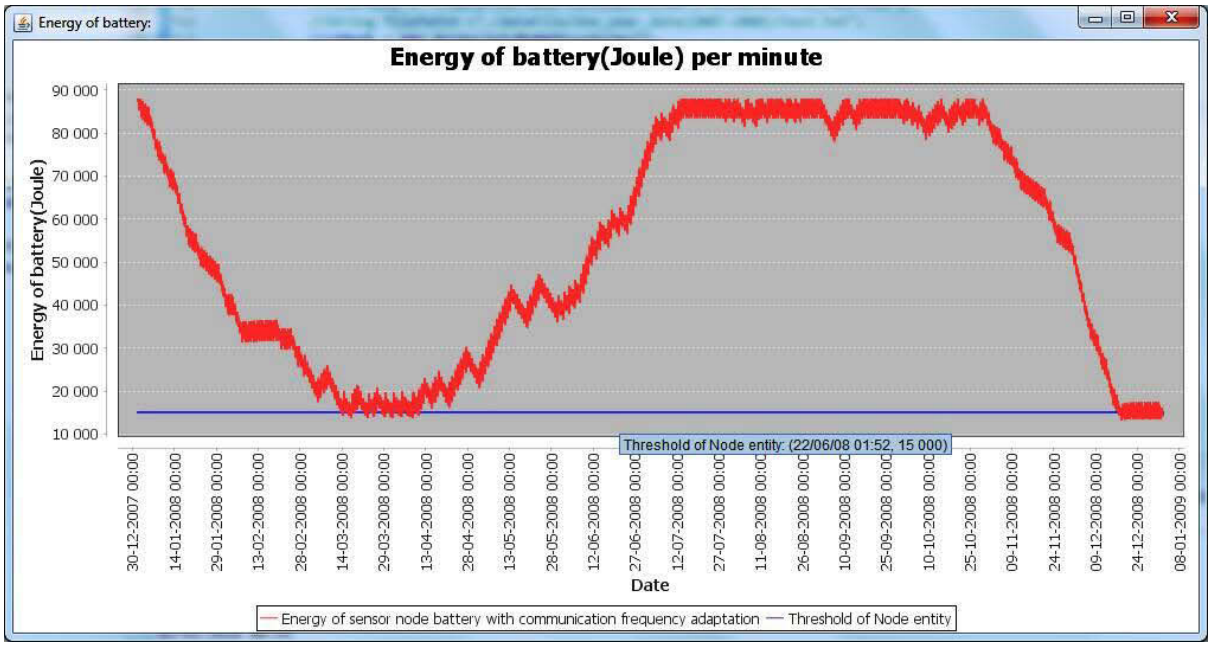


Figure 5.6.1: Energy level of a sensor node with communication frequency adaptation during the year 2008

In Figure 5.6.1, the curve seems similar to the one of Figure 5.4.7. However, we can see two differences during the time interval [March of 2008, A pril of 2008] and at the end of December of 2008. T he energy values during this two periods are equal to $Th_{NodeEnergy}$ as drawn in purple line in Figure 5.6.1. This is due to the adaptation part. When the energy level of the sensor node is smaller than $Th_{NodeEnergy}$, the communication frequency of the sensor node will change to once per 2 m inutes. Comparing the two cited figures, we can see that there is almost 15000 J energy saved at the end of year 2008 on Figure 5.6.1.

The Figure 5.6.2 has two curves and one straight line. This line in green represents the value of $Th_{NodeEnergy}$. The blue curve represents the energy level of the sensor node without communication frequency adaptation. That means there is no c ontext adaptation process in this sensor node. We can see that this sensor node cannot send all the messages especially during the nights and the mornings during the interval [January of 2009, March of 2009]. It is due to the fact that the energy consumption is equal or higher than the energy supplied by the solar panel. The energy level of this sensor node is increasing starting from March, 2009 because of a big increase of the peak radiation hours. The energy of the battery of the sensor node is almost reduced to "0" at the end of December, 2009. T he red curve represents the energy level of the sensor node with communication frequency adaptation. That means there is a context adaptation process. We can see that this sensor node can work more than two years. In comparison with the blue curve, the energy level of the sensor node is almost 15000 J at the end of year 2009.

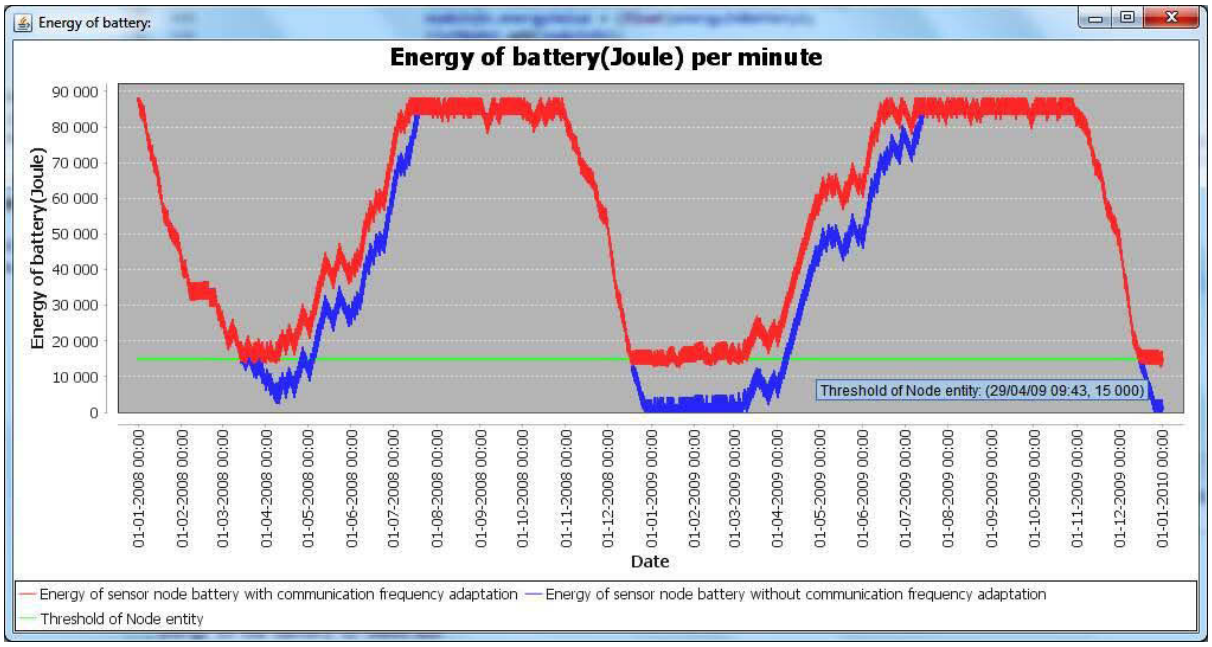


Figure 5.6.2: Comparison of the energy level of a sensor node, with and without communication frequency adaptation, for the years 2008 and 2009

5.6.3 Communication Frequencies of the Flood and Node Context-aware System

The communication frequencies of the Flood and Node context-aware system is based on the *F* entity states and the *N* entity states. Table 5.6.3 presents the value of the communication frequencies based on the states of these two entities. Each node can have different communication frequencies that may change over time depending of their energy level at the beginning of the simulation. When the Flood entity reaches the "Risky (*F3*)" or "Flood (*F4*)" state, all the nodes will have to communicate with the maximum frequency which is equal to the sample frequency (one transmission per minute) without considering its "own" *N* entity state. This is due to the fact that the priority of the flood context is higher than the one of the WSN as we presented in section 4.4.

Table 5.6.3: Communication frequencies based on the flood and the node entity states

State	Normal (*F1*)	Rainy (*F2*)	Risky (*F3*)	Flood (*F4*)
Stable (*NA*)	Sample Frequency	Sample Frequency	Sample Frequency	Sample Frequency
Critical (*NB*)	Sample Frequency	Sample Frequency	Sample Frequency	Sample Frequency

6 Simulations

The purpose of our experimentations is to test the feasibility of our formalization by simulating different context-aware systems. Moreover, we want to demonstrate that the communication frequency adaptations of the sensor nodes will improve the lifetime of any context-aware system according to the objective function of the application. We expect that our adaptation approach will have limited impact on the Quality of Service (QoS) of the system. We propose several context-aware systems with different adaptations based on entities: Flood entity or/and Node entities.

6.1 Evaluation Protocol

To simulate context-aware systems, we implement three scenarios. All these scenarios use the same WSN:

- Three agents represent three "Precipitation nodes",
- Two agents represent two "Watercourse nodes",
- One agent represents one "Outlet node",
- One agent represents the DSS.

To realize the context acquisition, these agents use the archive of the Orgeval basin dataset. It is to be highlighted that the energy measurements are provided by our energy model (c.f. Section 5.2.2). We limit our simulation to one month: February 2008. Each agent has different parameters. The sample frequency of all the nodes is fixed to one measurement per minute. The communication frequency will depend on the adaptation. It is presented in the Tables 5.6.1 to 5.6.3 provided in section 5.6.1. The system configuration specifies the aggregation function applied on the acquired measurements in order to compute the value sent to the DSS (see Table 6.1.1).

Table 6.1.1: The different sensor nodes configurations used in our simulation

	"Precipitation nodes" Configuration	"Watercourse nodes" Aggregation function	"Watercourse nodes" Configuration	DSS Aggregation function for "Watercourse nodes"	"Outlet node" Configuration
Configuration 4	Rainfall amount per 24h		Last water flow rate	Max Function	Last water flow rate
Configuration 10	Rainfall amount per	Max Function	Max slope of water flow	Max Function	Max slope of water flow

	24h		rate per communicatio n interval		rate per communicati on interval
Configuration 12	Rainfall amount per 24h	Max Function	Max slope of water flow rate per 6h	Max Function	Max slope of water flow rate per 6h

The three considered scenarios are:

- **Scenario 1:** defines the baseline of the flood context-aware system. This scenario has no adaptation as shown in Figure 4.2.4. The sample and communication frequencies are the same. Thus, in this baseline, the DSS computes the *F* entity state using more value than other scenarios. Scenario 1 is the baseline for the *F* entity state and the QoS.
- **Scenario 2:** is an adaptive flood context-aware system as shown in Figure 6.1.1. Compared with scenario 1, scenario 2 proposes context adaptation based on the *F* entity state. The adaptation modifies communication frequencies. All the nodes will have the same frequency at the same time.

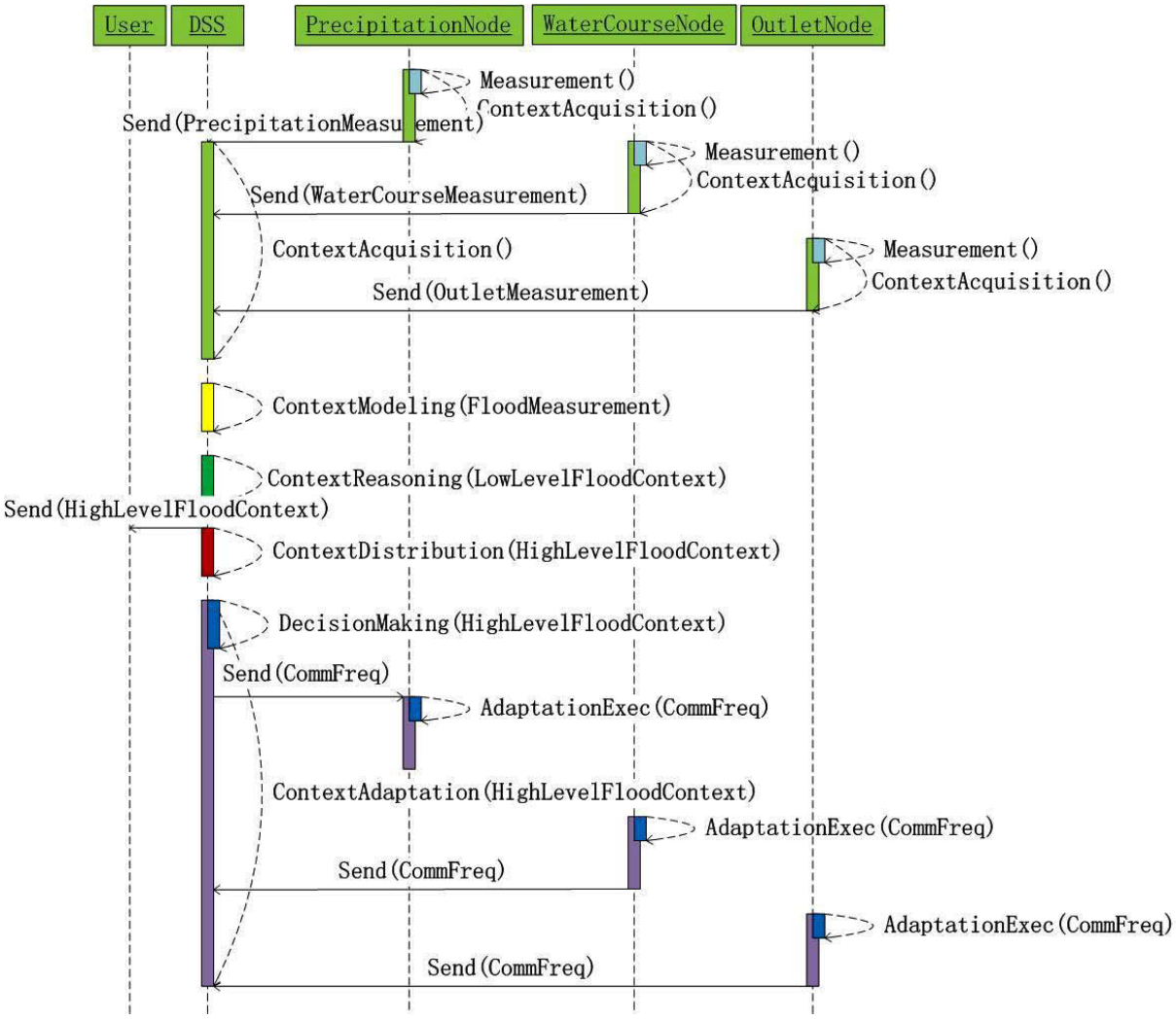


Figure 6.1.1 Simulation architecture sequence diagram of an adaptive flood context-aware system

- **Scenario 3:** is an adaptive context-aware system as shown in Figure 4.4.1. Compared with scenario 2, the adaptation is based on the *F* entity state and on the state of the node itself. The node state depends on its energy level. Our proposed energy model is only used in this scenario. Notice that the adaptation is presented in Table 5.6.3 in section 5.6.1. For our experimentation, the nodes of same type will have different initial energy levels. Each node will have a different communication frequency. Thus, we can study the node behaviors.

Table 6.1.2: Nodes initial energy level in scenario 3

Nodes	Energy level (J)	N entity state
melarchez-p35_RainGauge	20000	Stable
boissy-meteo_RainGauge	20000	Stable
boissy-p28_RainGauge	14950	Critical
lesAvenelles_StreamGauge	20000	Stable
melarchez_StreamGauge	14950	Critical
leTheill_StreamGauge	20000	Stable

Table 6.1.2 presents the initial energy level of the nodes in scenario 3. The last column of the table presents the initial state of the nodes based on its energy level. For our simulation, we have to configure some nodes with a "Stable" state and some others with a "Critical" state. We want also that some nodes change their state during the simulation. Thus, we choose, for some of them, an initial energy level close to the $Th_{NodeEnergy}$ threshold. This threshold is set to 15000 J which is approximately 17% of the battery capacity considered for our experimentation. Thus, we set:

- Precipitation nodes "melarchez-p35_RainGauge" and "boissy-meteo_RainGauge", Watercourse node "lesAvenelles_StreamGauge" and Outlet Node "leTheill_StreamGauge" have the same initial energy level: 20000 J, which is approximately equal to 23% of their battery capacity. The initial states of these nodes are "Stable".
- Precipitation node "boissy-p28_RainGauge" and Watercourse node "melarchez_StreamGauge" have the same initial energy level: 14950 J, which is approximately equal to 17% of their battery capacity. The initial states of these two nodes are "Critical".

The different metrics considered for our evaluation are presented in the following section.

6.2 Metrics

To compare these scenarios, different metrics have to be defined. First, metrics should help to identify the WSN with the longest lifetime. In our simulation, we decide to focus on energy consumption. Without the availability of a renewable energy (e.g., use of a solar panel), the energy consumption gives indication about the lifetime of the network. Therefore, our first metric has to establish the different energy consumptions of the whole network between the different scenarios. To study the energy consumption impact of our adaptation approach, a second set of metrics will estimate the QoS of a flood management system.

6.2.1 Metrics to Evaluate the Energy Consumption for Each Scenario

In most wireless applications, communication is the activity that consumes the most energy (Anastasi *et al.*, 2006). The energy consumed by communication is proportional to the number of transmitted and received bit. We assume that each packet has the same size, using padding if needed. Thus, we select, as a metric, the total amount of exchanged messages between all the nodes and the DSS.

6.2.2 Metrics to Evaluate the QoS

Based on our formalization of the context, we expect that the communication frequency adaptation will reduce the data exchange inside the network. However, this kind of dynamic adaptations should not be at the expense of the quality of the final application decision, called QoS of the application. Thus, studying potential impact on the QoS is also important for our adaptation approach. As presented in sections 4.3 and 4.4, the flood context has more priority than the node one. Therefore, the QoS at the level of the Flood entity is the most important.

In order to evaluate the QoS, we compare the scenarios in terms of the following two metrics:

- Total number of each entity state changes (Flood entity and Node entity).
- Timestamps of each entity state changes (Flood entity and Node entity).

Based on the timestamp metric, we also compute the total delay in the detection of state changes between different scenarios.

Considering the Node entity, these two metrics will only be studied in scenario 3. However, even in this scenario, the most important entity is still the Flood entity.

6.3 Baseline Systems Evaluation

Table 4.2.6 of section 4.2.3.4 shows all the possible configurations for the sensor nodes at the level of measurements computation. For scenario 1 of our experimentation, we implement configurations 4, 10 and 12 as shown in Table 6.1.1. This three baseline systems will be described and evaluated in the current section.

6.3.1 Baselines Specification

In scenario 1, sensor nodes send their aggregated value every minute. First, we focus on the two following systems: configurations 4 and 10. Precipitation nodes always send "**rainfall amount per 24h**" in these two systems. However, Watercourse nodes and Outlet node send "**last water flow rate**" to the DSS in configuration 4 and "**max slope of water flow rate per communication interval**" in configuration 10.

In configuration 4, the DSS computes the slopes of the last water flow rate per minute for all watercourse nodes. Then, the DSS computes the maximum slope for this Watercourse nodes

group. Then, this maximum slope is compared with the $Th_{watercourse}$ threshold one in order to establish the state of the Watercourse entity.

However, in configuration 10, the watercourse nodes send "**max slope of water flow rate per communication interval**". For scenario 1, the communication interval is one minute, which is equal to the acquisition period. Thus, the DSS receives the max slopes of water flow rate per minute from the watercourse nodes. For each node, this slope is exactly the same than the slope sent to the DSS in configuration 4. Then, the DSS also computes the maximum slope between the different Watercourse nodes to compare with the $Th_{watercourse}$ threshold in order to compute the state of the Watercourse entity.

This reasoning can be applied in the same way to the outlet node as there is only one outlet node in each system. Thus, configurations 4 and 10 will have the same result for scenario 1.

Now, considering configuration 12, precipitation nodes also send "rainfall amount per 24h" values to the DSS. However, Watercourse and Outlet nodes send their "max slope of water flow rate per 6h" to the DSS, which is an historical value of the slope as presented in section 4.2.3.2.

As, for the scenario 1, configurations 4 and 10 are the same, in the next section, we will focus on configurations 4 and 12 as two possible baseline systems.

6.3.2 Evaluation of the Baseline Systems

In this section, we present, in detailed, the evaluation of two baseline systems, scenario 1 configurations 4 and 12, in terms of the following metrics:

- Total amount of exchanged communication packets.
- Number of each Flood entity state changes.
- Timestamps of each Flood entity state changes.

6.3.2.1 Total Amount of Exchanged Communication Packets

Table 6.3.1 shows, for the configurations 4 and 12 of the scenario 1, the total number of messages exchanged using data of February, 2008 from the French Orgeval watershed.

Table 6.3.1: Total number of exchanged packets in scenario 1 configurations 4 and 12

	Scenario 1 Configuration 4	Scenario 1 Configuration 12
01/02/2008 00:00 to 01/03/2008 00:00	250,560	250,560

In scenario 1, the communication frequency is equal to the sample frequency: there is no adaptation. The consequence is that the total number of exchanged communication packets for scenario 1 configurations 4 and 12 are the same and equal to 250,560 packets.

6.3.2.2 Flood Entity State Changes

As shown in Figure 6.3.1, for the scenario 1, configurations 4 and 12 are different at the level of the F entity state changes.

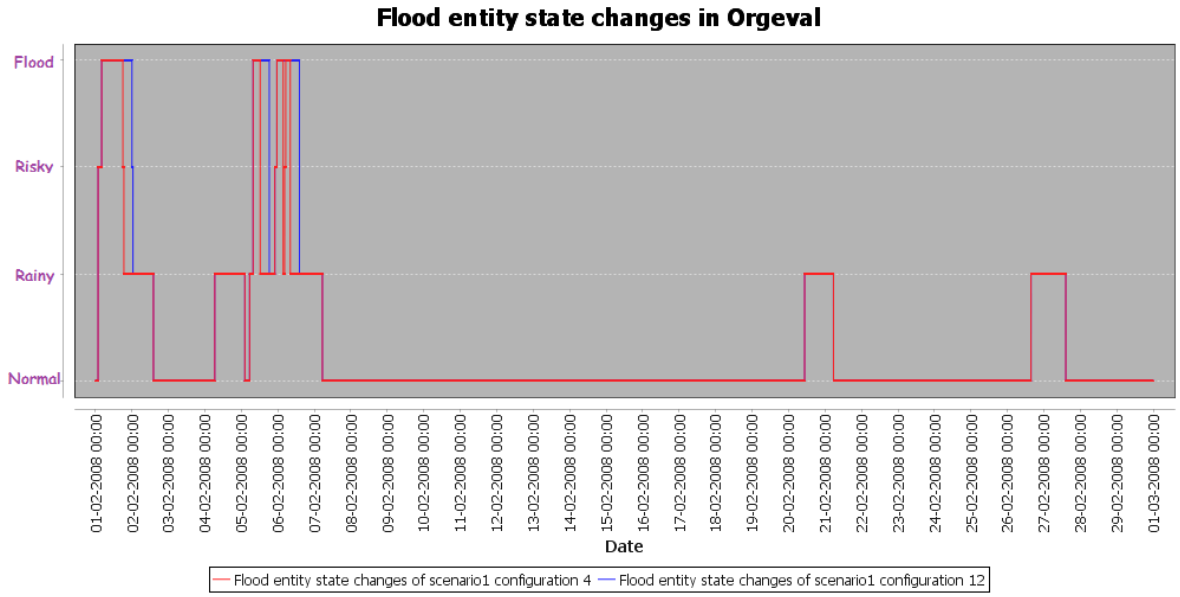


Figure 6.3.1: Flood entity states evolution for scenario 1 configurations 4 and 12

Figure 6.3.1 introduces some differences in the behavior for these two baseline systems. The main things to notice in this Figure 6.3.1 are:

- In the scenario 1 configuration 4, which is presented in red curve, the system detects more Flood entity state changes especially between time intervals: [05/02/2008, 06/02/2008] and [06/02/2008, 07/02/2008].
- In the scenario 1 configuration 12, which is presented in green curve, the system detects less Flood entity state changes compared with the configuration 4. However, the Flood entity state "Flood (F4)" stays longer in configuration 12 than in configuration 4 during for the following time intervals: [02/02/2008, 03/02/2008], [05/02/2008, 06/02/2008] and [06/02/2008, 07/02/2008].

Now, we will consider the total number of each Flood entity state changes.

6.3.2.2.1 Total Number of Flood Entity State Changes

From Figure 6.3.1, we can see differences between the two curves associated to configurations 4 and 12 of scenario 1. For the *F* entity state, a "Flood (*F4*)" state appears in the 1st, 5th and 6th of February, 2008.

For example, in the 6th of February, 2008, the curves of the two systems are very different. In the scenario 1 configuration 4, there are more entity state changes than in scenario 1 configuration 12. At the end of the 6th of February, 2008, the two systems reach the same state: Flood (*F4*).

Table 6.3.2 presents the number of each Flood entity state changes for the baseline systems, scenario 1 configurations 4 and 12.

Table 6.3.2: Number of each flood entity state changes of scenario 1 configurations 4 and 12 for February, 2008

	N>Ra	*N>Ri*	*Ra>Ri*	*Ra>F*	*Ri>F*	*F> Ri*	*F>Ra*	*Ri>Ra*	*Ra>N*
Scenario 1 Configuration 4	4	1	**2**	1	**3**	1	**3**	1	5
Scenario 1 Configuration 12	4	1	**1**	1	**2**	1	**2**	1	5

For scenario 1, the number of state changes in configuration 4 is bigger than the one in configuration 12 for the following transitions: "Rainy to Risky (*Ra* > *Ri*)", "Risky to Flood (*Ri* > *F*)" and "Flood to Rainy (F > Ra)".

The total number of Flood entity state changes in scenario 1 configuration 4 is 21. The total number of Flood entity state changes in scenario 1 configuration 12 is 18.

From this evaluation, we can see that scenario 1 configuration 4 is more reactive in the detection of *F* entity state changes. Overall, this advantage becomes a drawback when we make transition from the "Flood" state. Indeed, it can generate dangerous situations if we consider too soon that a Flood event is finished. At the opposite, scenario 1 configuration 12 is more stable and, in case of flood event, this is an advantage in terms of security.

6.3.2.2.2 Timestamps of Each Flood Entity State Changes

Table 6.3.3 presents the different timestamps of the state change, each line corresponding to a peak of the Figure 6.3.1. The state changes are presented in chronological order. In scenario 1, compared with configuration 4, configuration 12 has three less state changes. In Table 6.3.3, in order to highlight the differences between these systems, we filled them in bold. For the missing state changes for the scenario 1 configuration 12, we add the element "None" in bold format too.

Table 6.3.3: Timestamps of each flood entity state changes of the baseline systems scenario 1 configurations 4 and 12 for February, 2008

	N>Ra	*N>Ri*	*Ra>Ri*	*Ra>F*	*Ri>F*	*F>Ri*	*F>Ra*	*Ri>Ra*	*Ra>N*
Scenario1 Config 4		01/02 01:41			01/02 04:03	01/02 18:01		01/02 18:41	02/02 14:08
	04/02 06:30								05/02 02:16
	05/02 05:24			05/02 07:41			05/02 12:21		
			05/02 22:01		05/02 23:21		06/02 03:31		
			06/02 04:27		06/02 05:11		06/02 08:11		07/02 05:18
	20/02 10:35								21/02 05:34
	26/02 15:33								27/02 14:25
Scenario1 Config 12		01/02 01:41			01/02 04:03	**02/02 00:01 (+6h)**		**02/02 00:41 (+6h)**	02/02 14:08
	04/02 06:30								05/02 02:16
	05/02 05:24			05/02 07:41			**05/02 18:21 (+6h)**		
			05/02 22:01		05/02 23:21		**06/02 14:11 (+6h)**		
			None		**None**		**None**		07/02 05:18
	20/02 10:35								21/02 05:34

	26/02 15:33								27/02 14:25

From these different evaluations, the three baseline systems considered seem useful for us. One is more reactive than the other which implies advantages but also drawbacks. In the following section, we will consider and evaluate the configurations 4, 10 and 12 in scenario 2, which integrates phenomenon context-aware adaptation.

6.4 Flood Adaptive Context-aware System Evaluation

In this section, we will evaluate the configurations 4, 10 and 12 applied to scenario 2. These systems are all flood adaptive context-aware systems without taking into account the node entity. In scenario 2, the communication frequencies of the sensor nodes are not, all the time, equal to the sample one. Thus, the node will send all the aggregated measurements to the DSS. The aggregation may have an impact on the QoS. The purpose of this section is to study the impact of the adaptation and to select the configuration whose impact is the most limited on the QoS.

6.4.1 Comparison Between Scenarios 1 and 2 for the Configuration 4

First, we focus on the two following systems: scenarios 1 and 2 configuration 4. More precisely, this section evaluates scenario 2 configuration 4 using the baseline scenario 1 configuration 4.

6.4.1.1 Total Amount of Exchanged Communication Packets

Table 6.4.1 shows the total number of messages exchanged using data of February, 2008. Therefore, the total number of exchanged communication packets of scenario 2 configuration 4 is smaller than the one of scenario 1 for the same configuration. For scenario 2, the total of exchanged communication packets equals to 97,824 packets.

Table 6.4.1: Total number of exchanged communication packets in scenarios 1 and 2 for configuration 4

	Scenario 1 Configuration 4	Scenario 2 Configuration 4
01/02/2008 00:00 to 01/03/2008 00:00	250,560	97,824

6.4.1.2 Flood Entity State Changes

Figure 6.4.1 presents F entity state changes for scenarios 1 and 2 using configuration 4. We will first study the different Flood entity state changes and their number of occurrences.

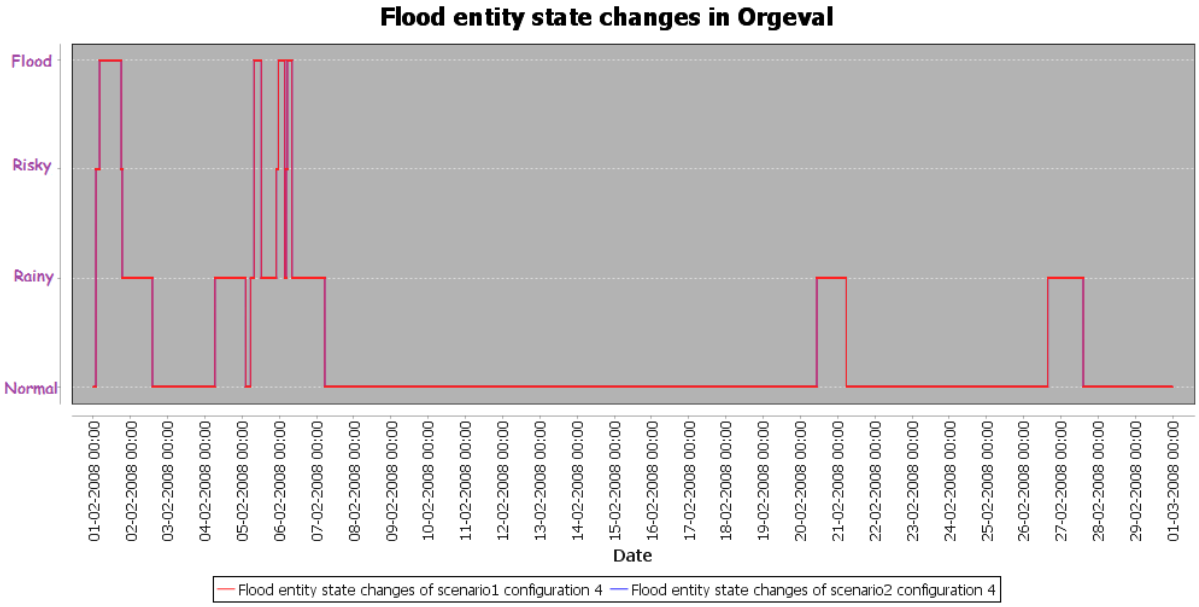


Figure 6.4.1: Flood entity state changes in scenarios 1 and scenario 2 using configuration 4 for February, 2008

From Figure 6.4.1, we can see that the two curves associated to each system are very close. They seem similar because the time scale is the day. We will therefore study the number of Flood entity state changes in the next section in order to confirm or invalidate this impression.

6.4.1.2.1 Total Number of Flood Entity State Changes

As shown on Table 6.4.2, the number of each Flood entity state changes in February, 2008 is the same for the two systems. The total number of Flood entity state changes for these two systems is 21.

Table 6.4.2: Number of each flood entity state changes in scenarios 1 and 2 using configuration 4 for February, 2008

	N>Ra	*N> Ri*	*Ra> Ri*	*Ra>F*	*Ri>F*	*F>Ri*	*F>Ra*	*Ri>Ra*	*Ra>N*
Scenario 1 Config 4	4	1	2	1	3	1	3	1	5
Scenario 2 Config 4	4	1	2	1	3	1	3	1	5

However, if the time scale is increased to the minute, some timestamp differences appear as shown in Table 6.4.3. This Table provides the timestamps of each Flood entity state changes in February, 2008. Compared with scenario 1 configuration 4, there are some delays for the state changes in scenario 2 configuration 4. This later has 9 delayed state changes. In Table 6.4.3, the delayed timestamps are presented in bold format. In brackets, the delay duration is specified.

Table 6.4.3: Timestamps of each flood entity state changes in scenarios 1 and 2 using configuration 4 for February, 2008

	N>Ra	*N>Ri*	*Ra>Ri*	*Ra>F*	*Ri>F*	*F>Ri*	*F> Ra*	*Ri>Ra*	*Ra>N*
Scenario 1 Config 4		01/02 01:41			01/02 04:03	01/02 18:01		01/02 18:41	02/02 14:08
	04/02 06:30								05/02 02:16
	05/02 05:24			05/02 07:41			05/02 12:21		
			05/02 22:01		05/02 23:21		06/02 03:31		
			06/02 04:27		06/02 05:11		06/02 08:11		07/02 05:18
	20/02 10:35								21/02 05:34
	26/02 15:33								27/02 14:25
Scenario 2 Config 4		**01/02 01:42 (+1m)**			01/02 04:03	01/02 18:01		01/02 18:41	**02/02 14:09 (+1m)**
	04/02 06:30								05/02 02:16
	05/02 05:25 (+1 min)			**05/02 07:43 (+2m)**			05/02 12:21		
			05/02 22:03 (+2m)		05/02 23:21		06/02 03:31		

			06/02 04:27		06/02 05:11		06/02 08:11		**07/02 05:19 (+1m)**
	20/02 10:37 (+2m)								**21/02 05:35 (+1m)**
	26/02 15:35 (+2m)								27/02 14:25

6.4.1.2.2 Timestamps of Each Flood Entity State Changes

In Figure 6.4.2, an example of state changes for scenarios 1 and 2 using configuration 4 is provided illustrating the delay. We take only into account the messages sent by the Watercourse nodes. This example is provided along a chronological order:

- Time instant "05/02 21:59", both scenarios 1 and 2 using configuration 4 deduce that the Flood entity state is "Rain (*F2*)". The communication frequency of scenario 1 configuration 4 is always one communication per minute. Based on Table 5.6.3, the communication frequency of scenario 2 configuration 4 is one communication every two minutes as, in scenario 2, all the nodes stay always in the state "Stable". So, the wireless sensors of scenario 2 configuration 4 will only send their next measurements to the DSS at the time instant "05/02 22:01".
- Time instant "05/02 22:01", in the scenario 1 configuration 4, the values of the water flow rate sent by the nodes at this time instant will generate a slope between [05/02 22:00, 05/02 22:01]. The DSS of scenario 1 configuration 4 detects a state change from "Rain (*F2*)" to "Risky (*F3*)". This change has also to be theoretically detected by the scenario 2 configuration 4. But, in the scenario 2 configuration 4, the values sent by the nodes at this time instant "05/02 22:01" will generate a slope between the time interval [05/02 21:59, 05/02 22:01] which is under the $Th_{Watercourse}$ threshold. The state change from "Rainy" to "Risky" is thus not detected here and the nodes are still sending messages every 2 minutes.
- Time instant "05/02 22:03", in scenario 2 configuration 4, based on the measurements sent by the nodes, the DSS finally switches the Flood entity state from "Rainy (*F2*)" to "Risky (*F3*)". Meanwhile, in scenario 1 configuration 4, the Flood entity stayed in the "Risky (*F3*)" state.

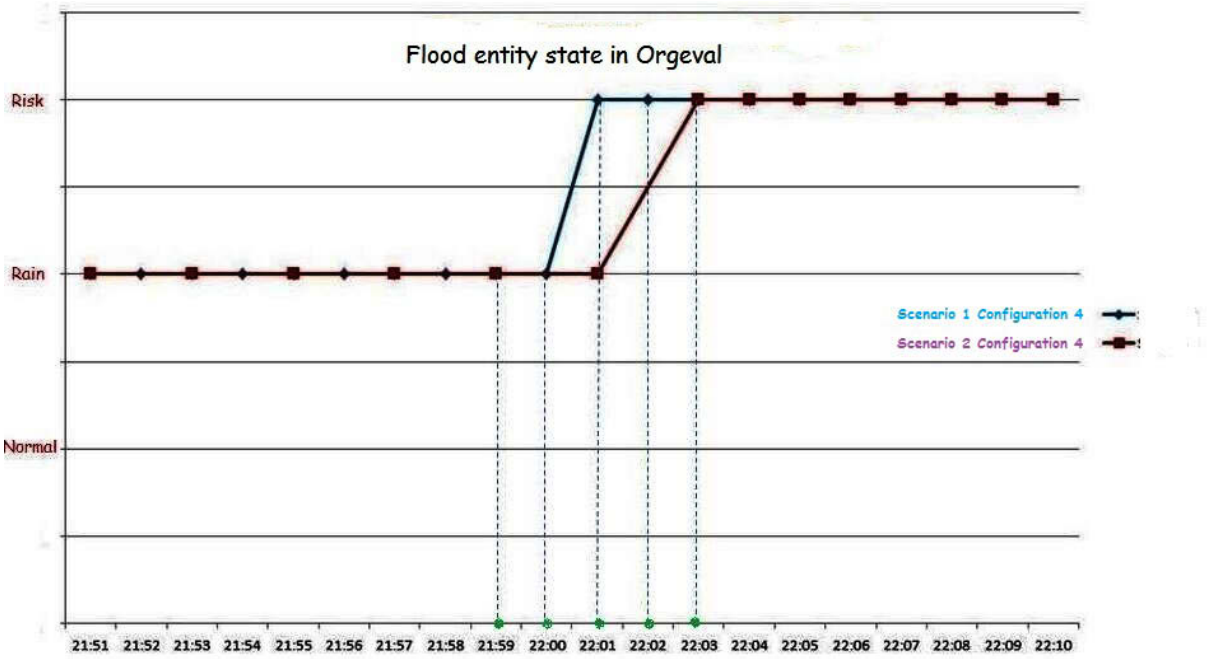


Figure 6.4.2: Example of evolution and delay of the flood entity states in scenarios 1 and 2 configuration 4 during February, 2008

6.4.1.3 Evaluation and Conclusion of Scenario 2 Configuration 4

Consequently, based on the metric **"total amount of exchanged communication packets"** presented in Table 6.4.1, compared with scenario 1 configuration 4, scenario 2 configuration 4 exchanges less communication packets.

As presented in section 5.2.4.1, node consumes 0.144 J when the communication unit is in "OFF" mode during a time cycle of one minute. This value is 3.77 J when the communication unit is in "ON" mode. Thus, using the "OFF" mode instead of the "ON" one makes a node saving 3.626 J during a one-minute time cycle. Thus, scenario 2 configuration 4 saves 553,820.736 J (*(250,560 - 97,824) × 3.626*). Therefore, the WSN in scenario 2 configuration 4 will have a theoretical longer lifetime than the one in scenario 1 configuration 4.

For the level of QoS of the flood entity, although the total number of Flood entity state changes (21) is the same in the two systems as presented in Table 6.4.2, scenario 2 configuration 4 has 9 delayed state changes in comparison with scenario 1 configuration 4. It means that scenario 2 configuration 4 is less reactive to detect the Flood entity state changes than scenario 1 configuration 4. Depending on the phenomenon, this additional delay can have important consequences. Compared with scenario 1 configuration 4, scenario 2 configuration 4 accumulates a delay of 13 minutes (1+1+1+2+2+1+2+1+2) to detect Flood entity state changes.

6.4.2 Comparison of Scenarios 1 and 2 for Configuration 10

In this section, we focus on the scenarios 2 using configuration 10. We evaluate scenario 2 configuration 10 using the scenario 1 configuration 4 as a baseline. We also compare scenario 2 configuration 10 with scenario 2 configuration 4.

6.4.2.1 Total Amount of Exchanged Communication Packets

Table 6.4.4 shows the total number of messages exchanged for scenarios 1 and 2 using configuration 10 based on the Orgeval watershed data of February, 2008. The total number of exchanged communication packets of scenario 2 configuration 10 (97,836 packets) is smaller than the one of scenario 1 configuration 4 (250,560 packets), but is a little bigger than the one of scenario 2 configuration 4 (97,824 packets). The reason is that, in scenario 1 configuration 10, the system detects earlier flood or risky states. Thus, it is more reactive in comparison with scenario 2 configuration 4 in terms of communication frequency adaptation.

Table 6.4.4: Total number of exchanged communication packets in scenario 1 configuration 10, scenario 2 configuration 4 and configuration 10 for February, 2008

	Scenario 1 Configuration 4	Scenario 2 Configuration 4	Scenario 2 Configuration 10
01/02/2008 00:00 to 01/03/2008 00:00	250,560	97,824	97,836

6.4.2.2 Flood Entity State Changes

Figure 6.4.3 presents the *F* entity state changes deduced by the systems for scenario 1 configuration 10 and scenario 2 configuration 10. First let us see the total number of Flood entity state changes.

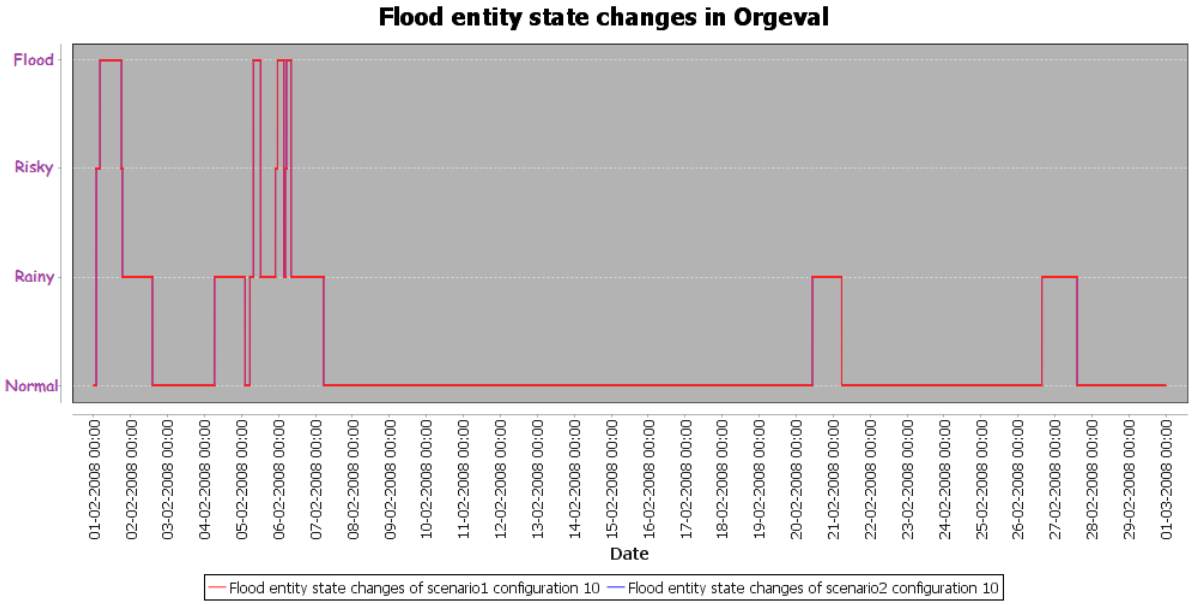


Figure 6.4.3: Flood entity states evolution for scenario 1 configuration 10 and scenario 2 configuration 10 for February, 2008

The curve associated to scenario 2 configuration 10 is very close to the one of scenario 1 configuration 10 as presented in Figure 6.4.3. They seem similar because the time scale is the day. As shown in Table 6.4.5, the number of each Flood entity state changes is the same (21) for scenario 1 configuration 10 and scenario 2 configuration 10 for February, 2008. However, if the scale is enlarged to the minute, some timestamp differences also will appear as shown in Table 6.4.6 w here the timestamps of each Flood entity state changes of scenario 1 configuration 10, scenarios 2 configuration 4 and scenario 2 configuration 10 for February, 2008 is presented.

6.4.2.2.1 Total Number of Each Flood Entity State Changes

Table 6.4.5: Number of each flood entity state changes in scenario 1 configuration 10, scenario 2 configurations 4 and 10 for February, 2008

	N>Ra	*N>Ri*	*Ra>Ri*	*Ra>F*	*Ri>F*	*F>Ri*	*F>Ra*	*Ri>Ra*	*Ra>N*
Scenario 1 Config 10	4	1	2	1	3	1	3	1	5
Scenario 2 Config 4	4	1	2	1	3	1	3	1	5
Scenario 2 Config 10	4	1	2	1	3	1	3	1	5

6.4.2.2.2 Timestamps of Each Flood Entity State Changes

Compared with scenario 1 configuration 10, scenario 2 configuration 10 has 7 state changes that are detected with delay, which is 2 less than scenario 2 configuration 4. In Table 6.4.6, in order to highlight the differences between among these 3 systems, we filled them in bold.

Table 6.4.6: Timestamps of each flood entity state changes in scenario 1 configuration 10 and scenario 2 configurations 4 and 10 for February, 2008

	N>Ra	N>Ri	Ra>Ri	Ra>F	R>F	F>Ri	F>Ra	Ri>Ra	Ra>N
Scenario 1 Config 4		01/02 01:41			01/02 04:03	01/02 18:01		01/02 18:41	02/02 14:08
	04/02 06:30								05/02 02:16
	05/02 05:24			05/02 07:41			05/02 12:21		
			05/02 22:01		05/02 23:21		06/02 03:31		
			06/02 04:27		06/02 05:11		06/02 08:11		07/02 05:18
	20/02 10:35								21/02 05:34
	26/02 15:33								27/02 14:25
Scenario 2 Config 4		**01/02 01:42 (+1m)**			01/02 04:03	01/02 18:01		01/02 18:41	**02/02 14:09 (+1m)**
	04/02 06:30								05/02 02:16
	05/02 05:25 (+1m)			**05/02 07:43 (+2m)**			05/02 12:21		

			05/02 22:03 (+2m)		05/02 23:21		06/02 03:31		
			06/02 04:27		06/02 05:11		06/02 08:11		**07/02 05:19 (+1m)**
	20/02 10:37 (+2m)								**21/02 05:35 (+1m)**
	26/02 15:35 (+2m)								27/02 14:25
Scenario 2 Config 10		**01/02 01:42 (+1m)**			01/02 04:03	01/02 18:01		01/02 18:41	**02/02 14:09 (+1m)**
	04/02 06:30								05/02 02:16
	05/02 05:25 (+1m)			05/02 07:41			05/02 12:21		
			05/02 22:01		05/02 23:21		06/02 03:31		
			06/02 04:27		06/02 05:11		06/02 08:11		**07/02 05:19 (+1m)**
	20/02 10:37 (+2m)								21/02 **05:35** (+1m)
	26/02 15:35								27/02 14:25

	(+2m)								

Figure 6.4.4 presents an example to show why scenario 2 configuration 10 is more reactive than scenario 2 configuration 4. As presented in Figure 6.4.2, scenario 2 configuration 4 failed to detect "Risky (*F3*)" at the time instant "05/02 22:01". Scenario 2 configuration 10 solves this kind of problem.

- Time instant "05/02 21:59", as scenario 2 configuration 4, scenario 2 configuration 10 deduces that the Flood entity is in state "Rainy (*F2*)". Also, based on Table 5.6.3, the communication frequency of scenario 2 configuration 10 is one communication every two minutes. Thus, the watercourse nodes of scenario 2 configuration 10 will only send their next measurements to the DSS at the time instant "05/02 22:01".
- Time instant "05/02 22:01", unlike scenario 2 configuration 4, the Watercourse nodes of scenario 2 configuration 10 will compute the slopes between time intervals [05/02 21:59; 05/02 22:00] and [05/02 22:00; 05/02 22:01] and send the maximum of this two values. The slope (**max slope of water flow rate per communication interval**) between the time interval [05/02 22:00; 05/02 22:01] is above the $Th_{Watercourse}$ threshold. Consequently, the state change will be detected at time instant "05/02 22:01", earlier than scenario 2 configuration 4.

Thus, scenario 2 configuration 10 is more reactive than scenario 2 configuration 4. This example also shows why the scenario 2 configuration 10 generates a little bit higher number of exchanged packets than the one of the scenario 2 configuration 4 as shown in Table 6.4.4.

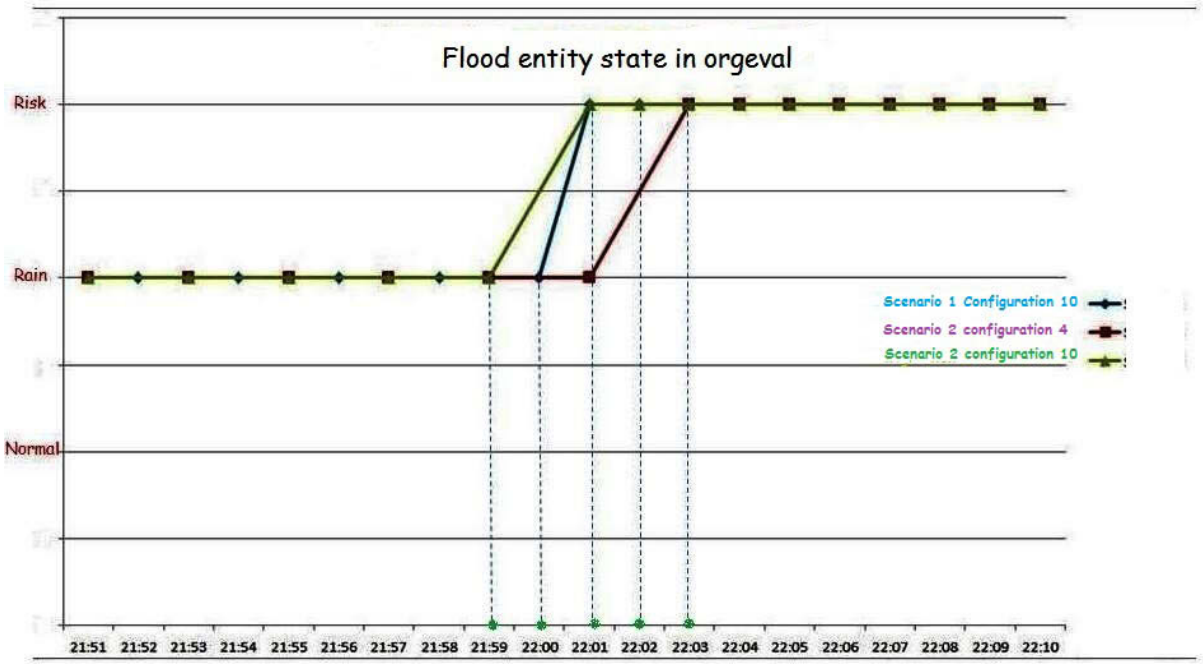


Figure 6.4.4: Example of evolution and delay of the flood entity states in scenario 1 configuration (4 or) 10 and 2 configurations 4 and 10 during February, 2008

6.4.2.3 Evaluation and Conclusion of Scenario 2 Configuration 10

So based on the metric **"total amount of exchanged communication packages"** presented in Table 6.4.4, compared with scenario 1 configuration 10, scenario 2 configuration 10 exchanges less communication packets. Therefore, scenario 2 configuration 10 save 553,777.224 J ((250,560 - 97,836) × 3.626). This value is little smaller than the one of scenario 2 configuration 4 (553,820.736 J) as presented in section 6.4.1.3. So, the WSN of scenario 2 configuration 10 has a longer lifetime than the one of scenario 1 configuration 4, but a shorter lifetime than the one of scenario 2 configuration 4.

For the QoS at the level of the Flood entity, the total number of Flood entity state changes is the same for the studied systems: scenario 1 configuration 4, scenario 2 configuration 4 and scenario 2 configuration 10. Compared with scenario 1 configuration 4, scenario 2 configuration 10 accumulates a delay of 9 (1+1+1+1+2+1+2) minutes to detect Flood entity state changes. Scenario 2 configuration 10 has a smaller cumulative delays than scenario 2 configuration 4. It means that scenario 2 configuration 10 is more reactive to detect the Flood entity state changes than scenario 2 configuration 4, but less reactive to detect the Flood entity state changes than scenario 1 configuration 4.

6.4.3 Comparison of Scenario 1 Configuration 12 and Scenario 2 Configuration 12

In this section, we evaluate scenario 2 configuration 12 using scenario 1 configuration 12 as baseline.

6.4.3.1 Total Amount of Exchanged Communication Packets

Table 6.4.7 shows the total number of messages exchanged in scenario 1 configuration 12 and scenario 2 configuration 12 using Orgeval basin data of February, 2008. The total number of exchanged communication packets of scenario 2 configuration 12 is smaller than the one of scenario 1 configuration 12.

Table 6.4.7: Total number of exchanged communication packets in scenario 1 configuration 12 and scenario 2 configuration 12

	Scenario 1 Configuration 12	Scenario 2 Configuration 12
01/02/2008 00:00 to 01/03/2008 00:00	250,560	101,244

6.4.3.2 Flood Entity State Changes

Figure 6.4.5 presents *F* entity state changes deduced by the systems associated to scenario 1 configuration 12 and scenario 2 configuration 12. The total number of flood entity state changes is presented in Table 6.4.8.

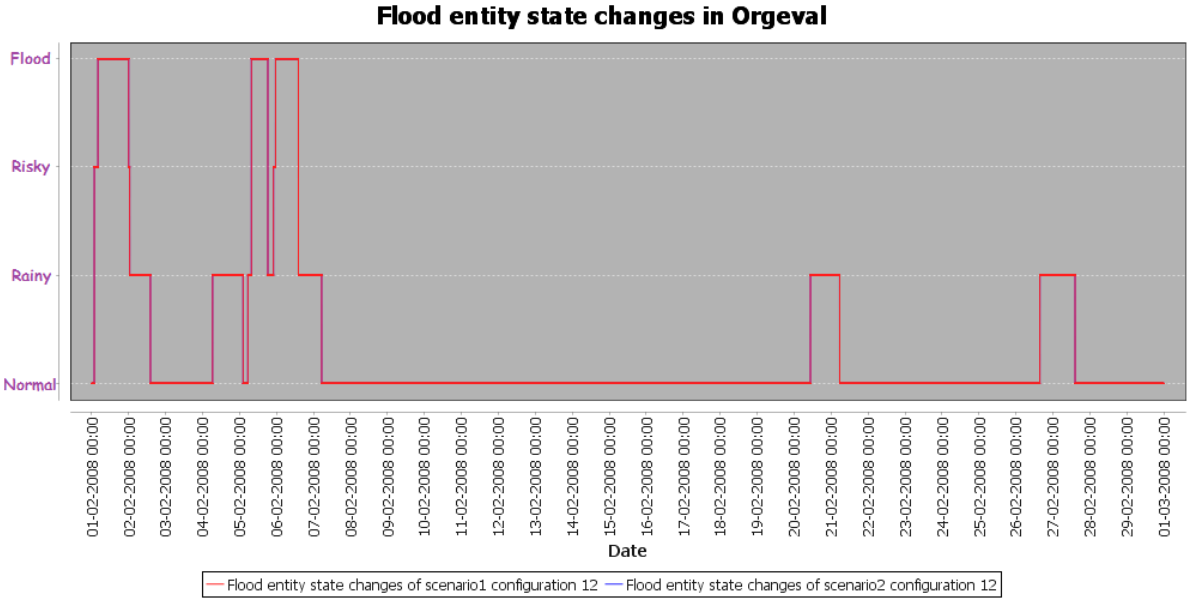


Figure 6.4.5: Flood entity states evolution for scenario 1 configuration 12 and scenario 2 configuration 12 in February, 2008

6.4.3.2.1 Total Number of Each Flood Entity State Changes

In Table 6.4.8, we can see the number of each Flood entity state changes in scenario 1 configuration 12 and scenario 2 configuration 12. For each kind of state changes, these two systems provide the same number. The total number of Flood entity state changes is also 18.

Table 6.4.6: Number of each flood entity state changes of scenario 1 configuration 12 and scenario 2 configuration 12 in February, 2008

	N>Ra	*N>Ri*	*Ra>Ri*	*Ra>F*	*Ri>F*	*F>Ri*	*F>Ra*	*Ri>Ra*	*Ra>N*
Scenario 1 Config 12	4	1	1	1	2	1	2	1	5
Scenario 2 Config 12	4	1	1	1	2	1	2	1	5

6.4.3.2.2 Timestamps of Each Flood Entity State Changes

Compared with scenario 1 configuration 12, there are some delays for the state changes in the scenario 2 configuration 12. The scenario 2 configuration 12 has 7 state changes that are

delayed. In Table 6.4.9, the delayed timestamps are presented in bold format with the delay duration indicated in brackets.

Table 6.4.7: Timestamps of each flood entity state changes of the systems of configuration 12 in scenario 1 and scenario 2 and scenario 3 in Feburary, 2008

	N>Ra	*N>Ri*	*Ra>Ri*	*Ra>F*	*Ri>F*	*F>Ri*	*F>Ra*	*Ri>Ra*	*Ra>N*
Scenario 1 Config 12		01/02 01:41			01/02 04:03	02/02 00:01		02/02 00:41	02/02 14:08
	04/02 06:30								05/02 02:16
	05/02 05:24			05/02 07:41			05/02 18:21		
			05/02 22:01		05/02 23:21		06/02 14:11		
									07/02 05:18
	20/02 10:35								21/02 05:34
	26/02 15:33								27/02 14:25
Scenario 2 Config 12		**01/02 01:42 (+1m)**			01/02 04:03	02/02 00:01		02/02 00:41	**02/02 14:09 (+1m)**
	04/02 06:30								05/02 02:16
	05/02 05:25 (+1m)			05/02 07:41			05/02 18:21		
			05/02 22:01		05/02 23:21		06/02 14:11		

									07/02 05:19 (+1m)
	20/02 10:37 (+2m)								**21/02 05:35 (+1m)**
	26/02 15:35 (+2 m)								27/02 14:25

In Figure 6.4.6, an example of state changes for the scenario 1 configuration 12 and scenario 2 configuration 12 is provided to illustrate a case of delay. We only take into account the messages sent by the watercourse nodes.

- Time instant "01/02/2008 01:39", scenario 1 configuration 12 and scenario 2 configuration 12 deduce that the Flood entity state is in state "Normal (*F1*)". Based on Table 5.6.3, the communication frequency of scenario 2 configuration 12 is one communication every three minutes. So, the wireless sensors of scenario 2 configuration 12 will only send their next measurements to the DSS at the time instant "01/02/2008 01:42".
- Time instant "01/02/2008 01:41", the system of scenario 1 configuration 12 detects a state change from "Normal (*F1*)" to "Risky (*F3*)". This change has also to be theoretically detected by the scenario 2 configuration 12. But, the water flow rate values of the scenario 2 configuration 12 nodes are not sent at this time instant.
- Time instant "01/02/2008 01:42", from the values sent by the watercourse nodes, the system of the scenario 2 configuration 12 computes the "**max slope of water flow rate per 6h"** between the time interval [01/02/2008 01:39, 01/02/2008 01:42]. The computed value is bigger than the $Th_{Watercourse}$ threshold and the Flood entity state changes from "Normal (*F1*)" to "Risky (*F3*)".

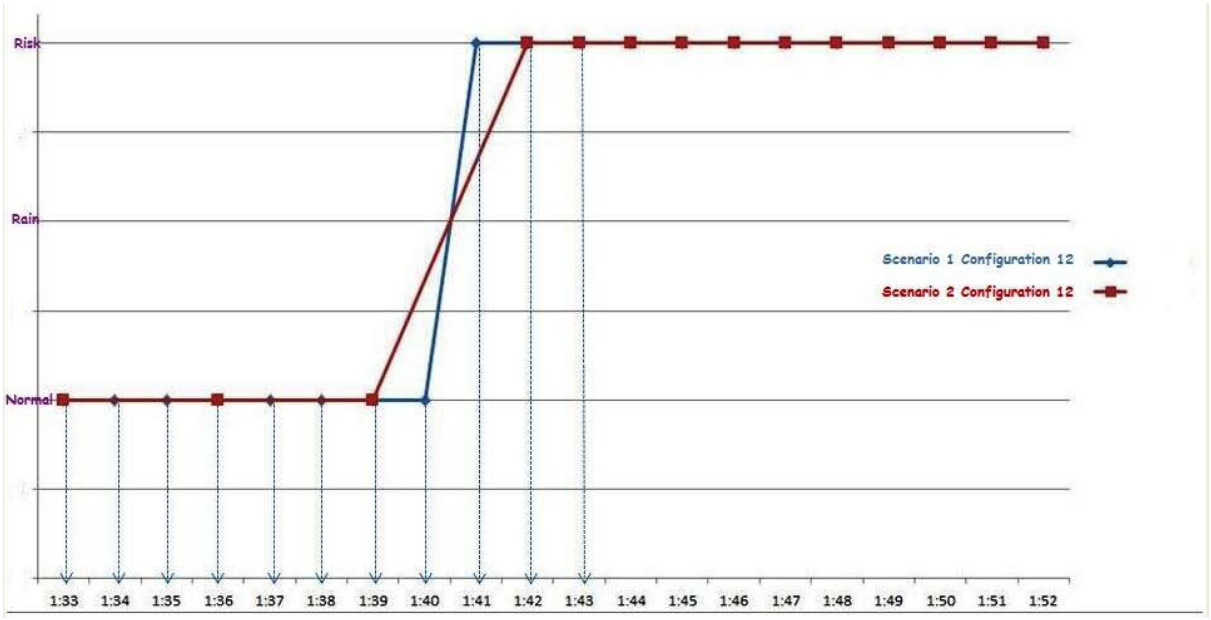


Figure 6.4.8: Example of evolution and delay of the flood entity states in scenario 1 configuration 12 and scenario 2 configuration 12 during February, 2008

6.4.3.3 Evaluation and Conclusion of Scenario 2 Configuration 12

So based on the metric **"total amount of exchanged communication packages"** presented in Table 6.4.7, compared with scenario 1 configuration 12, scenario 2 configuration 12 exchanges less communication packets. Thus, scenario 2 configuration 12 saves 541,419.816 J (*(250,560 - 101,244) × 3.626*). For the configuration 12, the WSN of scenario 2 has a longer lifetime than the one of scenario 1.

For the QoS at the level of the Flood entity, the total number of each Flood entity state changes is the same for the scenario 1 configuration 12 and scenario 2 configuration 12. Scenario 2 configuration 12 has 7 Flood entity state changes delayed compared with scenario 1 configuration 12. Scenario 2 configuration 12 accumulates a delay of 9 (*1+1+1+1+2+1+2*) minutes to detect Flood entity state changes compared with scenario 1 configuration 12.

6.5 Flood and Node Adaptive Context-aware System Evaluation

The scenarios with configuration 12 are more stable and avoid some risky situations at the level of the considered phenomenon. Therefore, in this section, we decide to compare scenario 3 configuration 12 with the scenario 1 configuration 12 as baseline. Moreover, we also compare scenario 3 configuration 12 with scenario 2 configuration 12. The purpose of this latter is to study the behavior of the adaptations based on many entity states (*F* entity state and *N* entities state).

6.5.1 Total Amount of Exchanged Communication Packets

Table 6.5.1 shows the total number of messages exchanged of scenario 1, scenario 2 and scenario 3 using configuration 12 in February 2008. We can see, in this table, the total number of transmitted packets in scenario 3 configuration 12 is the smallest because Node entity is involved in the decision of adaptation. A node with a low energy level can save more energy by decreasing its communication frequency.

Table 6.5.1: Total number of exchanged communication packets of scenarios 1 to 3 for configuration 12

	Scenario 1 Configuration 12	Scenario 2 Configuration 12	Scenario 3 Configuration 12
01/02/2008 00:00 to 01/03/2008 00:00	250,560	101,244	79,606

We can notice that the scenario 2 has half less number of exchange packet than the baseline. Scenario 3 has less than scenario 2. The adaptation differences between scenarios 2 and 3 explains the difference in the total number of exchanged communication packets. Section 6.5.4 presents in detail the communication frequency adaptation.

The impact of the adaptation on the Flood entity state changes will be studied in next paragraph.

6.5.2 Flood Entity State Changes

Figure 6.5.1 shows the evolution of the flood entity states for scenario 1 configuration 12, scenario 2 configuration 12 and scenario 3 configuration 12. We can see that the three curves are very close.

In scenario 3 configuration 12, when the Flood entity state is “Normal (*F1*)” or “Rain (*F2*)”, there is a delay in the reception of the values to compute the “**rainfall amount per 24h**” of Orgeval watershed due to different nodes states. For example, the *N* entity state of two precipitation nodes “melarchez-p35_RainGauge” and “boissy-meteo_RainGauge” is “Stable (*NA*)”. The third one: “boissy-p28_RainGauge” is in state “Critical (*NB*)”. So, based on Table 5.6.3, “melarchez-p35_RainGauge” and “boissy-meteo_RainGauge” have different communication frequency in comparison with “boissy-p28_RainGauge” when the Flood entity state is “Normal (*F1*) or “Rain (*F2*)”. At a certain time instant, the value of “**rainfall amount per 24h**” of the Orgeval watershed can be lower due to different node communication frequencies. There is a risk to miss the detection of a Flood entity state change from “Normal (*F1*)” to “Rain (*F2*)”. We propose a method in order to limit the impact

of this drawback. To present it, we consider the case where, at a given time instant, the DSS receives the value of "**rainfall amount per 24h**" for the "melarchez-p35_RainGauge" and "boissy-meteo_RainGauge" nodes but not the value from the "boissy-p28_RainGauge" node. In this situation, the DSS will aggregate the update values of "melarchez-p35_RainGauge" and "boissy-meteo_RainGauge" nodes plus the latest value of the "boissy-p28_RainGauge" node received by the DSS (the value received at the previous communication time instant). This computed value of "**rainfall amount per 24h**" of the Orgeval watershed will be a little different than the real one. But, we expect that it will decrease the possibility of missing the detection of a Flood entity state change from "Normal (*F1*)" to "Rain (*F2*)".

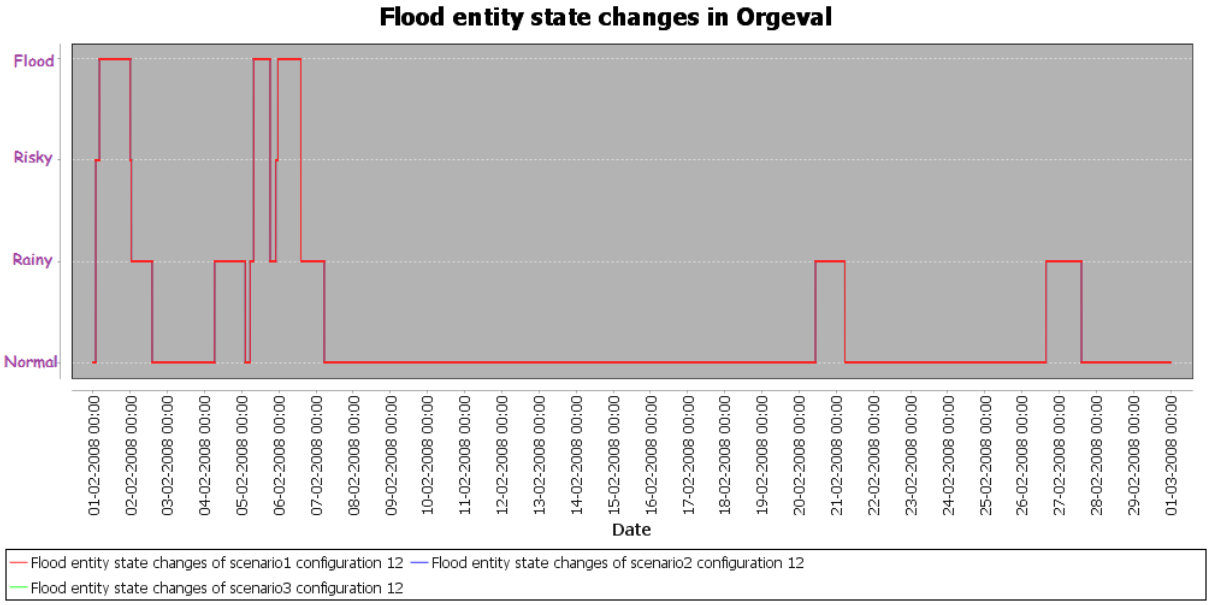


Figure 6.5.1 Flood entity states evolution in scenarios 1 to 3 for configuration 12

6.5.2.1 Total Number of Each Flood Entity State Changes

In Table 6.5.2, we can see the number of each Flood entity state changes in scenario 1 configuration 12, scenario 2 configuration 12 and scenario 3 configuration 12. For each state transition, the number is the same. The total number of Flood entity state changes in these three systems is 18.

Table 6.5.2: Number of each flood entity state changes in scenarios 1 to 3 for configuration 12 during February, 2008

	N>Ra	*N>Ri*	*Ra> Ri*	*Ra>F*	*Ri>F*	*F>Ri*	*F>Ra*	*Ri>Ra*	*Ra>N*
Scenario 1 Config 12	4	1	1	1	2	1	2	1	5
Scenario 2	4	1	1	1	2	1	2	1	5

Config 12									
Scenario 3 Config 12	4	1	1	1	2	1	2	1	5

6.5.2.2 Timestamps of Each Flood Entity State Changes

For the configuration 12, there are some delays for the state changes in both scenarios 2 and 3 compared with scenario 1. The scenario 2 configuration 12 has 7 delayed state changes that are delayed. The scenario 3 configuration 12 has 8 delayed state changes because the Node entity is involved in the adaptation decision. A node with low energy will decrease it communication frequency which can result in a delay in a Flood entity state change detection. The delayed timestamps are presented in bold format. Table 6.5.3 contains also in brackets the delay duration.

Table 6.5.3: Timestamps of each flood entity state changes in scenarios 1 to 3 for configuration 12 during February, 2008

	N>Ra	N>Ri	Ra>Ri	Ra>F	Ri>F	F>Ri	F>Ra	Ri>Ra	Ra>N
Scenario 1 Config 12		01/02 01:41			01/02 04:03	02/02 00:01		02/02 00:41	02/02 14:08
	04/02 06:30								05/02 02:16
	05/02 05:24			05/02 07:41			05/02 18:21		
			05/02 22:01		05/02 23:21		06/02 14:11		
									07/02 05:18
	20/02 10:35								21/02 05:34
	26/02 15:33								27/02 14:25
Scenario 2		01/02			01/02	02/02		02/02	02/02

Config 12		**01:42 (+1m)**			04:03	00:01		00:41	**14:09 (+1m)**
	04/02 06:30								05/02 02:16
	05/02 05:25 (+1m)			05/02 07:41			05/02 18:21		
			05/02 22:01		05/02 23:21		06/02 14:11		
									07/02 05:19 (+1m)
	20/02 10:37 (+2m)								**21/02 05:35 (+1m)**
	26/02 15:35 (+2m)								27/02 14:25
Scenario 3 Config 12		**01/02 01:45 (+4m)**			01/02 04:03	02/02 00:01		02/02 00:41	**02/02 14:09 (+1m)**
	04/02 06:30								05/02 02:16
	05/02 05:25 (+1m)			05/02 07:41			05/02 18:21		
			05/02 22:01		05/02 23:21		06/02 14:11		

									07/02 05:19 (+1m)
	20/02 10:37 (+2m)								**21/02 05:37 (+3m)**
	26/02 15:34 (+1m)								**27/02 14:26 (+1m)**

6.5.3 Node Entity State Changes

In this section, we analyze the Node entity state changes based on the energy level of the node itself. The purpose is to study the behavior of the wireless sensor node according to its level of energy and the flood management in the current section and in section 6.5.4.

Figure 6.5.2 presents the *N* entity state changes of the precipitation nodes in the scenario 3 configuration 12. Main indications about this figure are:

- The initial *N* entity state of the two nodes "melarchez-p35_RainGauge" and "boissy-meteo_RainGauge" is "Stable (*NA*)". These two nodes are represented by red curve and blue curve respectively. They have the same *N* entity state changes, which is much bigger than the one of "boissy-p28_RainGauge" node.
- The initial *N* entity state of the "boissy-p28_RainGauge" node is "Critical (*NB*)". This node is represented as a green curve.

Figure 6.5.2 clearly shows that the *N* entity state changes are different for the 3 precipitation nodes because their initial energy level ($Energy_{Node}$) is different. This results in a different behavior during the simulation.

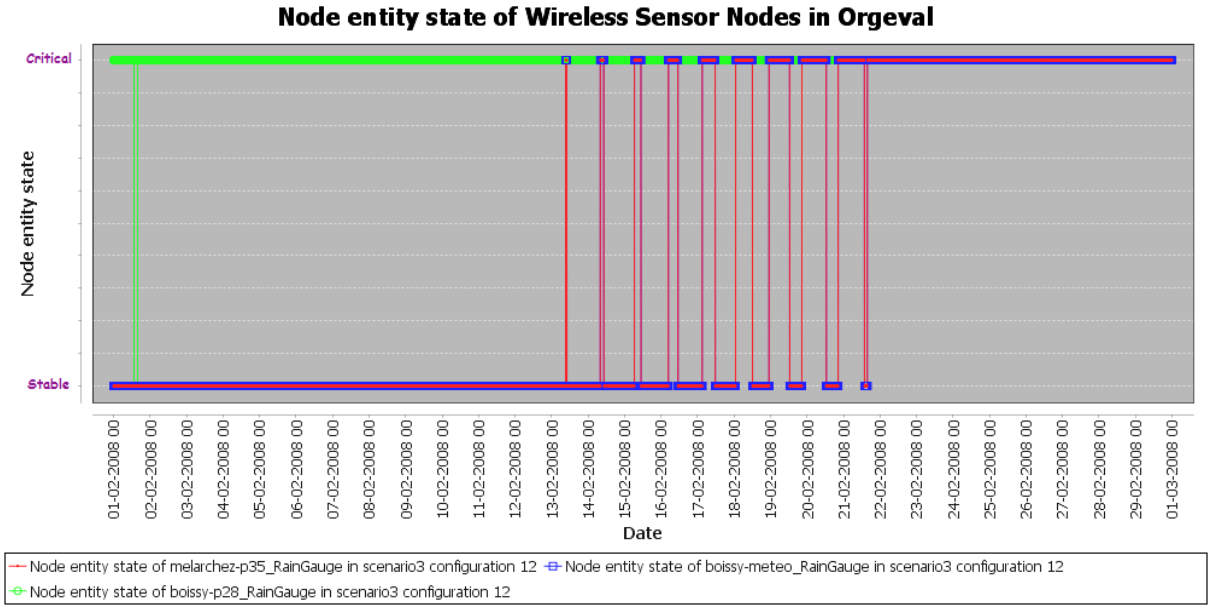


Figure 6.5.2: Node entity states evolution of the precipitation nodes in scenario 3 configuration 12 for February, 2008

Figure 6.5.3 presents the *N* entity state changes of the watercourse nodes in the scenario 3 configuration 12. The initial *N* entity states of the watercourse nodes are as followed:

- Node "lesAvenelles_StreamGauge" state is "Stable (*NA*)".
- Node "melarchez_StreamGauge" state is "Critical (*NB*)".

In Figure 6.5.3, the curve of the "lesAvenelles_StreamGauge" node is the same as the ones of "melarchez-p35_RainGauge" and "boissy-meteo_RainGauge" precipitation nodes of the Figure 6.5.2. The curve of the "melarchez_StreamGauge" node is the same as the "boissy-p28_RainGauge" one. Even if these nodes collect different measurements, they share the same energy model with the same parameters. Their behavior is related to the energy level of their battery at the beginning of the simulation.

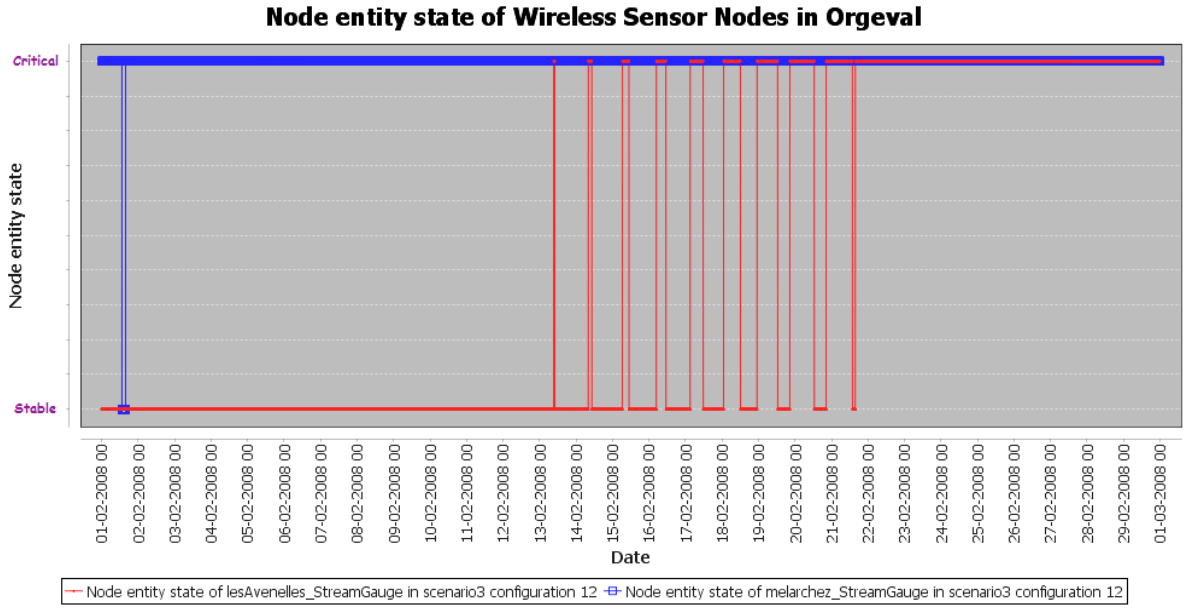


Figure 6.5.3: Node entity states evolution of the watercourse nodes in scenario 3 configuration 12 for February, 2008

Figure 6.5.4 presents the *N* entity state changes of the outlet node in scenario 3 configuration 12. The initial *N* entity state of this outlet node, called "leTheil_StreamGauge", is "Stable (*NA*)". The curve of this node is the same as the ones of "lesAvenelles_StreamGauge" node in Figure 6.5.3 a nd, "melarchez-p35_RainGauge" and "boissy-meteo_RainGauge nodes in Figure 6.5.2.

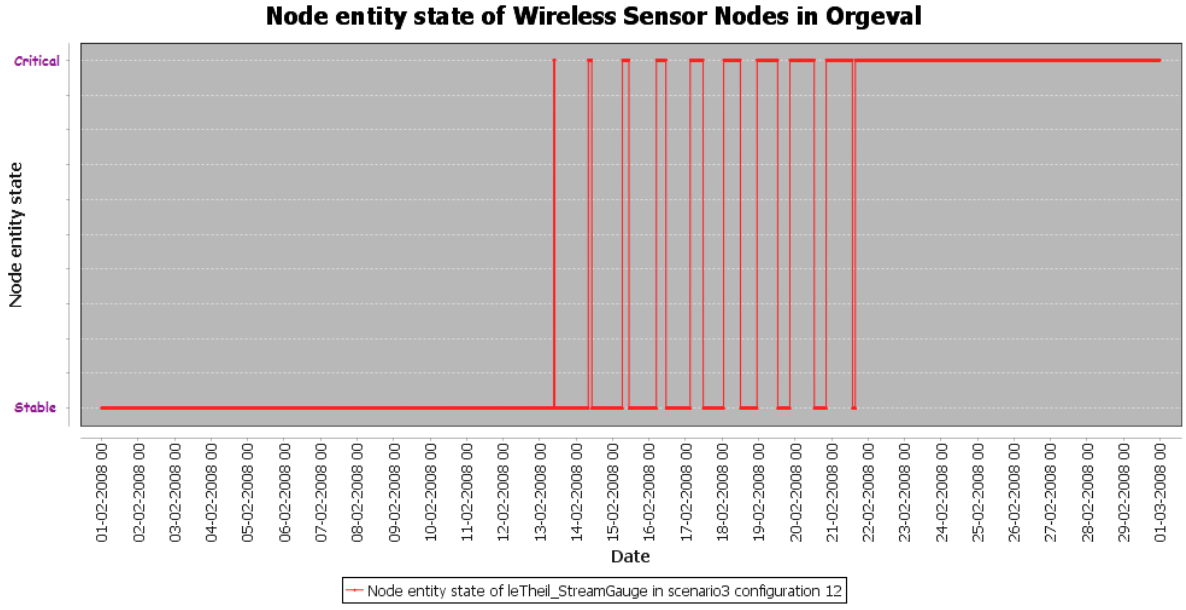


Figure 6.5.4: Node entity states evolution of the outlet node in scenario 3 configuration 12 for February, 2008

6.5.3.1 Total Number of Node Entity State Changes

Table 6.5.4 presents the total number of node entity state changes for each node in scenario 3 configuration 12 during February, 2008. This table is based on Figures 6.5.2 to 6.5.4. In this table, we can see that the nodes which have the same initial energy level have the same total number of node entity state changes.

Table 6.5.4: Total number of node entity state changes of each node in scenario 3 configuration 12 for February, 2008

	Initial Energy (J)	"Stable" > "Critical"	"Critical" > "Stable"
melarchez-p35_RainGauge	20000	11	10
boissy-meteo_RainGauge	20000	11	10
boissy-p28_RainGauge	14950	1	1
lesAvenelles_StreamGauge	20000	11	10
melarchez_StreamGauge	14950	1	1
leTheil_StreamGauge	20000	11	10

6.5.4 Communication Frequency

Studying the evolution of the communication frequencies enables to check if the adaptive context-aware system that we implement well operates. Figure 6.5.5 presents the communication period of the following precipitation nodes used in scenario 3 configuration 12: "melarchez-p35_RainGauge", "boissy-meteo_RainGauge", "boissy-p28_RainGauge".

In Figure 6.5.5, we consider, for example, the "boissy-p28_RainGauge" node presented as the green curve. At the beginning of 1st of February, 2008, the *F* entity is in "Normal (*F1*)" state as presented in Figure 6.5.1 and, the *N* entity state of the "boissy-p28_RainGauge" node is "Critical (*NB*)" as presented in Figure 6.5.2. Therefore, based on the Table 5.6.3, the communication frequency is the acquisition/sample frequency. Thus, the communication period is equal to 5 minutes what can be seen in Figure 6.5.5.

Figure 6.5.6 provides the same study as Figure 6.5.5 but for the watercourse nodes: "lesAvenelles_StreamGauge", "melarchez_StreamGauge". We can notice that the shape of the "melarchez_StreamGauge" node (in blue in Figure 6.5.6) is exactly the same as the "boissy-p28_RainGauge" node (in green in Figure 6.5.5) because they have the same energy level.

Finally, Figure 6.5.7 presents the evolution of the communication period for the outlet node "leTheil_StreamGauge" whose N entity state is "Stable (NA)" at the beginning of the simulation. The evolution of the communication period of this node is the same than the one of the "melarchez-p35_RainGauge", the "boissy-meteo_RainGauge" and the "lesAvenelles_StreamGauge" nodes, all these nodes having the same initial energy.

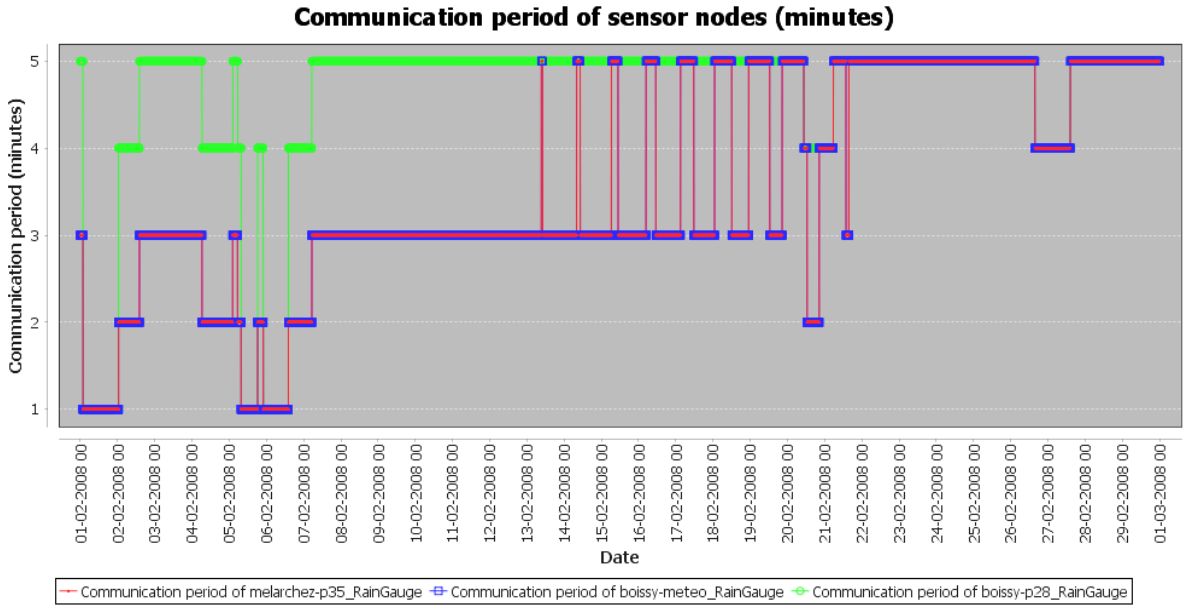


Figure 6.5.5: Communication period of the precipitation nodes in scenario 3 configuration 12 for February, 2008

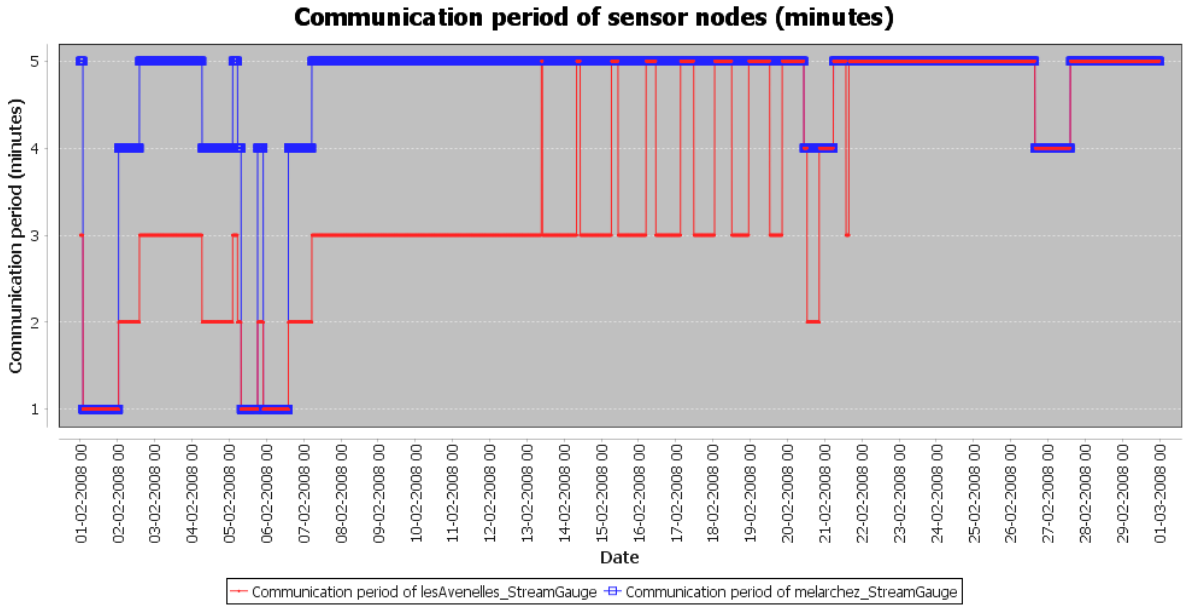


Figure 6.5.6: Communication period of the watercourse nodes in scenario 3 configuration 12 for February, 2008

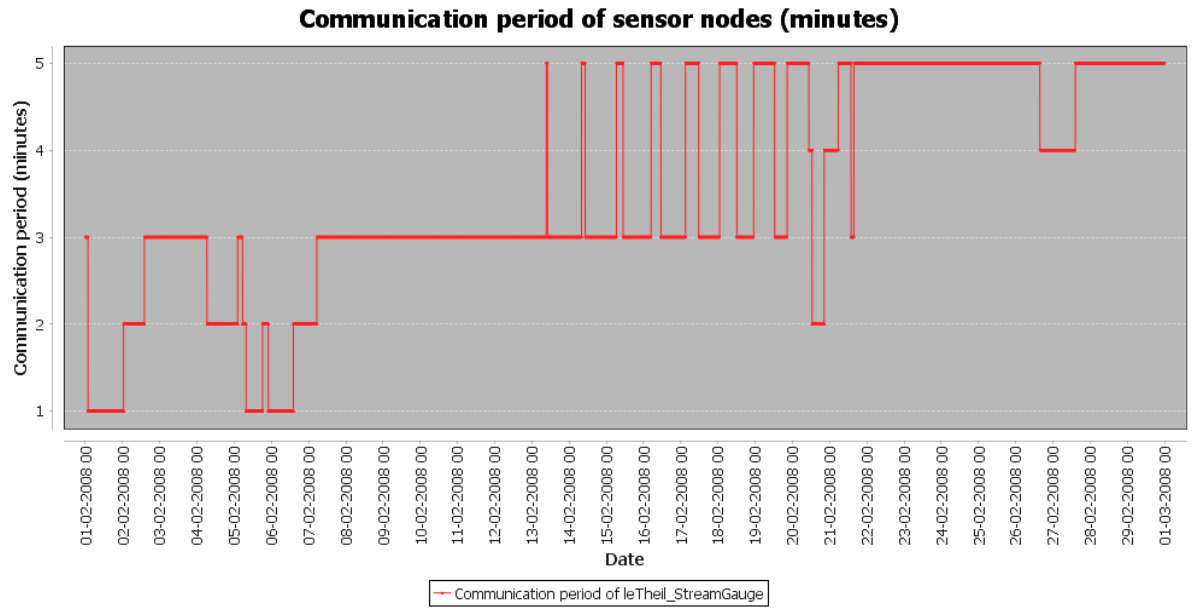


Figure 6.5.7: Communication period of the outlet node in scenario 3 configuration 12 for February, 2008

Figures 6.5.5 to 6.5.7 confirm that nodes with the same initial energy will have the same communication frequency adaptation. It also means that the adaptation of wireless sensor node is really according to its energy level and the flood management.

All these figures show also that, most of the time, the nodes have a communication frequency higher than every 1 minute. This explains why fewer amounts of communication packets is generated for this scenario 3 in comparison with scenario 1. For scenario 2, whatever the energy level of a node is, its communication frequency is "forced" by the *F* entity state which is different in scenario 3 as we mentioned before. This results in energy consumption for both scenarios 2 and 3 lower than the one in scenario 1. The energy consumption of the WSN in scenario 3 is also lower than the one of WSN in scenario 2. Indeed, nodes in scenario 3 have the possibility to "protect" themselves, by reducing their wireless communications, in some specific situations according to the state of the Flood entity. Finally, we can say that the performances of the WSN increase according to the complexity of the context adaptation.

6.5.5 Evaluation and Conclusion of Scenario 3 Configuration 12

So based on the metric **"total amount of exchanged communication packages"** presented in Table 6.5.1, compared with scenario 2 configuration 12, scenario 3 configuration 12 exchanges less communication packets. So, scenario 3 configuration 12 save 619,879.204 J ((250,560 - 79,606) × 3.626), which is a little bigger than the one of scenario 2 configuration 12 (541,419.816 J). The WSN of scenario 3 configuration 12 has a longer lifetime than the one of scenario 2 configuration 12, and longer lifetime than the one of scenario 1 configuration 12.

For the QoS at the level of the Flood entity, for configuration 12, scenarios 1 to 3 have the same total number of Flood entity state changes. For configuration 12, compared with scenario 1, scenario 2 and scenario 3 accumulates 9 and 14 (*4+1+1+1+2+3+1+1*) minutes of delay respectively to detect Flood entity state changes.

6.6 Conclusion

In this section, we evaluate our proposed formalization through three different scenarios. Our two reasoning steps to create the high-level context are also implemented in our simulation.

Based on the results of the comparison of scenario 1 configurations 4 and 12, we notice that:

- The lifetime of the WSN in these two systems is similar.
- The total number of *F* entity state changes of these two systems is the same.
- The scenario 1 configuration 4 is more reactive to detect the *F* entity state changes than scenario 1 configuration 12.

The latter observation can be interpreted differently. For some phenomena, this reactivity is an advantage. However, for the considered use case (flood management in a watershed), this is a drawback because it can generate potential dangerous situations if we think too earlier that a flood event is finished. So, for this use case, the stability of scenario 1 configuration 12 is more suitable.

Table 6.6.1: Comparative synthesis between scenario 2, configurations 4, 10 and 12 and, scenario 3 configuration 12 for February, 2008

	Saved Energy (J)	Flood entity QoS		Considered Baseline
		Flood entity state changes	Cumulative delays (min)	

Scenario 2 Config 4	553,820.736	21/21	13	Scenario 1 Config 4
Scenario 2 Config 10	553,777.224	21/21	9	Scenario 1 Config 4
Scenario 2 Config 12	541,419.816	18/18	9	Scenario 1 Config 12
Scenario 3 Config 12	619,879.204	18/18	14	

Table 6.6.1 presents the saved energy and two different metrics (entity state changes and cumulative delays in the detection) associated to the QoS at the level of the Flood entity for scenario 2 configurations 4, 10 and 12 and, scenario 3 configuration 12 during February 2008. For the Flood context-aware system related to scenario 2 configurations 4, 10 and 12, we notice that:

- Scenario 2 configuration 4 saves more energy than the two other systems. Therefore, the WSN of this system has a longer lifetime. However, it has more cumulative delays to detect Flood entity state changes, so it is less reactive than the two other systems.
- Scenario 2 configuration 12 saves less energy than the two other systems, so it has a shorter lifetime. However, it detects less Flood entity state changes that means it is more stable. Moreover, it has the same number of cumulative delays to detect Flood entity state changes than scenario 2 configuration 10, but less than scenario 2 configuration 4.

As we expected, the communication frequency adaptations of the sensor nodes based on the F entity help to improve the lifetime of the system. This adaption has also a limited impact on the QoS as shown the evaluation results of scenario 2 configuration 12.

In Table 6.6.1, for the Flood and Node context-aware systems represented by scenario 3 configuration 12, we can notice that:

- Scenario 3 configuration 12 saved more energy than scenario 2 configurations 4, 10 and 12. So, its associated WSN has a longer lifetime. Therefore, the communication frequency adaptations based on both F entity state and N entity state help to improve the lifetime of the system.
- Scenario 3 configuration 12 detects the same total number of Flood entity state changes than scenario 2 configuration 12 which means that it is also very stable.
- However, Scenario 3 configuration 12 has more cumulative delays to detect Flood entity state changes than scenario 2 configuration 12, so it is less reactive.

About this latter observation, we also know that nodes which by having the same initial energy level will have the same total number of node entity state changes and communication frequency adaptations. Therefore, the reason that scenario 3 configuration 12 is less reactive comes from the integration, in the decision adaptation process, of the Node entity. A node with low energy can delay to detect the Flood entity state changes by decreasing its communication frequency.

In our dissertation, we use WSN to collect environmental data and we also take care of this WSN. In some critical situations, node can be sacrificed to transmit collected data. However, if the node no longer works, no data are collected from the environmental phenomena.

We can see that scenario 1 is unrealistic or a theoretical perfect system. It also does not take care of the Node entity part. Scenario 2 just considers the Flood entity and does not take care of the Node entity, which is a mistake. Scenario 3 is the most realistic, the Flood entity and Node entity are both involved in this scenario. If we deploy a WSN in a real watershed, no doubt that scenario 3 will be considered.

We would like to highlight that the current delay to detect the Flood entity state changes is a little bit long. How to short this delay is an open research issue. For the ongoing work, we will investigate the following issues:

- Thanks to the development of ultra-low power MCU (e.g., ARM Cortex-Mx (TI-CC2650, 2016)), the maximum sample frequency of each wireless sensor node may be increased from 1 min to 1 s for example. Moreover, one or two states will be added (e.g., Flood trend high) in critical wireless sensor nodes. When the Flood entity state changes to Flood trend high threshold, the maximum sample frequency will be adopted to acquire and send the data. With the current implementation, we expect that the flood delay detection will decrease from 14 min to 14 s. To achieve this objective, we will investigate carefully the QoS of the deployed WSN.
- We will also investigate other techniques such neural network (Marina Campolo *et al*., 2003) and fuzzy logic (Q. Li, 2013) to compare the global performance of our proposal in terms of complexity, flood detection and WSN lifetime.

7 Conclusion and Future Works

In this dissertation, we first addressed the challenges of WSN while applying it into IoT paradigm. Context-aware computing is the solution that we have studied. Consequently, this dissertation focuses on the issues of applying WSN on context-aware system in environmental domain. The environmental context aware system studied is a flood risk management system that observes a watershed.

Therefore, based on existing solutions, we have proposed a new formalization of context associated with a design method of context-aware system. We have also implemented some flood context-aware systems. The main advantage of our context formalization is its genericity. It can be applied for multiple purposes. For example, a complex context-aware system can integrate the monitoring of the studied phenomenon (the feature of interest) and the management of the hardware and the software system used to observe it. More generally, our formalization provides a unified way to deal with all the components/entities of a context observation system. This formalization can be used in different application fields such as agriculture, environment, smart care smart home and industry 4.0. To illustrate its use, we have provided:

- A WSN use case application investigating the study of network management,
- An environmental use case focusing on the study of flood events in a watershed,
- An adaptive context-aware use case investigating the study of fusion of two previous use case.

Simulated data from the ideal configurations of a node are used to evaluate our WSN context-aware system. Besides, the Irstea institute has different real historical data related to watershed observation for flood monitoring. These datasets are used to design the environmental context-aware system. Then the two previous systems are merged to propose an adaptive context-aware system.

An architecture for simulation is provided to evaluate the proposed and developed systems using our context formalization. This architecture uses the multi-agent system named JADE to simulate all the interactions between the components of a context-aware system. The Jess rule-based engine is one component of our architecture. It performs the context reasoning phase. JADE and Jess are both Java language tools. The context is modeled by ontologies. We adopt the SSN ontology to build our ontologies network to model the context and build the context schema. JADE also use ontology to model the agent message content. Our JADE ontology is derived from our SSN ontology network. Thanks to our context modeling based on ontologies, we propose to divide the rules-based reasoning phase in two steps:

- Low-level context deduced from raw data,
- High-level context deduced from low-level context.

Different scenarios for this environmental application are proposed taking into account different states and extended wireless sensors reasoning capabilities. Our application is implemented with tools suitable for the limited resources of wireless sensors. The adaptation behavior of WSN is changing the communication frequency. Communication frequency policies are proposed according to different context-aware systems.

Based on the simulation results, we verify the feasibility of our formalization by simulating different context-aware systems. Moreover, the communication frequency adaptations of sensor nodes based on any context entity can help improve the lifetime of the system. Our adaptation approach has limited impact on the Quality of Service (QoS) of the system. No matter which types of nodes, which have the same initial energy level, will have the same communication frequency adaptation. That means that the adaptation of wireless sensor node is made according to its level of energy and the flood management.

In the future, we can imagine a system in which the communication frequencies are different depending of the nodes types, for example:

- Precipitation nodes are one communication per minute,
- Watercourse nodes are one communication every 2 minutes,
- Outlet node is one communication per 3 minutes.

Moreover, the sample frequency and communication frequency may be increased to reduce the flood detection delay (cf. Section 6).

Another scenario can be proposed, in which sensor nodes are more intelligent. Each sensor node is a context-aware system. They can deduce the state of Environmental Entities of Interest (EEI) through the context acquisition, context modeling and context reasoning phases. Sensor node can also deduce Sensor Node Entity of Interest (SNEI) states from its Sensor Node Observable Entities (SNOE) states. The adaptation is based on flood state (EEI state) and node state (SNEI state). In this scenario, sensor nodes make the final decision of adaptation.

In reality, there are massive sensor nodes in context-aware system, the communication channel could be blocked by massive sensor nodes send packets when F entity is RISKCY or FLOOD. However, our formalization is generic, it can add new entity such as Network Channel entity. Network Channel state be part of inference to reduce the energy consumption of WSN.

For the perspective of this work, we would like to investigate the global performance of our approaches by comparing particularly with neural network and fuzzy logic ones.

According to our current implementation, we think that fuzzy logic may be combined with our approaches to improve the flood detection.

7.1 Summary of Publications

1. Sun, J.; De Sousa, G.; Roussey, C.; Chanet, J.P.; Pinet, F. and Hou, K.M., A New Formalisation for Wireless Sensor Network Adaptive Context-aware System: application to an environmental use case, In Proceedings of the Tenth International Conference on Sensor Technologies and Applications SENSORCOMM 2016, 24-28 July 2016, Nice, France, http://www.thinkmind.org/index.php?view=article&articleid=sensorcomm_2016_3_20_10048

2. Sun, J.; De Sousa, G.; Roussey, C.; Chanet, J.P.; Pinet, F. and Hou, K.M., Intelligent Flood Adaptive Context-aware System: How Wireless Sensors Adapt their Configuration based on Environmental Phenomenon Events, Sensors & Transducers, Vol. 206, Issue 11, November 2016, pp.68-81(http://www.sensorsportal.com/HTML/DIGEST/New_Digest.htm)

8 References

Abowd, Gregory D. and Dey, Anind K. and Brown, Peter J. and Davies, Nigel and Smith, Mark and Steggles, Pete. 1999. "Towards a Better Understanding of Context and Context-Awareness" Proceedings of *the 1st International Symposium on Handheld and Ubiquitous Computing HUC'99*. Karlsruhe, Germary. 304–307

Akyildiz, Ian F., Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. 2002. "Wireless Sensor Networks: A Survey." *Computer Networks* 38 (4): 393–422.

Akyildiz, Ian F., Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. 2002b. "A Survey on Sensor Networks." *IEEE Communications Magazine* 40 (8): 102–14.

Aldabagh, Najla Badie, and Ban Sharief Mustafa. 2012. "A Comparative Study between Using OWL Technology and Jess Rule Based for Applying Knowledge to Agent Based System." *International Journal of Computer Science and Information Security* 10 (7): 31.

Anastasi, Giuseppe, M. Conti, M. Di Francesco, and Andrea Passarella. 2006. "How to Prolong the Lifetime of Wireless Sensor Networks." *Mobile Ad Hoc and Pervasive Communications*, 1–26.

Atzori, Luigi, Antonio Iera, and Giacomo Morabito. 2010. "The Internet of Things: A Survey." *Computer Networks* 54 (15): 2787–2805.

Aurora Energy. (http://www.aurorasolarenergy.com/average-daily-sun-hours/), 2012.

BDOH. (https://bdoh.irstea.fr/),2017.

Bellifemine, Fabio, Agostino Poggi, and Giovanni Rimassa. 1999. "JADE–A FIPA-Compliant Agent Framework." In *Proceedings of PAAM*, 99:33. London.

Bellifemine, Fabio, Giovanni Rimassa and Agostino Poggi. 2001. "Developing Multi-Agent Systems with a FIPA-Compliant Agent Framework." *Software-Practice and Experience* 31 (2): 103–128.

Bellifemine, Fabio, Federico Bergenti, Giovanni Caire, and Agostino Poggi. 2005. "JADE—a Java Agent Development Framework." In *Multi-Agent Programming*, 125–47. Springer.

Bendadouche, Rimel, Catherine Roussey, Gil De Sousa, Jean-Pierre Chanet, and Kun Mean Hou. 2012. "Extension of the Semantic Sensor Network Ontology for Wireless Sensor Networks: The Stimulus-WSNnode-Communication Pattern." In *Proceedings of the 5th International Conference on Semantic Sensor Networks-Volume 904*, 49–64. CEUR-WS. org.

Bittencourt, Ig Ibert, Pedro Bispo, Evandro Costa, João Pedro, Douglas Véras, Diego Dermeval, and Henrique Pacca. 2009. "Modeling Jade Agents from Gaia Methodology under the Perspective of Semantic Web." In *International Conference on Enterprise Information Systems*, 780–89. Springer.

Bizer, Christian, Tom Heath, and Tim Berners-Lee. "Linked data-the story so far." *Semantic services, interoperability and web applications: emerging concepts (2009)*: 205-227.

Borst, Willem Nico. 1997. *Construction of Engineering Ontologies for Knowledge Sharing and Reuse*. Universiteit Twente.

"Calculating the Kilowatt Hours Your Solar Panels Produce (Solar Panel Output)." Understand Solar. 22 Mar. 2016. Web. 17 July 2016.

Campolo Marina, Alfredo Soldati and Paolo Andreussi. 2003. "Artificial neural network approach to flood forecasting in the River Arno" Hydrological Sciences Journal, 48:3, 381-398.

Chandrasekaran, Balakrishnan, John R. Josephson, and V. Richard Benjamins. 1999. "What Are Ontologies, and Why Do We Need Them?" *IEEE Intelligent Systems and Their Applications* 14 (1): 20–26.

"Climate and Forecast (CF) Features." Climate and Forecast (CF) Features. Web. 09 Jan. 2017.

Compton, Michael, Payam Barnaghi, Luis Bermudez, RaúL GarcíA-Castro, Oscar Corcho, Simon Cox, John Graybeal, Manfred Hauswirth, Cory Henson, and Arthur Herzog. 2012. "The SSN Ontology of the W3C Semantic Sensor Network Incubator Group." *Web Semantics: Science, Services and Agents on the World Wide Web* 17: 25–32.

Corsar, David, Peter Edwards, Nagendra Velaga, John Nelson, and Jeff Pan. 2011. "Short Paper: Addressing the Challenges of Semantic Citizen-Sensing." In *Proceedings of the 4th International Conference on Semantic Sensor Networks-Volume 839*, 101–6. CEUR-WS. org.

Diao, Xunxing. 2011. "A Resource-Aware Embedded Commucation System for Highly Dynamic Networks." Université Blaise Pascal-Clermont-Ferrand II.

Doshi, Tilak K., Neil Sebastian D'Souza, Linh Nguyen, Teo Han Guan, and Nahim Bin Zahur. 2013. "The Economics of Solar PV in Singapore." *GSTF Journal of Engineering Technology (JET)* 2 (1): 53.

Efstratiou, Christos. 2004. "Coordinated Adaption for Adaptive Context-Aware Applications." Lancaster University.

Friedman, Ernest. 2003. *Jess in Action: Rule-Based Systems in Java*. Manning Publications Co.

Garnier, J., G. Billen, G. Vilain, M. Benoit, P. Passy, G. Tallec, J. Tournebize, Juliette Anglade, C. Billy, and B. Mercier. 2014. "Curative vs. Preventive Management of Nitrogen Transfers in Rural Areas: Lessons from the Case of the Orgeval Watershed (Seine River Basin, France)." *Journal of Environmental Management* 144: 125–34.

Goumopoulos, Christos, Brendan O'Flynn, and Achilles Kameas. 2014. " Automated Zone-Specific Irrigation with Wireless Sensor/Actuator Network and Adaptable Decision Support." *Computers and Electronics in Agriculture* 105: 20–33.

Gray, Alasdair JG, Raúl García-Castro, Kostis Kyzirakos, Manos Karpathiotakis, Jean-Paul Calbimonte, Kevin Page, Jason Sadler, Alex Frazer, Ixent Galpin, and Alvaro AA Fernandes. 2011. "A Semantically Enabled Service Architecture for Mashups over Streaming and Stored Data." In *Extended Semantic Web Conference*, 300–314. Springer.

Gruber, Thomas R. 1993. "A Translation Approach to Portable Ontology Specifications." *Knowledge Acquisition* 5 (2): 199–220.

Guillemin, Patrick, and Peter Friess. 2009. "Internet of Things Strategic Research Roadmap." *The Cluster of European Research Projects, Tech. Rep*.

Gutiérrez, Joaquín, Juan Francisco Villa-Medina, Alejandra Nieto-Garibay, and Miguel Ángel Porta-Gándara. 2014. "Automated Irrigation System Using a Wireless Sensor Network and GPRS Module." *IEEE Transactions on Instrumentation and Measurement* 63 (1): 166–76.

Heathcote, Isobel W. 1998. *Integrated Watershed Management: Principles and Practice*. Taylor & Francis.

Huang, Xiaoci, Jianjun Yi, Shaoli Chen, and Xiaomin Zhu. 2015. "A Wireless Sensor Network-Based Approach with Decision Support for Monitoring Lake Water Quality." *Sensors* 15 (11): 29273–96.

Hwang, Jeonghwang, and Hyun Yoe. 2011. "Study on the Context-Aware Middleware for Ubiquitous Greenhouses Using Wireless Sensor Networks." *Sensors* 11 (5): 4539–61.

Hydro.(http://www.hydro.eaufrance.fr/), 2017.

Indulska, Jadwiga, and Peter Sutton. 2003. "Location Management in Pervasive Systems." In *Proceedings of the Australasian Information Security Workshop Conference on A CSW Frontiers 2003-Volume 21*, 143–51. Australian Computer Society, Inc.

INFSO, D. 2008. “ Networked Enterprise & RFID INFSO G. 2 M icro & Nanosystems, in Co-Operation with the Working Group RFID of the ETP EPOSS, Internet of Things in 2020, Roadmap for the Future [R].” *Information Society and Media, Tech. Rep*.

Jackson, Alex. "Geography AS Notes." Flooding. Web. 09 Jan. 2017.

Janowicz, Krzysztof, and Michael Compton. 2010. “The Stimulus-Sensor-Observation Ontology Design Pattern and Its Integration into the Semantic Sensor Network Ontology.” In *Proceedings of the 3rd International Conference on Semantic Sensor Networks-Volume 668*, 64–78. CEUR-WS. Org.

Kabadayi, Sanem, Adam Pridgen, and Christine Julien. 2006. “Virtual Sensors: Abstracting Data from Physical Sensors.” In *Proceedings of the 2006 International Symposium on on World of Wireless, Mobile and Multimedia Networks*, 587–92. IEEE Computer Society.

Lenzerini, Maurizio. 2002. “Data Integration: A Theoretical Perspective.” In *Proceedings of the Twenty-First ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, 233–46. ACM.

Li, Q. 2013. “Fuzzy approach to analysis of flood risk based on v ariable fuzzy sets and improved information diffusion methods” Nat. Hazards Earth Syst. Sci., 13, 239–249.

Libelium PRO v1.2 2017. L ibelium PRO v1.2 development Website (http://www.libelium.com/v12/development/)

Libelium Smart Metering 2017. L ibelium Smart Metering documentation (http://www.libelium.com/v12/development/waspmote/documentation/smart-metering-board-technical-guide/), 2017.

Lilas, Damien, Charles Perrin, Vazken Andreassian, Laurent Coron, Julien Peschard, Patrick Ansart, Carina Furusho, Cécile Loumagne, and Lionel Berthet. 2012. “ Mise Au Point d’un Modèle de Prévision Des Crues Sur Le Bassin Versant de l’Orgeval.” *Sciences Eaux & Territoires*, no. III: 10–15.

Liu, Yang, Boon-Chong Seet, and Adnan Al-Anbuky. 2013. “An Ontology-Based Context Model for Wireless Sensor Network (WSN) Management in the Internet of Things.” *Journal of Sensor and Actuator Networks* 2 (4): 653–74.

Maciag, Timothy, and Daryl H. Hepting. "Constructing collaborative online communities for visualizing spimes." *Web Intelligence and I ntelligent Agent Technology (WI-IAT), 2010 IEEE/WIC/ACM International Conference on. Vol. 3. IEEE*, 2010.

Merriam-Webster. Merriam-Webster. Web. 09 Jan. 2017.

(MESIA, 2014). “A Guide to Solar Power Generation in the United Arab Emirates” http://www.mesia.com/wp-content/uploads/Solar%20for%20Students.pdf

Moraru, Alexandra, Dunja Mladenic, Matevz Vucnik, Maria Porcius, Carolina Fortuna, and Mihael Mohorcic. 2011. “Exposing Real World Information for the Web of Things.” In *Proceedings of the 8th International Workshop on Information Integration on the Web: In Conjunction with WWW 2011*, 6. ACM.

"OWL." OWL - Semantic Web Standards. Web. 28 June 2017.

OWL. https://www.w3.org/OWL/ , 2017.

"Orgeval Basin: 50 Years of Hydrological Studies." Orgeval Basin: 50 Years of Hydrological Studies | Irstea. Web. 25 Jan. 2017.

Othman, Mohd Fauzi, and Khairunnisa Shazali. 2012. “Wireless Sensor Network Applications: A Study in Environment Monitoring System.” *Procedia Engineering* 41: 1204–1210.

Perera, Charith, Arkady Zaslavsky, Peter Christen, and Dimitrios Georgakopoulos. 2014. “Context Aware Computing for the Internet of Things: A Survey.” *IEEE Communications Surveys & Tutorials* 16 (1): 414–54.

"Perlman, Howard. 2017. “What is a watershed?”, https://water.usgs.gov/edu/watershed.html"

"Raw Data." Wikipedia. Wikimedia Foundation. Web. 15 Jan. 2017.

Reddy, Adi Mallikarjuna V., AVU Phani Kumar, D. Janakiram, and G. Ashok Kumar. 2009. “Wireless Sensor Network Operating Systems: A Survey.” *International Journal of Sensor Networks* 5 (4): 236–55.

RDF Schema 1.1. Web. 28 June 2017.

RDF Schema 1.1. https://www.w3.org/TR/rdf-schema/,2017.

"RDFS vs. OWL." http://www.cambridgesemantics.com/semantic-university/rdfs-vs-owl

Roussey, Catherine, Vincent Soulignac, Jean-Claude Champomier, Vincent Abt, and Jean-Pierre Chanet. 2010. “Ontologies in Agriculture.” In *AgEng 2010, International Conference on Agricultural Engineering*, p.-. Cemagref.

Sanchez, Luis, Jorge Lanza, Rasmus Olsen, Martin Bauer, and Marc Girod-Genet. 2006. “A Generic Context Management Framework for Personal Networking Environments.” In *Mobile and Ubiquitous Systems-Workshops, 2006. 3rd Annual International Conference On*, 1–8. IEEE.

Schilit, Bill, Norman Adams, and Roy Want. 1994. “Context-Aware Computing Applications.” In *Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop On*, 85–90. IEEE.

"Sensor." Wikipedia. Wikimedia Foundation. Web. 16 Jan. 2017.

"Sensor Web Enablement (SWE)." Sensor Web Enablement (SWE) | OGC. Web. 25 July. 2017

Shi, Hong-Ling, Kun Mean Hou, Hai-Ying Zhou, and Xing Liu. 2011. “Energy Efficient and Fault Tolerant Multicore Wireless Sensor Network: E^2MWSN.” In *Wireless Communications, Networking and Mobile Computing (WiCOM), 2011 7th International Conference On*, 1–4. IEEE.

"Solar Electric Photovoltaic Modules." Solar PV Modules. Web. 09 Jan. 2017.

"Solar Panel 7.4V." Solar Panel 7.4V. Web. 09 Jan. 2017.

Sterling, Bruce. "When blobjects rule the earth." *Keynote at SIGGRAPH 4* (2004): 401-450.

Sterling Bruce. Shaping Things (Mediaworks Pamphlets)[J]. 2005.

Studer, Rudi, V. Richard Benjamins, and Dieter Fensel. 1998. “Knowledge Engineering: Principles and Methods.” *Data & Knowledge Engineering* 25 (1–2): 161–97.

Subercaze, Julien, and Pierre Maret. 2011. “Programming Semantic Agent for Distributed Knowledge Management.” *Semantic Agent Systems*, 47–65.

Sun, Jie, Gil De Sousa, Catherine Roussey, Jean-Pierre Chanet, François Pinet, and K. M. Hou. 2016. “A New Formalisation for Wireless Sensor Network Adaptive Context-Aware System: Application to an Environmental Use Case.” In *Tenth International Conference on Se nsor Technologies and Applications SENSORCOMM 2016*, 49–55.

Terrasson, Guillaume, Renaud Briand, Skandar Basrour, Valérie Dupé, and Olivier Arrijuria. 2009. “Energy Model for the Design of Ultra-Low Power Nodes for Wireless Sensor Networks.” *Procedia Chemistry* 1 (1): 1195–98.

Thomas S. The end of cyberspace and other surprises[J]. *Convergence*, 2006, 12(4): 383-391.

Toma, Ioan, Elena Simperl, and Graham Hench. 2009. “A Joint Roadmap for Semantic Technologies and the Internet of Things.” In *Proceedings of the Third STI Roadmapping Workshop, Crete, Greece*. Vol. 1.

Tomaiuolo, Michele, Paola Turci, Federico Bergenti, and Agostino Poggi. 2006. “An Ontology Support for Semantic Aware Agents.” *Agent-Oriented Information Systems III*, 140–53.

“TI-CC2650, 2016” Texas Instruments, CC2650 SimpleLink™ Multistandard Wireless MCU, swrs158b – February 2015 – revised July 2016.

Wang, Andrew Y., and Charles G. Sodini. 2004. “A Simple Energy Model for Wireless Microsensor Transceivers.” In *Global Telecommunications Conference, 2004. GLOBECOM’04. IEEE*, 5:3205–9. IEEE.

Wenzel, Ken, Jörg Riegel, Andreas Schlegel, and Matthias Putz. 2011. “Semantic Web Based Dynamic Energy Analysis and Forecasts in Manufacturing Engineering.” *Glocalized Solutions for Sustainability in Manufacturing*, 507-512.

"What Is a Watershed?" What Is a Watershed - Primrose Creek Watershed Association. Web. 09 Jan. 2017.

"Wireless Sensor Network." Wikipedia. Wikimedia Foundation, 23 July 2017. Web. 25 July 2017.

W3C. Web. 28 June 2017.

W3C. https://www.w3.org/standards/semanticweb/ontology ,2017

Yick, Jennifer, Biswanath Mukherjee, and Dipak Ghosal. 2008. “Wireless Sensor Network Survey.” *Computer Networks* 52 (12): 2292–2330.

Yu, Liang, and Yong Liu. 2015. “Using Linked Data in a Heterogeneous Sensor Web: Challenges, Experiments and Lessons Learned.” *International Journal of Digital Earth* 8 (1): 17–37.

Résumé

Système sensible et adaptatif au contexte pour la gestion intelligente de crues

A l'avenir, l'agriculture et l'environnement vont pouvoir bénéficier de plus en plus de données hétérogènes collectées par des réseaux de capteurs sans fil (RCSF). Ces données alimentent généralement des outils d'aide à la décision (OAD). Dans cette thèse, nous nous intéressons spécifiquement aux systèmes sensibles et adaptatifs au contexte basés sur un RCSF et un OAD, dédiés au suivi de phénomènes naturels. Nous proposons ainsi une formalisation pour la conception et la mise en œuvre de ces systèmes. Le contexte considéré se compose de données issues du phénomène étudié mais également des capteurs sans fil (leur niveau d'énergie par exemple). Par l'utilisation des ontologies et de techniques de raisonnement, nous visons à maintenir le niveau de qualité de service (QdS) des données collectées (en accord avec le phénomène étudié) tant en préservant le fonctionnement du RCSF. Pour illustrer notre proposition, un cas d'utilisation complexe, l'étude des inondations dans un bassin hydrographique, est considéré. Cette thèse a produit un logiciel de simulation de ces systèmes qui intègre un système de simulation multi-agents (JADE) avec un moteur d'inférence à base de règles (Jess).

Mots-clefs : ontologies, inférences à base de règles, formalisation, données hétérogènes, intégration de données issues de capteurs, RCSF, ressources limitées, OAD, systèmes sensibles et adaptatifs au contexte, QdS, agriculture, environnement.

Abstract

Intelligent Flood Adaptive Context-aware System

In the future, agriculture and environment will rely on more and more heterogeneous data collected by wireless sensor networks (WSN). These data are generally used in decision support systems (DSS). In this dissertation, we focus on adaptive context-aware systems based on WSN and DSS, dedicated to the monitoring of natural phenomena. Thus, a formalization for the design and the deployment of these kinds of systems is proposed. The considered context is established using the data from the studied phenomenon but also from the wireless sensors (e.g., their energy level). By the use of ontologies and reasoning techniques, we aim to maintain the required quality of service (QoS) level of the collected data (according to the studied phenomenon) while preserving the resources of the WSN. To illustrate our proposal, a complex use case, the study of floods in a watershed, is described. During this PhD thesis, a simulator for context-aware systems which integrates a multi-agent system (JADE) and a rule engine (Jess) has been developed.

Keywords: ontologies, rule-based inferences, formalization, heterogeneous data, sensors data streams integration, WSN, limited resources, DSS, adaptive context-aware systems, QoS, agriculture, environment.