# Data Annotation and Ontology Provisioning for Semantic Applications in Virtualized Wireless Sensor Networks

Rifat Jafrin

A Thesis in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Computer Science
at Concordia University
Montréal, Québec, Canada

November 2015

# CONCORDIA UNIVERSITY
# SCHOOL OF GRADUATE STUDIES

This is to certify that the thesis prepared

By: **Rifat Jafrin**

Entitled: "**Data Annotation and Ontology Provisioning for Semantic Applications in Virtualized Wireless Sensor Networks**" and submitted in partial fulfillment of the requirements for the degree of

**Master of Computer Science**

Complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair
Dr. Tiberiu Popa

_____ Examiner
Dr. Juergen Rilling

_____ Examiner
Dr. Joey Paquet

_____ Supervisor
Dr. Roch Glitho

Approved by: _____
Chair of Department or Graduate Program Director

_____2015

_____
Dr. Amir Asif, Dean
Faculty of Engineering and Computer Science

# ABSTRACT

## Data Annotation and Ontology Provisioning for Semantic Applications in Virtualized Wireless Sensor Networks

Rifat Jafrin

In recent years, virtualization in Wireless Sensor Networks (WSNs) has become very popular for many reasons including efficient resource management, proper sharing and using the same WSN physical infrastructure by multiple applications and services. Semantic applications are very much pertinent to provide situational awareness to the end-users. Incorporating semantic applications in the virtualized WSNs can play a crucial role in providing contextual information to understand the situation, increase usability and interoperability. However, provisioning of semantic applications in virtualized WSNs remains as a big challenge. The reason is the data collected by the virtual sensors needs to be annotated in-network, and the pre-requisite of the data annotation process is to have an ontology that needs to be provisioned, i.e., developed, deployed and managed. Unfortunately, annotating sensor data and ontology provisioning in virtualized WSNs is not straightforward because of limited resources of sensors, on-demand creation of virtual sensors, and unpredictable lifetime. As the existing researches do not consider data annotation in virtualized WSN infrastructure level, these solutions are domain specific and lack of providing support for multiple applications. Moreover, the major drawback of the current ontology provisioning mechanisms requires domain experts to develop, deploy, and manage the ontologies in WSNs. This thesis aims to propose a solution for provisioning of multiple semantic applications in the virtualized WSNs.

The main contribution of this thesis is twofold. First, we have proposed an architecture to annotate sensor data in the virtualized WSN infrastructure and defined an ontology in sensor domain to perform data annotation. Second, we have proposed an architecture for provisioning ontology in the virtualized WSNs that consists of an ontology provisioning center, an ontology-enabled virtualized WSN, and an ontology deployment protocol. The proposed architectures use overlay network as a foundation. We have built a proof-of-concept prototype for a semantic wildfire monitoring application in the cloud environment using the Google App Engine. In order to evaluate the viability of the proposed architecture, we have made performance measurement of the implemented prototype. We ran a simulation to justify our proposed ontology provisioning protocol.

# Acknowledgments

Alhamdulillah, all praises be to almighty Allah, who guided me throughout this research and beyond. Only due to his blessings I could finish my thesis.

I am deeply indebted to my supervisor, Dr. Roch Glitho, for the continuous support of my master's study and research. I thank Dr. Glitho for his continuous guidance, friendly advice during the project, motivation, immense knowledge and for always being fair. I could not have imagined having a better supervisor and mentor for my master study.

I would also like to express my gratitude and appreciations to Dr. Juergen Rilling and Dr. Joey Paquet for their insightful comments and constructive criticisms of this work. I am also grateful to Dr. Tiberiu Popa for chairing the thesis defense session.

I would like to take this opportunity to thank my colleagues at Concordia's lab for their help, cooperation, and encouragement. First, I am thankful to Dr. Imran Khan for his essential advice and suggestions on different aspects of my project. I would like to express my deepest appreciation to Carla Mouradian and Dr. Jagruti Sahoo for their continual supports and helping me on this journey.

I am grateful to Dr. Roch Glitho and Concordia University for their financial supports.

Last but not the least, my deepest thanks to my dear husband for his unconditional love, unfailing support, continuous encouragement, and sacrifice throughout my years of study. Finally, my deepest thanks and greatest love to my parents for always encouraging me, believing on my potentials, and keeping me in their prayers.

# Contents

## List of Figures

# List of Tables

## Acronyms and Abbreviations

| | |
|---|---|
| WSN | Wireless Sensor Networks |
| NIST | National Institute of Standards and Technology |
| IaaS | Infrastructure as a Service |
| PaaS | Platform as a Service |
| SaaS | Software as a Service |
| W3C | World Wide Web Consortium |
| RDF | Resource Description Framework |
| RDFS | Resource Description Framework Schema |
| VM | Virtual Machines |
| OS | Operating System |
| VSN | Virtual Sensor Network |
| OWL | Web Ontology Language |
| BBC | British Broadcasting Corporation |
| SPARQL | Simple Protocol and RDF Query Language |
| SQL | Structured Query Language |
| URI | Uniform Resource Identifier |

| | |
|---|---|
| HTML | Hyper Text Markup Language |
| XML | Extensible Markup Language |
| CO2 | Carbon dioxide |
| SSW | Semantic Sensor Web |
| SWE | Sensor Web Enablement |
| OGC | Open Geospatial Consortium |
| SWRL | Semantic Web Rule Language |
| S-SOS | Semantic Sensor Observation Service |
| O&M | Observation and Measurement |
| SML | Sensor Model Language |
| LOD | Linked Open Data cloud |
| SOA | Service Oriented Architectures |
| SenML | Sensor Markup Language |
| KB | Knowledge Base |
| IoT | Internet of Things |
| SNMP | Simple Network Management Protocol |
| ANMP | Ad Hoc Network Management Protocol |

| | |
|---|---|
| HTTP | Hypertext Transfer Protocol |
| GA | Genetic Algorithm |
| P2P | Peer-to-Peer |
| REST | REpresentational State Transfer |
| JSON | JavaScript Object Notation |
| API | Application Programming Interface |
| HTML | Hyper Text Mark-up Language |
| GAE | Google Apps Engine |
| URL | Uniform Resource Locator |

# Chapter 1

## 1 Introduction

In this chapter, we define the key concepts related to our research domain including Virtualized Wireless Sensor Network, Semantic Applications, and Data Annotations in the first section. The second section presents the motivation of our research and states the problem we have solved. Thesis contributions are summarized in the third section. Finally, we introduce the thesis organization in the fourth section.

### 1.1 Definitions

In this section, we discuss few key concepts related to this thesis.

#### 1.1.1 Virtualized Wireless Sensor Network

We first introduce two important concepts to define virtualized Wireless Sensor Network: Cloud Computing and Virtualization.

##### *1.1.1.1 Cloud Computing*

There are several well-known definitions for cloud computing. According to the definition proposed in [1] clouds are "*large pool of easily usable and accessible virtualized resources that can be dynamically reconfigured to adjust to a variable load (scale), allowing for an optimum resources utilization.*"

Another definition from National Institute of Standards and Technology (NIST), "*Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal*

*management effort or service provider interaction*". These resources can be networks, servers, storage, applications, and services [2].

Cloud computing consists of three layers: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). The IaaS is the lowest layer comprising of physical and virtualized computing resources. The PaaS is the second layer constructed over the infrastructure layer which offers a software platform to develop and manage applications. The third layer is SaaS which offers applications residing in the cloud.

#### 1.1.1.2 Virtualization

Virtualization is a process of generating abstraction of the physical computing resources into the logical units for better resource utilization by multiple users [3]. System virtualization encapsulates the software layer by covering underline physical operating system and provides the same functionalities and behaviors like the actual physical hardware could provide [4].

Wireless Sensor Network (WSN) consists of sensors that have the ability to sense environmental phenomena. However, traditional WSNs are domain specific and dedicated for a particular application. Virtualized WSNs can be considered as an infrastructure (IaaS) that consists of virtual sensors. Virtual sensors are created on top of physical sensors by applying virtualization technique. As a result, multiple application tasks can run concurrently on top of the physical sensor without interfering each other.

### 1.1.2 Semantic Applications

Semantic applications are mostly web based applications and developed using the technologies of the W3C Semantic Web. Semantic Web technologies are being used in various application domains

because of their usability and usefulness. One of the application domain where Semantic Web technologies used in WSN. The main benefit of incorporating Semantic Web in WSN applications is that it provides more powerful representation, advanced access, and formal analysis of sensor data [5]. As a result, those applications get the high-level details of the events monitored by the sensors and infer additional knowledge to gain situational awareness. In this thesis, our focus is to provision semantic applications in a virtualized WSNs. We introduce Semantic Web and few related concepts for the better understanding of the semantic applications.

#### *1.1.2.1 Semantic Web*

Second generation World Wide Web (Web 2.0) depends on the human capability to pool resources and shares information online. The semantic web is an extension of the current Web 2.0 which provides an easier way to find, share, reuse, and combine information. According to the W3C, "*The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries*" [6].

The Semantic Web offers a common format of data of various combinations or different formats which are drawn from diverse sources. This common format helps to manipulate data regarding the real world objects and allows a machine to understand the data format, search, and aggregate those data without taking any help from the human operator. We define two important key concepts (Semantic data annotation and Ontology) of Semantic Web in next sections.

#### 1.1.2.2 Semantic Data Annotation

Semantic Web puts metadata (additional data) to the existing data to process them automatically by the machines. The mechanism of generating metadata with the actual data is called annotation, and

the procedure of creating semantic metadata is called Semantic Data Annotation. It allows users to get the high-level details and expand the existing data by tagging with semantic descriptions [7]. RDF (Resource Description Framework) is a flexible approach for representing basic data into a structured data.

#### 1.1.2.3 Ontology

Ontology is one of the most important attributes of the Semantic Web technology. Ontology can be defined as a vocabulary which describes a set of concepts and explains the relationships between the concepts in a particular area of concern. There are many formal definitions of ontology. The most cited definition is given in [8] and according to the author *" An ontology is an explicit specification of a conceptualization"*.

Ontology offers a standard mechanism to understand some particular domains. It represents a real world phenomenon in a machine-understandable format. A set of concepts, ideas, and information from a particular field is classified with a defined relationship by an ontology that helps to understand the field better.

### 1.1.3 Sensor Data Annotation

Sensors are becoming very popular and adopted by many different domains, especially in those applications which are dedicated to providing real-time services [7]. In some real-time monitoring applications, it is critical to understand the analogous situations. However, raw sensor data are simply not enough to provide knowledge on the particular situation. We need additional metadata along with the sensor data to understand the situations properly. Metadata provides additional data such as description, contextual information related to the actual data. Sensor data can be enriched

by annotating them with semantic concepts. The process of annotating sensor data is called sensor data annotation.

## 1.2 Motivation and Problem Statement

Wireless Sensor Networks (WSN) are becoming ubiquitous and being used by many applications to monitor different events. WSNs consist of tiny devices that allow applications to observe various physical phenomena. However, traditional WSN deployments are usually tailored for predefined applications with no possibility for new applications to use them concurrently. For this reason, WSN virtualization which uses the concept of concurrent application tasks running on a sensor node and combines such nodes together to work for multiple applications simultaneously has gained considerable attention [11]–[13].

However, classical virtualized WSNs provide sensor data in raw format. As a result, these application cannot interpret the sensor data or understand its context completely. For example, a traditional fire monitoring application can only receive a simple fire notification without additional details which could help its user understand the meanings and context of the fire event (e.g., event status and the location of the fire event). On the other hand, semantic applications are becoming very popular in the WSN domain as they provide high-level details of the events monitored by sensors with additional knowledge. For example, a semantic fire monitoring application allows its users making queries such as "*what is the current status of the fire?"* and "*what is the current location of the fire?"* Virtualized WSNs typically monitor several real-time events simultaneously for different applications. Few end-users of these applications may wish to know the context of the

specific events. These requests bring us to the need for a mechanism that annotates the sensor data semantically in a virtualized WSN.

Semantic annotation of heterogeneous sensor data generates a standard data format. It allows multiple semantic applications to share, reuse, search, and exploration of sensor data without any prior knowledge of the data source. However, semantic data annotation requires the domain concepts and the relationships among them. An ontology is used to represent a domain formally with its concepts, and the relationships among the concepts. The main advantage of data annotation in the virtualized WSN is that it allows the same virtualized WSN infrastructure to be used by several applications and enables users to understand the context of the event captured by the sensors.

Semantic data annotation in IaaS depends on an ontology provisioning in the network level. During the operation stage, it is natural that sensors with new capabilities are added to the network. This mandates an update to the underlying ontology in the network. These throw several research challenges including (a) how the ontology will be developed, (b) how the ontology will be deployed, and (c) how the ontology will be managed. We need a mechanism for ontology provisioning to address these challenges.

## 1.3 Thesis Contributions

The contributions of this thesis are as stated below:

- A set of requirements on the data annotation architecture for semantic applications and ontology provisioning in the virtualized WSNs are defined.
- A review of the state-of-the-art solutions for sensor data annotation and ontology provisioning mechanism in WSN with respect to the defined requirements is presented.
- A Data Annotation Architecture for Semantic Applications in Virtualized Wireless Sensors Networks is proposed to allow standard representation of raw sensor data by annotating them with semantic concepts.
- An ontology is defined for sensor data annotation which can be extended based on the application domains in the SaaS layer.
- An architecture for ontology provisioning in virtualized WSNs is proposed which allows ontology development, deployment, and management with ease without the requirement of a domain expert. An ontology provisioning center and an ontology provisioning protocol are also defined as part of the architecture.
- A prototype of the proposed architecture is implemented and evaluated using different performance metrics. A simulation is run to justify the proposed approaches for the ontology provisioning protocol.

## 1.4 Thesis Organization

The rest of the thesis is organized as follows:

**Chapter 2** provides the background knowledge about the key concepts related to our research domain.

**Chapter 3** presents the scenario and a set of requirements derived from the scenario. The state-of-the-art solutions are reviewed and evaluated on the set of defined requirements.

**Chapter 4** describes the proposed data annotation architecture for semantic applications in Virtualized WSN and proposes a new ontology (we refer to 'base ontology') for sensor data annotation.

**Chapter 5** proposes an ontology provisioning architecture that contains (a) an ontology provisioning center for the development and management of the base ontology in the virtualized WSN, and (b) an ontology deployment protocol for the interaction between the ontology provisioning center and virtualized WSN.

**Chapter 6** describes the implementation architecture and prototype solution as a proof-of-concept. This chapter also includes performance measurement of the implemented prototype and simulation results.

**Chapter 7** concludes the thesis by highlighting the summary of the overall contributions and identifying the future research directions.

# Chapter 2

## 2 Background

This chapter describes the background information related to our research domain. The topics which are covered in this chapter are Virtualized Wireless Sensor Network, Semantic Applications, and Data Annotation.

### 2.1 Virtualized Wireless Sensor Network

This section presents a basic overview of the virtualized WSN. There are two important concepts regarding the virtualized WSN: Cloud Computing and Virtualization. We first describe the concepts of Cloud Computing and Virtualization. We define virtualized WSN based on these two concepts.

#### 2.1.1 Cloud Computing

This section illustrates a basic overview of cloud computing by defining a brief definition of the cloud computing, cloud layers, and different cloud types.

##### 2.1.1.1 Definition of Cloud Computing

Cloud computing is the outcome of the progression and implementation of existing technologies and standards. One of the main advantages of cloud computing is that it permits users to take advantages of the existing technologies without having in-depth knowledge or expertise related to these fields [12].

Several definitions have been discussed in [1]. After analyzing all those definitions and considering the cloud features, the authors propose an integrative definition. They define cloud as:

"*Large pool of easily usable and accessible virtualized resources (such as hardware, development platforms, and/or services). These resources can be dynamically reconfigured to adjust to a variable load (scale), allowing also for an optimum resource utilization. This pool of resources is typically exploited by a pay per-use model in which guarantees are offered by the Infrastructure Provider by means of customized SLAs"*.

#### *2.1.1.2 Cloud Layers*

Cloud computing services are divided into three layers: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS).

- ***Infrastructure-as-a-Service (IaaS)*** provides infrastructural resources on demand such as servers, computation, storage, network, and communication [13]. Examples of IaaS include Amazon Web Services EC2, GoGrid, Flexiscale.

- ***Platform as a Service (PaaS)*** provides a platform (operating system, software framework) to create, manage, and deploy applications and services. Examples of PaaS are Google App Engine, Microsoft Windows Azure.

- ***Software-as-a-Service (SaaS)*** provides on-demand application to the end user through the internet. Users do not need to install the applications on their local machine as they can access those services through a web portal. Examples of SaaS are Salesforce.com, Rackspace.

#### *2.1.1.3 Types of Clouds*

Cloud can be classified into the following categories based on their usabilities such as public cloud, private cloud, community cloud, or hybrid cloud [14], [15], [25].

- **Private Cloud** is designed for a single organization. It can be developed and managed by the same organization that uses it or by any third-party organization [14].
- **Public Cloud** provides services to the general public as pay-per-use manner. No initial capital investment in infrastructure is required for the public cloud.
- **Community Cloud** is a multi-tenant infrastructure shared by several organizations from a particular community and set up for their specific requirements [15].
- **Hybrid Cloud** is a combination of private and public clouds. It handles the limitations of both cloud by utilizing the advantages of both public and private clouds. [14].

### 2.1.2 Virtualization

We describe virtualization by defining the concept as well as discussing the advantages of virtualization in this section.

#### 2.1.2.1 Definition of Virtualization

Virtualization is a process of creating a virtual resource from physical resources such as a server, storage device, network or an operating system. It separates the virtual resources from the underlying physical resources by creating an abstract layer in between the computing hardware and

the applications running on it [17]. Virtualization permits efficient physical resource utilization by creating several logical instances on a single hardware that can be used by multiple independent users. As a result, virtualization reduces cost and complexity in order to manage the physical resources [3]. For example, in virtualized computing environment, several operating systems can be installed on a single hardware. In such situation, virtualization creates multiple Virtual Machines (VM) on the physical machine. Each of the virtual machines ran on different operating systems isolated from each other. These operating systems are called Guest operating systems. The software layer which is responsible for creating VM is called Hypervisor or Virtual Machine Monitor. The physical device where virtualization takes place is known as host machine while the virtual machine is known as Guest machine.

#### 2.1.2.2 Advantages of Virtualization

The advantages of virtualization are methodically explained [26] by differentiating the scenario of before virtualization and after virtualization.

Without virtualization:

- Each machine holds a single OS image.
- Software and hardware are strongly coupled.
- If several applications are run on the same machine, often it creates conflict.
- Resources are not utilized properly.
- Inflexible and costly infrastructure.

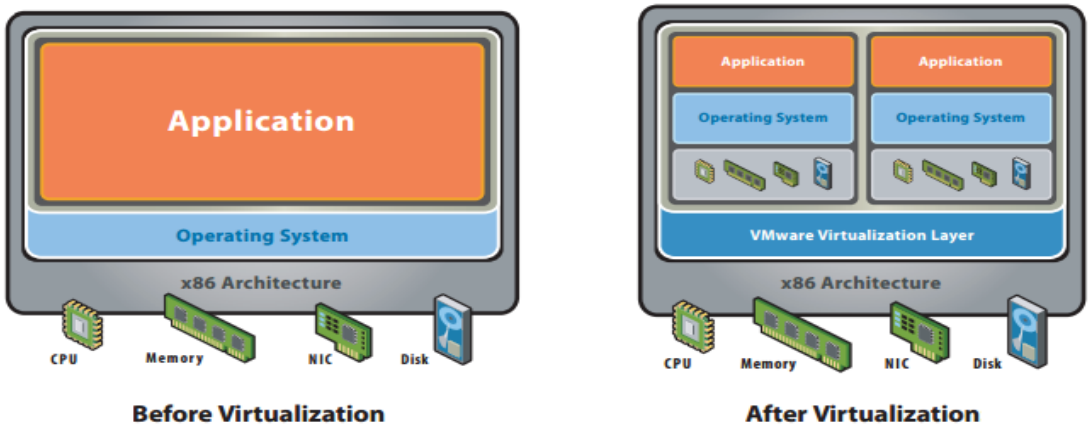


**Figure 2-1 Before and after virtualization [17]**

After Virtualization:

- Operating system and applications become hardware-independent.
- Virtual machines can be provisioned to any system.
- Become easier to manage the OS and application as a single unit by encapsulating them into virtual machines.

### 2.1.3 Virtualized WSNs

The Wireless Sensor Networks (WSNs) are now ubiquitous and using in the diverse application domains. A WSN can be considered as an Infrastructure-as-a-Service which consists of a vast number of sensor nodes deployed in a geographical area to detect various kinds of phenomena. The main limitation of traditional WSNs is that they are domain specific and devoted for a particular application. Virtualized WSN consists of virtual sensors as virtualization creates multiple virtual sensors on top of the physical sensor. Incorporating virtualization in the traditional WSNs allows

proper resource utilization of WSN deployment. As a result, multiple applications can be provisioned over the same virtualized WSN.

In the next subsections, we describe some benefits of virtualized WSNs and categories of WSNs virtualization. Finally, we discuss some applications that can utilize virtualized WSNs.

### 2.1.4 Benefits of Virtualized WSNs

Virtualized WSNs have several benefits over the traditional WSNs [18] [19]. A summary of the benefits of using virtualized WSNs is mentioned below:

- Virtualized WSNs allow multiple applications to share the same physical WSN infrastructure.
- As multiple applications can be provisioned over the same deployed WSN infrastructure, virtualized WSNs reduce the cost and complexity of redundant deployment [18].
- As virtualization in traditional WSNs creates an abstract layer over the physical network, virtualized WSNs eliminate the tight coupling between the WSN services and WSN deployments [19].
- Infrastructure provider of virtualized WSNs can resell the physical infrastructure to the third party to reuse the physical infrastructure [18].
- Virtualized WSNs offer scalability and flexibility to the network infrastructure.
- Virtualized WSNs increase business profitability as the same physical infrastructure is used by multiple applications and services [18].

### 2.1.5 Types of WSN Virtualization

WSN virtualization can be categorized into two types: Node-level virtualization and Network-level virtualization. In the following subsections, we describe each of them.

#### 2.1.5.1 Node-Level Virtualization

Node level virtualization is attained by isolating and partitioning the physical resources. The hardware resources of a physical node are partitioned into slices and allocated for a virtual node [20]. In WSN domain, node-level virtualization creates virtual nodes on a single physical sensor node. As a result, multiple applications can run their tasks simultaneously on different virtual nodes over the same sensor node [21]. Figure 2-2 shows the idea of WSN node-level virtualization.

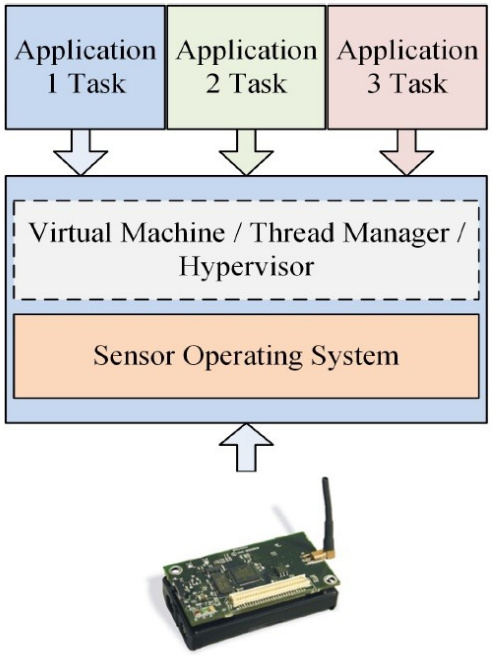


**Figure 2-2 Multiple application tasks running in a general purpose sensor node [19]**

Node-level virtualization can be attained by sequential or simultaneous execution. In sequential execution, application tasks execute in sequence basis (first come first serve). However, in simultaneous execution, there is a fixed time slot for each application task's execution process resulting the frequent switching of execution of tasks from one application to another one [19].

#### 2.1.5.2 Network-Level Virtualization

WSN network-level virtualization can be achieved by creating Virtual Sensor Network (VSN). A VSN can be created by establishing a logical connection among a subset of WSN's nodes. The subset of sensor nodes that create VSN is devoted to a certain task or an application at a given time [22]. In traditional WSN deployment, the whole network is dedicated to one application or service with little possibility to reuse the physical resource. However, in VSN, only a subset of physical nodes is committed to a certain task whereas the remaining nodes are available for other applications. As a result, physical resources are properly utilized and the overall cost and complexity of physical WSN deployment are reduced. There are two options to form WSN network-level virtualization [19].

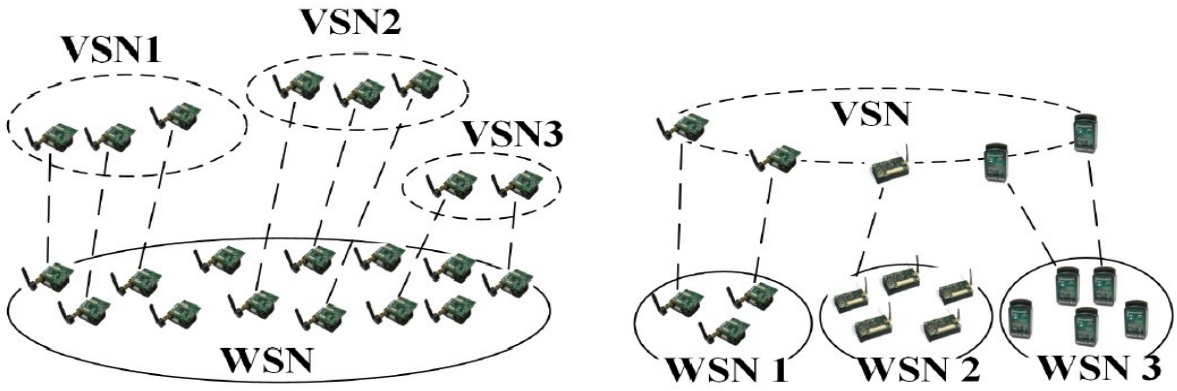


**Figure 2-3 VSN formation options [19]**

First, multiple VSNs can be formed on top of the same underlying WSN infrastructure (Figure 2-3:a). In the second choice, a single VSN can be formed by combining a subset of physical sensor nodes belonging to different physical WSNs (Figure 2-3: b).

### 2.1.6 Applications of Virtualized WSNs

The WSNs are now used in various application domains such as military, environment monitoring, smart home application, medical science and health care, traffic control, and car parking applications. WSN virtualization concept can be applied to these application areas.

The application of battlefield surveillance [23] uses various wireless sensors to detect and classify target objects (for example, civilians, enemies, soldiers, tanks, and animal). In this application, virtualized WSNs can play a crucial role by executing multiple tasks and detecting different objects by creating multiple VSNs on the same WSN infrastructure.

Recently WSNs are widely used to several real-time environmental monitoring applications such as weather monitoring, environmental disaster detection, wildfire detection, early flood detection, early earthquake warning applications. Instead of redundant WSN deployments, virtualized WSNs can be used where each sensor node can run multiple tasks on it and provides services to multiple applications. As a result, both cost and complexity are reduced in virtualized WSN approach.

Several promising scenarios are presented in Sensing-as-a-Service (SaaS) model [24] where virtualized WSN can enrich the SaaS model by proper physical resource utilization. In a smart home automation system [18], virtualized WSN allows to monitor and control different services of smart home remotely through the internet.

## 2.2 Semantic Applications

Applications integrating the tools and technologies of the W3C Semantic Web such as RDF, OWL, and other metadata standards are defined as Semantic Applications. Semantic applications are becoming very popular and using in different domains such as life Science, Sensor Domain, Supply Chain Management, Media Management, Data Integration in Oil and Gas, Web Search, and Ecommerce [25].

Medical and life science domain deals with diverse types of data about drugs, patients, diseases, proteins, cells, and pathways. Incorporating semantic technology in this domain facilitates proper data aggregation on different medicines and illnesses that have multiple names in various parts of the world. Semantic Web Health Care and Life Sciences Interest Group (HCLS IG) [26] provides support in Bioinformatics such as health care, life sciences, clinical research, and translational medicine.

Semantic Sensor Web is the outcome of combining Sensor applications and Semantic Web technologies. Sensors description and observation are encoded using Semantic Web languages. This allows more meaningful representation, advanced access, and formal analysis of sensors resources and data. A semantically enrich sensor network permits the sensor data to be structured, managed, queried, understood and controlled by adding semantics to the sensor data.

A well-known pharmaceutical producer Biogen Idec. manages its global supply chain by using Semantic Web technologies [27]. Many media companies such as British Broadcasting Corporation (BBC), Time Inc., Elsevier, and the Library of Congress use Semantic Web technology to manage their media contents. In 2010, BBC managed its World Cup website by utilizing the Semantic

Web technologies which were reported on SemanticWeb.com [25]. Joost online television service uses Semantic Web service to enable Joost users to understand the relationships between media contents and allow them to find their favorite media content.

For a better understanding of Semantic Application, we describe Semantic Web and some other related key concepts in the next subsection. We first discuss Semantic Web and then we describe ontology. Next, we describe SPARQL query language and semantic reasoning. Finally, we discuss the benefits of the Semantic Web.

### 2.2.1 Semantic Web

The Idea of Semantic Web (also known as Web 3.0) was invented by Tim Berners-Lee, the inventor of the World Wide Web and the director of the World Wide Web Consortium ("W3C"). According to Berners-Lee, the Semantic Web is "*A web of data that can be processed directly and indirectly by machines*" [28].

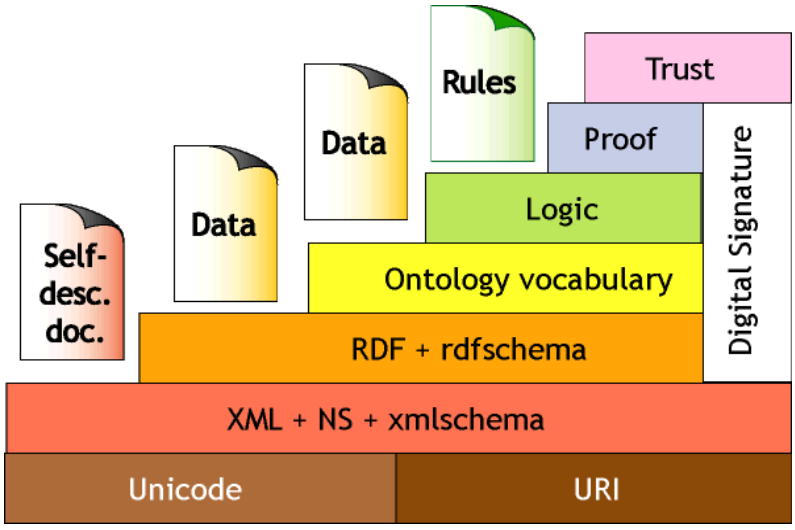


**Figure 2-4 Semantic Web architecture [28]**

The Semantic Web can be considered as a network of information linked up in such a way that it can be easily processed by machines. It is then considered as a web of data that enables data to be linked from an originating source to any other sources and understood by computers [29]. The main difference between Semantic Web technologies and other data related technologies (e.g., relational databases or the World Wide Web) is that the Semantic Web is concerned with the meaning of data whereas other technologies are concerned with the structure of data. Semantic Web provides a common format of data drawn from different sources. It also provides a common language that relates data on real world phenomena.

### 2.2.2 Ontology

The word 'ontology' originates from the Greek word *onto* (being) and *logia* (written or spoken discourse). In Semantic Web, ontology can be considered as a vocabulary that represents the concepts and corresponding relationships between those concepts in an area of concern. Ontology is one of the basic building blocks of inference technique on the Semantic Web. Precisely, ontology is a key enabling technique that annotates data with semantic description and provides a common, understandable foundation for describing resources [30].

Ontologies are becoming popular in many research areas such as Knowledge Engineering, Electronic Commerce, Knowledge Management, Artificial Intelligence, and Natural Language Processing. The primary objective of ontology is to attain a common and shared knowledge that can be reused in several application systems.

Ontology has four main components: concepts, instances, relations, and axioms [30].

- **Concepts**- A 'concept' is an abstract representation of a real-world object. It is one of the essential elements of a particular domain. In ontology, concepts can be defined as a group or class whose members share common properties. This component is similar to object-oriented systems. The highest level concepts are represented by “parent class” and under the “parent class”, the subordinates can be represented using “child class”. For example, "Sensor" could be represented as a class with many subclasses, such as "Temperature Sensor", "Humidity Sensor", and "Light Sensor".

- **Instances**- An 'instance' is also known as an individual. It describes the lowest level component of an ontology by representing a specific object of a concept. For example, “Java SunSpot” could be an instance of the class “Temperature Sensor” or simply “Sensor”.

- **Relations-** A 'relation' is used to describe the relationships between different concepts in a given domain. The relation between two concepts can be expressed using domain and range. For example, “measures” could be represented as a relationship between the concept “sensor” (which is a concept in the domain) and “temperature” or “humidity” (which is a concept in the range).

- **Axioms**- An 'axiom' sets constraints on the values of classes or instances. Axioms are expressed using general rules, logic-based languages such as first-order logic.

### 2.2.3 SPARQL Query Language

SPARQL (Simple Protocol and RDF Query Language) is a standard semantic query language recommended by W3C World Wide Web Consortium for exploring and manipulating data stored

in Resource Description Framework (RDF) format [31]. Data in RDF statements are characterized in RDF triple format (Subject-predicate-object). We can consider RDF triples as an SQL relational database containing a table with three columns –subject column, predicate column, and object column. The object column can be different (e.g., another resource or data type value). SPARQL allows querying over the RDF triples.

### 2.2.4 Semantic Reasoning

The reasoning is necessary for the situation where we need to find out implicit information. Semantic Reasoning is also known as reasoning engine, rules engine, or simply a reasoner. It infers implicit knowledge using inference rules or formal logics. Semantic reasoning identifies the subtyping and some other relationships between concepts in some particular areas of concern [32].

In WSN application domain, semantic reasoning plays a crucial role by providing new and implicit knowledge from the semantically annotated sensor data. In the following, we give an example to illustrate the importance of semantic reasoning.

Let us consider that a virtualized WSN is sending sensor measurement to a wildfire monitoring application. The end-user of this application is interested to know the context or the situation of the fire event. However, semantically annotated sensor data contains the temporal (time) and thematic (location) information along with the sensor measurements. Knowledge inference such as "No fire event", "Initial fire stage", and "Huge fire blaze" requires semantic reasoning and a fire domain ontology. Semantic reasoning uses formal logic to derive this implicit knowledge from the annotated sensor data and able to answer end-users complex queries such as (a) *What is the current fire situation?* or (b) *Is the fire event an initial stage or huge fire blaze?*

### 2.2.5 Benefits of Semantic Web

Semantic Web technologies have several advantages as mentioned below:

- Offers flexible data and information integration from diverse sources.
- Generates machine readable data by adding metadata with the actual data.
- Infers implicit knowledge of the data by applying semantic reasoning and ontology concepts.
- Resolves the ambiguity problem of the data or information that has the same name.
- Refines information retrieval and reduces information overload.
- Identifies relevant information for a given domain [33].
- Provides support for decision-making.

## 2.3 Data Annotation

Semantic Web enriches data models by adding semantics (metadata and knowledge) to the contents for the purpose of more efficient data management. Data annotation is a metadata generation process aiming towards more enriched and structured representation of the actual data. It provides additional information (metadata) about an existing data. Semantic data annotation is applicable for different types of data such as web pages, non-web documents, text fields in databases. [34]

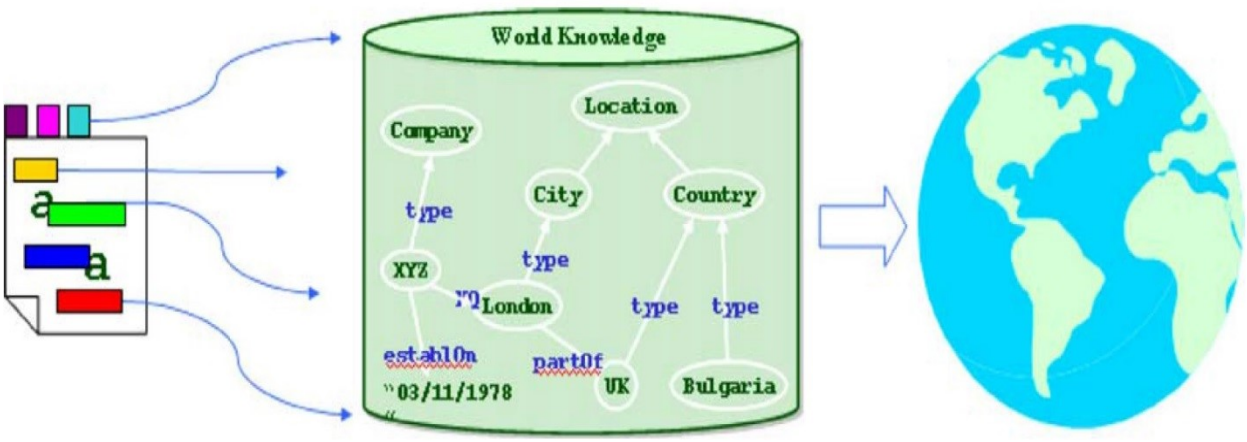


**Figure 2-5 Metadata referring to world knowledge [34]**

In Semantic Computing, every data is considered as a resource and represented by a uniform resource identifier (URI). Resource Description Framework (RDF) is a standard approach for processing metadata which specifies the semantics of the actual data in a standardized manner [35]. The main objective of RDF is to implement a mechanism to describe a resource in a way that the mechanism can describe information. RDF data model consists of the following components:

- **Resources** – Every component described by RDF expressions is called a resource. A resource can be anything such as object (e.g., a Sensor), data (e.g., sensor measurement), an entire Web page (e.g., HTML document). A resource can be a part of a web page (e.g., a specific HTML or XML element within the document source). Resources are always expressed by URIs.

- **Properties** - A property can be considered as a characteristic or a relation which is used to describe a resource.

- **Statements- A** RDF statement, also known as RDF triple, consists of three components: subject, predicate, and object. The subject of a statement should be a resource. However, the object can be another resource, or it can be a literal; (i.e., a resource specified by a URI or any data type such as string, double).

Figure 2-6 shows the graph representation of an RDF statement in a form of RDF triple (subject, predicate, and object)

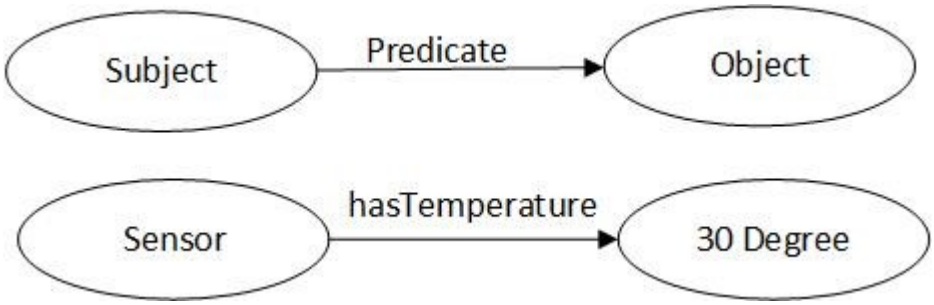


**Figure 2-6 Graph representation of RDF triple**

## 2.4 Chapter Summary

In this chapter, we have discussed the background concepts related to this thesis. First, we introduced the concept of virtualized WSNs along with cloud computing and virtualization. Second, we have provided few example applications that can be built on the virtualized WSNs. Third, semantic applications are defined by describing Semantic Web and its technologies including ontology, SPARQL query language, and semantic reasoning. Finally, we concluded this chapter by presenting the concept of data annotation.

# Chapter 3

## 3 Scenario, Requirements, and State-of-the-Art Evaluation

This chapter consists of three sections. In the first section, we present a motivating scenario. In the second section, we derive a set of requirements based on the scenario. Next, we review the state-of-the-art solutions and evaluate them based on our sets of requirements. Finally, we summarize the chapter.

### 3.1 Motivating Scenario

This section presents a motivating scenario in the virtualized WSN domain and shows how WSN applications can be provisioned in a cloud environment. We start this section by stating the early assumptions that we have made. The actors are identified in the next subsection. The third section presents a scenario in the virtualized WSN application domain. We describe the interactions between different actors in the fourth section. Finally, we apply the scenario in a wildfire monitoring application.

#### 3.1.1 Assumptions

Let us consider the case of a cloud environment where there is a virtualized WSN infrastructure provided by a WSN IaaS provider. Let assume that there are WSN application developers who develop and provide different WSN applications by using the infrastructure according to their requirements. Our proposed scenario is based on the following two assumptions:

- The first assumption is that all the WSN applications are offered as a Software-as-a-Service (SaaS) in the cloud platform. The key benefit is that it allows easy access to the WSN applications from anywhere and anytime.
- The second assumption is that the same virtualized WSN infrastructure serves multiple WSN applications residing in the cloud platform. This means that several WSN applications will share and re-use the same WSN infrastructure.

### 3.1.2 Actors

In our motivating scenario, we have identified four types of actors: the end-user, WSN SaaS provider, WSN PaaS provider, and WSN IaaS provider. Each actor may play several roles through interactions. In the following, we describe each of them.

#### *1) The End-User*

The end-user plays the role of discovering and using the WSN applications.

#### *2) WSN SaaS Provider*

The WSN SaaS provider is the WSN application developer who offers WSN applications to the end-users to discover and use the applications.

#### *3) WSN PaaS Provider*

The WSN PaaS provider allows the development and deployment of WSN applications by providing a platform that consists of the necessary development frameworks, libraries, and tools. After developing a WSN application, WSN SaaS provider deploys and manages the applications in the cloud PaaS.

#### *4) WSN IaaS Provider*

The WSN IaaS provider is responsible for deploying, installing, and managing heterogeneous sensors in different locations.

### 3.1.3 Virtualized Wireless Sensor Network Application Domain

Wireless sensors are becoming very popular in various application domains and being used all around the world to collect various data about the different environmental phenomenon. Wireless sensors play a crucial role in the environmental monitoring applications as it is crucial to recognize the context of the sensor data and understand the event or situation in such applications. We consider a virtualized wireless sensor network to resolve heterogeneity issues of the physical resources (e.g., sensors) as well as eliminate the tight coupling between WSN services and WSN deployments. The applications that use wireless sensors are named as WSN applications. In the following, we describe a motivating scenario in a virtualized wireless sensor network domain.

Let us consider a city administrator who is interested in monitoring wildfire in the forest in the nearby countryside for the early detection of the brush fire eruption. Wildfire monitoring is very crucial since late detection of fire leads to a rapid destruction. Status of a wildfire can be categorized into different states including *No fire*, *Tends to fire*, *Beginning of fire* and *Huge fire blaze* based on the fire size, spreading speed, and direction [36].

Let us assume that the city administrator informs the WSN IaaS provider who then deploys different brands of sensors having different sensing capabilities all around the city as well as the nearby forests to detect different physical phenomena to monitor fire situations. Examples of deployed

sensors can be Java SunSpot, TelosB, DHT11, AdaFruit, HTU21D-F, and ADXL335. These sensors have several sensing capabilities including temperature, humidity, light intensity, CO2, CO, and accelerometer.

Let us consider that the same city administrator wants to monitor a weather condition to generate alarms in the worst weather situation. In the weather monitoring application, the weather situations can be categorized into different groups such as *Heat warning, Sunny*, *Blizzard*, *Haze*, *Cold*, *Hot*, *Partial cloudy* and *Snow*. So, the city administrator again informs the WSN IaaS provider. As both the fire monitoring and weather monitoring applications use sensor data, we can refer them as WSN applications. Later, we categorize the WSN applications into two groups: (a) Semantic applications, and (b) Non-Semantic applications.

### a) Semantic Applications

This group of WSN applications uses semantically annotated sensor data for better understanding of the observed phenomenon. Semantically annotated data also provides contextual information and allows end-users to understand different complex events or situations. Examples of semantic applications are wildfire monitoring application and weather monitoring application. These applications use semantically annotated data to determine several complex events or situations. For example, a traditional fire monitoring application allows its users to detect the fire event. On the other hand, semantic fire monitoring application not only just detects the fire event but also can identify the spreading direction of the fire event and allow the end user to know the contexts of fire situations such as *Tends to fire*, *Beginning of fire*.

### b) Non-Semantic Applications

This group of applications is traditional WSN applications that use raw sensor data according to their requirements. For example, a smart home application has several features including smoke detection, temperature monitoring feature. The smoke detector application is quite simple and straightforward application. In this application, if a smoke detector sensor detects smoke, it generates an alarm. Similarly, in temperature monitoring application, a temperature sensor senses temperature event and display it.

## 3.1.4 Interactions among Actors

This section describes the interactions between the different actors in the proposed scenario. Interactions among the actors are categorized into three groups: end-user interactions, WSN SaaS provider interactions, and WSN PaaS provider interactions.

Figure 3-1 shows the interactions that happen among the actors.

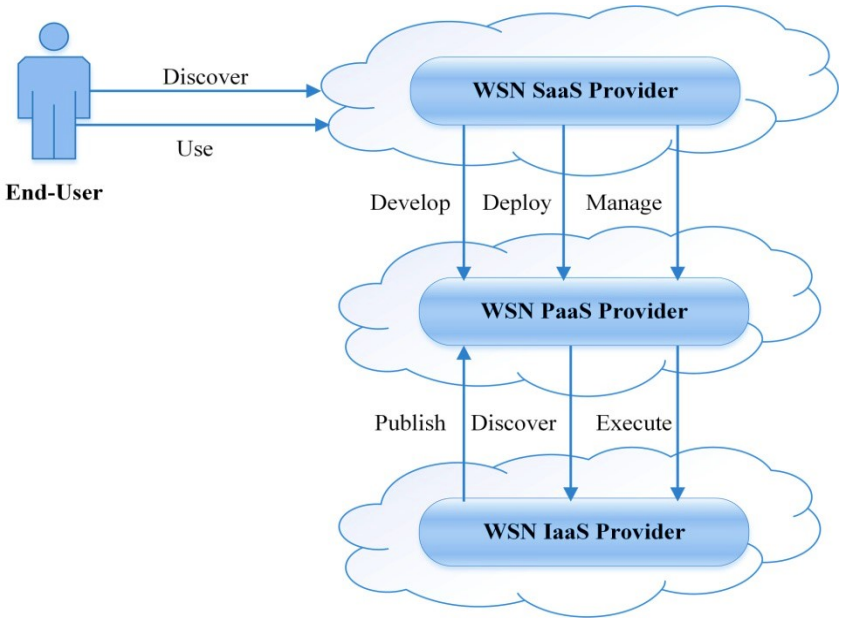


**Figure 3-1 Different roles of the actors and the Interactions**

### *A. End-user Interaction*

In the proposed scenario, the end user interactions can be defined into two types that happen between the end-user and the WSN SaaS provider.

- **"Discover" WSN Application Interaction**

  This interaction permits the end-users to discover the appropriate WSN application that is offered in the cloud SaaS. There can be an intermediate agent that bridge the gap between the end-users and the WSN application provider.

- **"Use" WSN Application Interaction**

  End-user uses the WSN application through this interaction. The use of the WSN application involves monitoring, detecting events, and taking appropriate courses of actions.

### *B. WSN SaaS Interactions*

This group of interactions occurs between the WSN SaaS and the WSN PaaS. We define three types of interactions which are as follows:

- **"Develop" WSN Application Interaction**

  The WSN SaaS provider or the WSN application developer is responsible for developing the applications. Upon being developed, the WSN application is offered by the WSN SaaS provider to the end-users. This interaction consists of different steps such as analyzing the requirements, designing the proposed solution, implementing it, and finally testing the developed application.

- **"Deploy" WSN Application Interaction**

  After developing the WSN application, WSN SaaS provider deploys it in the cloud platform.

Deployment interaction consists of installing, configuring, and activating the WSN application.

- **"Manage" WSN Application Interaction**

  This interaction is used to manage the WSN application by WSN SaaS Provider.

### *C. WSN PaaS Interactions*

These interactions occur between the WSN PaaS provider and the WSN IaaS provider.

- **"Publish" WSN infrastructure Interaction**

  WSN IaaS may consist of heterogeneous sensors having different sensing capabilities and serves several WSN applications. However, some of those applications may not need all the deployed sensor measurements. This interaction involves publishing different sensor capabilities to the cloud PaaS and allows WSN applications residing in PaaS to discover them.

- **"Discover" WSN infrastructure Interaction**

  As the WSN applications rely on sensor data, these applications need to discover the WSN infrastructure. This interaction allows the WSN PaaS provider to find the deployed sensors and their capabilities.

- **"Execute" WSN Application Interaction**

  This interaction describes the execution, utilization, and run-time management of the WSN IaaS. The WSN IaaS executes the sensing tasks by sensing different events, generating sensor measurements, and sending same sensor data to multiple WSN applications residing in the cloud PaaS.

### 3.1.5 Wildfire Monitoring Application

We now apply the described scenario in the wildfire monitoring use case. The actors are the WSN SaaS provider, WSN PaaS provider, WSN IaaS provider, and end-user is the city administrator.

Figure 3-2 shows the sequence of steps that occur among the actors and end-to-end execution of the fire monitoring scenario. We describe each of the interaction step by step in the following.

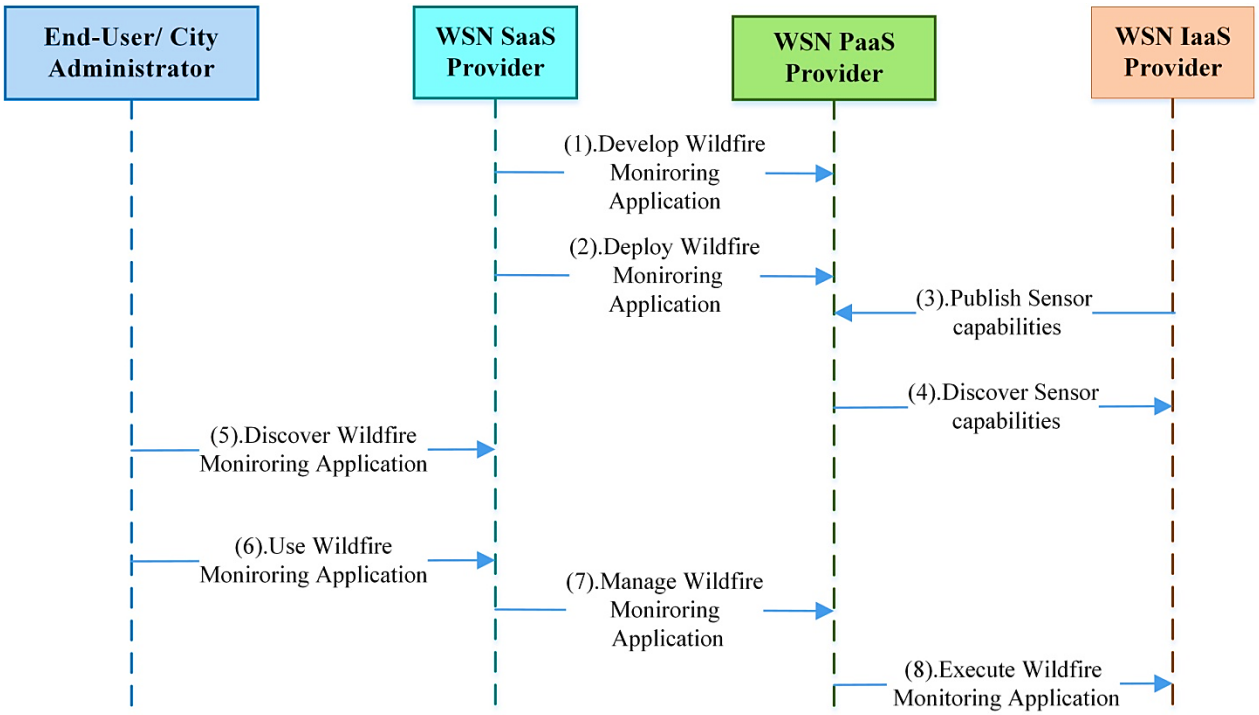


**Figure 3-2 Interactions among the actors and end-to-end execution of the Wildfire Monitoring scenario**

(1) The WSN SaaS provider develops a wildfire monitoring application in order to allow the city administrator to monitor the fire situation of the forest.

(2) The WSN SaaS provider deploys the wildfire monitoring application in the cloud PaaS which is offered by the WSN PaaS provider.

(3) The different capabilities of sensors are published in the cloud platform by the WSN IaaS Provider to allow WSN applications to use the sensor measurements.

(4) The wildfire monitoring application residing in cloud PaaS discovers sensors with different capabilities from the deployed area.

(5) When the wildfire monitoring application is in operation stage, the city administrator discovers it using discover interaction as the cloud platform contains several WSN applications.

(6) City administrator uses the wildfire monitoring application and monitors the fire situation.

(7) The SaaS provider manages the fire monitoring application.

(8) At the operation stages, WSN PaaS provider operates with the WSN IaaS by making proper setup and configurations of WSNs. As a result, sensors belonging to the WSN IaaS, execute the sensing tasks and send sensor data to the fire monitoring application. Fire monitoring application further processes these data to detect the fire events and understand the context of the fire situation.

## 3.2 The Requirements

In this section, we derive the requirements based on the motivating scenario described in the previous section. The first section describes the basic requirements and explains the importance of semantic data annotation and ontology requirements. Based on the basic requirements, we

categorize the requirements further into three different groups: (a) general requirements, (b) requirements for the virtualized WSNs IaaS, and (c) ontology provisioning requirements.

### 3.2.1 Basic requirements

We have identified two basic requirements: (a) Semantic data annotation requirement, and (b) Ontology requirement. In the following, we explain these requirements regarding the scenario described in the previous section.

#### a) Semantic Data Annotation Requirement

The first basic requirement is semantic data annotation. According to our assumption, data from the same virtualized WSN IaaS is sent to several WSN applications. Usually, the sensor produces low-level data either in the binary or plain text. However, the end-user is interested in the context or the situation of the detected event. For example, in the case of wildfire monitoring application, the end-user may be interested in the following queries:

(a) *What events are being observed near Lucille Forest currently*?

(b) *Find the temperature sensor reading near Lucille Forest?*

(c) *What is the current fire situation?*

(d) *Is it initial stage or huge fire blaze?*

(e) *What is the current CO2 level in the fire affected area?*

(f) *Which areas have been affected by the fire event?*

If the sensors send raw data in plain text or binary format (for example, data is "40"), these applications will not understand the context and meaning of the raw data (for example, it might be temperature, humidity or wind speed which is not clear from the data "40"). This ambiguity is

eliminated by data annotation which is a process of metadata generation that can be added to the raw data by labeling or tagging (for example, <temp>40</temp>) [37].

XML and JSON are popular approaches for data annotation. However, these data representations lack to provide the contexts and situations of the data. For example, the data '<temp>40</temp>' does not provide intuition whether this temperature represents a situation like *Tends to fire* or *No fire* [38]. These representations also lack more complex contexts involving multiple sensor measurements. In the scenario, we stated that the weather situations could be categorized into different groups such as *snow alert, heat warning, partially cloudy*. These situations depend on not only temperature measurement but also humidity, wind speed, etc. measurements. Now, the question is how to build knowledge from the low-level sensor data? This problem can be solved by performing semantic data annotation. Data annotation using semantic metadata provides contextual information. Using semantic data annotation, sensor observation can be expressed in terms of the sensing time, location and measurement unit.

### b) Ontology Requirement

Since our focus is to create an abstraction of the heterogeneous sensor data that come from the WSN IaaS, hence we need an ontology that will hold the concepts related to the WSN IaaS sensors capabilities and observations. As a result, the second basic requirement is the use of ontology for semantic data annotation. There are several reasons for using an ontology for data annotation.

*First*, data annotation using ontology provides more powerful and enrich representation of the observations having temporal and geographic contexts [39]. For example, a Java SunSpot sensor in

location *X* sent temperature output 40 degrees celsius on 09/02/2015 02:03:56 am. It is possible to annotate the raw data with respect to sensor type, location, measurement type, measurement value, unit of measurement, and measurement time. The fire monitoring application can determine a fire situation using the measurement value and unit of measurement (e.g., 50 C). It can also determine the direction of the fire using the sensor location and time.

*Second,* using an ontology at IaaS level allows the domain applications applying reasoning and extracting implicit knowledge from the captured information. For example, in the fire monitoring application, different fire situations (e.g., *Tends to fire*) and other fire domain-related concepts (such as *High temperature*, *Low humidity*) can be expressed with a *fire domain ontology*. If this application receives the semantically annotated data from the IaaS level, it can inherit additional knowledge or context of the environment by further annotating the data using reasoning and the *fire domain ontology*. Let us consider the case of *Tends to fire* situation (when *Very high temperature*, *Low humidity*, *High CO2*, *High CO*). The fire monitoring application receives the annotated data (temperature, humidity, CO2, CO) from the WSN IaaS and applies reasoning rules to determine knowledge such as *Very high temperature*, *Low humidity*, *High CO2*, *High CO*. After applying further reasoning using the domain ontology, it determines the situation *Tend to fire*.

However, applying ontology to annotate sensor data at WSN IaaS level is not straightforward. Since the ontology in the WSN IaaS domain contains concepts related to the deployed infrastructure, hence if any change happens to the WSN IaaS, the corresponding changes should be made to the ontology as well. After that, the ontology should be again deployed in the WSN IaaS. This situation arises the ontology provisioning requirement (explained in Section 3.2.4).

Based on the basic requirements, we identify several requirements and categorize them into three different sets which are described in the next sections.

### 3.2.2 General Requirements

In addition to the basic requirement of semantic data annotation, the *first* general requirement should be the real-time in-network annotation. As the same virtualized WSN would be used by several semantic applications, the sensor data should be annotated real-time before leaving the network. This requirement eliminates the costs and efforts of redundant annotations performed by the SaaS application developers.

The *second* requirement implies the need for an interface that allows proper interaction between WSN IaaS and WSN PaaS. The interface should be based on the standard technologies. For example, if we consider the scenario described in the previous section, WSN IaaS should be able to interact with the cloud PaaS through an interface which is based on standard technology.

The *third* requirement is that the proposed solution should be domain and application independent. For the scenarios (Fire monitoring application and Weather Monitoring application) described in the previous section, both applications require temperature and humidity sensor reading. In order to reuse the same infrastructure by both applications, the proposed framework should generate a standard data format that can be re-used and enhanced by multiple real-time applications.

The *fourth* requirement is scalability. The WSN Infrastructure should maintain the performance when the number of sensors increases at a large scale in the network.

The *fifth* requirement is that the annotation should be done in a distributed manner without relying on a central node. This ensures that single point of failure will not occur in the annotation process. When the number of nodes in the network increases, distributed annotation process can accelerate the annotation and handle the scalability issue.

### 3.2.3 Requirements for Virtualized WSNs IaaS

The proposed solution should consider the infrastructure heterogeneity to ensure interoperability. Any large scale WSN infrastructure will contain different sensor nodes having different sensing capabilities, data formats, and other properties. The proposed solution should be able to deal with the sensor heterogeneity issue.

### 3.2.4 Requirements for Ontology Provisioning

The *first* requirement is that a standard ontology is needed at the WSN IaaS level to perform data annotation and handle sensor heterogeneity issue. Using ontology at an IaaS level for data annotation eliminates the cost and effort of redundant annotations performed by several end-user applications. In this thesis, we refer our standard ontology as base ontology. Another benefit of using ontology at WSN IaaS is that the end-user application can extend the base ontology for the development of domain ontology by re-using the base ontology concepts.

The *second* requirement implies that the base ontology needs to be provisioned at WSN IaaS in the initial stage to enable semantic applications provisioning. Base ontology provisioning includes the development, deployment, and management of the base ontology. This requirement raises the need for an ontology provisioning center.

The *third* requirement is that it should be easy for the WSN IaaS provider or any novice user to interact with the ontology provisioning center without knowing technical knowledge or protocol details.

The *final* requirement is that the proposed solution should be technology independent and can be implemented using any standard technology.

## 3.3 The State-of-the-Art Review and Evaluation

Recent researches on the sensor data annotation focus on annotating the sensor measurements outside the network or at the gateway node. However, based on our knowledge, none of these solutions can provide in-network real-time sensor data annotation. Current solutions are domain specific, and there is a very little possibility for other applications to reuse the infrastructure.

In this section, we present current state-of-the-art solutions similar to our research area and evaluate them critically. We classify the state-of-the-art solutions into two groups: The first group is the existing frameworks for annotating sensor data for semantic applications, and the second group involves the research work related to the ontology development, deployment, and management. After reviewing the state-of-the-art solutions, we evaluate them based on our requirements.

### 3.3.1 Data Annotation Frameworks

There are a number of applications available in the literature that presents the semantic enhancement of sensor data. The first effort to generate the idea of annotating sensor data with semantic metadata has been proposed in Semantic Sensor Web (SSW) framework [5]. It is based on the Sensor Web

Enablement (SWE) effort by W3C Open Geospatial Consortium (OGC) Semantic Web. SWE annotates the sensor data using temporal, spatial and thematic concepts. OGC SWE languages are used for temporal and spatial annotation of sensor data. The authors use the RDF for the semantic annotation of the sensor data and domain ontologies for providing concepts and relationships. Semantic Web Rule Language (SWRL) is used to reason over the annotated data and infer knowledge. The authors develop two proof-of-concept prototype applications using their proposed architecture.

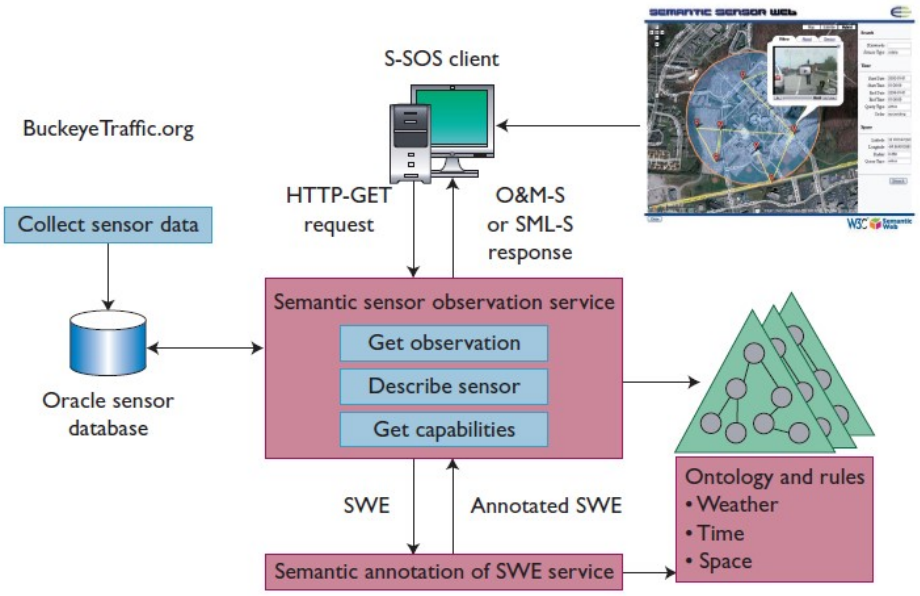


**Figure 3-3 Semantic Sensor Observation Service (S-SOS) architecture [5]**

Figure 3-1 is an implementation architecture named Sensor Observation Service (S-SOS) which is involved in requesting, filtering, and retrieving observations over the weather data. S-SOS collects weather reading (such as temperatures, wind speed, wind direction, and precipitation) from the website named BuckeyeTraffic.org that provides traffic and weather reading collected from 200 sensors. They convert those reading in O&M and SML formats. They use various domain

ontologies to annotate O&M and SML formatted document. In this work, no performance result is presented and the solution is domain specific. Moreover, the provided solution does not allow real-time sensor data annotation and uses a centralized approach to perform the annotation tasks. Finally, there is no discussion on the interface between the semantic annotation service and the knowledge base that contains ontologies.

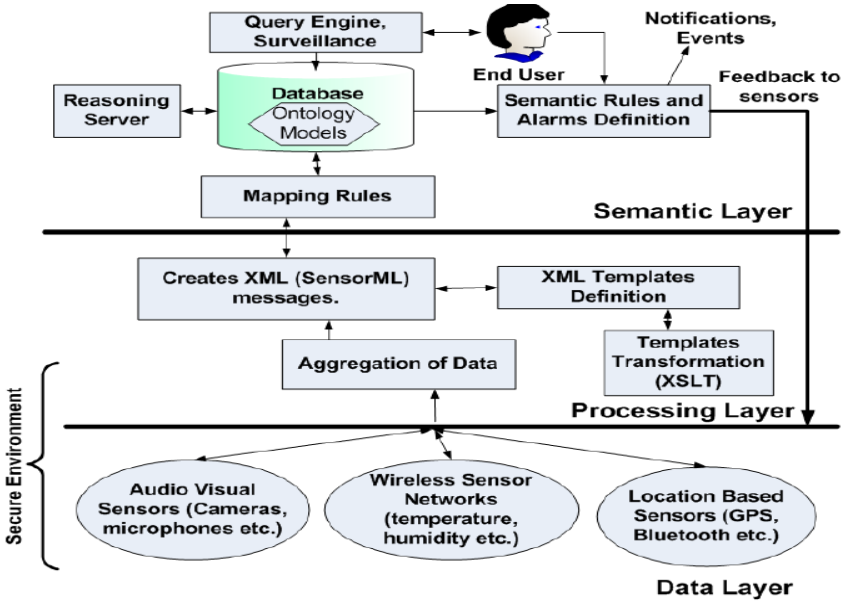


**Figure 3-4 Semantic-based architecture for sensor data fusion [40]**

A three layer architecture based on OGC SWE and the Semantic Web is presented in [40], that assists the gathering, processing, and exploiting sensor data in real-time. It uses Observations and Measurements (O&M), and SensorML specifications for semantic specification of the sensors, properties, and raw data. The first layer (data) involves in collecting raw data from heterogeneous sensors. The second layer (processing) aggregates those raw measurements, transforms into XML and forwards to the next layer. The third layer (semantic) processes those aggregated data by mapping them into ontology model contained in a database. The annotations are created on the third

layer and stored in a knowledge base. However, their proposed framework is domain specific that conflicts our *third* general requirement. The proposed annotation mechanism is centralized. Moreover, there is no discussion regarding the scalability issue. In this framework, ontology files are stored in a database, but there is no such discussion regarding ontology provisioning (development, deployment, and management).

A sensor data annotation architecture is presented in [41] which is similar to [40].

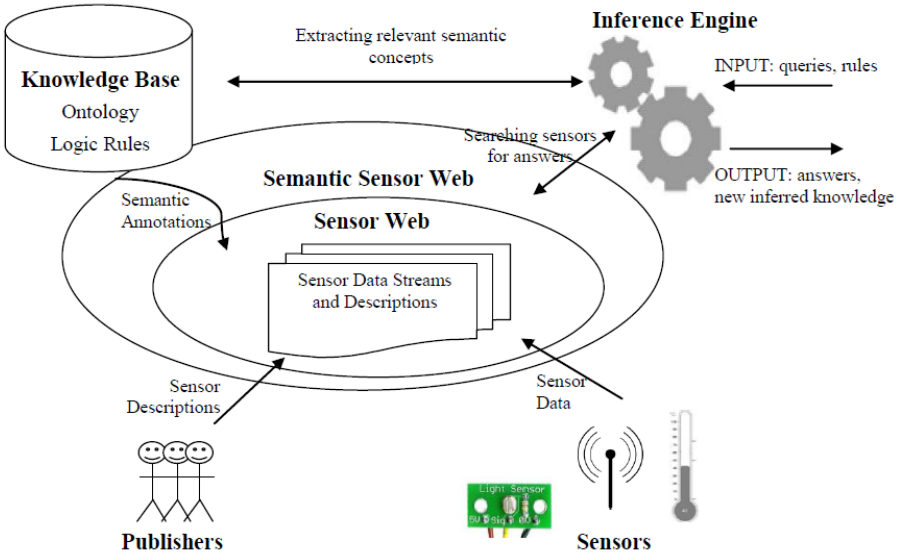


**Figure 3-5 Architecture proposed in [41]**

In the proposed architecture, sensors description, physical location, and data are published in XML format in Sensor Web by their publishers. The ontology concepts are stored in a knowledge base which is used to annotate sensor data by using the description provided by their publishers. However, there is no discussion how the ontology concepts are developed, managed, and deployed in that knowledge base. Work from Cyc project [42] is used in this framework by reusing the

existing concepts of the Cyc ontology to describe the sensors. A case study illustrating participatory sensing from Pachube [43] is presented. Each Pachube sensor provides its XML file containing the properties, measurements, and details of the events. Finally, user queries are answered using the Cyc inference engine.

A large-scale European project SPITFIRE is proposed in [44] to allow transition from Semantic Sensor Web to Semantic Web-of-Things.

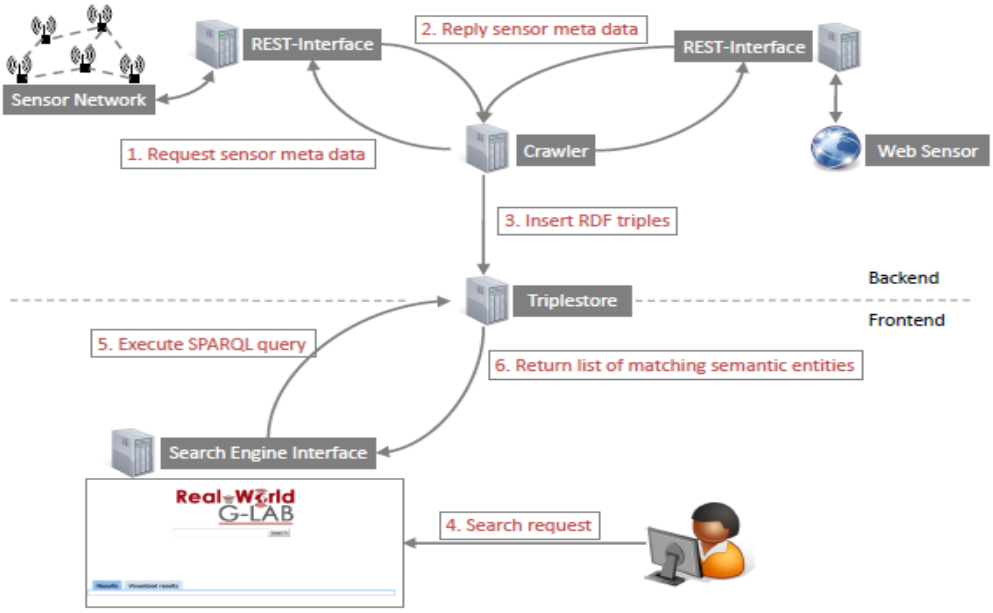


**Figure 3-6 Implementation architecture of SPLITFIRE [44]**

The work provides three main contributions: (a) A new sensor description mechanism that easily integrates with the Linked Open Data cloud (LOD). The data from the LOD can be used by different applications or services. (b) A semi-automatic creation process for semantic sensor descriptions. The sensor data is collected in a pattern dictionary to generate patterns along with the semantic

annotations. The patterns help to determine the type of a new sensor and automate annotation of its data, (c) An efficient search mechanism to find the sensors and things based on their current state. A crawler is used to retrieve the sensor data from multiple sources (sensors and web pages). The gathered data is stored in an RDF triple store and later queried SPARQL query engine. A reference implementation architecture is presented as a proof-of-concept but performance measurement for validation is absent in this paper. In this architecture, there is no discussion regarding the scalability. Moreover, the annotation process is centralized that raise the problem of single point of failure.

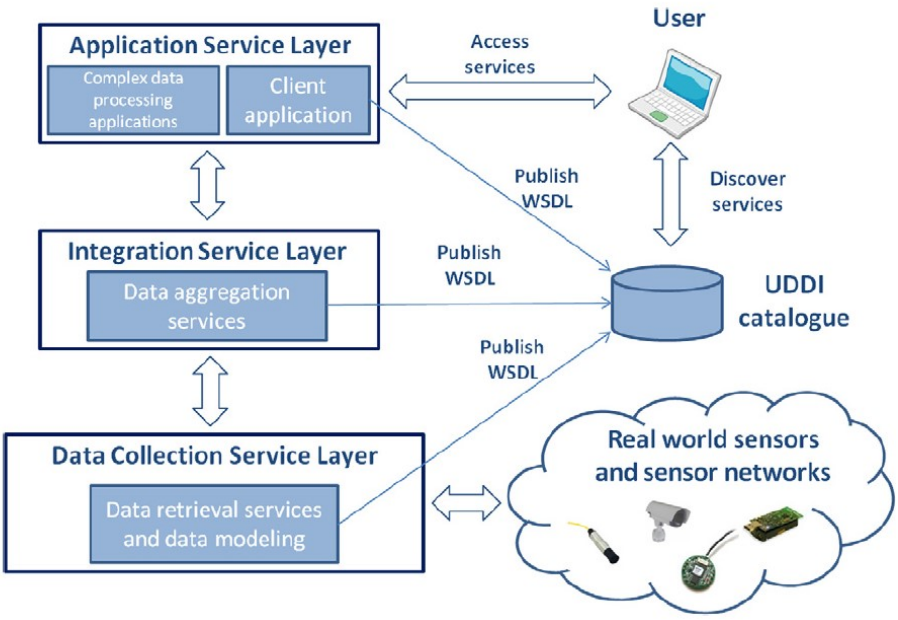


**Figure 3-7 Data annotation architecture proposed in [45]**

The architecture proposed in [45] is based on the Service-Oriented Architecture (SOA) and consists of three horizontal layers. The first layer is the Data Collection Service Layer that provides a homogenous view of different networks to the second layer. Authors use the ontological modeling technique to enrich sensor data by converting the raw sensor measurement into a common

XML/RDF data format using ontology technique. The second layer is responsible for aggregating the dataset of the same event observed by different networks and forwarding it to the third layer for a better understanding of the complex situation. The third layer consists of different types of real-time monitoring applications which utilize the lower layers and common formatted sensor data to implement a decision support system. The proposed architecture uses various ontologies (Sensor data ontology, Sensor observation Ontology, and domain ontology) but there is no such discussion how these ontologies are managed and deployed.

Authors propose a new ontology SenMESO for sensor data annotation in [46] which is a combination of various domain ontologies covering sensor data and features of interest.

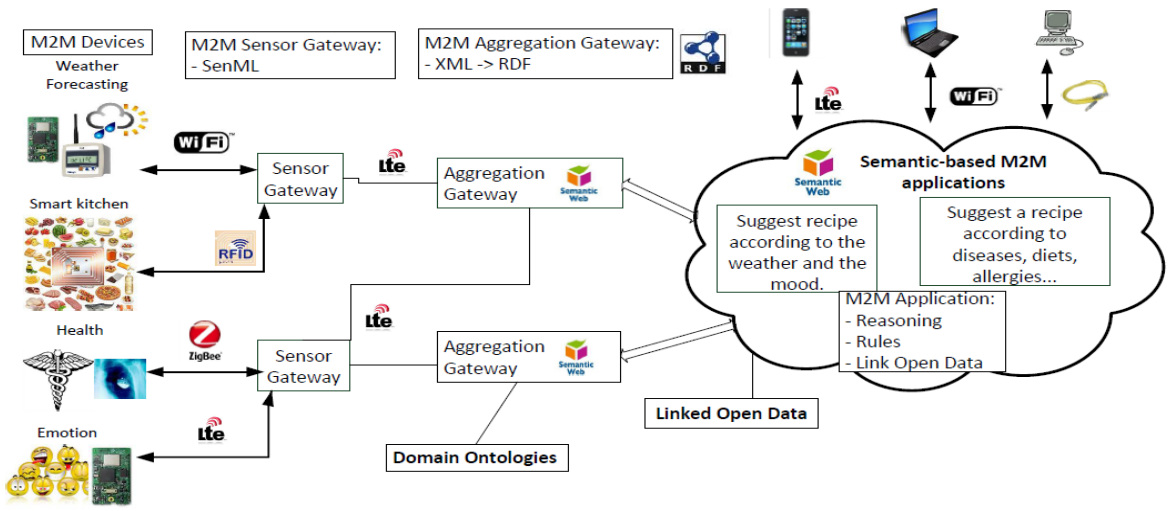


**Figure 3-8 M2M architecture [46]**

The gateway nodes receive sensor data in different formats and convert the data into an XML format to support interoperability. The aggregation gateways incorporate semantics to XML sensor data using RDF, RDFS, OWL, and domain ontologies. In this work, after creating the annotation, the

sensor measurements are linked to the LOD (Link Open Data) Cloud where additional information can be inferred using various domain ontology.This architecture does not use node-level or network-level virtualization. As a result, the proposed solution requires redundant sensor deployments for multiple applications. The scalability has not been discussed in this work. Since the annotation process is done at the aggregation gateway in this architecture, hence there is no discussion regarding standard interface through which ontology can be deployed in the WSN IaaS.

An approach to convert the sensor data in SenML format to RDF format is provided in [47]. A basic mapping of SenML elements to RDF types is provided. A SenML reading from a sensor is first transformed into RDF elements, which are used to generate an array of RDF triples. Finally, these RDF triples are serialized in different formats. A prototype implementation for monitoring water quality of fish farms is presented. The sensor data collected by an aggregation gateway is forwarded to a knowledge-based (KB) system. The KB system converts SenML data to RDF and applies domain ontology concepts in a centralized fashion to generate alerts for the client application. The implementation results show the performance gains while using SenML as compared to other data formats including RDF, N-triple, and N3. Only the data from IoT devices (sensors) that send data in SenML can be transformed in RDF which is the main limitation of this work. Instead of using a standard ontology, SenML is converted to RDF format that conflicts our requirement. Finally, the proposed solution is domain specific.

Table 3-1 shows a summary evaluation of the state-of-the-art solution for data annotation frameworks. The following table shows none of the solutions fully satisfies our derived requirements.

| Requirements / State-of-the-Arts | General Requirement | | | | | IaaS Requirement |
|---|---|---|---|---|---|---|
| | Real-time annotation | Standard Interface | Domain independent | Scalability | Annotation Mechanism | Infrastructure Heterogeneity |
| [5] | No | Not Discussed | No | Yes | Centralized | Yes |
| [40] | Yes | Not Discussed | No | Not discussed | Centralized | Yes |
| [41] | No | Not Discussed | Yes | Yes | Centralized | Not Applicable |
| [44] | Yes | Yes | Yes | Not discussed | Centralized | Yes |
| [45] | Yes | Yes | Yes | Not discussed | Centralized | Yes |
| [46] | Yes | Not Discussed | Yes | Not discussed | Decentralized | Yes |
| [47] | Yes | Not Discussed | No | Not discussed | Decentralized | Yes |
| Our Work | Yes | Yes | Yes | Yes | Decentralized | Yes |

**Table 3-1 Summary evaluation of the state-of-the-art solutions for data annotation frameworks**

### 3.3.2 Ontology Development, Deployment and Management Framework

In this section, we describe existing related works for ontology development, deployment, and management.

Several knowledge management systems have been proposed for the development and management of ontologies. Authors in [48] propose a framework *ONKI* for the collaborative development and management of ontologies. According to this framework, a domain expert first develops an ontology using existing ontology editor and then publishes it to the *ONKI* Library. User applications can access those ontologies using web service. This framework does not address our requirements: (a) a domain expert is needed for the development and management of ontology, and (b) ontology deployment process is not discussed.

An ontology management framework proposed in [49] allows ontology developers to manage multiple ontologies as well as create new ontologies reusing the concepts of existing ontologies. However, there is no mechanism proposed for ontology deployment. Moreover, domain experts or ontology developers are required to interact with the framework which does not satisfy our ontology provisioning requirements.

Another ontology development framework named *SOFA* has been proposed in [50] which is a Java API for representing concepts and developing an ontology. This is a technology-dependent solution and discusses only the ontology development process. However, the ontology management and deployment process are not addressed in this work.

Authors in [51] propose a tool for ontology generation which uses reverse engineering technique to develop an ontology. First, they extract database metadata information and then analyze the relationship between those data. Authors develop a prototype implementation using Apache Jena API. This framework does not meet our requirements as the proposed framework is technology dependent. As there is no easy way for the novice user to interact with the system, a domain expert is needed to process the data stored in the database and develop an ontology. Finally, ontology management and deployment procedures are not addressed in this work.

There are few protocols exist for managing network Infrastructures. Simple Network Management Protocol [52] is the most widely used management protocol that runs over TCP/IP protocol stack and applicable for wired networks. This protocol allows managing network performance, finding, and solving network problems. Ad Hoc Network Management Protocol (ANMP) [53] is mainly designed for managing mobile wireless ad hoc networks. This protocol is compatible with SNMPv3

and uses similar protocol data unit structure and management information base as in SNMP. Both the SNMP and ANMP have been designed for managing the network but according to our requirement, we need an ontology management mechanism in WSNs. These protocols do not fulfill our ontology management requirement. Also, SNMP and ANMP do not cater to develop and deploy ontology in WSNs. As a result, our requirements are not fulfilled by these protocols

Table 3-2 shows a summary evaluation of the state-of-the-art solution for Ontology development, management, and deployment frameworks. We find none of these solutions satisfy all the ontology provisioning requirements.

| Requirements / State-of-the-Arts | Ontology Provisioning Requirement | | | | |
|---|---|---|---|---|---|
| | **Standard Ontology** | **Ontology Development and Management** | **Ontology Deployment** | **User-friendly Solution** | **Technology Independence** |
| [48] | Yes | Yes | Not Discussed | No | No |
| [49] | Yes | Yes | Not Discussed | No | Yes |
| [50] | Not Applicable | Partially Discussed | No | Not discussed | No |
| [51] | Not Applicable | Partially Discussed | No | No | Yes |
| [52] | Not Applicable | No | No | No | Not Applicable |
| [53] | Not Applicable | No | No | No | Not Applicable |
| Our Work | Yes | Yes | Yes | Yes | Yes |

**Table 3-2 Summary evaluation of the state-of-the-art solutions regarding ontology development, deployment, and management framework**

## 3.4 Chapter Summary

In this chapter, first, we presented the motivating scenario. Then we derived the basic requirements based on the scenario and demonstrated the need for semantic data annotation and using ontology. Based on the basic requirements we derived some other requirements and classified them into three groups: general requirements, requirements on the virtualized WSN Infrastructure and requirements on the ontology provisioning. The state-of-the-art solutions related to our research domain are evaluated based on the requirements described in section 3.2. We found that none of the state-of-the-art solutions satisfies our requirements fully.

# Chapter 4

## 4 Data Annotation Architecture and Base ontology[1]

We derived our precise requirements for sensor data annotation in virtualized WSNs in Chapter 3. This chapter presents the data annotation architecture for semantic applications in the virtualized WSNs. In this regard, we have utilized a recently proposed WSN virtualization architecture [54] by our research team as a basis. Since semantic data annotation process requires an ontology to represent the corresponding domain concepts and the semantic relationships among the observed concepts, hence we define the required ontology (we refer it 'base ontology') for the sensor domain.

This chapter consists of the following sections: We explain the overall architecture for data annotation and describe the components of the architecture including layers and functional entities in the first section. In the second section, we present our base ontology that contains the concepts related to common sensing phenomena. Data annotation procedures are discussed in the third section. A wildfire monitoring use case is presented in the fourth section. We discuss how the requirements are met by the architecture in the fifth section. Finally, a summary of the chapter is presented in the sixth section.

[1] This chapter extends the architecture presented in the paper " I. Khan, R. Jafrin, F. Errounda, R. Glitho, N Crespi, M Morrow, P Polakos, A Data Annotation Architecture for Semantic Applications in Wireless Sensor Networks, *IFIP/IEE International Symposium on Integrated Management (IM 2015),* vol., no., pp. 27, 35, 11-15 May 2015, Ottawa, Canada" (acceptance rate: 27.2%)

## 4.1 Overall Architecture

In this section, we begin with stating the assumptions and describing the architectural principles that we adopted for designing the architecture. We explain the layers and functional entities of the overall architecture based on these assumptions and principles.

### 4.1.1 Assumptions

The proposed architecture is based on several assumptions which are stated below:

- The virtualized WSN consists of heterogeneous sensors and it is offered as a WSN infrastructure (IaaS) in cloud paradigm.
- We assume that the sensors have already been discovered and are stored in a registration server. For sensor discovery, there are existing works including [55], [56] can be reused for this purpose.
- Our proposed solution does not have storage to store the sensor data. Since it is an application specific requirement, hence we leave it for end-user applications to decide on the sensor data storage.
- We assume that the base ontology is already provisioned in the WSNs.

### 4.1.2 Architectural Principles

The *first* architectural principle is that a standard ontology is used to annotate the sensor data and stored in WSNs. We named it as base ontology because it holds a minimal set of concepts of sensor observations that can be used as a basis to build an application domain ontology. The base ontology consists of concepts related to the deployed sensors and their capabilities. However, annotated

sensor data can further be annotated in semantic applications using domain ontology. A domain ontology consists of a domain and application-specific concepts. This fundamental principle allows the solution to become independent of any application domain.

The *second* architectural principle is that we use two separate overlays: one for data annotation and the other for storing the base ontology. Overlays have several advantages: they are distributed, they do not rely on centralized control, and they allow resource sharing [57].

The *third* architectural principle is that every virtual sensor created for semantic applications is represented in the annotation overlay by a corresponding entity that annotates sensor data. This means that every sensor sending data to semantic applications will have a dedicated entity for annotation purpose.

The *fourth* principle is that the annotations will be performed by capable sensors and other powerful nodes (e.g., gateways).

### 4.1.3 Layers and Functional Entities

The overall data annotation architecture is represented in the Figure 4-1. It is based on the WSN virtualization architecture [54] proposed by our research team. The virtualization architecture contained four layers (physical, virtual sensor, virtual sensor access, and application overlay). In the data annotation architecture, the physical layer remains the same as in the virtualization architecture. The rest of the layers are enhanced by adding different functional entities. The virtualization architecture only supported traditional sensor applications (non-semantic applications) whereas our proposed data annotation architecture supports both semantic and non-semantic applications.

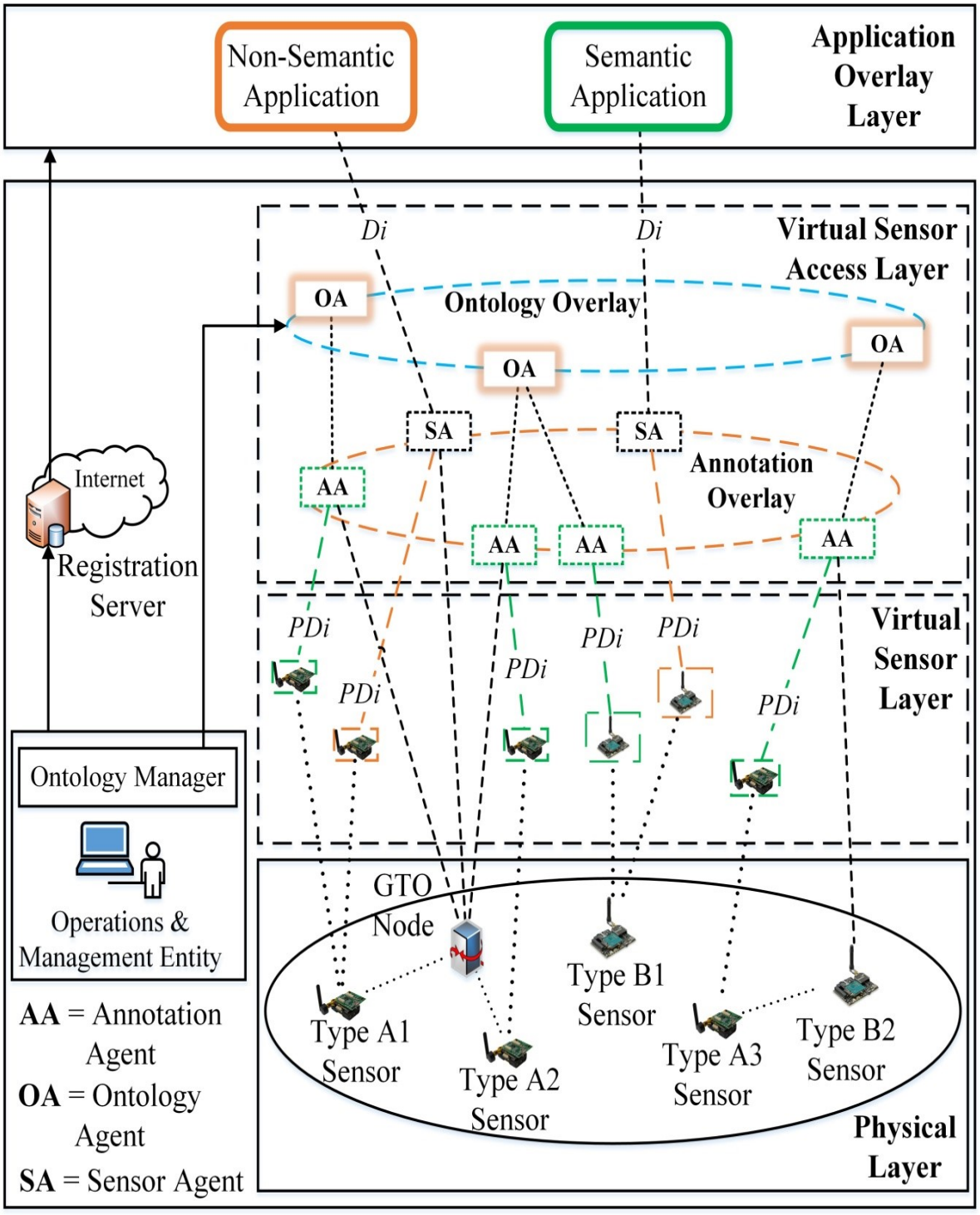


**Figure 4-1 Data annotation architecture**

We describe each component of the data annotation architecture in the following:

### A. *Physical Layer*

This is the first layer which contains the physical WSN infrastructure and consists of different types of sensors. Based on the processing capability, energy, and storage capacity, we categorized them into the following three groups:

#### 1. *Type A* sensors

This kind of sensor is resource constraint device that has very limited processing and storage capabilities (e.g., TelosB motes). Since *Type A* sensors may not be capable enough to work together with other sensors in the overlay, they rely on more powerful nodes such as Gate-to-Overlay (GTO) nodes or *Type B* sensors.

#### 2. *Type B* sensors

This type of sensors is a new generation powerful sensor node (e.g., Java SunSpots) and capable of executing multiple tasks.

#### 3. *GTO Node*

In order to facilitate *type A* sensors to work along with capable nodes, WSN infrastructure has specialized node called Gate-to-Overlay (GTO) node (e.g., base station, sink node). GTO nodes are more powerful than *type B* sensors. Figure 4-1 shows that *Type A*1 and *Type A*2 sensors are connected to the GTO node. However, *Type A3* sensor is connected to the *Type B2* sensor as *Type B* sensors are also capable nodes.

### *B. Virtual Sensor layer*

This is the second layer which contains virtual sensors. Virtual sensors are the logical representation of the physical sensors that execute different application tasks simultaneously. For example, in Figure 4-1, *Type B1* nodes have two virtual sensors that execute two application tasks (AA and SA which are explained in the next section). However, without virtualization, a physical sensor can execute single application task. Figure 4-1 also shows that *Type B*2 sensor has no virtual sensor and execute only one application task (AA).

We categorize the virtual sensors into two groups:

1. **Semantic virtual sensors**

   Semantic virtual sensors are created for semantic applications (marked as the green box in the Figure 4-1) which execute different semantic tasks such as annotating sensor data or storing ontology.

2. **Non-semantic virtual sensors**

   Non-semantic sensors are marked as the orange box in the Figure 4-1 which execute non-semantic tasks. For example, these sensors directly send raw data to the non-semantic applications or annotated data to the semantic applications.

### *C. Virtual Sensor Access Layer*

The sensor data annotation is performed in the virtual sensor access layer. This layer has three functional entities and two overlays.

Functional entities are pieces of software codes which define different application tasks and run either on virtual sensors or physical sensors. Following are the functional entities:

1. **Annotation Agent (AA)**

   Nodes acting as Annotation Agent are primarily responsible for annotating sensor raw data and sending the annotated data to the nodes acting as Sensor Agent.

2. **Ontology Agent (OA)**

   Nodes acting as Ontology Agent hold the base ontology and serve ontology request that comes from Annotation Agents.

3. **Sensor Agent (SA)**

   Nodes acting as Sensor Agent send annotated or raw sensor data to the semantic and non-semantic applications, respectively.

It is important to mention that a powerful node can play the role of multiple functional entities. For example, in the Figure 4-1, a GTO node acts as both AA and SA while a *Type B* sensor may act either AA or OA. The reason behind is that GTO nodes are much more resourceful with compared to *Type B* sensors.

We have adopted two overlays instead of one for two major reasons. First, if all the functional entities reside in the one overlay, it would be hard to manage tasks between them. Second, the solution becomes less scalable with one overlay when the network size (e.g., the number of sensors) increases. The overlays with their tasks are described below:

### 1. Annotation Overlay

The annotation overlay is dedicated for sensor data annotation. This layer consists of Annotation Agents (AAs) and Sensor Agents (SAs). The semantic virtual sensors send raw data to the AAs. Then, AAs annotate the received data using the base ontology. The detailed procedure of sensor data annotation is described later in Section 4.3. AA communicates with the SAs to send the annotated data to the semantic applications. SAs also send the raw data received from the non-semantic virtual sensors to the non-semantic applications. For this reason, SAs can be considered as the exit gates of the virtual sensor access layer.

### 2. Ontology overlay

The ontology overlay is responsible for storing the base ontology in a distributed manner and serving ontology when a request comes from the annotation overlay. This overlay consists of Ontology Agents (OAs). OAs act as super-peers which are responsible for storing the base ontology and providing the requested ontology to the AAs. The OAs require sufficient storage space and an efficient request/response mechanism. There are two types of nodes which can act as OAs: (a) GTO nodes, and (b) *Type B* sensors. As GTO node is more powerful, it stores the complete base ontology. On the other hand, *Type B* sensors store a part of the base ontology. The functional entities belonging to ontology overlay are not involved in processing sensor data.

## *D. Application Overlay Layer*

The final and fourth layer, Application Overlay Layer comprises with multiple applications including semantic and non-semantic applications and shares the same virtualized WSN

deployment. Semantic applications receive annotated data while non-semantic applications receive raw data from the SA in the Virtual Sensor Access Layer.

Apart from the four layers, there is an Operations & Management (O&M) entity which is usually the WSN IaaS provider and responsible for providing the base ontology. Since O&M entity is aware of the type of sensors deployed in the WSN, it can easily develop and deploy the base ontology to the ontology overlay.

### 4.1.4 Interfaces

The proposed architecture provides two standardized interfaces (*Pdi* and *Di*) for the interactions between the end-user applications (both semantic and non-semantic applications) and with the different functional entities in the overlay network.

1. **Proprietary Data interface (*Pdi*)**

   It is used by the virtual sensors to send the sensor data to the functional entities (AA, SA) in the annotation overlay.

2. **Data interface (*Di*)**

   It is used by Sensor Agent (SA) to send the raw data received from the virtual sensors to the non-semantic applications. The same *Di* interface is also used by SA to send the annotated data received from the Annotation Agent (AA) to the semantic applications.

*Pdi* and *Di* use RESTful interface for the interaction. The reason behind choosing RESTful web services are as follows: (a) lightweight, standards-based, supports different data formats (e.g., plain text, JavaScript, JSON, and XML), (b) provides a uniform interface as the REST resources can be accessed and manipulated in a standard way.

## 4.2 Base Ontology

In order to provision semantic applications, we need to send additional metadata along with the raw sensor data. For example, the raw sensor data for a fire monitoring application can be annotated with concepts such as observed property and location (e.g., temperature, longitude, and latitude in this case). Semantic data annotation has been a popular approach for this purpose. However, the data annotation process requires domain concepts and the relationships that exist between them to annotate data. An ontology is used to represent formally a domain, its concepts and the relationships that exist between them [5]. Within sensor domain, there are several efforts to develop ontologies. For example, the Semantic Sensor Network (SSN) Ontology developed by the W3C Semantic Sensor Network Incubator Group [58] and SensorML from the Open Geospatial Consortium (OGC) [59]. SSN ontology is more general purpose because it is application domain independent.

We develop our 'base ontology' by extending the SSN ontology since it is a well-known, standard, and widely used to describe sensors descriptions and their observations. The base ontology contains the concepts of different capabilities and types of sensors deployed in the WSN infrastructure. Since a single physical sensor may have multiple sensing capabilities (e.g., JAVA SunSpot sensor has both light and temperature capabilities), we need to add all the related concepts in the base ontology. If WSN infrastructure consists of temperature, humidity, light, and carbon sensors, then the base ontology contains the concepts of these types of sensors and their observations. Figure 4-2 shows part of the base ontology associated with the temperature sensors. Some of these concepts and properties are reused directly from the SSN ontology as these concepts already exist. We created the rest of the concepts based on our data annotation requirement.

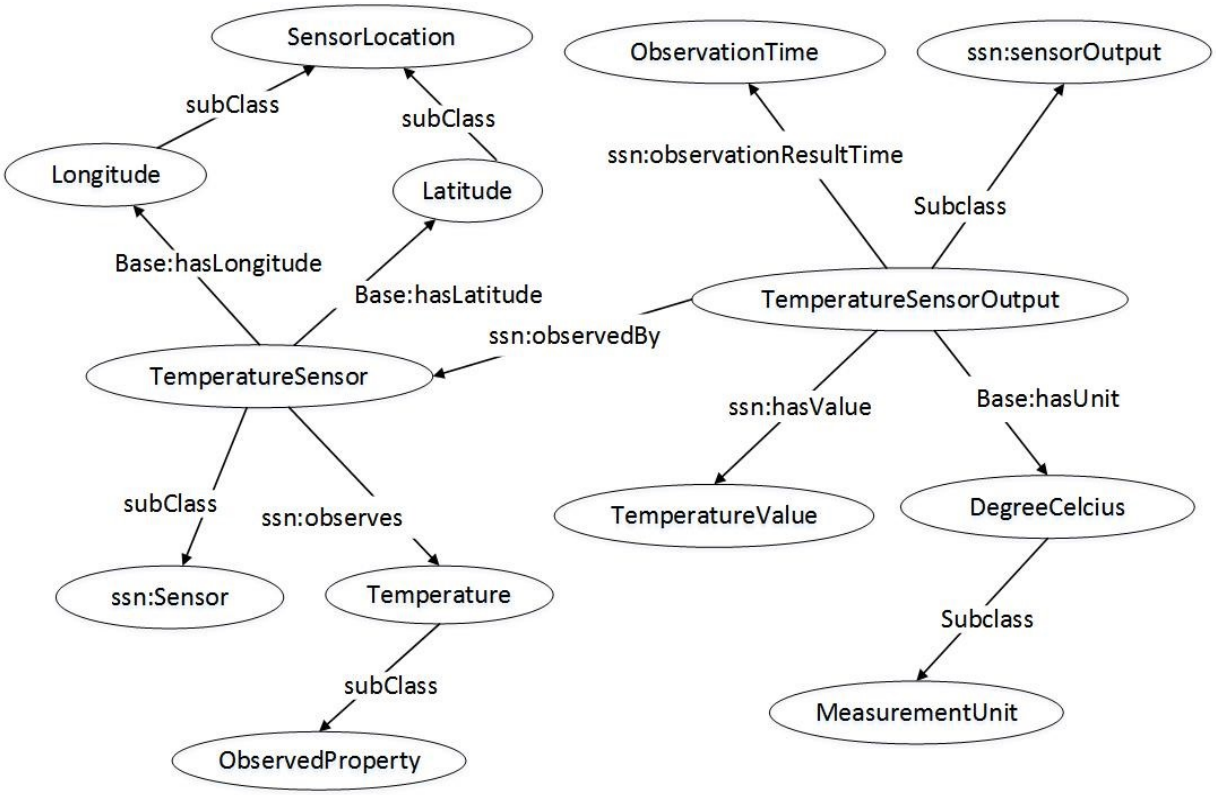


**Figure 4-2 Partial base ontology for Temperature Sensor**

A base ontology consists of six main parent classes: Sensor, ObservedProperty, SensorOutput, MeasurementUnit, ObservationTime, and Sensor location. We define each class below:

- **Sensor**

  This class defines information about the sensors having various capabilities. When a sensor with new capability is added in the WSNs, the corresponding concept is added as a subclass of the Sensor class. In Figure 4-2, TemperatureSensor is a subclass of the Sensor class. If the network size grows with the different capability sensors, sensor classes for new capabilities are added to the Sensor class.

- **ObservedProperty**

  This class defines the physical event that is observed by sensors. Sensor observes various events based on its sensing capability. If we consider the example of a temperature sensor, then the physical event would be the temperature that is observed by TemperatureSensor. In our base ontology, Temperature class is added as a subclass of the ObservedProperty.

- **SensorOutput**

  This class defines the sensor output. As WSNs consist of sensors having different capabilities, this class is introduced to distinguish different sensor output. For example, TemperatureSensorOutput and HumiditySensorOutput are the two concepts added under SensorOutput class to differentiate between the temperature and humidity sensor output.

- **MeasurementUnit**

  This class contains information about unit of measure of the sensor data. For example, if temperature sensor gives temperature reading in degree celsius, then the DegreeCelcius class is added as a subclass of MeasurementUnit in the base ontology.

- **ObservationTime**

  This class defines the time when a sensor senses an event.

- **SensorLocation**

  This class indicates sensor's position. As a sensor's position is determined by the longitude and latitude, these two concepts are added under the SensorLocation class.

## 4.3 Procedures for Data Annotation

Data annotation process starts in the operating phase after the deployment of WSNs with all necessary configurations are completed. Figure 4.3 shows the sequence diagram of the data annotation procedure in an abstract way.

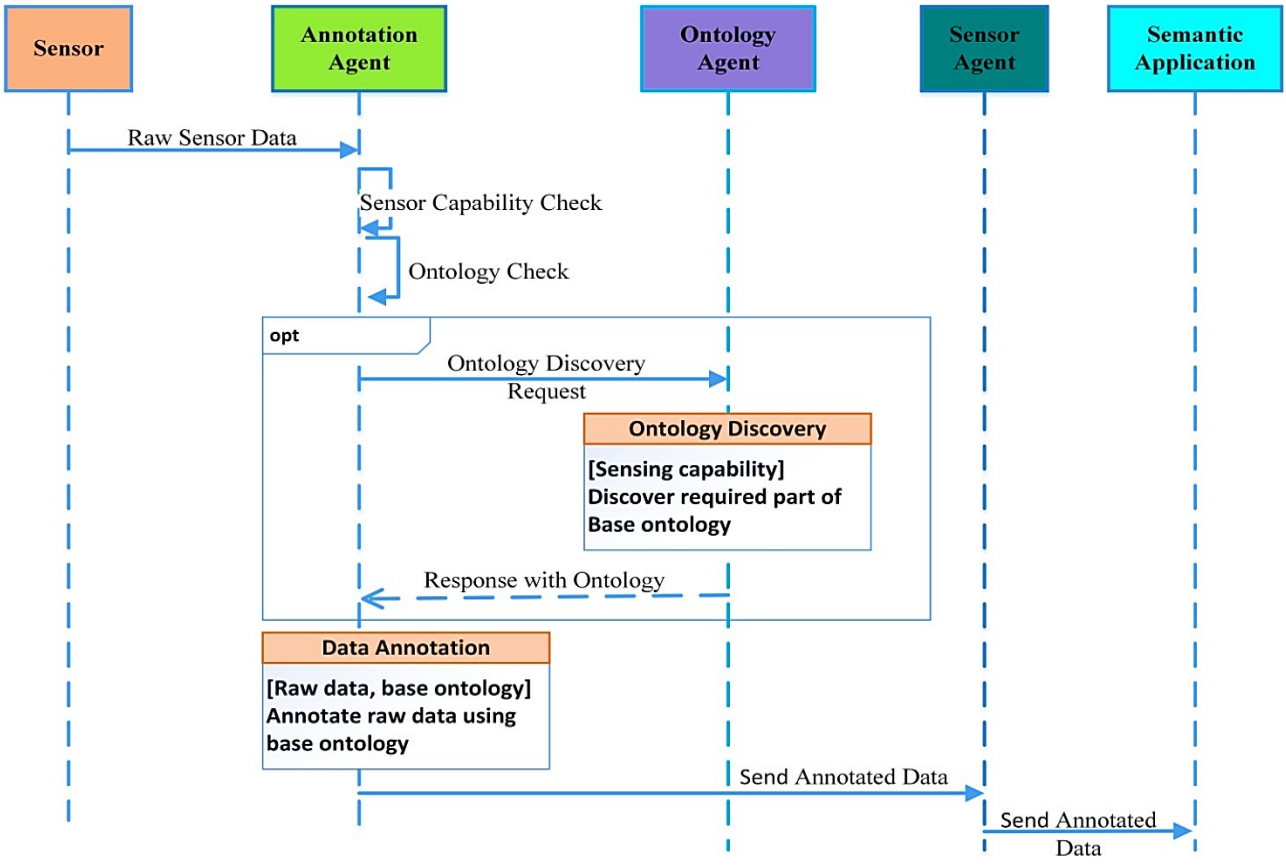


**Figure 4-3 Data annotation procedure**

First, when a physical sensor (e.g., Java Sunspot, AdvanticSys sensor) senses a phenomenon (e.g., temperature), it sends the data (*Raw Sensor Data*) to the Annotation Agent (AA).

Second, Annotation Agent (AA) determines (*SensorCapabilityCheck*) the type of phenomenon (e.g., temperature) recorded in the received *Raw Sensor Data.*

Annotation Agent (AA) checks whether it has the corresponding partial base ontology (e.g., temperature ontology) or not. If it does not have the partial ontology, it initiates the ontology discovery process which is explained in Section 5.4. In brief, it sends an *OntologyDiscoveryRequest* to the Ontology Agent (OA) and receives the required partial ontology.

Third, Annotation Agent (AA) annotates the data using the partial ontology which is explained later in this section. It then sends the annotated data to the Sensor Agent (SA).

Finally, Sensor Agent (SA) sends the annotated data to the Semantic Applications.

Algorithm 1 represents the pseudo code of the data annotation procedure that is executed in the Annotation Agent (AA).

**Algorithm 1** Data Annotation

Inputs:

$\Re$ : Raw sensor data

$\Theta_p$ : Partial ontology for phenomenon p

Outputs:

$\Lambda$ : Model representing annotated sensor data

Externals:

$\gamma(.)$: Parses inputs

$\beta(.)$: Extracts concepts

$\delta(.)$: Finds a concept in $\Lambda$ and populates it from a given keyvalue

Procedure:

1:$\{kv\} \leftarrow \gamma(\Re)$
2:$\{p\} \leftarrow \beta(\Theta_p)$
3:$\Lambda \leftarrow null$
4:**for** *each concept p in* $\{p\}$ **do**
  **for** *each keyvalue kv in* $\{kv\}$ **do**
   $\Lambda \leftarrow \delta(kv)$
  **end for**
 **end for**
5:**return** $\Lambda$

**Figure 4-4 Pseudo code for data annotation**

This algorithm takes raw data, a partial ontology as inputs and produces annotated data as output. However, the raw data can be plain text or any standard formatted data. In Line 1, raw data is parsed and stored as key-value pairs. Line 2 determines the concepts from the given partial ontology. The ontology will be used to create semantic relationship among the observed data. A model (represents the ontology concepts and holds their values) is initialized which represents the annotated data in Line 3. In Line 4, each key-value from the key-value pairs (Line 1) is mapped to the corresponding ontology concept (extracted in Line 2) to create the semantically annotated data. Finally, the annotated data is returned in Line 5. Section 4.4.2 presents an example to illustrate the algorithm step by step.

## 4.4 Wildfire Monitoring Use case

In this section, we describe the wildfire monitoring application based on the proposed architecture. Let us assume that the WSN IaaS providers deploy sensors of brands Java SunSpot (e.g., temperature & light sensors) and AdvanticSys Kit (humidity sensors) in a large geographic area including nearby forest, public streets, and private homes, respectively. These sensors run multiple application tasks concurrently using virtual sensors and semantic virtual sensors. In the following subsections, we describe the base ontology and data annotation procedure.

### 4.4.1 Base Ontology

As WSN IaaS provider is aware of the deployed sensors, he develops a base ontology. The base ontology development mechanism is discussed in the next chapter. In the current sensor

deployment, a base ontology contains concepts related to the temperature, humidity and light sensors. Figure 4-3 shows the concepts of the base ontology.

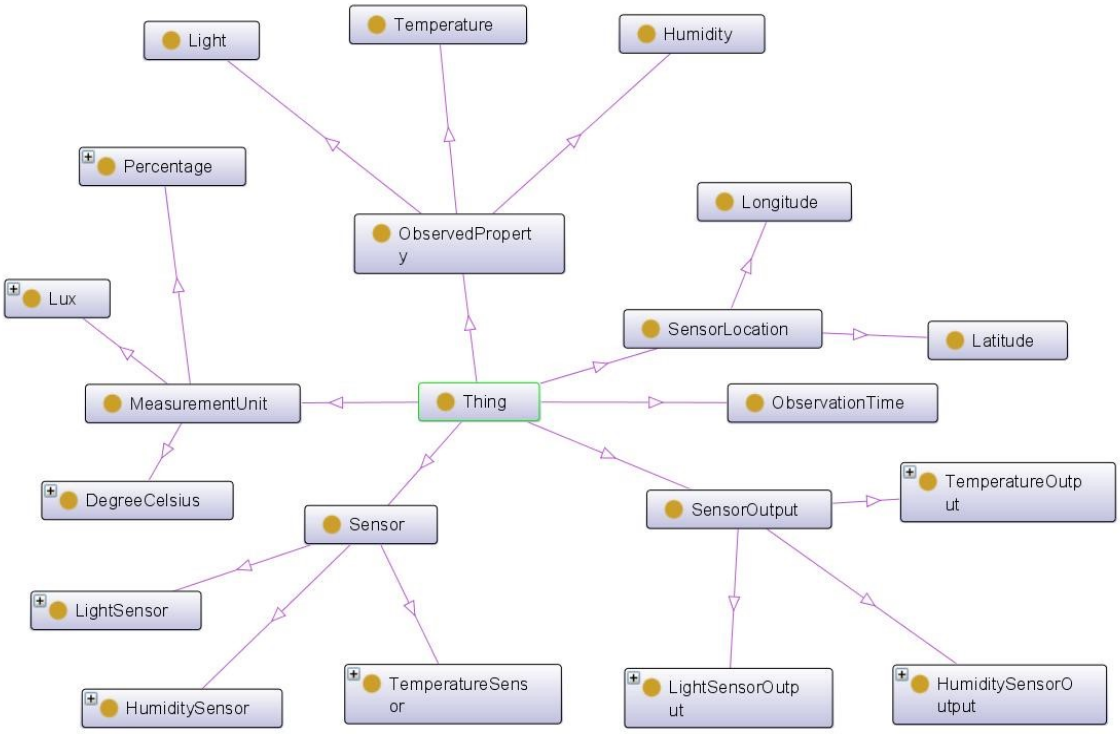


**Figure 4-5 Concepts of base ontology**

As we mentioned earlier, we extended SSN ontology by re-using some of the concepts. Figure 4-5 shows the parent-child class hierarchy relationships among different concepts. While developing an ontology, all the concepts should be under the concept of *Thing*. In our base ontology, we have five parent classes (*Sensor, ObservedProperty, SensorOutput, MeasurementUnit,* and *SensorLocation*). As WSN IaaS contains sensors of three different capabilities (e.g., temperature, light, humidity) we have three subclasses (*TemperatureSensor, LightSensor*, and *HumiditySensor*) under the parent class *Sensor*. Concepts of various sensing capabilities (e.g., *Temperature, light,*

and *Humidity*) are presented as subclass under the *ObservedPropoerty* class. In order to distinguish the sensor outputs, *TemperatureOutput, HumiditySensorOutput,* and *LightSensorOutput* subclasses are incorporated in the parent class *SensorOutput.* Java Sunspot temperature sensors produce output in degree Celsius, and light sensors produce output in lux unit. However, AdvanticSys Kit humidity sensors provide relativity measure of humidity. concepts for these measurement units (*DegreeCelcius, Lux,* and *Percentage*) are presented as a subclass of the *MeasurementUnit* class. As a sensor's physical position is determined by longitude and latitude value, *longitude* and *latitude* concepts are presented under the class *SensorLocation.* It is very critical to know the sensing time of an event in any WSN application. *ObservationTime* class defines the timestamp when a sensor senses an event.

In our proposed architecture, we assumed that the base ontology has already been provisioned in the WSNs after deploying the sensors. However, the detail description of the ontology provisioning in WSN can be found in Section 5.3. According to this assumption, the OAs in the ontology overlay already holds the base ontology. This means that the GTO node has the full base ontology and the *Type B* (Java Sunspot) sensors have part of the base ontology. As AdvanticSys Kits are more resource-constrained sensors, these are connected to the GTO node and send sensor reading to the AAs in virtual sensor access layer. GTO node can be the gateway or a sink node.

### 4.4.2 Procedures for Data Annotation

After the deployment of the WSNs and all other setup, Java Sunspot, and AdvanticSys sensors start giving their reading and send their raw data in a standard format to the AAs in the virtual sensor access layer. Let consider a JAVA Sunspot temperature sensor (Spot1) is giving the reading in the

SenML format [60] and an AdvanticSys humidity sensor (TlosB2) is giving the reading in the plain text. Figure 4-6 shows the different type of input data received by the AA from the virtual sensors of Spot1 and TlosB2.

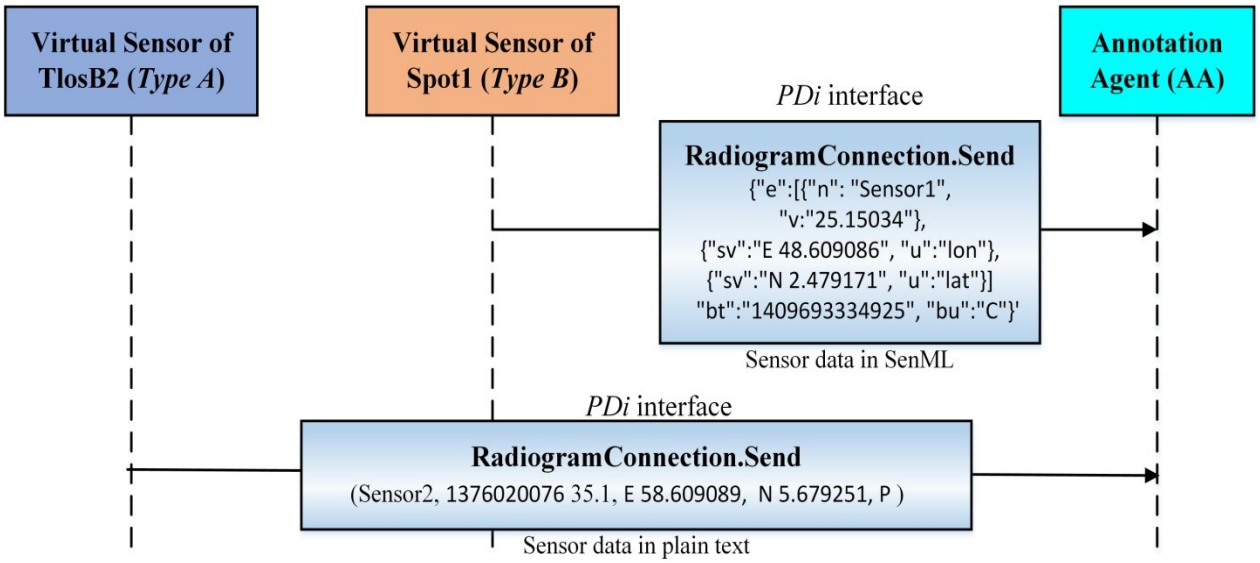


**Figure 4-6 Example of sensor data before semantic data annotation**

For each sensor reading, the following steps are executed by the AA. We consider the example of SenML data receiving from the Spot1 sensor.

*First,* AA determines the sensing capability (e.g., temperature or humidity reading) and finds it as a temperature reading. Sensor capability check depends on the configuration setup of the WSNs performed by the WSN provider. In our implementation, we used sensor id to determine the capability of the sensor.

*Second*, AA looks for the temperature part of the base ontology in its local storage. If the ontology is not found, AA sends an ontology discovery request for the required base ontology (temperature

sensor related concepts) to the OA in the ontology overlay. OA replies back with the requested temperature ontology.

*Third*, AA executes the data annotation algorithm by performing the following steps:

**Step 1:** The input data is parsed and stored as key-value pairs. Figure 4-7 shows how the input data can be extracted and stored as a key value pair.

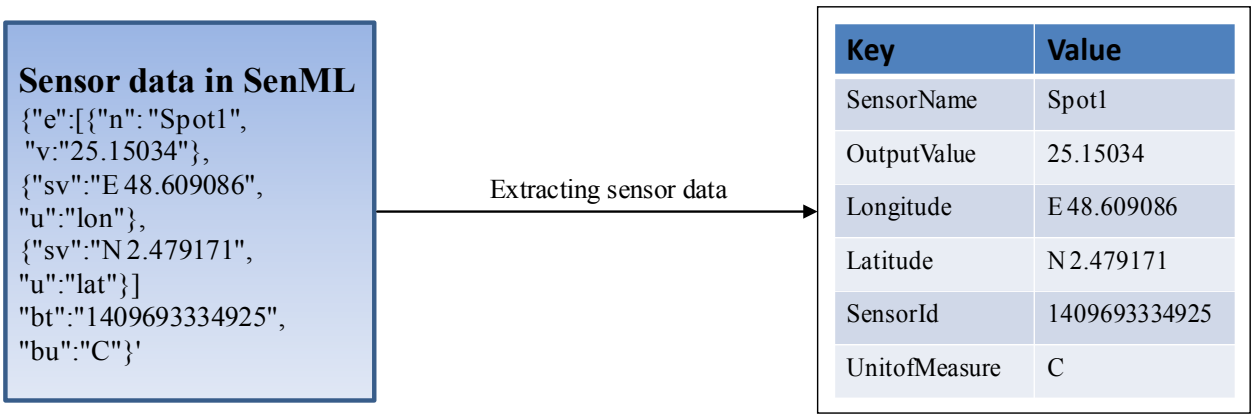


**Figure 4-7 Extracted sensor data**

**Step 2:** AA extracts the concepts of the temperature part of the base ontology as these concepts would be used to add the metadata to the actual data. The temperature part of base ontology has already been shown in the Figure 4-2.

**Step 3:** A new data model is initialized in which the annotated data would be stored. The new data format of the data model depends on the implementation choice. There are several data formats (e.g., RDF/XML, OWL/XML, Turtle, and so forth) which can be used to represent semantically enriched annotated data. In this implementation, we have chosen the RDF/XML format as this is the most popular and widely used all around the world.

**Step 4:** Each key-value pair is taken out and mapped with the corresponding concepts of the ontology. Properties are used to create a semantic relationship between the concepts and observed data. For example, AA receives temperature data from the sensor named 'Spot1' located in X(E 48.609086, N 2.479171) position. The partial ontology has related concepts including '*TemperatureSensor*', '*Temperature*' as well as properties including '*observes*', '*hasLongitude*', and '*hasLatitude*'. Using semantic annotation, AA can represent the sensor data similar to the following set of information;

- Spot1 is a type of TemperatureSensor
- Spot1 observes Temperature
- Spot1 has longitude E 48.609086
- Spot1 has latitude N 2.479171

This piece of information can be expressed in the RDF/XML format presented in the Figure 4-8. As we mentioned earlier, data can be simply tagged using existing standards (e.g., XML, JSON) but that representation cannot create semantic relationship among the observed data.

```
<rdf:Description rdf:about="http://BaseOntology.owl#Spot1">
<rdf:type rdf:resource="http://BaseOntology.owl#TemperatureSensor"/>
<ssn:observes rdf:resource="http://BaseOntology.owl#Temperature"/>
<base:hasLongitude> E 48.609086 E 48.609086</base:hasLongitude>
 <base:hasLatitude> N 2.479171</base:hasLatitude>
 </rdf:Description>
```

**Figure 4-8 Annotated sensor data**

There are several open source APIs (e.g., Apache Jena, OWL API) that can be used to create semantically annotated data. Figure 4-9 shows the annotated data in RDF/XML generated by the AA using base ontology.

**Step 5**: After generating the annotated data, AA forwards it to the Sensor Agent (SA) in the same overlay.

**Step 6:** Finally, SA sends the annotated data to the wildfire monitoring semantic application.

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:base="http://BaseOntology.owl#"
    xmlns:ssn="http://purl.oclc.org/NET/ssnx/ssn#"
   <rdf:Description rdf:about="http://BaseOntology.owl#SpotOutput1">
   <base:hasUnit rdf:resource="http://BaseOntology.owl#DegreeCelsius"/>
   <base:hasSensingTime rdf:datatype=
   "http://www.w3.org/2001/XMLSchema#dateTime">2014/09/01T09:12:55</base:hasSensingTime>
   <ssn:hasValue
rdf:datatype="http://www.w3.org/2001/XMLSchema#double">25.15034</ssn:hasValue>
   <ssn:observedBy rdf:resource="http://BaseOntology.owl#Spot1"/>
   <rdf:type rdf:resource="http://BaseOntology.owl#TemperatureOutput"/>
    </rdf:Description><rdf:Description rdf:about="http://BaseOntology.owl#Spot1">
   <rdf:type rdf:resource="http://BaseOntology.owl#TemperatureSensor"/>
   <ssn:observes rdf:resource="http://BaseOntology.owl#Temperature"/>
   <base:hasLongitude> E 48.609086 E 48.609086</base:hasLongitude>
   <base:hasLatitude> N 2.479171</base:hasLatitude>
   </rdf:Description>
</rdf:RDF>
```

**Figure 4-9 Annotated data received from Spot1 Sensor**

The wildfire monitoring application can further infer implicit knowledge from the received annotated data. The type of knowledge inference depends on the application domain and end-user requirements. In this thesis, we have created a fire domain ontology along with a set of reasoning rules to infer the knowledge like "*No fire*", "*Tends to fire*" or "*huge fire"*. The details can be found in Section 6.1 where we present our prototype implementation.

## 4.5 How the Architecture meet the Requirements

The proposed architecture fulfills all the data annotation requirements that we mentioned in Chapter 3. First, our proposed data annotation architecture can support near real time in-network annotation in the virtualized WSNs. We have used two overlays: (a) base ontology is deployed among the

nodes in the ontology overlay, and (b) sensor data annotation is performed by the nodes in the annotation overlay. In both cases, we use distributed approaches for storing the base ontology and performing sensor data annotation. Our proposed architecture allows the node-level-virtualization of the sensor nodes. As a result, multiple application tasks can be run concurrently on top of a physical node. For the standard representation of the sensor data and management of the sensor heterogeneity issue, we have proposed base ontology following the existing standards. We extend the SSN ontology to develop our base ontology. The virtualized WSNs send the annotated data to the semantic applications through a standard interface. We have used REST interface to establish a connection between the WSN infrastructure and Semantic Wildfire Monitoring Application. The architecture is platform independent and supports both semantic and non-semantic applications. Our proposed architecture is applicable for a large-scale sensor deployment while the physical layer can support heterogeneous sensors. In this way, the scalability and sensor heterogeneity requirements are satisfied.

## 4.6 Chapter Summary

In this chapter, we described our proposed data annotation architecture which is based on the virtualization and network overlay concepts. We first presented the architectural principles and then we explained the architecture in detail including the layers and functionalities. We explained the base ontology and sensor data annotation procedure. We described the use case of the wildfire monitoring application and showed the concepts of the base ontology and data annotation procedures step-by-step. Finally, we explained how our proposed architecture meets and satisfy all the requirements presented in Chapter 3.

# Chapter 5

## 5 Ontology Provisioning Architecture

We presented data annotation architecture for provisioning semantic applications in virtualized WSNs in Chapter 4. We explained about base ontology which was used to annotate the sensor data. However, in-network sensor data annotation requires base ontology provisioning in the virtualized WSNs. Ontology provisioning includes ontology development, deployment, and management in virtualized WSNs.

In Chapter 4, we assumed that the base ontology was provisioned in the virtualized WSNs. This chapter tackles the challenge of ontology provisioning. We extend the architecture presented in Chapter 4 for ontology provisioning and refer it as ontology provisioning architecture. We have introduced "ontology provisioning center" as a part of the ontology provisioning architecture and proposed a protocol for deploying base ontology in the virtualized WSNs.

This chapter consists of the following items: first, we present the ontology provisioning architecture which is an extension of our data annotation architecture described in the previous chapter to allow ontology provisioning in WSNs. The "ontology provisioning center" for base ontology development and management is outlined in the second section. In the third section, we propose a new protocol for ontology deployment in the virtualized WSNs which is used in several procedures for base ontology provisioning. A wildfire monitoring use case is presented to show the overall workflow of the ontology provisioning. Finally, we discuss how the requirements mentioned in Chapter 3 are met by the architecture.

## 5.1 Ontology Provisioning Architecture

In Chapter 4, we proposed an in-network data annotation architecture assuming that base ontology was provisioned. We have utilized our previous architecture as a foundation and extended it for provisioning base ontology in the virtualized WSNs. The virtual sensor layer and application overlay layer remain unchanged as in the data annotation architecture. We have introduced new functional entities in the physical layer and virtual sensor access layer. The overall architecture is presented in the Figure 5-1 which is the final architecture for supporting both ontology provisioning and data annotation in the virtualized WSNs.

In this section, we describe only the new components of the architecture which are summarized in the Table 5-1.

- **WSN IaaS Manager**

  This entity is a powerful physical node (e.g., server) with the overall knowledge of the deployed WSN IaaS, which belongs to the physical layer. It represents the WSN Infrastructure Manager (WIM) in the virtual sensor access layer. In our architecture, the functional entity WIM is very crucial as it executes the most important tasks related to the ontology deployment. In general, sensors are not reliable since they are resource constraint devices with limited battery life. In this regard, we chose WSN IaaS manager as a powerful node. The primary responsibility of this component is to provide 24/7 reliability to the WIM.

- **Ontology Manager (OM)**

  This entity belongs to the ontology overlay in the virtual sensor access layer. The primary responsibility of OM is to store the complete base ontology and send it to the OAs upon request from them. However, OAs store the partial base ontology. In order to ensure

availability of the ontology in the case of node failures, we replicate the base ontology among multiple OMs in this architecture. Powerful nodes such as GTO nodes can act as OMs.

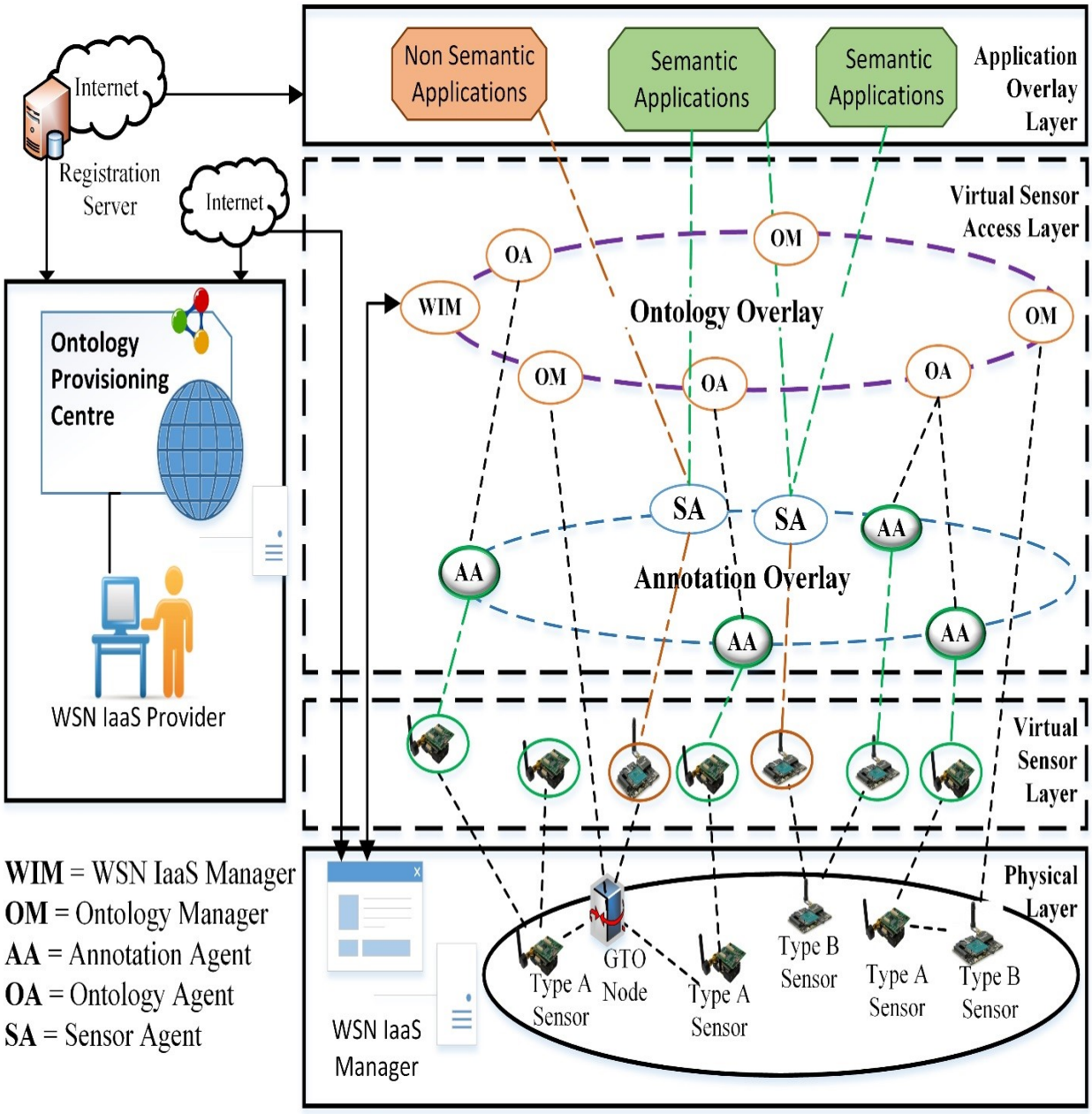


**Figure 5-1 Ontology provisioning architecture in a virtualized WSN**

- **WSN Infrastructure Manager (WIM)**

The WSN Infrastructure Manager (WIM) is another new functional entity added in the ontology overlay. The WIM is the logical entity of WSN IaaS Manager belonging to the physical layer as a centralized entity. Deployed WSNs may contain both resource constraint and resourceful (we use the term 'capable') sensor nodes. In our architecture, only capable nodes can act as OM or OA, which store the base ontology. It is very crucial to find the capable nodes among the different types of the nodes belongs to the deployed WSNs. WIM performs the following two tasks:

(i) *Selects the capable nodes to act as either OM or OA*.

In this regard, we have adopted a genetic algorithm recently proposed by our research team [61]. WIM executes the algorithm to find out the most appropriate nodes which can play the role of OM and OA.

(ii) *Deploys the ontology concepts on the selected node*.

We have explained the ontology deployment protocol in Section 5.3 which is executed by WIM.

| **Functional Entity** | **Position** | **Responsibility** |
|---|---|---|
| **WSN IaaS Manager** | Physical Layer | Provide reliability support to WIM |
| **Ontology Manager (OM)** | Virtual Sensor Access Layer | Holds the full base ontology |
| **WSN Infrastructure Manager (WIM)** | Virtual Sensor Access Layer | i) Selects the capable nodes that can act as OMs or OAs ii) Deploy entire base ontology to OMs and partial base ontology to OAs |

**Table 5-1 New functional entities introduced in the ontology provisioning architecture**

## 5.2 Ontology Provisioning Center

Base ontology can be considered as an abstract image of the physical sensors deployed in the WSNs because it holds all the related concepts of the physical infrastructure. For this reason, it is very necessary to develop, deploy, and manage the base ontology after deploying WSN infrastructure. We have proposed an ontology provisioning center to develop and manage the base ontology. Figure 5-2 shows the components of the ontology provisioning center.

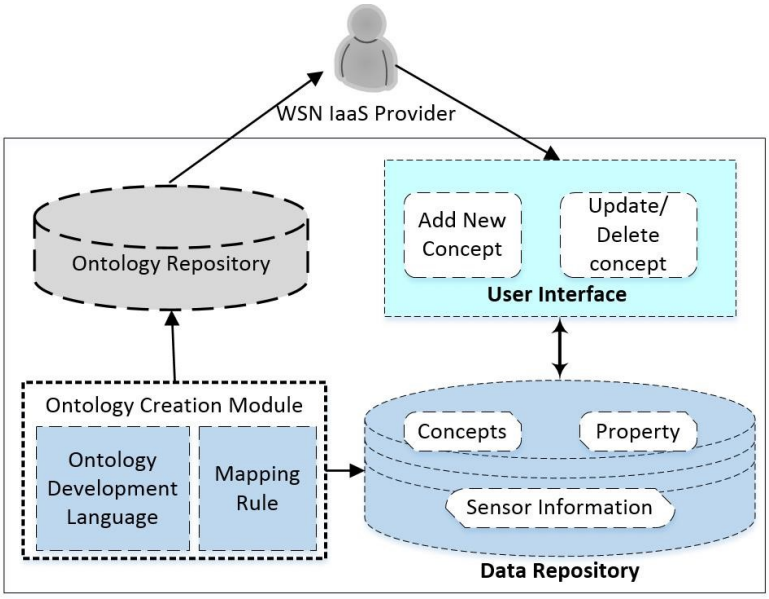


**Figure 5-2 Components of the ontology provisioning center**

WSN IaaS provider interacts with the ontology provisioning center through a user interface. Figure 6-10 shows a user interface for the implemented prototype. WSN IaaS provider adds new concepts after deploying the sensors in the WSN IaaS. The ontology creation module is responsible for generating the base ontology by taking the concepts stored in the data repository. We develop a set of mapping rules and use the ontology development language to create the base ontology. Finally,

the base ontology is stored in an ontology repository. IaaS provider can store, modify or remove concepts of the base ontology. In the case of removing some of the existing sensors from the network, WSN IaaS provider can remove the concepts related to those sensors.

The process of creating base ontology using the ontology provisioning center is described in Figure 5-3 (illustrated in Figure 5-6 and 5-7 in Section 5.5) involving the following steps:

- ***Add Concept***

  Infrastructure Provider defines the concepts according to the sensor domain. For each capability, a small piece of information needs to be specified including sensor type, output type, output unit and observed property.

- ***Add Sensor Details***

  In the case of deployment of a new kind of sensor, its information can be easily added to the ontology. For example, its name, sensor id, sensor type, the number of attached sensors, and the domain & range values it supports.

- ***Load Default Ontology***

  Once the new concepts are included in the system, a default ontology is used to incorporate these new concepts in this step. We use some of the concepts directly from standard SSN ontology [58] in the default ontology.

- ***Apply Mapping Rules***

  The new concepts (we refer these as 'child concepts') are included with the existing concepts by applying mapping rules.

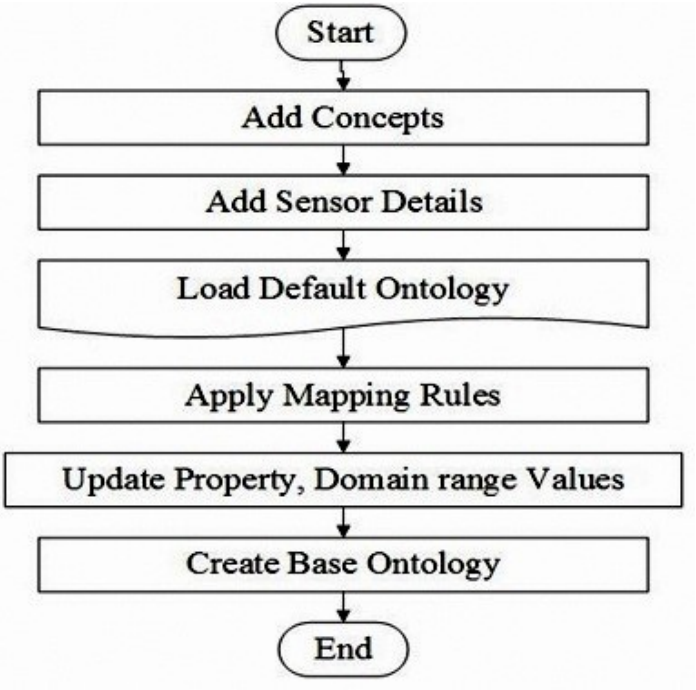


**Figure 5-3 Steps for Base ontology development**

- ***Update Property, Domain, and Range values***

  The values of the properties, domains, and ranges are updated to reflect the new additions or modifications to the concept.

- ***Create base ontology***

  The base ontology is created from the updated concepts, properties, domains, and ranges. The base ontology can be expressed as the collection of all the added concepts as below:

$$\text{Base ontology} = \text{concept}_{\_\text{temperature}} + \text{concept}_{\_\text{humidity}} + \ldots + \text{concept}_{\_k}$$

## 5.3 Ontology Deployment Protocol

This section discusses our proposed protocol for deploying the base ontology in the overlay network. The proposed protocol exchanges messages:

a) From the ontology provisioning center to the WIM belonging to the ontology overlay (*Deploy Ontology* and *Ontology Deploy Request*), and

b) WIM to other nodes in the ontology overlay

The ontology provisioning center uses the *Deploy Ontology* message to deploy a base ontology to the WIM. This situation might happen when the base ontology is created, or WIM requests the most updated version of the base ontology.

As the ontology overlay is responsible for holding the base ontology, the rest of the messages are exchanged between nodes of the ontology overlay. Table 5-2 summarizes the messages exchanged for the proper deployment of the base ontology.

| **Message Name** | **Message Description** | **Message Address** |
|---|---|---|
| *Deploy Ontology* | A full base ontology is deployed by the ontology provisioning center to WIM upon being developed. | Unicast |
| *Ontology Deploy Request* | A request message is sent from WIM to the ontology provisioning center to get the most updated version of the base ontology | Unicast |
| *Discover Request* | Discover the capable nodes that form the ontology overlay. Send the message by WIM to the nodes in the ontology overlay | Broadcast |
| *Discover Response* | Send the response to WIM by individual capable nodes and include the node ID in the response message. | Unicast |
| *Notification* | WIM notifies the selected OAs and OMs by sending the notification message. | Multicast |
| *Acknowledgement* | Send an acknowledge message to WIM by individual nodes. | Unicast |
| *Ontology Request* | A request message is sent from OM to WIM to get the ontology. | Unicast |
| *Set full Base Ontology* | WIM sets an entire base ontology to OM. | Unicast |
| *Set Partial Ontology* | WIM sets a partial base ontology to OA | Unicast |

**Table 5-2 Message exchanges for the deployment of ontology**

Sensors are tiny devices with low battery life to serve a set of specific tasks including detecting events, sending data, and maintaining communication in the WSN. These tasks require a large amount of energy which minimizes the battery lifetime. We find the average battery lifetime of the Java SunSpot sensors is around 10 hours (experimental results are shown in the Figure 6-11). Most of the deployed sensors in a WSN do not have a facility for battery recharging or replacement. Once a battery life finishes for a sensor, it cannot participate in the network anymore. For this reason, extensive research works are done on the efficient use of sensors' energy consumption. For the same reason, developing routing protocol, defining topology and designing every other context of a WSN need to consider the battery life of the sensors in the beginning.

In our architecture, we assume the sensors in the WSNs have different levels of battery life, processing power, and hardware ability. The intuition behind this is the sensors with the higher abilities perform more tasks including computation, routing, communication, and reporting. Ontology provisioning requires holding the ontology and serving to other nodes in case of ontology request. For this reason, we need to select a set of sensors which meet certain requirements. We refer these sensors as 'capable nodes'. These sensors meet pre-defined requirements such as battery life should be greater than X days, storage capacity should be greater than Y KB, processing abilities (such as memory, operating system, hardware) should be higher than a certain threshold. We assume that only these capable nodes can act as OM or OA in our architecture and hold the ontology completely or partially.

We have introduced following procedures that rely on our proposed ontology deployment protocol:

a) Discover potential candidates as capable nodes

b) Select the capable nodes

c) Notify the capable nodes

d) Deploy base ontology to the selected nodes

### a) Discover potential candidates as capable nodes

In large-scale WSN deployments, we select the potential candidates first in this procedure then finalize the capable nodes in the next procedure. The intuition behind this is to filter out a large number of sensors which can't meet the requirements to become capable nodes. The mechanism for candidate discovery uses the third and fourth messages presented in the Table 5-2. At the initial stage, WIM broadcasts the discovery request message to its neighbors. Table 5-3 summarizes the contents of the discovery request message. The first field, *broadcast_id*, defines the broadcast message ID, which is set by WIM. The *source_address* holds the ID of the sender of the *discovery request* message. The third and fourth fields (*node_storage_capcity* and *node_battery_life)* are used to discover the potential candidates.

| |
|---|
| *broadcast_id* |
| *source_address* |
| *node_storage_capcity* |
| *node_battery_life* |

**Table 5-3 Content of the *discovery request* message**

Each overlay node maintains a small cache consisting of <*source_address*, *broadcast_id*> pairs to identify and reject the duplicate messages upon receipt. When a node receives a *discovery request* message, it inspects the message using *broadcast_id* to determine the duplicate messages. If the received message is not duplicate, it updates the cache and disseminates it to its neighbors. As a

result, each node may receive a *discovery request* message from its neighbors multiple times but broadcasts the message once. In this way, *broadcast_id* filters redundant messages to avoid 'broadcast storm' problem. A node also compares its storage capacity and energy level with the condition specified in the *discovery request* message. If it satisfies the requirements, it forwards a *discovery response* message with its ID to the immediate source node from which the *discovery request* message was received. Upon receipt of a *discovery response* message from a neighbor node, the immediate source nodes also perform a similar task. If an immediate source node meets the energy and storage requirements, it adds its ID to the ID list found in the receipt message. It also forwards the updated message to its immediate source node. On the other hand, if it does not meet the energy and storage requirements, it simply forwards the message to the immediate source node. This process continues until the response message is reached to the WIM. WIM can receive multiple response messages from the same neighbor nodes. However, it will consider the messages with the highest number of node IDs. In this way, WIM will get the aggregated response messages with the node ID of potential candidates. There will be the very low probability of an 'ACK explosion' problem as the message aggregation technique reduces the number of messages in the network.

### *b) Select capable nodes*

WIM executes the heuristic-based Genetic Algorithm (GA) recently proposed by our research team [61] to select the capable nodes from the potential candidate nodes discovered in the previous procedure. GA provides an optimal solution for selecting capable nodes that can hold ontology in the virtualized WSNs. GA takes the node ID of the potential candidates as input and produces a list of the capable node IDs that can act as OA and OM. GA starts with a random population of individuals (i.e., the solutions). The individuals are evaluated using a fitness function, and the fitted

ones are selected to undergo crossover and mutation operations to produce a new generation of individuals. This procedure is repeated for many generations until a terminating condition (e.g., maximum number of generations) is reached.

### *c) Notify capable nodes*

WIM sends the *Notification* message to the selected capable nodes. The message indicates that the targeted node defined in the content is chosen to act as OA and/or OM. Each OA and OM node send an *Acknowledgement* message to the WIM stating that it is ready to act as OA and/or OM upon reception of the *notification* message.

### *d) Deploy ontology to the selected nodes*

WIM sets the complete ontology to the OMs after receiving the Acknowledgement messages from the OMs and OAs. Since OAs are resource-constraint devices and may not need all concepts, WIM divides the base ontology into multiple parts in a way that each section contains partial ontology for one capability. Each of these parts is randomly sent to the selected OAs.

## 5.4 Base Ontology Discovery

In our architecture, OM nodes hold the complete base ontology and OA nodes hold the partial base ontology where AA nodes annotate data it receives from the semantic virtual sensors. As AA nodes annotate data, these nodes require partial ontology. In the previous section, we have presented the procedures for ontology deployment to OM and OA nodes. AA nodes receive the necessary partial base ontology from the OA nodes through the base ontology discovery which can be either proactive or reactive. We describe the procedures for base ontology discovery below:

### 1. *Proactive Base Ontology Discovery*

Figure 5-4 shows the sequence diagram of ontology discovery procedure in the proactive approach.

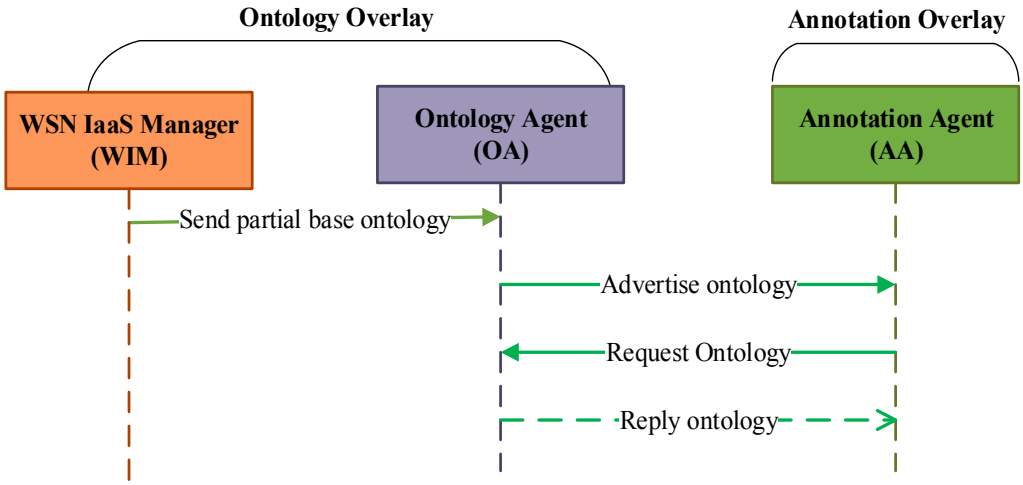


**Figure 5-4 Base ontology discovery in proactive approach**

In the previous section, we showed how the WIM deployed partial base ontology to the selected OAs. Each OA periodically advertises the partial base ontology it holds so that the AA nodes get the information. Each AA in the annotation overlay sends an ontology discovery requests in response to the advertisement. OAs reply back the requested ontology after receiving the request.

### 2. *Reactive Base Ontology Discovery*

The reactive approach is needed when an AA does not have the required part of the base ontology for annotation. The base ontology discovery procedure is illustrated in the Figure 5-5.

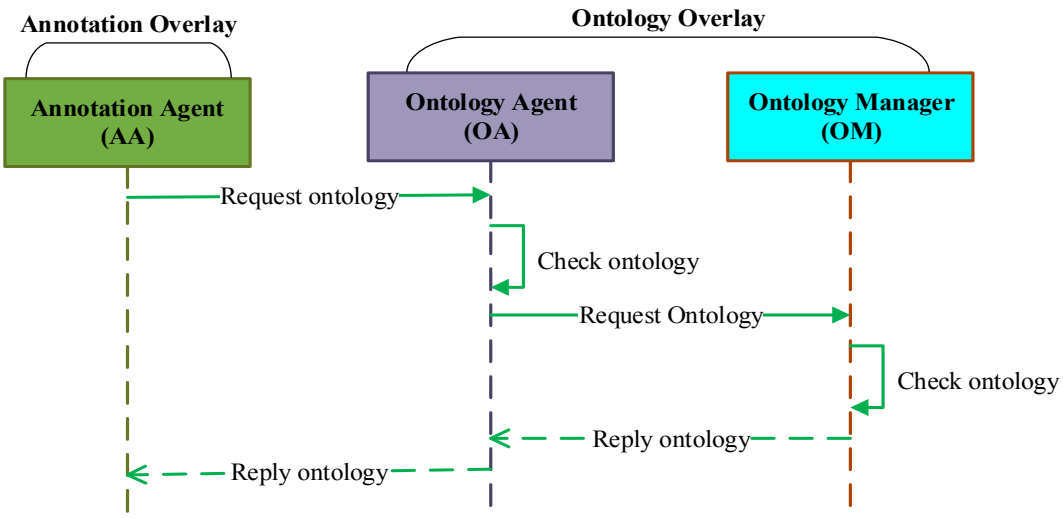


**Figure 5-5 Base ontology discovery in the Reactive approach**

The AA node sends an ontology discovery request message for the required part of the base ontology to an OA known from its cached advertisement. If the OA does not have the requested part of the base ontology, it sends an ontology discovery request to the OMs in the same ontology overlay as OMs contain the complete base ontology. Finally, OM replies back the request partial base ontology (such as ontology for temperature sensor) to the OA, which is then forwarded to the requesting AA.

## 5.5 Wildfire Monitoring Use case

We presented the wildfire monitoring use case to illustrate the semantic data annotation architecture in Chapter 4. We have extended the architecture in this chapter to allow ontology provisioning in the virtualized WSNs. In this section, we present the wildfire monitoring use case again to illustrate the new architecture focusing on the ontology development and deployment.

As we mentioned before, initially the WSN IaaS provider deploys sensors of brands Java SunSpot (temperature and light sensors) and AdvanticSys Kit (humidity sensors) on a large geographic area. WSN IaaS developer may not be an ontology developer or domain expert to develop base ontology by himself. He can develop the base ontology using our proposed ontology provisioning center by adding few information. Then he deploys the base ontology in the virtualized WSNs using ontology provisioning protocol.

Based on the scenario, we describe the ontology provisioning process. We divide the work process into two sections:

(i) Ontology development process using ontology provisioning center, and

(ii) Ontology deployment procedure from ontology provisioning center to the virtualized WSNs.

### 5.5.1 Base Ontology Development

The first step of the base ontology provisioning is to develop a base ontology. The base ontology development using ontology provisioning center is illustrated in the Figure 5-6.

In this scenario, we show the steps performed by the IaaS provider to develop a base ontology by adding concepts related to the Java Sunspot temperature sensors:

1. IaaS provider adds the concepts related to the temperature sensor (e.g., sensor type: TemperatureSensor, observed property: Temperature, output type: TemperatureOutput, unit: DegreCelcius) using the user interface.

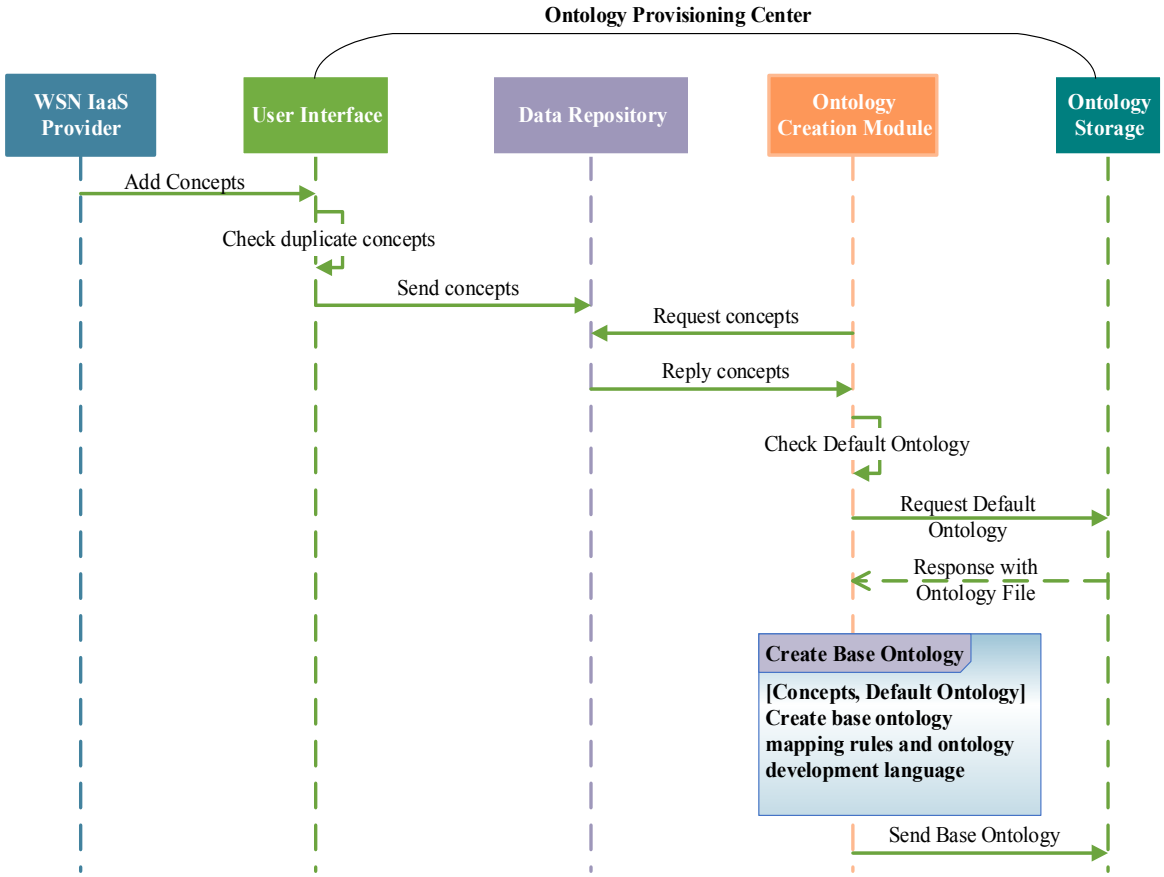


**Figure 5-6 Sequence diagram for base ontology development**

2. In the user interface of the ontology provisioning center, there is a process for checking duplicate concepts. When the IaaS provider wants to add new concepts, the system first checks whether the concepts are unique or not. If a duplicate concept is found, IaaS provider is notified that duplicate concept cannot be added to the system. For example, if there are already some existing concepts related to the temperature sensor in the data repository, the IaaS provider would not be allowed to add the redundant concepts. We have not addressed the issue when relevant concepts already exist in the data repository using different names.

Because it requires human effort to match and identify relevant concepts having different names (e.g., location: washroom, restroom). We believe that this type of situation will less likely happen in the sensor domain as most of the sensing phenomena are well-known.

3. The newly added concepts related to the temperature sensor are stored in the data repository. Following concepts are added for this use case:
    - Temperature
    - TemperatureSensor
    - TemperatureSensorOutput
    - DegreeCelcius
4. Ontology creation module requests for the concepts related to the temperature sensor in the data repository for creating a base ontology and checks its local storage for the default ontology. If the default ontology is not found, a request message is sent to the ontology storage.
5. Ontology storage replies back the requested ontology. The default SSN ontology already contains some core classes (e.g., Sensor, ObservedPropoerty, SensingUnit) and properties (e.g., observes, hasUnit) that create semantic links between the classes.

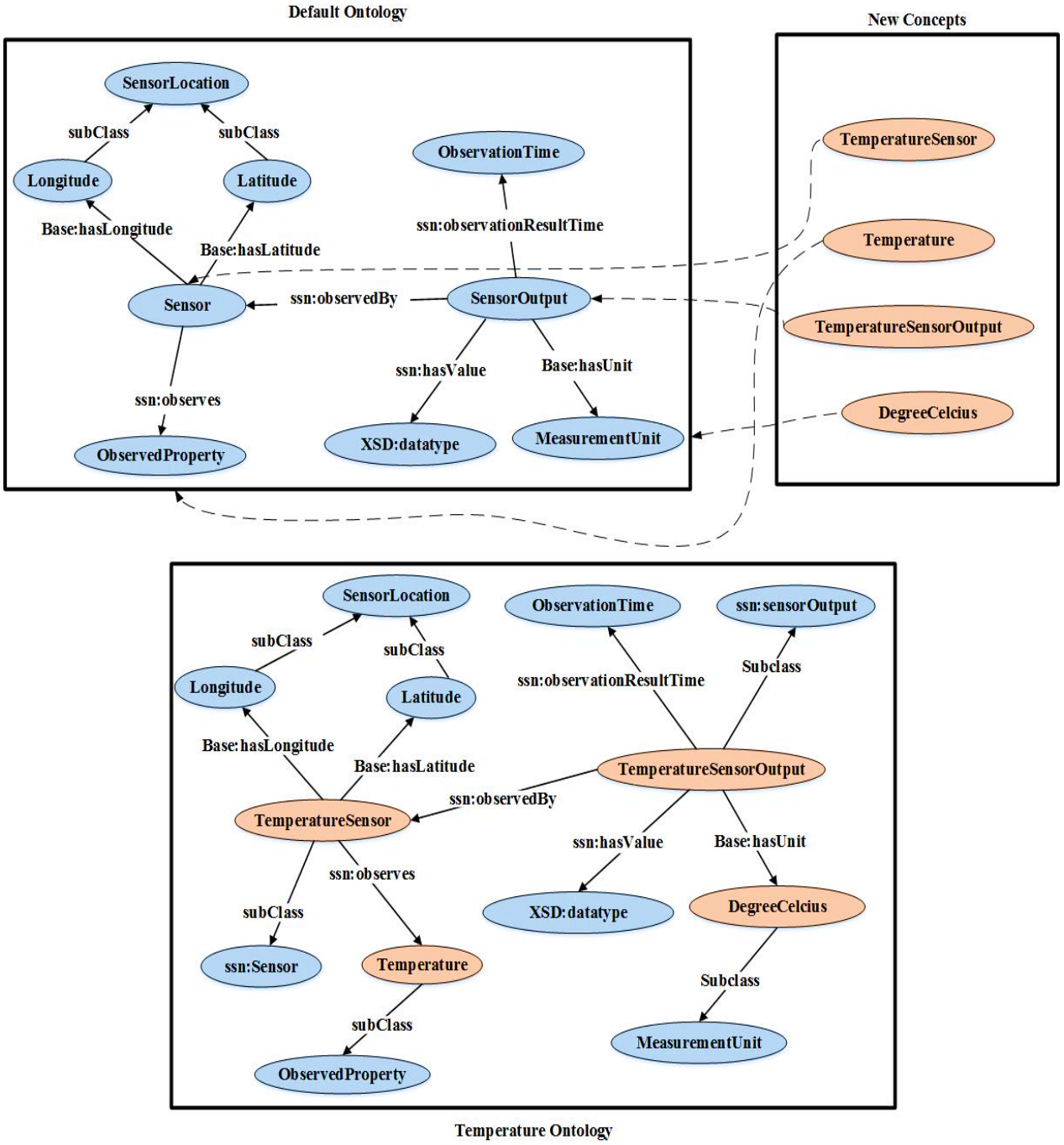


**Figure 5-7: Ontology development for temperature sensor**

6. The ontology creation module incorporates the new concepts in the ontology by creating a parent-child relationship between the concepts from SSN ontology and the new concepts.

Figure 5-7 shows the default ontology, newly added concepts and generated ontology for the temperature sensor. We see that the "TemperatureSensor" is added as a subclass of the "Sensor" class whereas the "Temperature" class is added to the "ObservedProperty" class. In order to create a semantic link (for example, "TemperatureSensor observes Temperature") between the TemperatureSensor and Temperature class, domain and range values of the property "observers" are updated. An Ontology development language (such as OWL, Apache Jena) is used to create a base ontology when all the newly added concepts are mapped to the corresponding parent classes and properties are updated for domain and range values.

7. Finally, the base ontology is sent to the ontology storage which is used to deploy in the ontology overlay.

### 5.5.2 Base Ontology Deployment

Once the base ontology development is completed, base ontology deployment is performed. Figure 5-8 shows the sequence diagram of base ontology deployment. We describe the deployment process for the wildfire monitoring use case below.

At the initial stage, the WSN IaaS provider adds the concepts related to the Java SunSpot (temperature and light) and AdvanticSys Kit (humidity) sensors and develops the base ontology using the ontology provisioning center that we have discussed in the previous subsection.

As the developed base ontology contains concepts of sensors having three different capabilities, the ontology provisioning center splits the complete base ontology into the following three parts in such a way that each portion contains all the related, capability-specific concepts:

a) Ontology for only Temperature Sensor

b) Ontology for Light Sensor

c) Ontology for Humidity Sensor

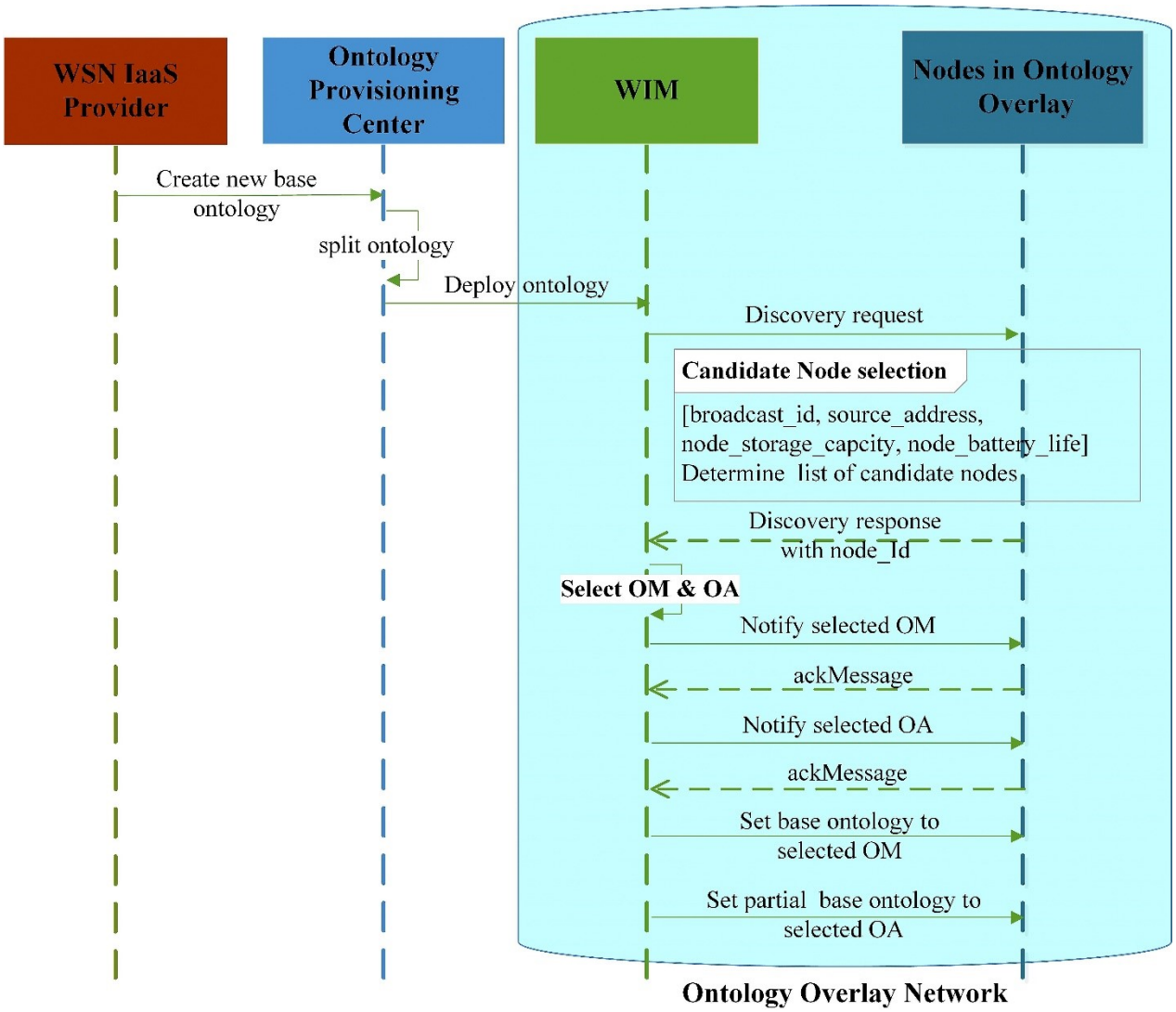


**Figure 5-8 Sequence diagram for ontology deployment process**

The common concepts and properties (e.g., "sensor", "observedProperty", "sensorOutput", "observes") are present in all the three parts. The ontology provisioning center deploys the full and parts of the base ontology to WIM belonging to the ontology overlay through *deploy ontology*

message. A *discovery request* message is sent from WIM to the neighboring nodes residing in the ontology overlay. Only the potential candidates, i.e., those with the higher energy and storage capacity, give the response messages along with their IDs to WIM. In the next step, WIM selects capable nodes that can act as OM or OA by executing a genetic algorithm. WIM notifies the selected OM and OA nodes. Upon receiving the *notification* message, OM and OA nodes send an *acknowledgment* message to WIM, stating their readiness to work accordingly. Finally, WIM sends the complete base ontology to the selected OMs and the partial base ontology to the selected OAs.

## 5.6 How the Architecture meet the Requirements

The refined architecture fulfills all the ontology provisioning requirements that we mentioned in Chapter 3. First, our proposed ontology provisioning center allows WSN IaaS provider to easily develop and manage the base ontology. It can be implemented using any standard technology. We use distributed approach to deploy the base ontology in order to avoid a single point of failure. We propose an ontology deployment protocol that allows the interaction between the ontology provisioning center and the virtualized WSNs. Our proposed protocol re-uses the genetic algorithm to find a set of capable nodes and deploy the base ontology among those nodes. It results in efficient resource utilization of the WSN IaaS. Base ontology is replicated among the selected capable nodes to ensure fault tolerance. Our proposed solution is applicable to the large-scale sensor deployments, thus, the scalability requirement is also satisfied.

## 5.7 Chapter Summary

In this chapter, we have described an ontology provisioning architecture in the virtualized WSNs, which was built upon the data annotation architecture presented in Chapter 4. We first described the enhanced architecture focusing on the new components and functionalities, then presented the ontology provisioning center. We discussed the ontology provisioning protocol to deploy the base ontology in the virtualized WSNs. An illustrative scenario showing the workflow of different components of the architecture was presented based on the wildfire monitoring use case. Finally, we explained how our proposed architecture satisfied all the ontology provisioning requirements presented in Chapter 3.

# Chapter 6

## 6 Validation: Prototype, Simulation, and Evaluation

We presented an architecture for data annotation in Chapter 4 which was extended in Chapter 5 for ontology provisioning. In this chapter, we describe the implemented prototype and simulation to evaluate them based on the different performance metrics. We implemented a semantic wildfire monitoring application that uses the annotated data from sensors. We also implemented an application for ontology provisioning which was used for developing and managing the base ontology. This chapter consists of two main sections. The first section validates the data annotation architecture. The second section validates the ontology provisioning architecture. Finally, we summarize the chapter.

### 6.1 Validation of Data Annotation Architecture [2]

We begin this section by representing the implementation scenario. The scope of the prototype application is defined in the second subsection. We discuss the prototype setup and different configurations that we used to implement our prototype in the third subsection. The fourth subsection describes the performance matrices along with the performance result of the implemented prototype.

[2] This section is an extended version of the validation presented in the paper " I. Khan, R. Jafrin, F. Errounda, R. Glitho, N Crespi, M Morrow, P Polakos, A Data Annotation Architecture for Semantic Applications in Wireless Sensor Networks, *IFIP/IEE International Symposium on Integrated Management (IM 2015),* pp. 27, 35, 11-15 May 2015, Ottawa, Canada" (acceptance rate: 27.2%)

### 6.1.1 Implemented Scenario

We have implemented a semantic wildfire monitoring application inspired by the scenario presented in Chapter 3. In this implementation, we consider that the base ontology has already been provisioned into the WSNs. Figure 6-1 depicts the workflow of the implementation scenario.

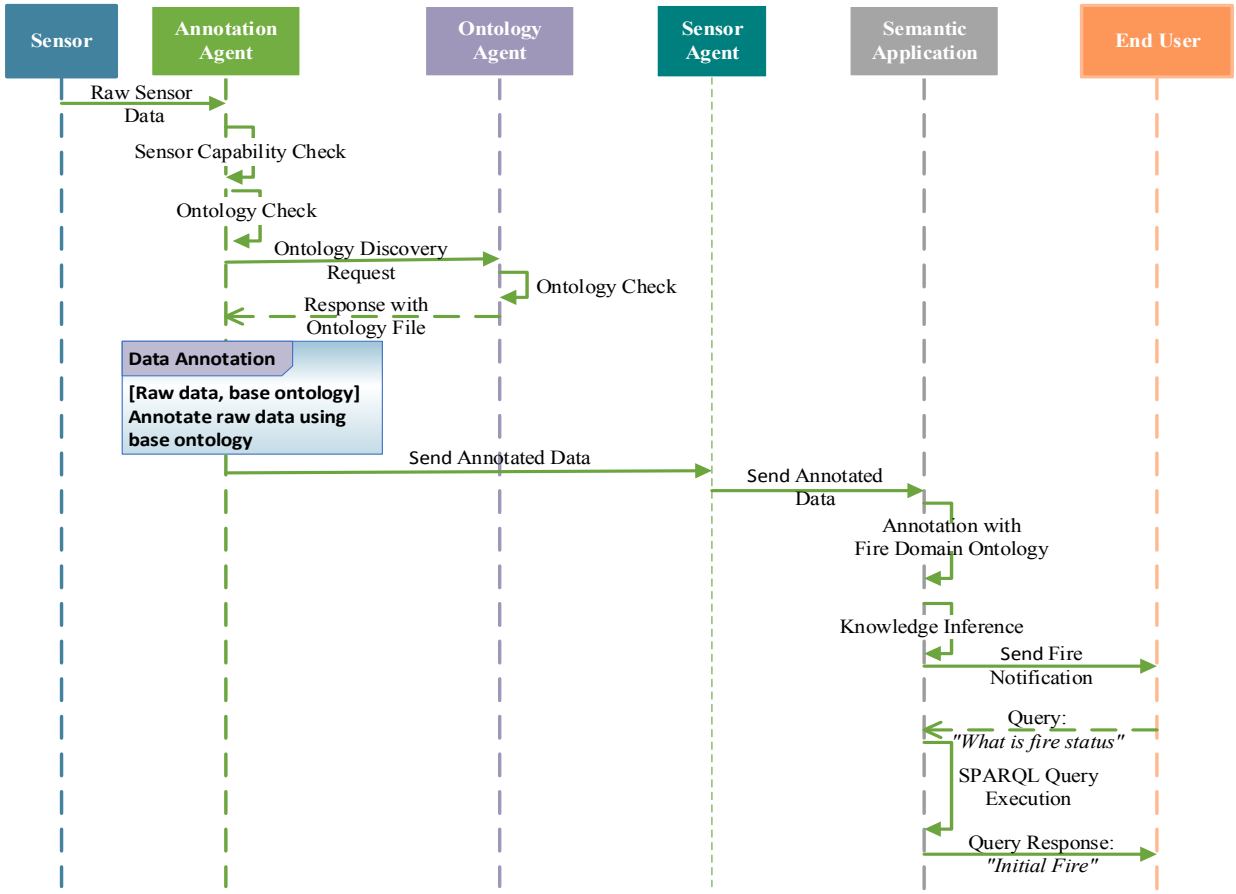


**Figure 6-1 Sequence diagram of the implemented scenario**

Semantic virtual sensors send their raw data to the AA. Once an AA receives the raw sensor data, it first checks locally to determine whether it has the required ontology to annotate the data. If AA does not have the necessary ontology, it sends a request message to an OA for the ontology. The

OA sends the requested ontology to the AA. The AA annotates the raw sensor data using the ontology and sends the annotated data to the SA. The SA sends the annotated data to the semantic wildfire monitoring application.

Wildfire monitoring application receives annotated data and applies a set of reasoning rules along with domain ontology to infer additional knowledge. If the application detects any fire event, it immediately sends a fire notification message to the end user. The user can ask for some detail information such as *where the fire event has occurred?* Or *what is the current fire situation?* Semantic wildfire monitoring application has an SPARQL query execution engine that finds the required information from the annotated data and replies back to the end-user.

### 6.1.2 Prototype High-Level Description

Figure 6-2 is a snapshot of the user interface of the semantic wildfire monitoring application that enables the end-user to monitor fire situation.

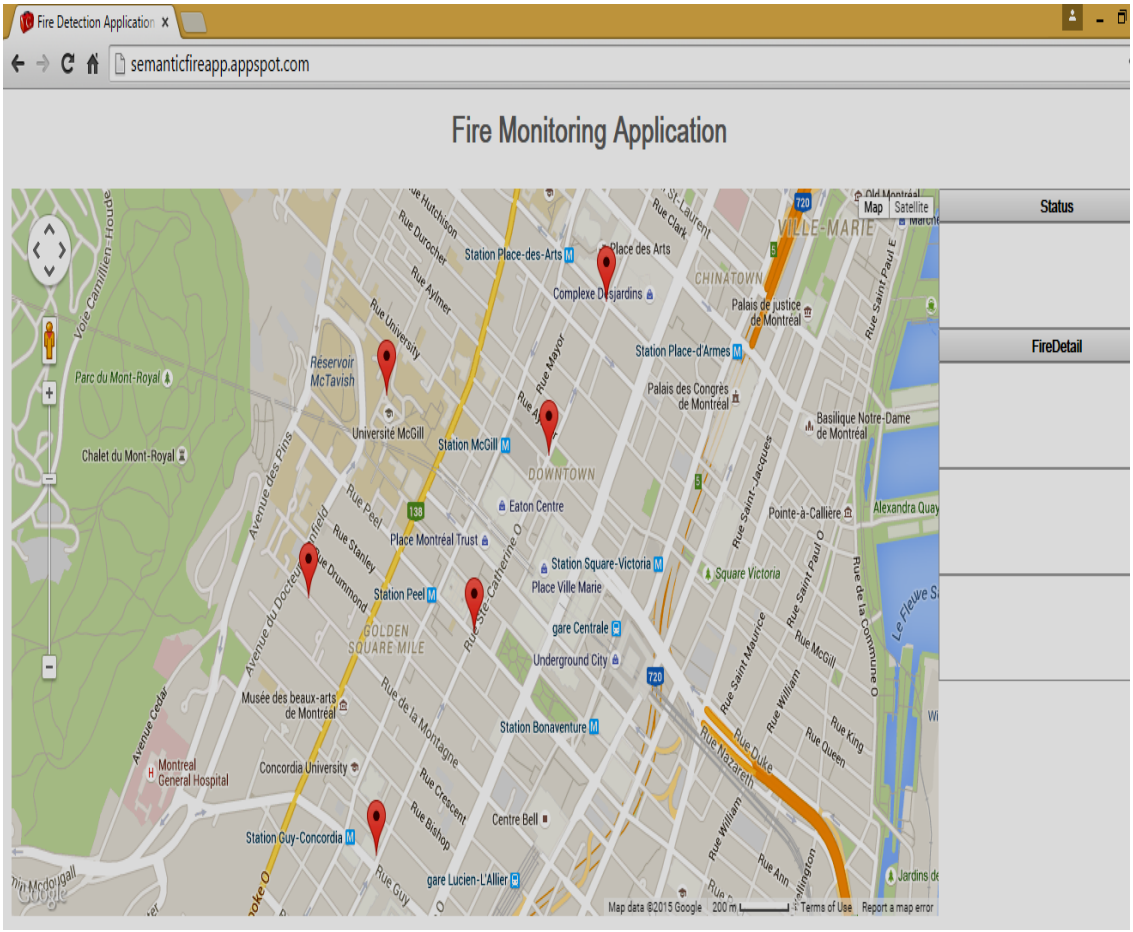


**Figure 6-2 A user interface for the semantic wildfire monitoring application**

There are two buttons in the user interface. The prototype application continuously checks the fire status by taking sensor measurement from virtualized WSN. If any fire situation occurs, then the application immediately triggers a fire notification message to the user. After getting the fire notification message, the user can query for additional information by pressing the button "Status" to understand the context of the fire situation such as *is it initial fire* or *massive fire* in that place. The user can also get the detail information such as *time, location,* and the *current temperature* of the fire-eruption area by clicking the next button "FireDetail".

### 6.1.3 Prototype Setup

We used the following software and technologies to implement the prototype:

1) *Google App Engine* [62] was used as the PaaS Layer for implementing and hosting the semantic wildfire monitoring application.
2) *Apache Jena* [63] was used for annotating sensor data, reasoning, and executing SPQRQL queries.
3) *RESTlet framework* [64] was used for the interaction between the WSN IaaS and semantic wildfire monitoring application in PaaS.
4) *JXTA protocol* [65] was used to implement the annotation and ontology overlays.

We used two different sensor kits for the prototype: Java SunSpot and TelosB motes from AdvanticSys Kit. In total, we used 6 SunSpots (two of them as base stations), 4 TelosB motes (one of them as border router) running Contiki OS [66]. All these sensors have multiple on-board sensing capabilities but differ in their processing and storage abilities. In our implementation, TelosB motes are *Type A* sensors and Java SunSpots are *Type B* sensors. All of the sensors were running multiple application tasks. The Java SunSpots had three application tasks running concurrently and periodically measured the temperature, light and blinking LEDs. The TelosB motes had the temperature, light, and humidity tasks running concurrently. *Type B* sensors send their data in SenML [60] format which is a lightweight standard data model and suitable for sending sensor data. *Type A* sensors send their data in a simple string format.

The wildfire monitoring semantic application is a RESTful web service that uses the following components:

### *1) Fire domain ontology:*

We developed a fire domain ontology. Figure 6-3 shows few concepts of the fire domain ontology. Fire domain ontology contains the concepts related to the fire event situations, states of the sensing events, and location (city, area). Examples of the fire event situations include "no fire", "initial fire", and "fire blaze". States of the sensing events might be "low temperature" or "high temperature, "low humidity" or "high humidity", etc.

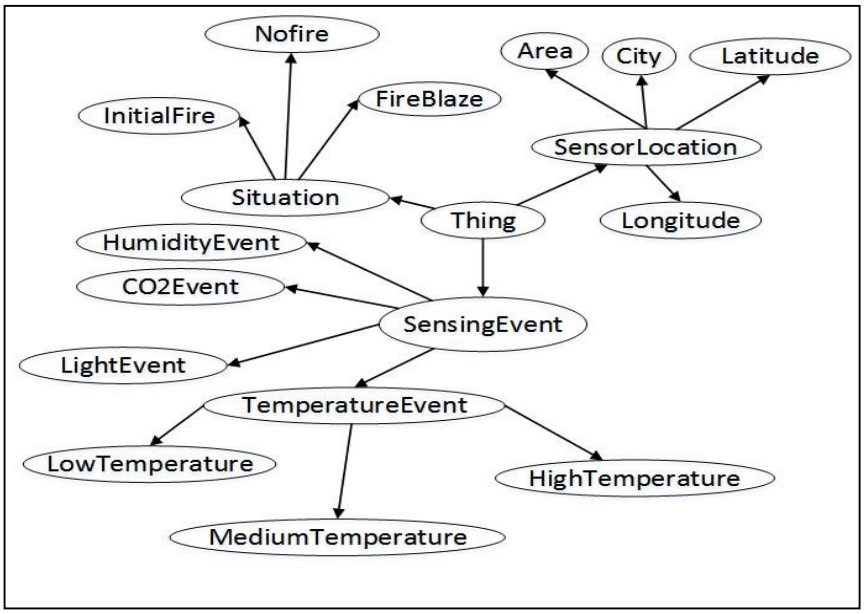


**Figure 6-3 Few concepts of the fire domain ontology**

### *2) Jena Inference API:*

We used Jena inference API to reason over the annotated data and infer implicit knowledge using a set of rules. We developed several rules for our wildfire monitoring

application to provide information to the end-users about the fire events. Two examples of the rules are given in the Figure 6-4.

```
[Rule1: (?output ssn:hasValue ?Value) greaterThan(?Value,80),
        (?output rdf:type base:TemperatureOutput),
        (?output base:hasUnit base:DegreeCelsius) ->
        (?output fda:hasTemperatureType: fda:HighTemperature)
       ]
[Rule2: (?output fda:hasTemperatureType  fda:HighTemperature),
       (?output fda:hasHumidityLevel fda:LowHumidity),
        (?output fda:hasCO2Level fda:HighCO2) ->
        (? output fda:hasFireSituation fda: fireBlaze)
      ]
```

**Figure 6-4 Example of Reasoning rules**

The first rule states that if the sensing temperature is more than 80-degree celsius, then it is a *high temperature*. The second rule states that if the temperature is high, relative humidity is low, and the CO2 level is high then there is a *fire blaze* situation.

### *3) Query Engine:*

SPARQL query engine is used to query the annotated data. An example query is shown in the Figure 6-5 to get the event information like fire event's occurrence time, current temperature value, location and the status (fire blaze in this case).

```
SELECT  ?Time ?Temperature ?Longitude ?Latitude ?Firesituation
WHERE {
        ?SunSpotOutput base:hasSensingTime ?Time.
        ?SunSpotOutput ssn:hasValue ?Temperature.
        ?Sunspot base:hasLongitude ?Longitude.
        ?Sunspot base:hasLatitude ?Latitude.
        ?SunSpotOutput fda:hasFireSituation ?Firesituation.
FILTER ( regex(str(?Firesituation),
        'http://www.semanticweb.org/WirelessSensor/FireApplication#FireBlaze', 'i' )
      }
```

**Figure 6-5 Example of SPARQL query**

The functional entity AA belongs to the annotation overlay and has the following components:

- **Web Server**: receives the sensor data
- **JXTA Edge Peer**: participates in the overlay and requests the required parts of the base ontology
- **RDF Generator**: annotates sensor data using the base ontology
- **Web Client**: sends annotated data to the semantic wildfire monitoring application

The functional entity OA in the ontology overlay has the following components:

- JXTA Rendezvous Peer: stores the base ontology and sends it to the requesting AA. We used the JXTA Content Management System (CMS) to advertise the base ontology available in each OA and send it to the requesting AAs.

The proposed architecture is implemented as Infrastructure as a Service (IaaS) which allows us to link our solution to the IaaS, PaaS, and SaaS aspects of the cloud computing paradigm. Figure 6-6 shows the three implementation configurations we used for evaluation purpose. The details of these configurations are as follows:

### *1) Configuration A*

We used *Type A* sensors (TelosB) at the infrastructure level. The semantic virtual sensors sent their raw data to a GTO node. The GTO node (acting as an AA) downloaded the required ontology from an OA and annotated the raw sensor data. Finally, the annotated data was sent to the semantic wildfire monitoring application via SA.

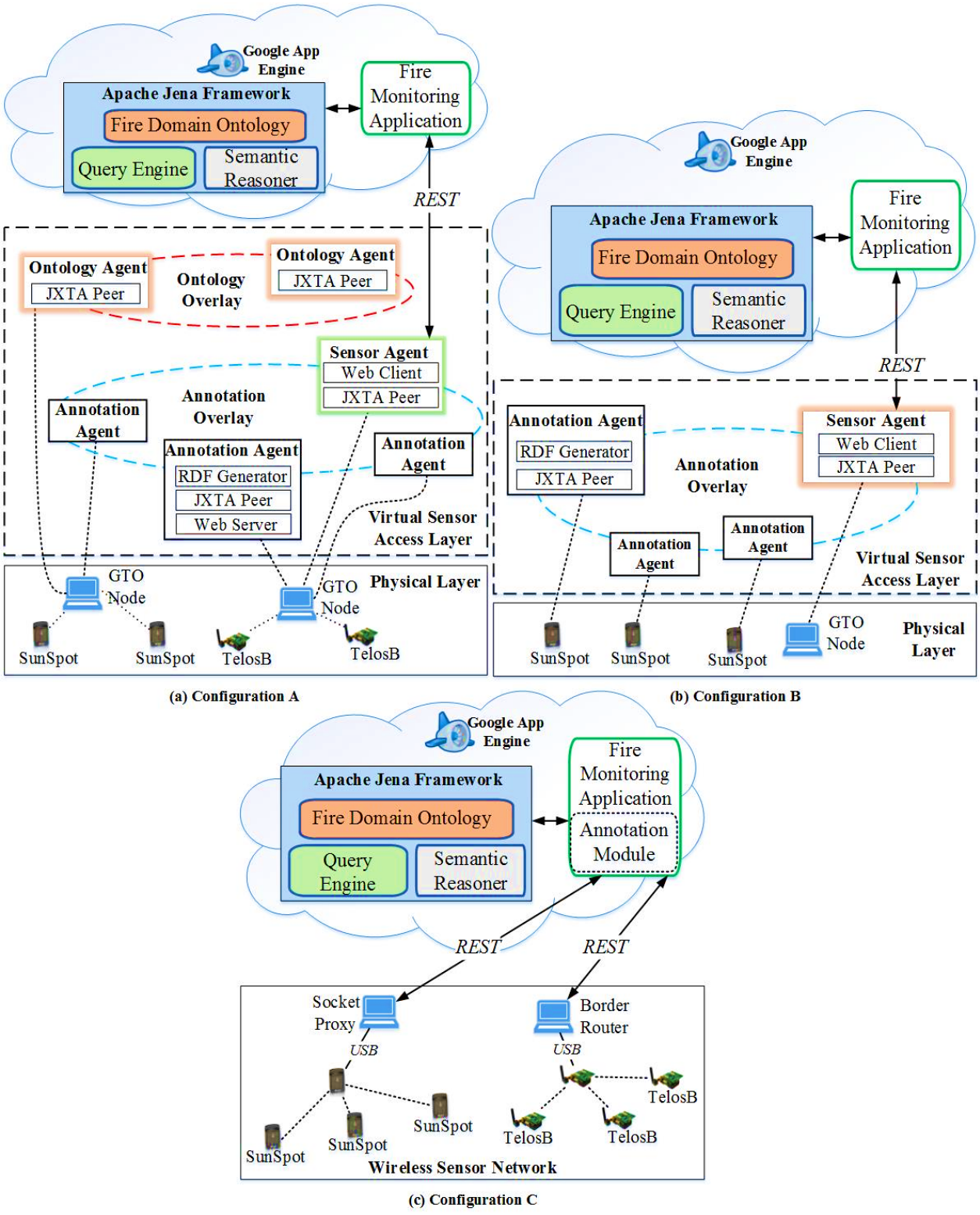


**Figure 6-6 Implementation architecture of the semantic wildfire monitoring application**

### 2) *Configuration B*

We used *Type B* sensors (Java SunSpots). The ontology used to annotate the data was stored locally in the *Type B* sensors as there was no ontology overlay. We implemented the AA in the Type B sensors using μJena library [67]. In this way, those AAs did not need any GTO node to perform annotation. Each semantic virtual sensor sensed and generated the raw data, annotated it and sent it to the semantic wildfire monitoring application via SA.

### 3) *Configuration C*

We used both *Type A* and *Type B* sensors. All of the sensors sent their raw data over the internet. For *Type A* sensors, we used a Contiki border router to allow them directly communicate with the wildfire monitoring application. For *Type B* sensors, we used Java Socket-Proxy, which communicates with the wildfire monitoring application on their behalf. In this configuration, the wildfire monitoring application performed the annotation itself. This allowed us to measure the extra delay introduced by our approach.

## 6.1.4 Performance Metrics and Results

We evaluated the performance of the implemented prototype with the following metrics:

### 1) *End-to-End Delay (E2ED)*

E2ED is defined as the duration of sending the raw data by virtual sensors and receiving acknowledgment by SA from the wildfire monitoring application. It includes the time taken by all the following intermediate steps:

- A semantic virtual sensor sends raw data to AA

- Discover ontology (for configuration A) when ontology is not present in AA
- Annotation performed by AA
- AA sends annotated data to the semantic wildfire monitoring application
- SA receives acknowledgment from the application

### 2) *Ontology Download Time (ODT)*

ODT is the duration between the request for an ontology by AA and reception of the required ontology from an OA.

### 3) *Scalability of AAs*

The number of AAs are varied to observe the ontology discovery time and ODT.

### 4) *Expected Operation Time (EOT)*

EOT is the duration a sensor can execute tasks (semantic and non-semantic) and its battery lifetime.

The delays were measured in milliseconds. The experiments were repeated 50 times.

Figure 6-7 shows the individual E2ED of the three configurations. Configuration A had an average E2ED of 3566ms. The actual annotation delay was negligible (less than 10ms) since the AA was implemented on a laptop computer. The E2ED of configuration B was the highest, at 4575ms. The average annotation delay was 525ms since the Java SunSpots were annotating data themselves. We found that this longer time was due to the low RAM size, only 1MB. Despite this, SunSpots were able to annotate sensor data and run other tasks concurrently without any other issues. The E2ED of configuration C was 3187ms. As expected, the wildfire monitoring application was able to annotate the sensor data quickly but at the expense of developing the base ontology and then implementing it in addition to the application logic.

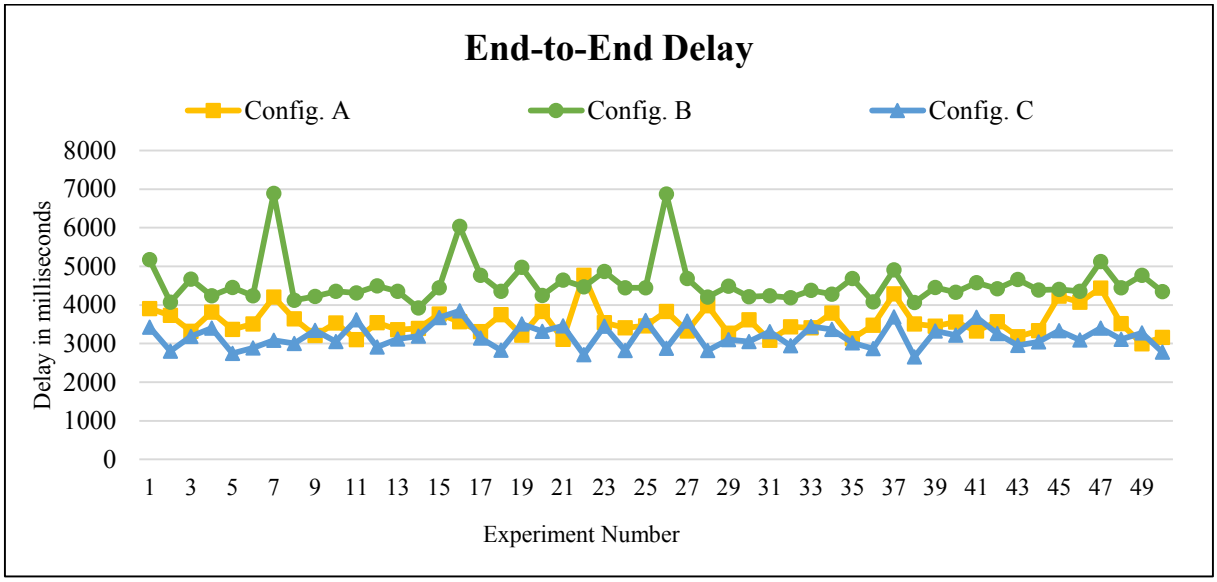


**Figure 6-7 End-to-End Delay**

Figure 6-8 shows their average E2ED of all configurations after 50 repetitions.

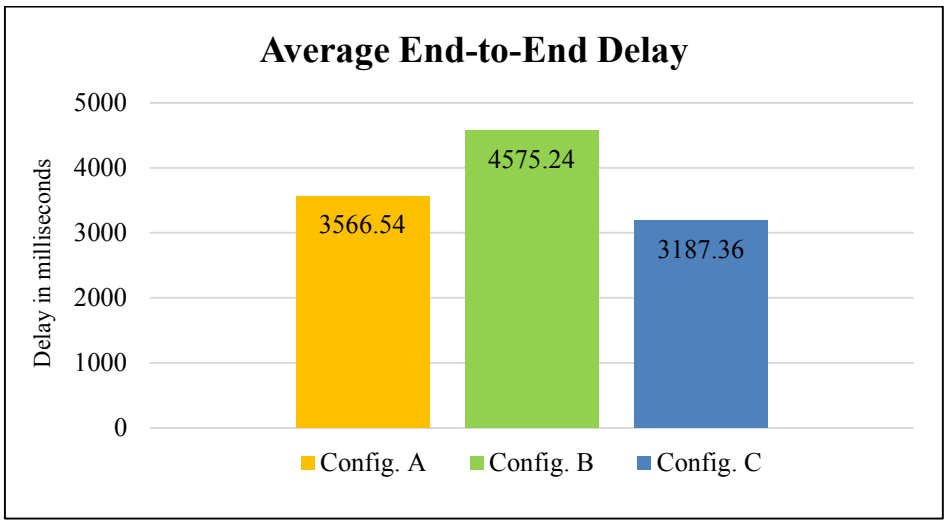


**Figure 6-8 Average End-to-End Delay**

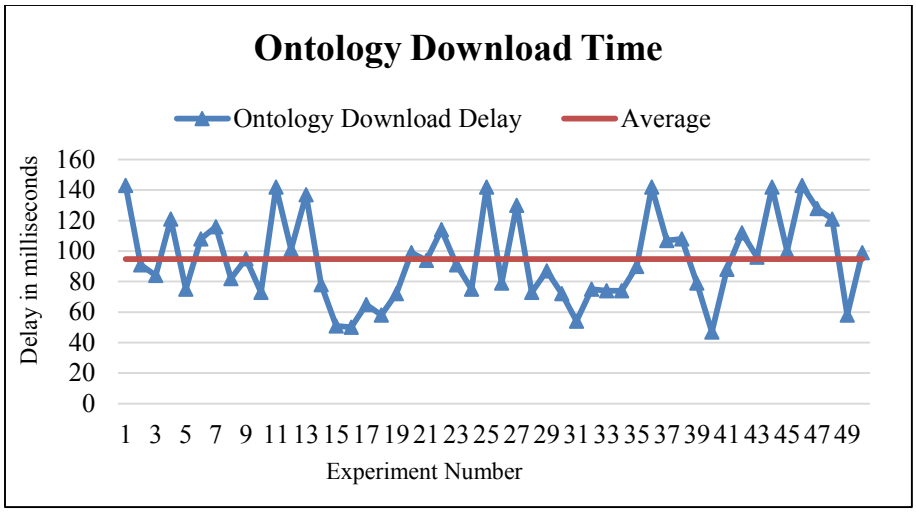


**Figure 6-9 Ontology Download Time**

Figure 6-9 shows the ODT when a particular AA requests for the required part of the base ontology and received the corresponding OWL file. The average ODT found from 50 experiments was *94ms* which was typically found in private LAN settings using JXTA protocol.

The results in Figure 6-10 show how the OA discovery time increases when the number of AAs increases.

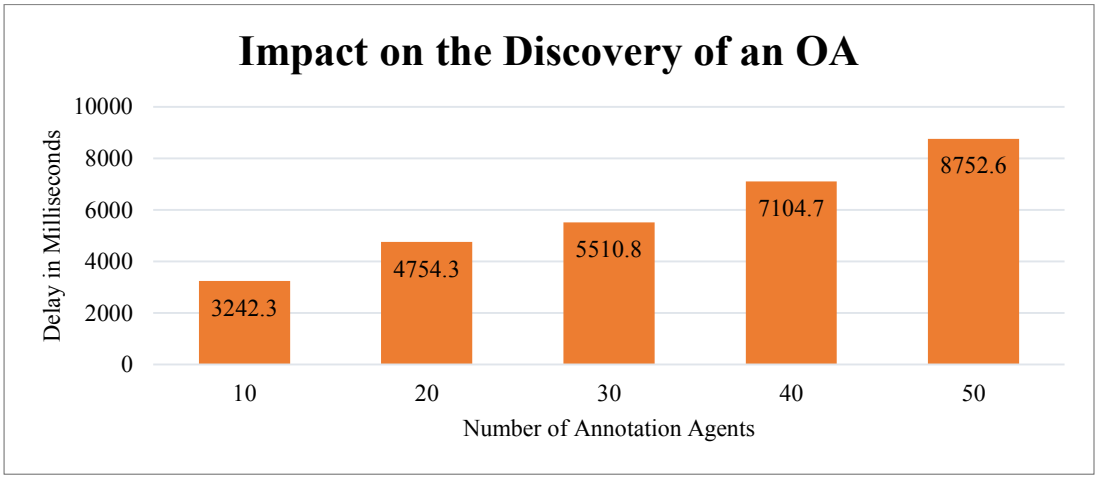


**Figure 6-10 OA discovery time when AAs increase**

Since JXTA was used for implementation, it had a direct impact on the scalability. JXTA is known to perform poorly when peers in the network increase which is also found in this work. However, the increase in AAs does not impact the ODT mainly because OA is already discovered. Here, the average ODT was around 100ms, which is almost similar to the one shown in the Figure 6-9.

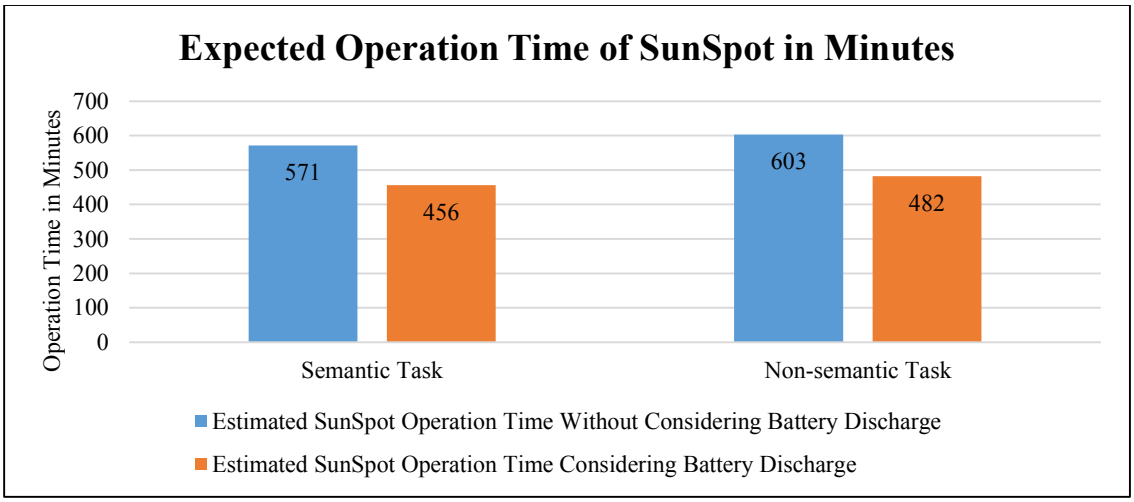


**Figure 6-11 Expected operation time of Java SunSpots (always on)**

Figure 6-11 shows the EOT of the Java SunSpots while running a semantic and a non-semantic task without using any sleep mechanism. Without considering normal battery discharge, SunSpots lasted around 571 and 603 minutes operation time for the semantic and non-semantic tasks, respectively. Using 0.8 as a constant multiplier for normal battery discharge reduced the operation time to 456 and 482 minutes, respectively

For all the three configurations, we have experienced delay due to the circumstances beyond our control. For example, from time to time, GAE would start a new process for the wildfire monitoring application and reload it thereby incurring an unnecessary delay. We determined this from the log files of our wildfire monitoring application.

## 6.2 Validation of Ontology Provisioning Architecture

In this section, we discuss how we validate our ontology provisioning architecture. The first section states implementation scenario. In the second section, we give a high-level description of the prototype. The prototype setup for ontology provisioning center and the implementation architecture is presented in the third section. The fourth section describes the performance matrices and performance results of the implemented prototype. The fifth section presents the simulation setup and performance matrices to validate our ontology deployment protocol.

### 6.2.1 Implemented Scenario

In the data annotation implementation scenario, we assumed that the base was already provisioned. Our scope of this implementation is to develop, manage, and deploy base ontology to the WSNs. We extended the same implementation scenario discussed in the Section 6.1.1. After deploying the sensors, WSN IaaS owner needs to create the corresponding base ontology. In this regard, we built a prototype application named ontology provisioning center that allows WSN IaaS owner or any novice user to interact with the system and develop a base ontology without knowing any technical knowledge or protocol details. Finally, the base ontology is provisioned using the ontology deployment protocol to the virtualized WSNs.

### 6.2.2 Prototype High-Level Description

We need a user-friendly application that permits WSN IaaS provider to create, manage, and deploy the base ontology to the virtualized WSNs to accelerate the automatic annotation process. In this thesis, we have implemented an ontology provisioning center which is a web-based application.

This application allows base ontology development and management which can also be used by a novice user.

Figure 6-12 represents the user interface of the ontology provisioning center application. From the figure on the left side, there is a list of buttons dedicated to performing some specialized tasks.

- The button "Add Concept" is used to create new concepts when a new type of sensors is deployed in the WSN infrastructure.
- WSN IaaS provider can add new sensor information by using the button "Add Sensor". Existing concepts can be updated or deleted by using the button "Update Concept".
- The button "Base Concepts" simply shows the current list of the concepts that forms the base ontology.
- Ontology can be developed by using the button "Create Ontology". The button "Ontologies" presents a list of already created ontologies by WSN IaaS provider.
- WSN IaaS provider can view the existing sensors information residing in the WSN infrastructure by clicking the button "Sensor Repository".
- WSN IaaS provider can also download the base ontology to his local machine with the help of "Download" button.

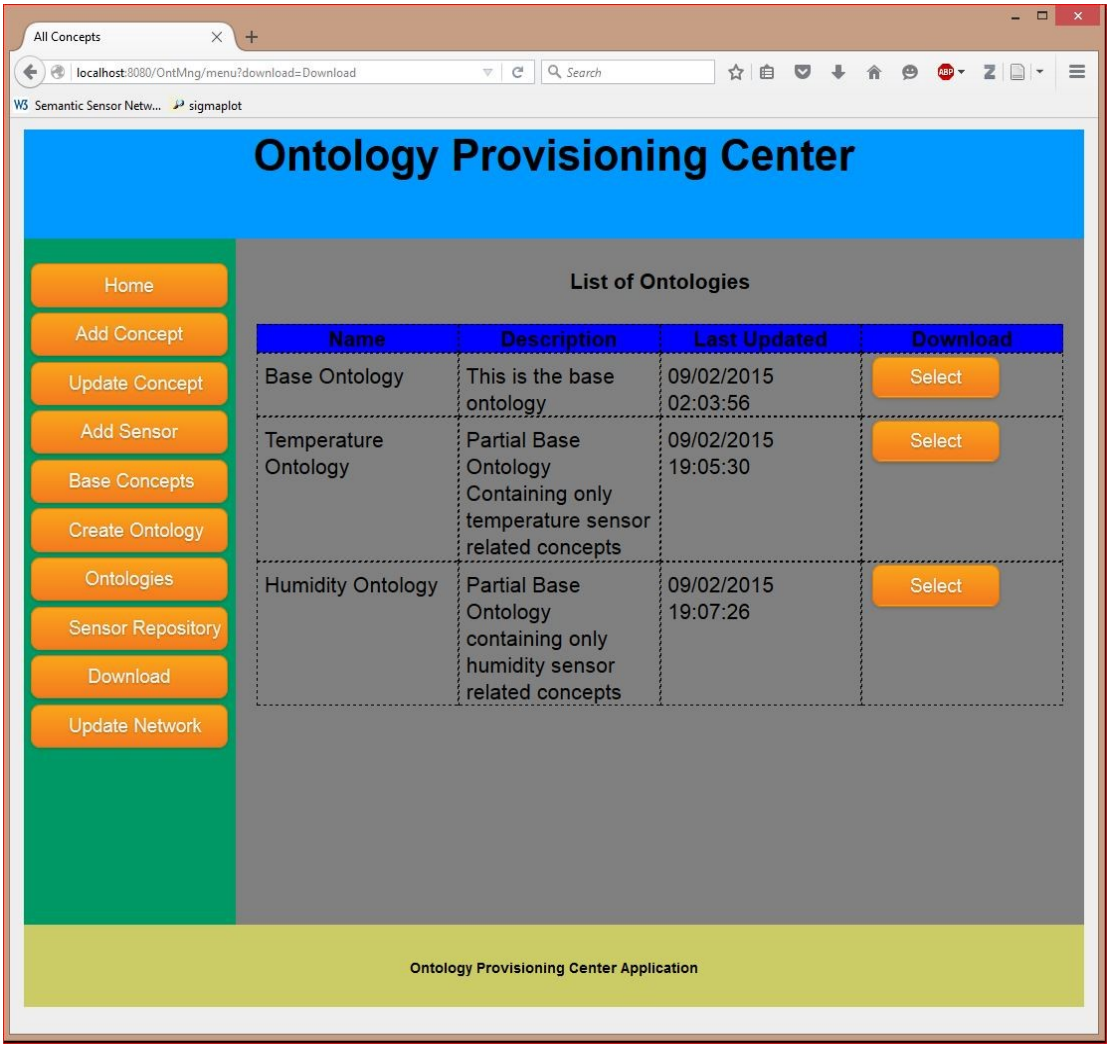


**Figure 6-12 User interface for ontology provisioning center application**

### 6.2.3 Prototype Implementation Architecture

The ontology provisioning center was developed by using the following software and technologies:

***1)*** *Java* was used to develop the whole application using JAX-WS web services API

***2)*** *MySQL Database* was used to store the base ontology concepts

***3)*** *Apache Tomcat* web server was used to host the application

*4)* In order to generate the base ontology using the stored concepts in the database, we used Protégé 3.8 API which is an open-source Java-based library for OWL and RDFS.

To implement the annotation and ontology overlay, we used the Java-based JXSE implementation of JXTA protocol. The WSN IaaS Manager, OM, and OA were implemented by the JXTA Rendezvous Peer functionality to store the base ontology. We used the JXTA Content Management System (CMS) to send the base ontology from WIM to OM, then from OM to OA and finally, from OA to AA.

Figure 6-13 shows the implementation architecture for provisioning base ontology. We extended the data annotation implementation architecture and added the ontology provisioning center application in this new architecture.

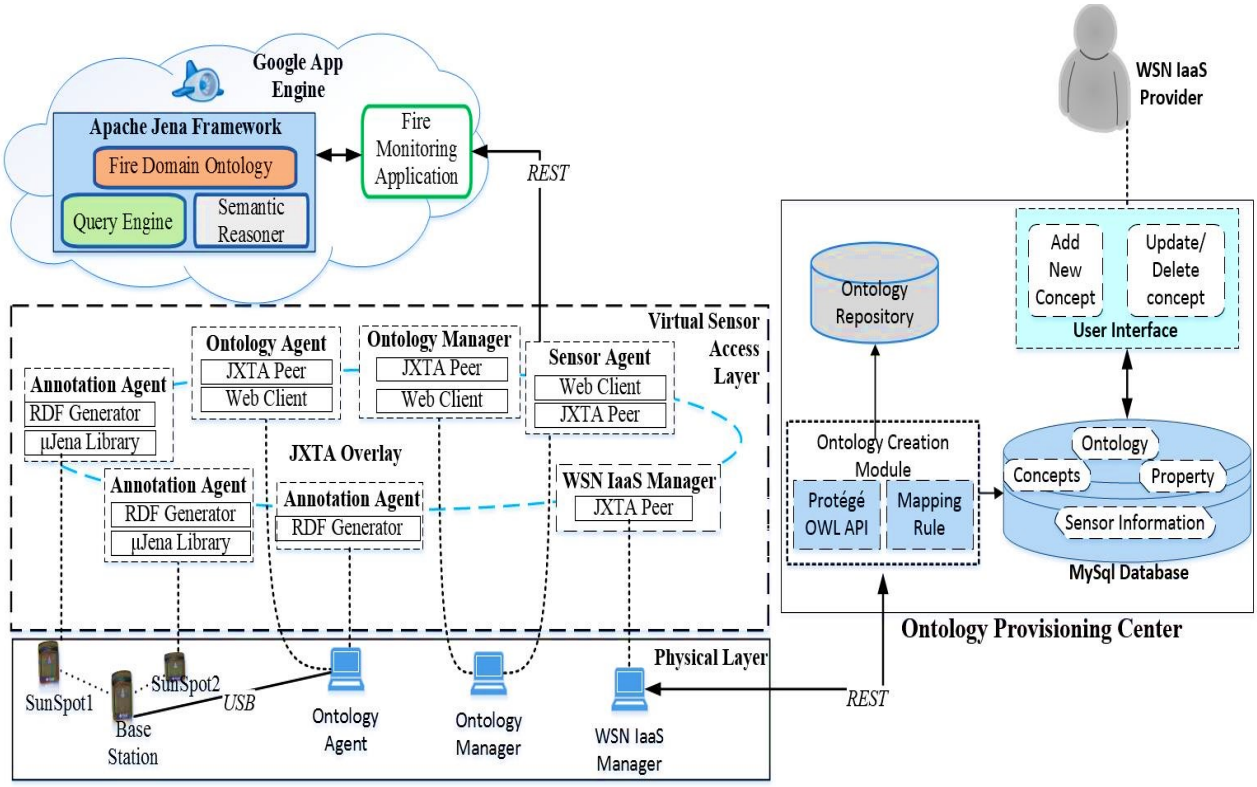


**Figure 6-13 Prototype implementation architecture for ontology provisioning**

### 6.2.4 Performance Metrics and Results

We used the following two metrics to evaluate the performance of the prototype:

#### *1) Overlay Creation Delay (OCD)*

OCD is the time to create JXTA overlay from a non-existent state to a ready state when it is ready to accept join requests. We measured this delay inside the Java code to ensure that the OCD does not include the JVM start-up delay.

#### *2) Ontology Distribution Time (ODisT)*

ODisT is the combination of the following delays:

- The delay from ontology provisioning center application to WSN IaaS Manager,
- The delay from WSN IaaS Manager to OM and Delay from OM to OA.

All these experiments were repeated 50 times with 95% confidence interval.

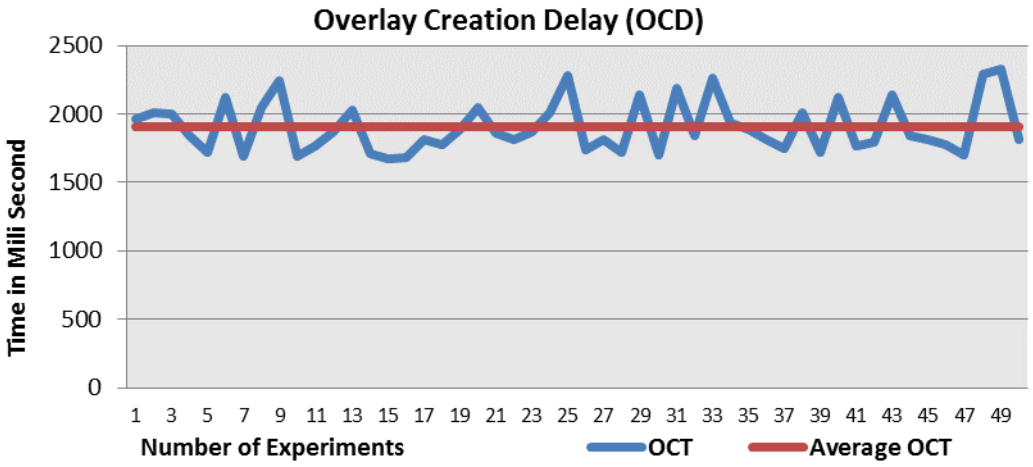


**Figure 6-14 Overlay creation time**

Figure 6-14 presents the OCD of 50 experiments and its average value. We figured out that the average OCD was 1906ms, based on the 50 experiments. It is important to note that the OCD depends on the configurations of the machines acting as JXTA peers and is unavoidable. As it was

experienced only during the overlay initiation phase, it did not make much impact on the sensor data annotation process.

Figure 6-15 shows the ODisT. Since the ontology provisioning center and WIM were implemented on the same laptop, hence the delay between this two entities was minor. For this reason, we did not include this delay in the result. The average delay from WIM to OM was ~56ms. The average delay from OM to OA was about 54ms. In total, the average ODisT from 50 experiments was around 109ms.

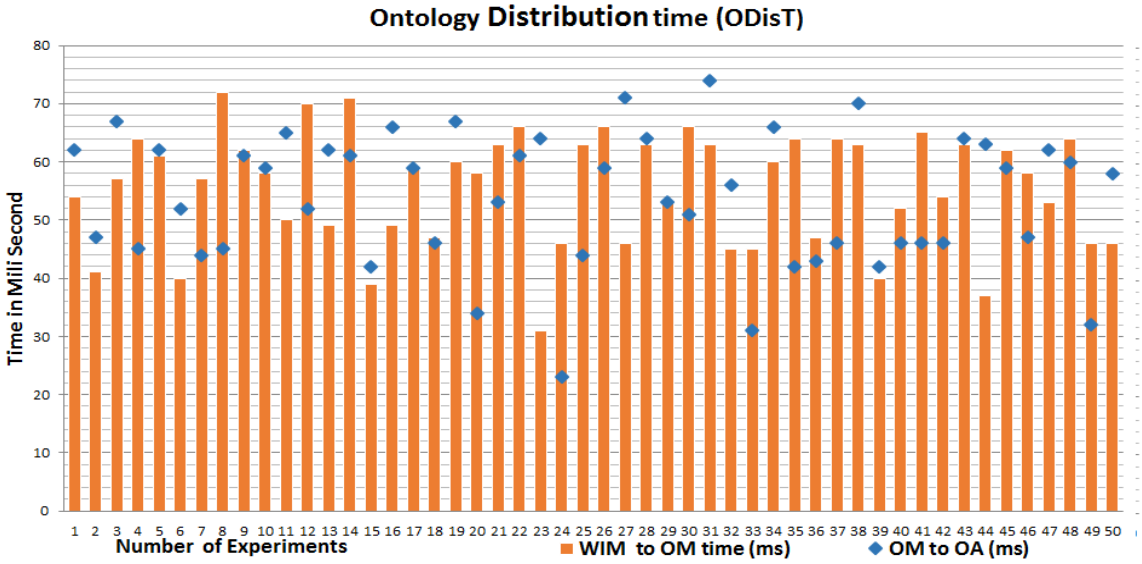


**Figure 6-15 Base ontology distribution time**

### 6.2.5 Simulation Setup and Performance Result

We developed a discrete event simulator [68] to validate our ontology deployment protocol. The primary objective was to compare our proposed protocol with the simple flooding protocol. In this

section, we first present the simulation setup and then describe the performance metrics. Finally, we will show the simulation results.

#### 6.2.5.1 Simulation Setup

We developed a discrete event simulator [68] to select the proper protocol for finding OA and OM nodes that create the ontology overlay.

We considered following two protocols to discover the OA and OM nodes:

(i) *No Delay Simple Flooding*

(ii) *Delayed Aggregation Flooding*

We selected following two types of network topologies:

(i) Grid topology

(ii) Random topology

In both protocols, upon receiving a message for the first time, a node updated its cache and sent it to its neighbors. If the received message was duplicate, it was discarded. In the *No Delay Simple Flooding* approach, the capable node sent a reply message immediately upon receipt of a *discovery request* message. On the other hand, each node waited for a while and then sent the *discovery response* message to have the opportunity to aggregate responses from neighbor nodes in the *Delayed Aggregation* approach.

In the Grid topology, we assumed that the nodes were in a fixed distance rectangular area, which represented a planned deployment. The positions of the nodes were determined by the Gaussian distributions [69] in the Random topology representing the ad hoc deployment of a vast number of sensors.

#### 6.2.5.2 Performance Matrices and Results

We evaluated our protocol using the following two performance matrices:

- ***Convergence Time (CT)***

  Convergence time is the total number of discrete events required to discover all the potential candidates that can act as OM or OA.

- ***Discover Response Messages (DRM)***

  Discover Response Messages (DRM) is the total number of response message received by the WIM node from the potential candidate nodes.

CT and DRM were calculated for both *No Delay Simple Flooding* and *Delayed Aggregation Flooding* protocols. While calculating CT and DRM, several factors were considered. We varied the network size, the number of directly connected neighbors of the nodes by changing the connection range R, and the percentage of capable nodes C. We observed how the values of CT and DRM changed on these factors. We varied the network size from 1000 to 5000, set the connection ranges (R) 18 or 25 to change the directly connected neighbors, and fixed the percentage of capable nodes (C) 40% or 60%. In Grid topology, we considered the node distance was 10. The mean and standard deviation of the Gaussian distribution were $\frac{\sqrt{network\_size}}{2}$ and $\frac{\sqrt{network\_size}}{4}$, respectively.

Figure 6-16 shows the convergence time of the Grid and Gaussian topologies using both the *No Delay Simple Flooding* and *Delayed Aggregation* approaches. As the figure makes it clear, for both topologies, the network convergence time is higher in the *Delayed Aggregation* approach, where each node waits a certain time before sending a *response message*. In the *No Delay Simple Flooding* approach, the potential nodes instantly reply back with the response message results in less delay

compared to the *aggregation* approach. Since this discovering process will run single time during the network initial setup phase, we can ignore that the *Delayed Aggregated* approach needs more time to converge, compared to the other approach, to reduce the number of response messages in the network, which will, in turn, resolve the ACK explosion problem.

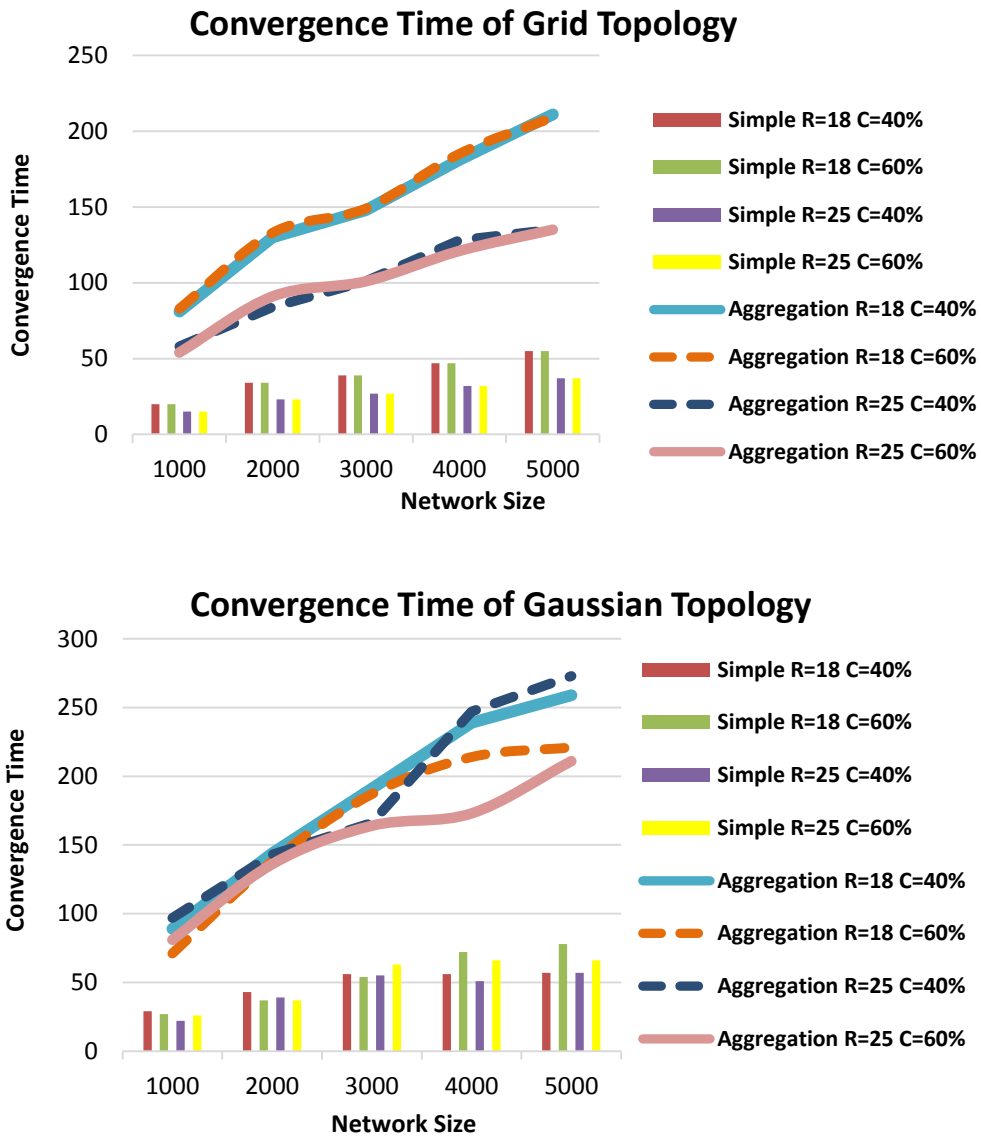


**Figure 6-16 Total convergence time in Grid and Gaussian topologies**

Figure 6-17 shows the total *Discovery Response* messages in the Grid and Gaussian topologies. From the figure, it is clear that the *Delayed Aggregation* approach has lower *Discovery Response* messages in both topologies. The reason behind the better performance of *Delayed Aggregated* approach is that each node gets the opportunities to aggregate response messages from the huge number of neighbors.

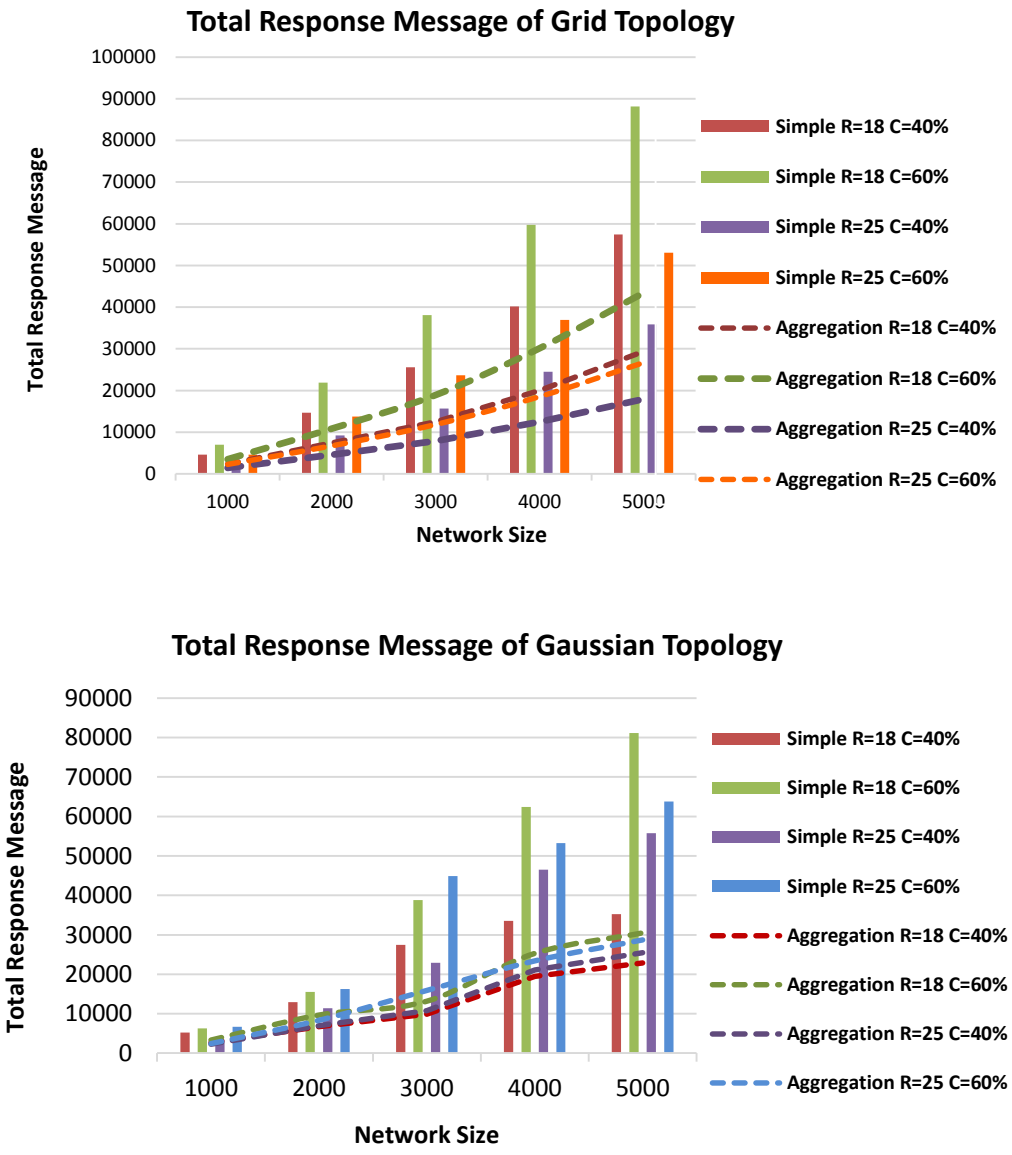


**Figure 6-17 Total response message in Grid & Gaussian topologies**

## 6.3 Chapter Summary

In this chapter, we validated both of the proposed architectures presented in Chapter 4 and 5. In order to validate the data annotation architecture, we presented an implementation scenario, the scope of the prototype, prototype setup including different configuration settings and the performance measurements. After that, we validated the ontology provisioning architecture by describing the high-level prototype description, implementation architecture, and performance measurements. Finally, we presented the simulation setup and performance measurements to justify our proposed ontology deployment protocol.

# Chapter 7

## 7 Conclusion and Future Work

In this chapter, first we summarize the overall contributions we made and then give some research directions for future work.

### 7.1 Summary of Contributions

Wireless sensor networks (WSN) are ubiquities. Virtualization in WSNs allows an efficient resource usage through the sharing of the same WSN physical infrastructure by multiple applications. Semantic applications are very much momentous to provide situational awareness to the end-users. Incorporating semantic applications in the virtualized WSN can potentially play a critical role as these applications are situation aware. However, provisioning semantic applications in virtualized WSNs remains big challenges. The first challenge is the data collected by the virtual sensors need to be annotated semantically in-network so that multiple semantic applications can use those data. The pre-requisite of semantic data annotation is to have an ontology to a particular area of interest. The second challenge is that the defined ontology needs to be provisioned (i.e., developed, deployed, and managed) in the virtualized WSNs to enable the in-network data annotation.

This thesis tackles the semantic application provisioning challenges. At first, we identify the user requirements and show that none of the existing research works meet all the user requirements. Our

proposed solutions overcome the challenges and satisfy all the user requirements including platform and technology independence.

We identified appropriate user requirements. At first, we define two basic requirements: (a) semantic data annotation requirement, and (b) ontology requirement. In the basic requirements, we define the motivation for using semantically annotated data and the need for an ontology to perform the sensor data annotation. Based on the basic requirements we defined three different sets of requirements: general requirements, requirements on the virtualized WSN Infrastructure, and requirements for ontology provisioning. We have reviewed the existing related works in our research domain. We have classified these related works into two broad categories: sensor data annotation framework and ontology provisioning in WSNs. Afterward, We have evaluated these related works based on our defined requirements. We found that none of them satisfied all of our requirements.

In order to address the first challenge, we have proposed a data annotation architecture that allows in-network, distributed, real-time annotation of sensor data at the WSN IaaS level. We have proposed an ontology in sensor domain and referred it as “base ontology”. This ontology contains concepts related to the basic sensing phenomena. Moreover, the proposed architecture is built upon the notions of overlay and super peer to store the base ontology and annotate data at the IaaS level. A proof-of-concept prototype has been implemented based on the wildfire monitoring scenario. The wildfire monitoring application uses annotated data receiving from Java SunSpot and AdvanticSys kit. We have evaluated this application based on the defined metrics.

In order to address the second challenge, we have extended our data annotation architecture for provisioning base ontology to the virtualized WSN. The new architecture consists of an ontology provisioning center for base ontology provisioning and an ontology deployment protocol for interactions between the provisioning center and the WSN. We have implemented a proof-of-concept prototype along with an ontology provisioning center application for developing and deploying the base ontology. Some measurements have been presented to validate the proposed architecture. Finally, we performed a simulation to justify our proposed ontology provisioning protocol.

## 7.2 Future Work

This section presents some key research issues and some new work items as a future work of this thesis.

- ***Publication and Discovery of VWSN capabilities***

In this thesis, we have focused on the sensor data annotation and ontology provisioning in the virtualized WSNs. It would be interesting to find whether semantic web can help in publishing and discovering sensors and their services from a virtualized WSN IaaS or not. This will provide a standard way of advertising the capabilities and services of sensor deployments and make it easier for interested applications to discover easily sensors according to their requirements.

- ***Libraries for semantic annotations in resource constraint environment***

There are limited libraries for semantic annotation that can be used by resource-constraint devices. We found an old J2ME-based µJena library.After several modifications, we managed to use the library with the Java SunSpots. However, it only annotates data in the N-Triple format whereas

standard Apache Jena Framework supports multiple formats. Developing new libraries to allow suitable data annotation in resource constraint environment can be an interesting future work.

- ***Propose a new PaaS for provisioning WSN applications***

Another interesting research would be the possible integration of our proposed architecture with PaaS for the rapid provisioning of WSN applications. As a future work, we plan to design and implement a new PaaS, which will raise the level of abstraction of the virtualized WSN and include specialized features for the provisioning of semantic applications.

- ***Algorithmic Extension***

We can also evaluate the performance of our proposed architecture by comparing the used GA algorithm with other existing algorithms. We can adopt a Round-Robin algorithm for scheduling the role of OM and OA nodes among the selected capable nodes to consider the availability and energy efficiency of the sensors.

# Bibliography


[1] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A Break in the Clouds: Towards a Cloud Definition," *ACM SIGCOMM Computer Communications review*, vol. 39, no. 1, pp. 50–55, Dec. 2008.

[2] P. Mell and T. Grance, "The NIST Definition of Cloud Computing", NIST Special Publication 800-145 (Draft), *US National Institute of Standards and Technology*, Gaithersburg, Maryland, 2011.

[3] S. Loveland, E. M. Dow, F. LeFevre, D. Beyer, and P. F. Chan, "Leveraging virtualization to optimize high-availability system configurations," *IBM Systems Journal*, vol. 47, no. 4, pp. 591–604, 2008.

[4] M. Pearce, S. Zeadally, and R. Hunt, "Virtualization: Issues, Security Threats, and Solutions," *ACM Computing Surveys (CSUR)*, vol. 45, no. 2, pp. 17:1–17:39, Mar. 2013.

[5] A. Sheth, C. Henson, and S. S. Sahoo, "Semantic Sensor Web," *IEEE Internet Computing*, vol. 12, no. 4, pp. 78–83, Jul. 2008; ISSN:1089-7801.

[6] Koivunen, Marja-Riitta, and Eric Miller. "W3C Semantic Web Activity," in *Proceedings of the Semantic Web Kick-off Seminar*, Helsinki, Finland, pp. 27–44, Nov. 2001.

[7] A. Kiryakov, B. Popov, I. Terziev, D. Manov, and D. Ognyanoff, "Semantic annotation, indexing, and retrieval," *Web Semantics: Science, Services, and Agents on the World Wide Web*, vol. 2, no. 1, pp. 49–79, Dec. 2004.

[8] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowledge Acquisition*, vol. 5, no. 2, pp. 199–220, 1993.

[9] S. Abdelwahab, B. Hamdaoui, M. Guizani, and A. Rayes, "Enabling Smart Cloud Services Through Remote Sensing: An Internet of Everything Enabler," *IEEE Internet Things Journal*, vol. 1, no. 3, pp. 276–288, Jun. 2014.

[10] A. Merentitis, F. Zeiger, M. Huber, N. Frangiadakis, K. Mathioudakis, K. Sasloglou, G. Mazarakis, V. Gazis, and Z. Boufidis, "Wsn trends: Sensor infrastructure virtualization as a driver towards the evolution of the internet of things," presented at the *UBICOMM 2013, The Seventh International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies,* Porto, Portugal, 2013, pp. 113-118.

[11] I. Khan, F. Belqasmi, R. Glitho, and N. Crespi, "A multi-layer architecture for wireless sensor network virtualization," in *Wireless and Mobile Networking Conference (WMNC), 6th Joint IFIP*, 2013, pp. 1–4.

[12] M. Hamdaqa, and L. Tahvildari, "Cloud Computing Uncovered: A research landscape". *Advances in Computers,* Elsevier, 2012, vol. 86, pp. 41-85, ISSN 0065-2458.

[13] R. Buyya, J. Broberg, and A. M. Goscinski, "Cloud Computing: Principles and Paradigms", *John Wiley & Sons publications*, vol. 87, 2011.

[14] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of Internet Services and Applications.*, vol. 1, no. 1, pp. 7–18, Apr. 2010.

[15] I. Sriram and A. Khajeh-Hosseini, "Research agenda in cloud technologies," In *Proceedings of the 1st ACM Symposium on Cloud Computing*, SOCC 2010, Indianapolis, IN, USA, pp. 10–11, Jun. 2010.

[16] S. Kuyoro, F.Ibikunle, and O.Awodele, "Cloud Computing Security Issues and Challenges," *International Journal of Computer Networks (IJCN)*, Vol. 3, No. 5, pp. 247-255, Dec. 2011.

[17] VMWare,"Virtualization Overview". White Paper: http://www.vmware.com/pdf/virtualization.pdf.

[18] M. M. Islam, M. M. Hassan, G.-W. Lee, and E.-N. Huh, "A Survey on Virtualization of Wireless Sensor Networks," *Sensors Journal*, vol. 12, no. 2, pp. 2175–2207, Feb. 2012.

[19] I. Khan, F. Belqasmi, R. Glitho, N. Crespi, M. Morrow, and P. Polakos, "Wireless Sensor Network Virtualization: A Survey," *I EEE Communications Surveys & Tutorials,* vol.PP, no. 99, pp. 1–1, 2015.

[20] J. Carapinha and J. Jiménez, "Network virtualization: a view from the bottom," in *Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures*, New York, NY, USA, 2009, pp. 73–80.

[21] Y. Yu, L. J. Rittle, V. Bhandari, and J. B. LeBrun, "Supporting Concurrent Applications in Wireless Sensor Networks," in *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*, New York, NY, USA, 2006, pp. 139–152.

[22] A. P. Jayasumana, Q. Han, and T. H. Illangasekare, "Virtual Sensor Networks - A Resource Efficient Approach for Concurrent Applications," in *Fourth International Conference on Information Technology, 2007. ITNG '07*, 2007, pp. 111–115.

[23] T. Bokareva, W. Hu, S. Kanhere, B. Ristic, N. Gordon, T. Bessell, M. Rutten, and S. Jha, "Wireless Sensor Networks for Battlefield Surveillance," in *Proceedings of the Land Warfare Conference 2006 (LWC 2006)*, Brisbane, Australia, Oct. 2006.

[24] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Sensing as a service model for smart cities supported by Internet of Things," *Transactions on Emerging Telecommunications Technologies*, vol. 25, no. 1, pp. 81–93, Jan. 2014.doi: 10.1002/ett.2704

[25] Example Semantic Web Applications, http://www.cambridgesemantics.com/semanticuniversity/example-semantic-web-applications.

[26] Semantic Web Health Care and Life Sciences (HCLS) Interest Group, http://www.w3.org/2011/09/HCLSIGCharter.

[27] Pharmaceutical Supply Chain Management Using Semantic Web Technology: Case Study, http://www.americanlaboratory.com/914-Application-Notes/35621-Pharmaceutical-Supply-Chain-Management-Using-Semantic-Web-Technology-Case-Study/

[28] T. Berners-Lee, J. Hendler, and O. Lassila, 'The Semantic Web', *Scientific American.*, vol. 284, no. 5, pp. 28–37, May 2001.

[29] Introduction to the Semantic Web, http://www.cambridgesemantics.com/semantic-university/introduction-semantic-web.

[30] M. M. Taye, "Understanding Semantic Web and Ontologies: Theory and Applications," *Journal of Computing*, vol. 2, no. 6, June 2010, ISSN 2151-9617.

[31] T. Segaran, C. Evans, J. Taylor, "Programming the Semantic Web". O'Reilly Media, Inc., 2009, ISBN 978-0-596-15381-6.

[32] J. Rao, P. Kungas, and M. Matskin, "Logic-based Web services composition: from service description to process model," in Proceeding of *IEEE International Conference on Web Services,* 2004*,* pp. 446–453.

[33] J. Kuriakose, "Understanding and Adopting Semantic Web Technology," *Cutter IT Journal (Cutter Information Corporation.)* white paper, vol. 22, no. 9, pp. 10–18, September 2009.

[34] A. Kiryakov, B. Popov, I. Terziev, D. Manov, and D. Ognyanoff, "Semantic annotation, indexing, and retrieval," *Web Semantics: Science, Services, and Agents on the World Wide Web*, vol. 2, no. 1, pp. 49–79, Dec. 2004.

[35] O. Lassila and R. Swick, "Resource Description Framework (RDF) Model and Syntax Specification," W3C Recommendation, World Wide Web Consortium, Feb. 1999; www.w3.org/TR/REC-rdf-syntax

[36] National Interagency Fire Center, http://www.nifc.gov/fireInfo/fireInfo_main.html.

[37] A. Sheth and K. Thirunarayan, "Semantics empowered Web 3.0: managing enterprise, social, sensor, and cloud-based data and services for advanced applications," *Synthesis Lectures on Data Management*, vol. 4, no. 6, pp. 1–175, 2012.

[38] W. Wei and P. Barnaghi, "Semantic Annotation and Reasoning for Sensor Data," in *Proceedings of the 4th European Conference on Smart Sensing and Context*, Berlin, Heidelberg, 2009, pp. 66–76.

[39] K. Thirunarayan and J. Pschorr, "Semantic information and sensor networks," in *Proceedings of the 2009 ACM Symposium on Applied Computing*, 2009, pp. 1273–1274.

[40] A. Zafeiropoulos, N. Konstantinou, S. Arkoulis, D.-E. Spanos, and N. Mitrou, “A Semantic-Based Architecture for Sensor Data Fusion,” in *The Second International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies, 2008. UBICOMM ’08*, 2008, pp. 116–121.

[41] A. Moraru, C. Fortuna, and D. Mladenić, ‘Using semantic annotation for knowledge extraction from geographically distributed and heterogeneous sensor data’, *4th SensorKDD, ACM*, 2010.

[42] D. B. Lenat, “CYC: A Large-scale Investment in Knowledge Infrastructure,” *Communications of the ACM*, vol. 38, no. 11, pp. 33–38, Nov. 1995.

[43] U. Haque, "Pachube," http://www.pachube.com

[44] D. Pfisterer, K. Romer, D. Bimschas, O. Kleine, R. Mietz, C. Truong, H. Hasemann, A. Kröller, M. Pagel, M. Hauswirth, M. Karnstedt, M. Leggieri, A. Passant, and R. Richardson, “SPITFIRE: toward a semantic web of things,” *IEEE Communication Magazine*, vol. 49, no. 11, pp. 40–48, Nov. 2011.

[45] F. Amato, V. Casola, A. Gaglione, and A. Mazzeo, “A semantic enriched data model for sensor network interoperability,” *Simulation Modeling Practice and Theory*, vol. 19, no. 8, pp. 1745–1757, Sep. 2011.

[46] A. Gyrard, “A Machine-to-machine Architecture to Merge Semantic Sensor Measurements,” in *Proceedings of the 22Nd International Conference on World Wide Web*, Republic and Canton of Geneva, Switzerland, 2013, pp. 371–376.

[47] X. Su, H. Zhang, J. Riekki, A. Keränen, J. K. Nurminen, and L. Du, “Connecting IoT Sensors to Knowledge-based Systems by Transforming SenML to RDF,” *Procedia Computer Science*, vol. 32, pp. 215–222, 2014.

[48] V. Komulainen, A. Valo, and E. Hyvönen, “A collaborative ontology development and service framework ONKI,” *Paper, Helsinki University of Technology, Laboratory for Media Technology,*2005.

[49] N. F. Noy and M. A. Musen, “Ontology versioning in an ontology management framework,” *IEEE Intelligent Systems*, vol. 19, no. 4, pp. 6–13, Jul. 2004.

[50] A. Alishevskikh and G. Subbiah, *"*Simple Ontology Framework API*",*
*http://sofa.projects.semwebcentral.org*

[51] S. Zhou, H. Ling, M. Han, and H. Zhang, “Ontology Generator from Relational Database Based on Jena”, *Computer and Information Science*, vol. 3, no. 2, Apr. 2010

[52] J. D. Case, M. Fedor, M. L. Schoffstall, and J. Davin, “Simple Network Management Protocol (SNMP),” RFC Editor, RFC1157, May 1990; DOI: http://dx.doi.org/10.17487/RFC1157

[53] W. Chen, N. Jain, and S. Singh, “ANMP: ad hoc network management protocol,” *IEEE Journal on Selected Areas in Communications,* vol. 17, no. 8, pp. 1506–1531, Aug. 1999; ISSN: 0733-8716

[54] I. Khan, F. Belqasmi, R. Glitho, N. Crespi, M. Morrow, and P. Polakos, "Wireless Sensor Network Virtualization: Early Architecture and Research Perspectives," *IEEE Network Magazine*, vol. 29, pp. 104–112, 2015.

[55] M. Liu, T. Leppanen, E. Harjula, Z. Ou, A. Ramalingam, M. Ylianttila, and T. Ojala, "Distributed resource directory architecture in Machine-to-Machine communications," in *2013 IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, 2013, pp. 319–324.

[56] J. Mäenpää, J. J. Bolonio, and S. Loreto, "Using RELOAD and CoAP for wide area sensor and actuator networking," *EURASIP Journal on Wireless Communications and Networking*, vol. 2012, no. 1, pp. 1–22, Mar. 2012.

[57] E.K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A survey and comparison of peer-to-peer overlay network schemes," *IEEE Communications Surveys & Tutorials*, vol.7, no.2, pp.72,93, Second Quarter 2005.

[58] M. Compton, P. Barnaghi, L. Bermudez, R. GarcíA-Castro, O. Corcho, S. Cox, J. Graybeal, M. Hauswirth, C. Henson, and A. Herzog, "The SSN ontology of the W3C semantic sensor network incubator group," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 17, pp. 25–32, 2012.

[59] M. Botts, G. Percivall, C. Reed, and J. Davidson, "OGC® sensor web enablement: Overview and high level architecture," in *GeoSensor Networks*, Springer, 2008, pp. 175–190.

[60] C. Jennings, Z. Shelby, and J. Arkko, "Media types for sensor markup language (SenML)," *IETF Network Working Group,* Internet-Draft, Jan. 2012; https://tools.ietf.org/html/draft-jennings-core-senml-00.

[61] I. Khan, S. Sahoo, R. Glitho, and N. Crespi, "A genetic algorithm-based solution for efficient in-network sensor data annotation in virtualized wireless sensor networks," presented *at the 13th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, Las Vegas, USA, 2016.

[62] A. Zahariev, "Google App Engine," in *TKK T-110.5190 Seminar on Internetworking*, pp. 1-5, Apr 27, 2009.

[63] Apache Jena, "A free and open source Java framework for building Semantic Web and Linked Data applications," 2011; http://jena.apache.org/

[64] H. Li, "RESTful Web service frameworks in Java," in *Signal Processing, Communications and Computing (ICSPCC), 2011 IEEE International Conference on 1*4-16 Sept. 2011, pp. 1–4.

[65] L. Gong, "JXTA: A network programming environment," *IEEE Internet Computing,* vol. 5, no. 3, pp. 88–95, 2001.

[66] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki - a lightweight and flexible operating system for tiny networked sensors," *in 29th Annual IEEE International Conference on Local Computer Networks, 2004*, pp. 455-462, 16-18 Nov. 2004; doi: 10.1109/LCN.2004.38

[67] F. Crivellaro, "μ Jena: Gestione di ontologie sui dispositivi mobile," Thesis, M.Sc., Politecnico di Milano, Milan, Italy, 2007.

[68] R. Hoare, J. Ahn, and J. Graves, "Discrete event simulator", *Google Patents*, 2002.

[69] Weisstein, W. Eric, "Normal Distribution." From *MathWorld* A Wolfram Web Resource. http://mathworld.wolfram.com/NormalDistribution.html