TRUST MANAGEMENT IN OPPORTUNISTIC NETWORKS: A SEMANTIC WEB-BASED APPROACH

By
Elvira Noelly Bonilla Tamez
B.Sc., La Salle University, Mexico City, Mexico, 1994.

A thesis
presented to Ryerson University
in partial fulfillment of the
requirements for the degree of
Master of Science
in the Program of
Computer Science

Toronto, Ontario, Canada, 2009

I hereby declare that I am the sole author of this thesis.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

ABSTRACT

TRUST MANAGEMENT IN OPPORTUNISTIC NETWORKS: A SEMANTIC WE-BASED APPROACH

Elvira Noelly Bonilla Tamez

MSc. Computer Science, Ryerson University, 2009

The need for having a mechanism to automatically interpret content available on the Web without a human intervention has lead to the development of a new vision for the next generation of the Web, known as the Semantic Web. This new paradigm advocates the use of ontologies to achieve a common language for communication among humans, computers, and programs. In this thesis, a novel Semantic Web-based solution called SCOW-Q (Semantic Capability discovery With QoS) model, is proposed, which provides an architectural basis for representing trust and trust management in Opportunistic Networks. The model is validated by means of a Use Case Scenario using a well-defined Semantic Web Service framework.

ACKNOWLEDGMENTS

I offer my sincerest profound gratitude to my supervisor, Dr. Isaac Woungang whose invaluable guidance, encouragement, and support enabled me to develop an understanding of the subject materialized in this thesis. Without Dr. Woungang's knowledge and wise advice, the completion of this work would not have been possible.

I am also deeply grateful to my co-supervisor Dr. Cherie Ding who always posted the appropriate questions and suggestions to stimulate my thinking. I would also like to thank Dr. Leszek Lillien for providing feedback and useful suggestions to my work.

The elaboration of this research work is the culmination of two years of intensive learning that were guided by the experience and knowledge of my professors. In addition to my supervisor and co-supervisor, I gratefully thank Dr. Alex Ferworn for making every class a joyful experience, Dr. Marcus Santos for helping me to discover the world of Evolutionary Programming and to Dr. Eric Harley for teaching me how to excavate knowledge mines using data mining techniques; and to all the faculty members and staff for their support in making this moment a reality.

I would also like to thank my family for all their unconditional support and comprehension, but most importantly for being the motivation that keeps me looking forward to give them the best in me.

DEDICATION

To my husband for supporting me and encouraging me every step on the way

To my daughter for her infinite patience and understanding

To my Mom for always being there to help me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

CC	Distributed Control Center
CEE	COCOA Execution Engine
COCOA	Conversational-base Service Composition in Pervasive Computing Environments with Quality of Service Support
COCOA-CI	COCOA Dynamic Service Composition
COCOA-L	COCOA Language (supported by OWL-S ontology)
COCOA-SD	COCOA Service Discovery
CR	COCOA Repository
CRS	Centralized Reputation System
CSP	Client Server Paradigm
DAML	DARPA Agent Markup Language
DARPA	Defense Advanced Research Projects Agency
DRS	Distributed Reputation System
FOL	First Order Logic
HTTP	Hypertext Transfer Protocol
IOPE	Input, Outputs, Preconditions and Effects
KAS	Kerberos Authentication Server
KDC	Key Distribution Center
MANETs	Mobile Ad hoc Networks
OIL	Ontology Inference Layer
Oppnets	Opportunistic Networks

OWL	Ontology Web Language
P2P	Peer-to-Peer
PriUH	Private Unknown Helpers
PUH	Public Unknowns Helpers
QoS	Quality of Service
RDF	Resource Description Framework
SCP	Adaptive Services to Client Paradigm
SEP	Spontaneous Service Emergence Paradigm
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SWS	Semantic Web Services
TGS	Ticket Granting Server
TKH	Trusted Known Helpers
UML	Unified Modeling Language
UPnP	Universal Plug and Play
URI	Uniform Resource Identifier
WCN	Wireless Community Network
WS	Web Service
WSA	Web Service Architecture
WSDL	Web Service Description Language
WWW	World Wide Web
XML	Extensible Markup Language

Chapter 1 INTRODUCTION

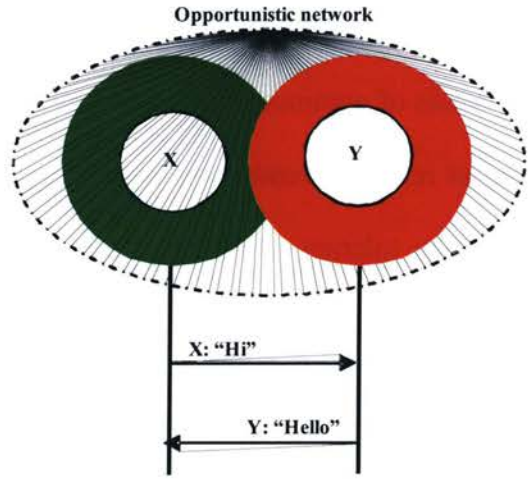
1.1 Opportunistic Networking Paradigm

Over the years, we have witnessed the evolution of the need for expanding the benefit of computer resources to larger audiences through the implementation of wired networks, wide area networks and wireless networks presented in a form of cell phones or hand held devices that deliver computing services at the fingertips of users. These advancements have led to continuous research and development of technologies aimed at enabling ubiquitous communications and bringing to light the paradigm of pervasive computing. In this environment, access to systems and information processing could occur throughout everyday life devices at any time. Some instances of pervasive computing environments include distributed computing environments, ubiquitous computing environments, mobile and ambient networking environments, and most recently opportunistic networking environments.

The opportunistic networking paradigm has mostly been motivated by the desire to move from a network of devices to which users must adapt, to a network that adapts itself to user needs and behaviors. Opportunistic networks are self-organized wireless networks that use opportunistically all kinds of communication possibilities that wired or wireless devices can offer. In the current research trends, one can distinguish two major types of opportunistic networks: (1) Delay tolerant networks - also referred to as network of regional networks [1], and (2) Opportunistic capability utilization networks (or Oppnet for short) [2] – which can be defined as specialized ad hoc networks and systems (SAHNS), where diverse external systems, not originally employed as nodes of the

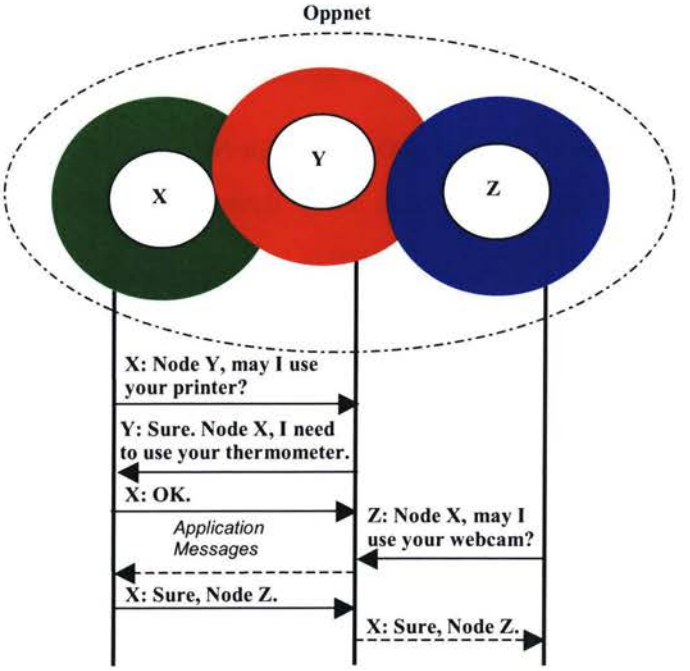
networks, can join dynamically in order to perform certain tasks for which they have been called or ordered to participate in.

Oppnet [2] enables an opportunistic growth of networks and an opportunistic use of resources gained by this growth. These authors advocated three classes of Oppnets [3]: (1) Class 1 referred to as *Opportunistic Communication Networks* – Their goal is to avail communication capabilities by interconnecting previously disconnected devices when there is a need for such communication (typically, they dynamically configure an ad hoc network), (2) Class 1.5 referred to as *Opportunistic Data Dissemination Networks* – Their goal is to opportunistically disseminate data. Here, devices or networks initiate a dynamic interconnection with other detected components (devices or networks), with the goal to propagate data, and finally (3) Class 2- *Opportunistic Capability Utilization Networks* – This class aims at using all sorts of capabilities (including Class 1 & Class 1.5). Here, capabilities refer to resources, services, skills, such as processing power, storage, sensing and actuating, software, hardware, to name a few. The above three classes of Oppnets are depicted in Fig. 1.1. This thesis focuses on Class 2 Oppnets (denoted Oppnets for short).

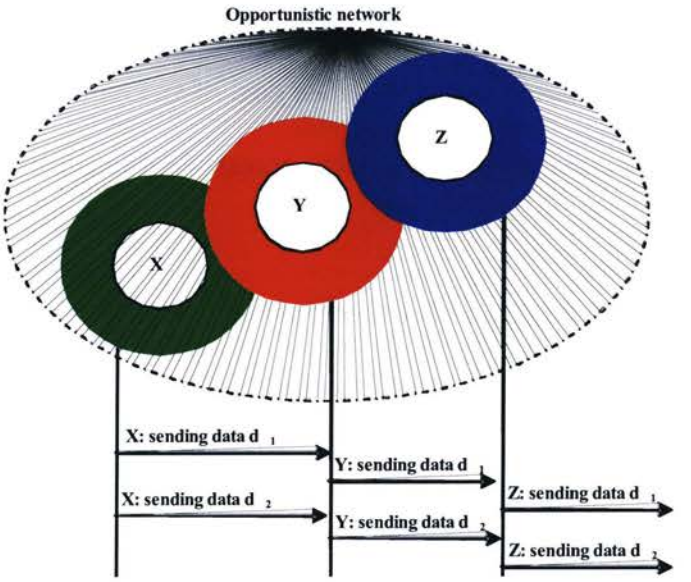


(a) Class 1 Opportunistic Networks: opportunistic *communication* when devices X and Y are within communication range of each other.

Communication ranges of X and Y, indicated by intersecting circles centered on X and Y, overlap. X and Y work together opportunistically, forming a Class 1 Opportunistic Network indicated with the oval outline



(b) Class 1.5 Opportunistic Networks: opportunistic *dissemination* of data.
Communication range of X does not intersect with the communication range of Z, therefore data must be propagated to Z via Y that has range overlapping with X and Z. X, Y and Z work together opportunistically, forming a Class 1.5 Opportunistic Network indicated with the oval outline.



(c) Class 2 Opportunistic Networks: opportunistic use of *any needed and available capabilities* (including but not limited to communication capabilities or data dissemination). Communication range of X does not intersect with the communication range of Z, therefore capabilities of X can be used by Z only indirectly with the help of Y, forwarding Z's messages. X, Y and Z work together opportunistically, forming a Class 2 Opportunistic Network indicated with the oval outline.

Figure 1-1: Classes of Opportunistic Networks [3]

One of the difficulties that the Oppnet paradigm presents occurs when Oppnets are being formed through its opportunistic nature. Typically, highly heterogeneous devices with diverse capabilities within each others reach are considered as candidate nodes. A seed node which has been spontaneously deployed (for instance a typical ad hoc wireless network), discovers and recruits candidate nodes, these nodes once invited to join the network are added to the Oppnet architecture, thereby becoming its "*helpers*". It should be noticed that candidate helpers might or might not accept the invitation to join the Oppnet. However, in emergency situations, helpers are ordered to join. Some other nodes in an Oppnet known as *Oppnet Reservists* [3] are those helpers that are always available to offer their services to the Oppnet. These helpers can facilitate the discovery, contact, and collaboration among nodes in an Oppnet. The goal of an Oppnet is to be able to take advantage of the various capabilities and services accessible through its helpers to realize its tasks.

In order for seamless interaction to occur between candidate nodes and a growing Oppnet, a common basis for communication must first be established. The communication challenges identified at this point in expanding an Oppnet are twofold: (1) Establishing a trusted relationship between seed nodes and candidate nodes, and (2) Designing a mechanism through which nodes in an Oppnet could communicate with

each other and the Oppnet in order to discover potential helpers and evaluate their capabilities in contributing to achieve the Oppnet goal.

1.2 Motivation

The motivation for this research work is to propose a solution to address the two challenges mentioned above in the Oppnet paradigm. This thesis advocates that by learning from the contexts of trust and trust management in the Semantic Web [4] using ontologies and Service Oriented Architectures [5], it is possible to introduce trust and trust management with QoS specifications into the Oppnet context. Typically, this research argues that it is possible to establish a communication pathway to identify available capabilities in trustworthy candidate nodes and helpers. To this effect, (1) a mechanism for establishing a "secure and trusted" communication must be put in place to exploit resources and services offered by heterogeneous devices in such a pervasive environment, and (2) this mechanism must be followed by the discovery of capabilities through the implementation of Semantic Web services with support of QoS specifications.

1.3 Objectives

The objectives of this thesis are twofold: (1) proposing a novel Semantic Web solution called SCOW-Q, which stands for "Semantic Capability discOvery With QoS" model. The objective of this model is to provide an architectural basis for representing trust and trust management in Oppnets. In this intended solution, software agents are expected to use trust information from the Semantic Web framework, to make security decisions, for instance, on access control. The design of the SCOW-Q model will be

based on two well-known principles: the use of Semantic Web as underlying technology and the use of ontologies as the language for communication among agents in order to discover and securely share potential services. (2) Illustrating the proposed SCOW-Q model through a use case scenario – In this task, it will be shown how Semantic Web Services can be used to realize trust and trust management in an Oppnet environment using the capability discovery process of the SCOW-Q model.

1.4 Thesis Outline

The rest of this document is composed of the following chapters:

Chapter 2: Related Work

In this chapter, the definition of Oppnet (started in Chapter 1) is complemented by describing the various elements of an Oppnet architecture. This description is particularly important in the context of this thesis since the type of helper's capability plays a central role within the design of the SCOW-Q model. Related works on trust management for the Semantic Web are presented, and an analysis of the applicability of trust management to Oppnet.

Chapter 3: SCOW-Q Model

This chapter constitutes the core of this thesis. The main contribution of this research is presented therein: (1) the SCOW-Q model is introduced, and an in-depth description of its capability discovery process is provided; and (2) the functionality of the SCOW-Q model is provided through a series of use cases.

Chapter 4: Illustrating the SCOW-Q Model

This chapter illustrates how the SCOW-Q model will perform by proposing a demonstrative use case scenario using a well-known Semantic Web Services framework.

Chapter 5: Conclusions

This chapter concludes this research work by analyzing and summarizing the results, as well as highlighting future enhancements to the SCOW-Q model. Also, a discussion of a few interesting questions that have arisen from the work in this thesis is provided.

Chapter 2 BACKGROUND RESEARCH

2.1 Elements of an Oppnet Architecture

Oppnets differ from traditional networks, in which the nodes of a single network are all deployed together, with the network size and locations of its nodes pre-designed. In an Oppnet, the initial seed Oppnet grows into an expanded Oppnet by enrolling foreign nodes, which become helpers, assisting the Oppnet in the realization of its goals. Due to the nature of this Oppnet growth, candidate helpers may be highly heterogeneous devices with diverse software and hardware capabilities.

In an Oppnet, nodes are typically mobile devices owned by users, but can be fixed devices too. Connection links via which nodes can discover each other can be made of diverse technologies such as Bluetooth WiFi, RFID, cellular-based technologies, ham radio, satellite, to name a few. Finally, the network itself is composed of several partitions named "regions". Therefore, an end-to-end path between the source and destination may never exist, or even if it exists, it may last only for a brief and unpredictable period of time. This means that a forwarding path between a node pair must be designed on the fly in order to achieve packets transmission. Hence, the Oppnet paradigm advocates (1) opportunistic communication – which enables nodes and user devices to self-configure and exploit the resources dynamically, and (2) pervasive networking scenarios, which promote epidemic data exchanges among devices in temporary proximity. Due to these requirements, a semantic, service-oriented middleware such as the COCOA model [6] that deals with the dynamics and

heterogeneity inherent to pervasive computing environments is appropriate for studying the service provisioning and management between entities in an Oppnet environment.

A typical Oppnet architecture (Fig. 2.1) [1] is made of: (1) a seed Oppnet – often a self-configured ad hoc network; (2) distributed Control Center nodes (CC nodes) – a subset of seed nodes with a flexible size based on the number of admitted or expelled nodes. This distinguished set of nodes look into the overall operations of the Oppnet; (3) helpers – wired or wireless entities able to capture, communicate, and transmit information or signals. For instance, a sensor, a cell phone, a vehicle or an appliance could be considered potential helper nodes. Within the Oppnet, helpers have the ability to discover other potential helpers and invite them to join the network. However, the discovery of helpers goes beyond locating resources, as it requires the design of some negotiation techniques to identify trustworthy and good quality helpers; (4) lites – helpers with limited capabilities.

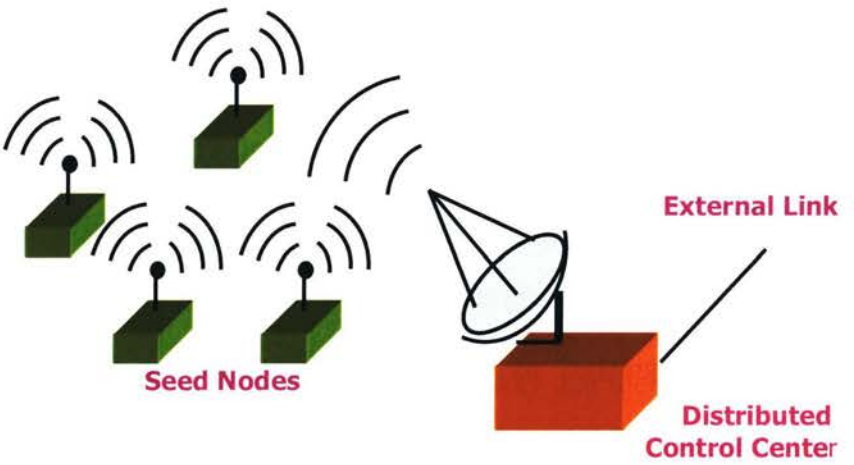


Figure 2-1: Oppnet Seed nodes [1]

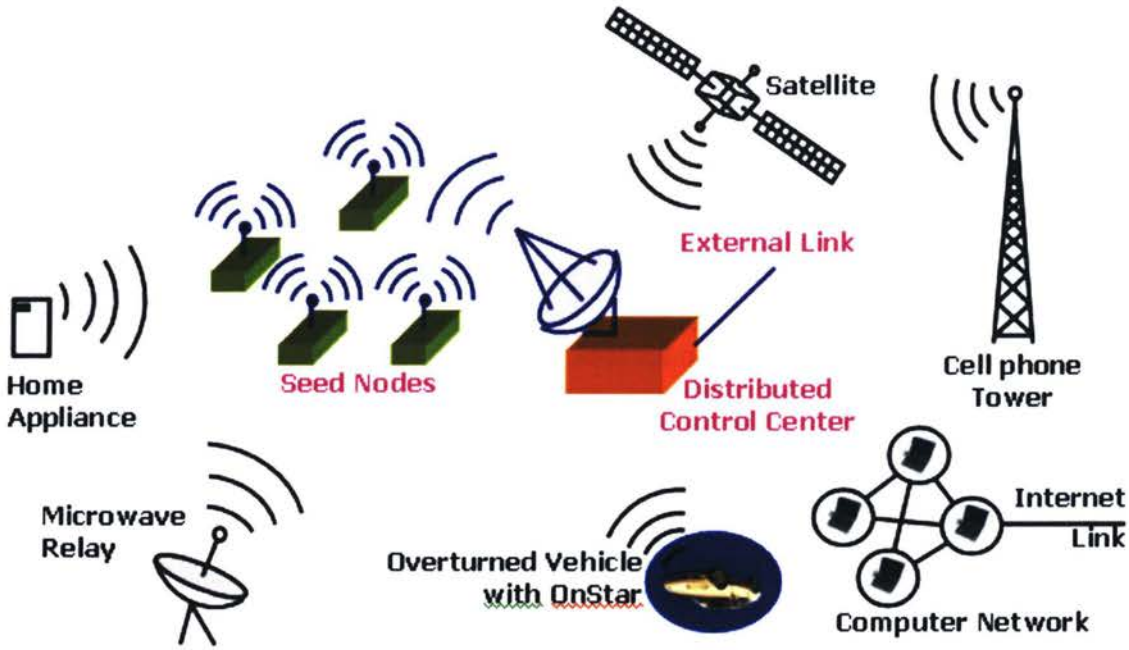


Figure 2-2: Expanded Oppnet [2]

It should be noted that in an Oppnet environment, a seed Oppnet is first deployed; then once operational, its first task is to analyze the jobs in hand and the capabilities (e.g., storage resources, skills, etc.) necessary to complete these tasks. Afterwards, the Seed scans the nearby ambient environment and detects helpers (e.g., devices or networks) that offer the needed capabilities, then invite or order them to join the Oppnet, in order to help the Oppnet in fulfilling its goal (forming an Expanded Oppnet (Fig. 2.2). In this exercise, helpers can be distinguished as of three types, based on how easy Oppnet can communicate with each of them: (1) *Oppnet-enabled insiders* – nodes that already belong to an Oppnet, (2) *Oppnet-enabled outsiders* – this category of helpers are those that are easy for an Oppnet to communicate with because they are equipped with primitives and protocols defined by the Oppnet Virtual Machine [7]. *Oppnet reservists* is a subclass of this class of helpers who have agreed a-priori to help an Oppnet; and (3) *Oppnet-*

unenabled outsiders – those helpers that cannot communicate with the Oppnet. This distinction is important because from a communication point of view, before an Oppnet get use of an outsider helper, it must first obtain this outsider's agreement. Once granted, the Oppnet has to deal with issues generated by the enrollment of outsiders in its architecture.

Two problems (among others) for which the work proposed in the context of this thesis can help towards finding some solutions are: (1) the issue of the quality of service (QoS) and resource utilization by these outsiders and (2) the issue of trustworthiness of insiders and outsiders in Oppnets.

2.2 Quality of Service in Oppnets

It is often necessary to ensure that certain QoS parameters are met in a computing system in order for jobs to be efficiently completed in a timely fashion. Several definitions of QoS exist depending on the context where the concept of quality is applied. Typically, QoS refers to the definition of an *expected behavior*. In this thesis, we adopt the definition of QoS provided in [3], which considers *throughput* and *delay* as the most fundamental QoS measures of the QoS to the consumers in an Oppnet environment. To this effect, an Oppnet realizing users' jobs needs to invoke its own resources as well as helpers' capabilities, and then utilize them in such a way that the QoS demands are met.
In an Oppnet, resources are scattered across the network and may be offered to a node requesting them by one or several other nodes of different processing power and varying communication bandwidths. In [3], a method for identifying the resources in an Oppnet, coupled with a Service Location and Planning (SLP) mechanism for optimizing resource utilization in Oppnets have been proposed.

2.3 Trustworthiness of Insiders and Outsiders in Oppnets

The key purpose of Oppnets is to benefit from the resources and capabilities available via its helpers to realize its goal. To this effect, one of the major challenges of an Oppnet is to timely detect and identify the malicious devices, and to prevent them from joining the network, while ensuring the privacy and security of the Oppnet. Of course, the ability to talk and disseminate information among nodes securely is of paramount importance. As pointed out in [3], Oppnets have two mechanisms of defense: (1) preventive mechanisms- which prevent the incorporation of malevolent helpers into the network; and (2) reactive mechanisms - which monitor the behavior of already incorporated nodes to identify malicious entities. The former can be strengthened by mechanisms that allow Oppnet to maintain a list of untrusted entities as well as a list of trusted devices that are assessed based on experience and reputation.

This thesis focuses on proposing a possible solution to address trust and trust management in an Oppnet environment, by arguing that one can learn from the context and experience of the Semantic-Web frameworks. To this effect, ontologies play a crucial role since they provide a means for nodes in an Oppnet to achieve a common communication language with outsiders, and they can be used to enable the resources to specify their QoS requirements. It is therefore important to explore the literature on how trust has been used as one of the security pathways for the Semantic Web.

2.4 Trust Management for the Semantic Web

Trust has been used in various ways as a solution for enhancing security in the World Wide Web. Using trust, devices can guide their behaviors. To this effect, a method should

be designed to evaluate the level of trust between devices, while reflecting the relationships between devices. The mechanism that deals with the evaluation, collection, and propagation of trust is referred to as *trust management* [8].

Designing a trust management solution in dynamic and pen computing environments such as multi-agent systems, Web services and pervasive networks such as Oppnets require an appropriate definition of trust [9], [10], [11]. A common basic definition used for trust is as follows: "the trust that device A places in device B is the strength of device A's belief that device B will behave without malicious intent, and the service that device B provides to A will satisfy device A's request, within a specified context" [7]. In fact, trust is not only viewed as a measure of a device's faithfulness, but also as an indicator of the QoS that a device can provide. Trust management design also involves dealing with specific issues such as trust models design, trust evaluation techniques, to name a few.

Several authors have investigated trust management solutions in the World Wide Web, which are suitable for the Semantic Web. Without loss of generality, these solutions can be classified into four categories: (1) Policy-based trust management models [12], [13], [14], [15], [16], [17], (2) Reputation-based trust management models [15], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], (3) Trust models based on information resources [31], [32], [33], [34], and finally (4) General trust management models [35], [36], [37], [38], [39], [40], [41],. It is worth mentioning that only solutions that are the most relevant to the context of the work carried in this thesis have been cited here. This thesis advocates using policy-based models.

2.4.1 Policy-Based Trust Management Models

Policy-based trust management approaches for securing the Semantic Web have attracted a lot of attention [14], [17]. In these schemes, a common approach has been to use trust management with deontic notions of rights, obligations, and prohibitions, expressing the authorization decisions through the application of relevant security, trust policies, and high-level declarative languages such as RDF-schema, DAML, XML, OWL, to name a few. These languages form a necessary basis for the construction of the Semantic Web. In these settings, policies are used to define rules and constraints on agents, and the actions that these agents can take on objects in terms of their credentials and properties. In this context, credentials rely mainly on security certificates with digital signatures to trust entities for exchange of information, and policies are applied to facilitate administrative tasks such as configuration, security or specification of QoS requirements. One key feature of policy-based approaches is that they allow the separation of roles that govern the behavior from the functionality provided by an entity. Typical illustrative examples of this class of trust models are as follows. The Kerberos protocol [12], [13] – a network authentication-based protocol which uses a third party (known as Key Distribution Centre –KDC-) to securely exchange credentials (digital signatures) between two entities across an insecure network. Once the identity of the two parties involved in a communication is known, the protocol issues encryption keys in a form of messages (known as tickets) to enable a communication session. These tickets are valid for designated period of time and can be used by the same parties until they expire. The Web Service community [42] has adopted the Kerberos protocol to securely exchange SOAP messages. In this environment, including the Semantic Web

environment, policies provide the description of security rules in a machine understandable way to control access to resources. In [43], Olmedilla et al. have categorized the policies based on the mechanism used to expose them, leading to two distinguished policy languages commonly used in the Semantic Web that allow an undisclosed access of policies, the KAoS [44], [45] and the REI [46] frameworks. The former provides the ability for establishing a distributed policy exchange and changing the policies dynamically, whereas the latter describes policies based on the entities' capabilities of expressing what is doable or what is not doable. As a follow up of the work in [43], the authors in [47] suggested the *PeerTrust* [48] and the *Protune* [49] policy languages as alternative solutions to overcome the drawbacks of the REI framework.

2.4.2 Reputation-Based Trust Management Models

Reputation is a measure of trust where an entity retains information on the reputation of other entities, creating what is known as a *Web of trust*. Through this concept, each node is considered as an entity and each edge is assigned a weight in the form of a trust value. This trust value is based on a predefined trust metric that reflects the reputation that one entity assigns to another entity [31]. Typically, "experience" is the key element based on which decision on the trustworthiness of an entity is made. An entity's experience alone or combined with others entities' experiences are the determining factors to accessing the reputation of what is expected to be trustworthy resources. The basic idea behind this concept is to let parties rate each other after a transaction has been completed with the purpose of compiling local and external ratings to build a reputation score. In [18], Krukow et al. have introduced the notion of "experience-based trust management" models, where trust is determined based on past behaviors of entities in the

environment. A similar approach was described in [22], where trust models are based on the concept of "entity's rating". A rating on the performance of a given entity is collected from other members who have had previous experience of interactions with that particular entity. This rating is used by a central authority to make a decision on the entity's reputation. The reputation's trustworthiness can further be determined by implementing for instance a mechanism of referrals [23]. Well-known examples of this type of reputation-based trust models are the EBay [50] and Amazon Auctions [51] websites. Using the same concept of rating, some distributed and decentralized agent architectures were introduced in [24] in order to disseminate the ratings that entities can give to each other. Similarly, in [25], Agrawal, et al. proposed a distributed approach where each user can specify a group of users that he/she could trust, resulting in a Web of trust that could collect trusted entities from other trusted user's list, building what they refer to as a "trust graph". Zhang and Cohen [28] present a trust model based on the concept of social network analysis. This model calculates trust values based on pair wise trust ratings and reliable factors of acquaintances as key information. Similarly, Golbeck et al. [27] propose a trust model that integrates the applicability of the social network analysis into the Semantic Web, based on multi-dimensional networks using ontologies. In the same vain, Pennock et al. [29] apply social network algorithms to Webs of trust to identify users with high network influence.

2.4.3 Trust Models Based on Information Resources

A few trust models have been investigated in this category. In [31], Gil et al. addressed trust by analyzing the nature of the content provided by entities, and then used it to make trust judgments made on specific segments of content provided by Web

resources which are defined by a unique URI. Similarly, in [34], the authors derive a trust model based on user annotations of information sources, but the resulting trustworthiness values are not personalized for the individual using them. In a different approach [33], Chklovski et al proposed a system called "*Trellis*", and then used it to obtain consensus trust on information resources based on the experiences gained on a community of users.

2.4.4 General Trust Management Models

Trust models in this category focus on defining conditions, components, requisites, etc, that are based on the analysis of human-based trust decisions, looking into elements available within psychology and sociology, cryptography, to name a few, in order to identify the elements of trust. In [36], a trust management model based on a sociological viewpoint of trust was designed for authentication purpose. In [35], [37], some trust management models based on cryptographic schemes and the development of policies and security mechanisms at the lower infrastructure levels have been described. In [30], a model of trust was presented by Kamvar et al. which compute global trusts as a function of local trust values in a peer-to-peer network. In [39], Vijanen et al. proposed a trust ontology that enables various trust models to share trust relationship details and information. Their analysis of trust is centered in the information needed in the trust decision process.

This thesis advocates that a combination of the above trust models with a well-defined Semantic Web service framework can be coupled together to allow the establishment of QoS requirement specifications in an Oppnet environment. In this setting, a criteria model such as the one proposed by Kauster *et al.* [52] could be used to evaluate the Semantic Web services or to comprehensively analyze them.

2.5 Concept of Semantic Web Services

Most information on Web has been designed to be read and understood by humans. Computer programs and pages on the Web have the ability to interpret only some attributes of pages or contents. However, there is a limitation in understanding and processing the meaning of contents of the Web. To address this deficiency, a vision known as the *Semantic Web* was introduced [15]. The goal of the Semantic Web is to foster an environment where software agents (programs) can perform sophisticated tasks on behalf of the users using information with a well-defined meaning that can be processed automatically, resulting in a collaborative effort among humans and computers.

2.5.1 Semantic Web Paradigm

In the Semantic Web paradigm, online services and software agents, assisted by the use of *ontologies* (describing the application domain, determining its vocabulary, and describing the relationships of its elements), are able to interact in an autonomous way to satisfy users' requirements. Services are described by service specifications. A user that needs a particular service can find the service he/she needs. The service specification is not limited to service *functions* or characteristics; it also looks into *non-functional* properties that define requirements, such as QoS.

The Semantic Web uses two technologies that complement each other: the eXtensible Markup Language (XML) [53], and the Resource Description Framework (RDF). The former allows building a structure for contents and documents, but does not provide a clear meaning of what that structure is about. The latter compensates this deficiency, providing meaning to the content and documents by using sets of triples to describe data.

The Semantic Web is challenged in its capacity to express data and rules for reasoning. It faces a problem where one meaning is defined by two or more different terms. This issue can be addressed by including a collection of ontologies (i.e., structured vocabularies) that define the relationships among terms or concepts (also called *classes*). Ontologies enhance the communication and identification of services that are available to satisfy a given service request.

The ontology for the Web is composed of a "taxonomy and a set of inference rules" [15]. The taxonomy provides the definition of classes of objects and their relationships, and the set of inference rules provides the processing mechanisms to manipulate terms in a meaningful way.

Nowadays, ontologies are commonly expressed by the Web Ontology Language (OWL) which has its foundation in the Description Logics (DL) [54]. OWL allows for inferring of relationships between concepts and their definitions. It is an evolution of its predecessors, DARL and OIL.

Other approaches have also been developed to provide alternatives for enabling the semantic description of ontologies. The W3C organization developed the Web Service Architecture (WSA) [55] framework to provide standard definitions for the Web Services model, and the relationships among its different components. Web Services (WS) have been defined as "a software system designed to support interoperable machine-to-machine interaction over a network".

WS are characterized by two main entities: the providers and the requesters of services. These entities exchange messages through the use a Web Service Description Language (WSDL), which defines the functional specifications of services (for instance,

their data types, formats, and protocols). Semantics in WS refers to the expected behavior during service interactions. It represents a contract or an agreement between the participants.

Services are typically identified by a Unified Resource Identifier (URI), and rely on technologies for message exchange (such as XML) and transport protocols (such as HTTP and SOAP (Service Oriented Architecture Protocol). A URI can be used to identify physical resources, such as cell phones, TV sets, etc. The RDF (Resource Description Framework) language can describe capabilities of these resources, and the way in which they can operate together as if they were software agents.

Service descriptions include information on the service's interface and metadata that can be used during the service discovery process, all these ideally written in a machine-understandable language.

The term *profile* refers to the "document(s) exchange between devices that describe the capabilities of a device" [56]. The use of ontologies in profiles enables automatic search and discovery of needed functionality in devices, and utilization of well-described services.

An *agents* is a tangible "piece of software or hardware that sends and receives messages," while *services* are bundles of functionality that are advertised to be used or requested. Agents play a key factor in the success of the Semantic Web because they are responsible for collecting the content, processing it, and exchanging it; they can even allow agents that were not designed to work together to collaborate on data transfer when it is determined by the semantics to do.

2.5.2 Approaches for Semantic Web Services

Web Services and Semantic Web complement each other, evolving into what is known as Semantic Web Services [3], [57]. *Semantic Web Services* are self-contained and self-described entities that semantically advertise their capabilities and descriptions, and assist in the discovery, composition and binding of services. Semantic Web Services use machine-understandable language supported by ontologies, and function in an open, heterogeneous environment.

Service discovery is usually performed by agents that look for service descriptions matching the desired functional or semantic criteria specified in the user's service request. There are two important aspects of service discovery: the *service discovery architecture*, determining where services can be broadcast or advertised [58], and the *service matchmaking mechanism*, determining how services are compared against user's requests until a proper match is found. Typically, only the *functional* characteristics of a service are used in service matching, ignoring the *non-functional* properties, such as QoS requirements. Non-functional properties add extra service-matching criteria, thus narrowing down the number of functionally equivalent Web Services. Many approaches facilitating a semantic specification of Web Services have been proposed, including the following:

- The DARPA program [59] proposed the *DARPA Agent Markup Language* (*DAML*) as an extension of XML and RDF. The *Ontology Inference Layer* (*OIL*) adds the underlying ontology layer to Semantic Web. DAML and OIL were succeeded by DAML+OIL, a language that included features of both its predecessors, and eventually evolved into the Ontology Web Language (OWL).
- The DAML organization proposed the use of OWL as the "representation

language of choice for the OWL-S proposal". (OWL-S was formerly known as DAML-S.) The ontology of services, proposed by the OWL Service Coalition [84], provides information on: (a) the service profile that presents each service and describes its functionality and characteristics; (b) the service model that describes how the service works; and (c) the service grounding that provides details on how a service can be accessed.

- The Web Services Modeling Ontology (WSMO) [60] allows for service specifications and provides support for addressing inaccuracies in describing services. It uses the Web Service Modeling Language (WSML).
- In an effort to incorporate semantics into descriptions of the Web services, the W3C organization extended their WSDL language into the WSDL-S. WSDL-S is a standard format in XML, and is used to describe the "*network services as a set of endpoints*" [58] based on the syntax of the services. The WSDL-S language enhances the WSDL descriptions by adding to it semantics, and by incorporating into it some concepts from OWL, WSMO, to name a few. Recently, a standard called WSDL 2.0 has been introduced [58], providing "*a model and an XML format for describing Web Services*". This standard distinguishes between the *abstract description* of a service, and the *concrete specification* of the offered functionality (i.e. when, where, and how).

Another proposal is the *First-order Logic Ontology for Web Services* (*FLOW*). In addition to using XML and URIs for Web support, it utilizes semantics defined by means of the first-order logic. FLOW incorporates standards such as WSMO, OWL-S [61], and PSL (ISO 18629). In addition, it supports a direct mapping to the *Rules Ontology for Web*

Services (*ROWS*), a language from the DAML consortium based on logic-programming semantics.

2.5.3 Semantic Web Paradigm in Different Environments

This section concentrates on the implementation of the Semantic Web Services paradigm in different environments, as well as on the implementation of QoS with focus on Web service discovery and service selection [62].

- ***Semantic Web Services in Ubiquitous and Pervasive Computing***

Although there is a tendency to consider ubiquitous computing and pervasive computing as synonyms, Gaber [63] distinguishes between the two based on the environment they operate in and their interaction with it. *Pervasive computing* is considered to be adaptable and interacts with the closest environment, enhanced by context awareness and emergent functionalities. The purpose of *ubiquitous computing* is to provide users with a global access to services and devices anytime and anywhere. Gaber [63] provides the following classification of interaction paradigms: (i) the *client server paradigm* (*CSP*) is a traditional approach, in which a user places a request for a service already known as available; (ii) the *adaptive services to client paradigm* (*SCP*), suitable for ubiquitous computing, uses a "decentralized and self-organizing" agent-based approach to deliver the service to the user; and (iii) the *spontaneous service emergence paradigm* (*SSE*) deals with unexpected and spontaneous creation of services, which are provided by nodes interacting in ad hoc connections, and is suitable for pervasive computing environments.

There have been other approaches to handle Semantic Web Services in other environments. For instance, for pervasive computing environments, Amigo-S [64] is used to extend the OWL-S framework by integrating features that characterize the heterogeneity and richness of pervasive environments.

Due to the heterogeneous nature and user-centric goals of pervasive computing, the Semantic Web Services approach enables ad hoc relationships between service providers and requesters using semantics. However, challenges such as the limitation of resources entering the network and the absence of a centralized mechanism to maintain registries and ontologies for service discovery make the implementation of this technology difficult. Ben Mokhtar *et al.* [65] indicate that significant computational effort is one of the major disadvantages of using semantic technologies for service discovery within pervasive environments. They present a competitive scheme called EASY (Efficient Semantic Service Discovery) for pervasive environments with QoS context and support. EASY is not yet another service discovery protocol but can instead be layered on top of existing ones, leveraging semantic abstractions at a higher level. This approach is composed of two parts: (i) the EASY Language (EASY-L) ontology used for semantic service description, which assures independence of underlying layers (or middleware infrastructure) and addresses the specification of non-functional properties such as QoS; and (ii) the EASY Matching (EASY-M), which can be used to support matchmaking of non-functional services.

According to Chakraborty *et al.* [66], service discovery in pervasive computing should be decentralized, autonomous, self-advertised and adaptable, in order to reflect environmental challenges. The authors presented a novel approach to service discovery in

pervasive computing, in which discovery architectures and service matchmaking tasks are coupled, building upon the concepts of peer-to-peer dynamic caching, service advertising, and a group-based forwarding of service discovery requests. The service description is supported by OWL.

Gagnes *et al.* [67] indicated that not only a richer description of services is needed in dynamic environments but also better mechanisms to distribute these descriptions are required. Although UDDI can be an instrument for providing Web service descriptions, in their opinion it is not appropriate for delivering "semantic service advertisements in dynamic environments," where "dynamic" could be interpreted as a rapid and spontaneous change in topology or location, and where information sharing occurs in ad hoc modality. The authors proposed a generic service discovery architecture applicable to dynamic environments, which leverages concepts of Web services and a distributed multi-registry topology. They also present a categorization of service discovery topologies.

Mokhtar *et al.* [6] presented COCOA, a solution for a conversation-based service composition with QoS support. This approach consists of two mechanisms: (i) COCOA-SD for QoS-aware semantic service discovery; and (ii) COCOA-CI for QoS-aware service integration. The authors also discuss the issue of *syntactic heterogeneity* of service descriptions in a pervasive environment, assuming that most agreements between service requesters and providers are made based on a common service description syntax. To resolve this issue, they suggested implementing the semantic modeling of functional and non-functional service features through ontology-based semantic reasoning. COCOA-L is an OWL-S based language for semantic service specification and semantic-

aware description of services. This language is used by COCOA-SD to enable matching of service functionalities complemented by QoS-based matching.

- ***Semantic Web Services in Mobile Ad Hoc Services***

Rapidly changing characteristics as well as the autonomous nature and decentralized topologies of mobile ad hoc networks (MANETs) makes discovery of services a very challenging task. The dynamics of MANETs prevents the use of agreed upon or predefined service interfaces, while most of the existing models, architectures and languages have been developed considering a universally-connected environment, such as the one available in the Web. For this reason, we need to design a mechanism where an exchange of service representations can take place without using a formal representation. The use of Semantic Web Services is an alternative. However, it is necessary to have appropriate technologies that can handle the distribution of ontologies despite the spontaneous nature of MANETs.

Nedos *et al.* [68] presented a model for autonomous semantic service discovery, assuming a role symmetry, which means that each node can potentially be both a service producer and a service consumer. This model differs from others since the semantic representation is not shared by the nodes, but instead derived through node interactions. The authors also indicate that in order to apply standards for service discovery (such as WSMO, OWL-S or WSDL-S) in a mobile ad hoc environment, the following requirements should be satisfied: (i) nodes should interpret discovery queries with heterogeneous ontologies and maintain their own ontologies to describe their own services (since semantic interpretation is needed, an ontology matching process should be put in place to provide a common understanding); (ii) it is mandatory to have centralized

service registries that provide details on the longevity of ontology references, URIs, etc., and self-contained ontologies within the nodes should be used.

An important contribution of Nedos *et al.* [68] is the implementation of a *gossip protocol*, which is at the core of discovering and matching heterogeneous ontologies. Each node stores concepts in a buffer. Then, a lightweight ontology-matching mechanism matches those concepts with the ones received. In summary, the discovery process first identifies candidate nodes with compatible ontologies, and then uses those nodes to perform the service-matching step.

- ***Semantic Web Services in Peer-to-Peer (P2P) Environments***

Le-Hung *et al.* [69] present a distributed approach to the semantic discovery of Web services in peer-to-peer-based registries, considering QoS characteristics. They argue that their scheme is scalable, efficient and reliable. The scalability is achieved through the use of P2P overlays as a service repository network. The considered QoS is determined by the users' feedback for a given service. They emphasize evaluation of the credibility of the reporting users. According to the authors, their QoS model is unique and robust against malicious behaviors, by using known solutions for trust and reputation management in P2P systems [22], [70].

Verma *et al.* [71] propose the METEOR-S WSDI architecture, an environment for publication of Web services, and their discovery in multiple registries. It follows an ontology-based approach that facilitates organization of a registry into domains. Each registry is related to a specific domain using semantics for domain association. It is kept in a custom-made ontology known as the *Registries ontology*, containing the relationships between domains and registries.

- ***Semantic Web Services in Grid Computing***

Systems configured in a grid environment allow for deployment of data, resources, etc., in a virtual working environment, such as the Internet. This infrastructure takes advantage of heterogeneous resources that can be geographically dispersed. Grid technology display lends itself well to implementation of Semantic Web Services.

Ren *et al.* [72] present a model for grid-based semantic service discovery with QoS constrains. The model introduces a QoS-ranking approach for matching user-specified preferences rather than the traditional semantic matchmaking capability as used in engine-based service discovery. The paper presents an efficient QoS model using OWL QoS ontologies to meet the needs of non-functional requirements and QoS information collection. It also presents a classification of QoS parameters, grouping them into four categories: (i) *network* QoS parameters (bandwidth, delay, etc.); b) *system* QoS parameters (reliability, capacity, etc.); c) *task* QoS parameters (memory, CPU usage, response time, etc.); d) *extension* QoS parameters (reputation, security, etc.).

A summary of the above findings is given in Table 1. It is worth mentioning that in Chapter 4, we have adopted the COCOA frameworks [6] as a mean for illustrating the newly proposed SCOW-Q model for trust and trust management in Oppnet.

Table 1: Semantic Web and Web Services in Different Networking Environments [3].

Paradigm/Proposal	Service Description	Service Specification	Service Discovery
Mobile ad hoc Networks			
Nedos et al. [68]	- RDFS	- OWL-S	- Concepts are represented through a "network representation" - Gossip Protocol - Discovery Query - Syntactical Matching
Kopena et al. [73]	- OWL-S	- OWL-S	- Random walk service discovery agents, also described as a set of service monitoring agents - Profile matching
Pervasive Computing Paradigm			
EASY Ben Mokhtar *et al.* (2007) [6]	- Proprietary EASY-L based on OWL. Support for functional and non-functional services		
Peer-to-peer (P2P) Paradigm			
P2P–based registries with QoS Support Le-Hung *et al.* [69]	- WSMO - QoS requirements are described as normalized values in a set of triples $\{q_i, n_i, v_i\}$, where: q_i= *a QoS parameter* n_i= *order of importance of* q_i *vi= user's minimal required value*		- SD, selection and ranking based on the matching of service advertisements considering QoS, trust and reputation.
METEOR-S WSDI Verma *et al.* [71]	- WSDL + semantic publication of Web services in UDDI. Other developed algorithms are: SAWS to map concepts in WSDL description to an ontological concept returning the degree of similarity.	- Customized ontology called "Registries Ontology" where ontology/registry mapping occurs and properties such as QoS are stored.	- In the Operator Service layer of the proposed architecture conventional UDDI query and matching are supported. - It uses SOAP for non-UDDI registry implementation.
Grid Computing Paradigm			
Web service Discovery model with QoS constrains	- Service description is transformed onto the OWL-S profile specification	- Parsing of the request specifications to identify concepts and properties.	- Concepts are captured in a vector list (input/output) for matching purpose. - Extension of a previously semantic matching algorithm ([74])

2.5.4 Use of Semantic Web to Specify QoS Requirements in Oppnets

In the Oppnet context, the QoS requirements describe the behavior of a capability (e.g., a service) expected by a user (whether a human or an entity such as software, agent, etc). Moreover, QoS requirements are used to further refine the *capability discovery* and the subsequent *helper-matchmaking process* (which matches user needs with the descriptions of capabilities helper candidates) to narrow down the selection of helpers — potential capability providers. The main contribution of this thesis is the introduction of a novel model for implementing trust and trust management using Semantic Web capabilities (e.g. services) and QoS requirement specifications within the Oppnet paradigm. This is the subject of the next chapter (Chapter 3)

Chapter 3 The SCOW-Q Model

This chapter describes how the concepts of QoS requirements, expressed by the means of Semantic Web in the service delivery process, could fit into the Oppnet context. We propose a new model, referred to as the *Oppnet SCOW-Q* ("Semantic Capability discOvery With QoS in Oppnets"), which can be used (1) for implementing trust and trust management in Oppnets, and (2) for adapting QoS requirements to Oppnets.

3.1 The Oppnet SCOW-Q Model

Based on our desire to fulfill the goal of establishing a trustworthy communication among nodes in an Oppnet while providing a means for establishing QoS specification in such an environment, the design, establishment and functionality of the SCOW-Q model rely on a predefined classification of helper nodes (helpers), according to their access, capability, and QoS specifications. It is worth mentioning that this classification also relies on the basic distinction among helpers as introduced in Section 2.1 of Chapter 2 (i.e. whether a helper is an outsider node or an insider node to the Oppnet).

3.1.1 Classification of Helpers According to Their Access, Capability, and QoS Specifications

The pervasive and ubiquitous characteristics of an Oppnet expose it to a large variety of candidate helpers (including lites), which vary in terms of the nature and the amount of the capabilities they can provide. The Oppnet SCOW-Q model provides the following classification of helpers according to their access, capability, and QoS specifications:

- *Public Unknown Helpers (PUHs)* – PUHs are considered as candidate nodes or services (entities) that expose their access policies, services, QoS functional and

non-functional specifications to the network with the purpose of being identified and establish new relationships with those other entities that are looking for the provision of a particular service. Their access policies, capability descriptions, and QoS parameters are open to the public. Once the candidate node is invited to join the Oppnet and accepts the invitation by following a policy-based access mechanism, it becomes a formal PUH. The assumption made for nodes in this category is that, non pre-existing transactions have occurred between the PUH and the requester of the service (i.e. a seed node or a helper). Hence, there is no precedent on the performance of this node. If the outcome of the established interaction results in a successful transactional relationship, the performance of such a service will be recorded, retaining a reputation score, which is rated based on the feedback received by requesters of the service and made available to the Oppnet. These nodes have the potential to become trusted known helpers if a set of requisites is met (the definition of the requisites to become a trusted known helper are yet to be defined, this can be accomplished in later work) including a high reputation score. Examples of entities in this category are: sensor networks, appliances, GPS services, to name a few.

- *Private Unknown Helpers (PriUH)* – PriUHs are considered as candidate nodes or services (entities) located on restricted environments and that only advertise minimum information about the services provided with the purpose of establishing first contact with other nodes on the network. However, in order to use the services of PriUH, capabilities and QoS specification must be disclosed. First, it will be required to identify if the needed capabilities are available.

Second, it is required to establish a protocol through the negotiation process. Typically, the access policies of PriUHs are private. They do not exchange this information unless a negotiation takes place, and an agreement is reached. Helpers of this type publicly advertise capabilities and QoS parameters. However, their use must be preceded by negotiations. Once the candidate node is invited to join the Oppnet and accepts the invitation following a negotiation-driven access mechanism, it becomes formally a PriUH. These nodes have the potential to become formal trusted known helpers if a set of requirements is met. For example, a private network in a university, access to a personal account in a financial institution or access to Intranet networks in a governmental institution, to name a few.

- *Trusted Known Helpers (TKHs)* – TKHs or services (entities) are located within a known trusted environment and it is assumed that there has already been a pre-established relationship among them. Typically, their access policies, capabilities and QoS parameters are known to the Oppnet, and they can use private keys to identify each other. Seed nodes or helpers identified as trusted are, e.g., those located within a known trusted environment whose capabilities have already been successfully used by the Oppnet. This category of helpers is based on the concept that establishing a Web of trust (based on social network theories applied to the Semantic Web using ontologies) will result in self-sufficient Oppnet nodes able to manage their network of trusted Oppnet nodes based on the previous experience with them [75]. Examples of this class of helpers are: police services, emergency

preparedness and response services, governmental agencies that provides accurate weather information such as Environment Canada, to name a few.

- *Oppnet Reservists (Reservists)* – are considered as formal helper nodes primarily representing some volunteers (it should be noticed that these nodes are not considered as candidates since they are already formal helpers). The identities and services provided by the Reservists are previously known and could be considered Trusted Known Helpers. If there is a match between the application category of a Reservist and the Oppnet (e.g. both are tagged for "Emergency" applications), they are highly trusted by the Oppnet, more than the previous group of "Trusted known helpers". Once a Reservist has been identified and invited to join the Oppnet, an identity-based access mechanism can be utilized to access its services. The access policies, capabilities and QoS parameters of Reservists are known a-priori (from the Oppnet Reservist directory for the matching application category) to the Oppnet. In case of an emergency scenario, a Reservist does not have a choice but to accept joining the Oppnet. Some examples of Reservists are: traffic cameras in a local community, local weather stations, public networks at a library, to name a few.

Default QoS parameters can be established to ensure the minimum QoS level for interaction with helpers of a given class in case the requestor does not provide its own QoS specifications. The defaults are saved within service profiles.

One major parameter used in the above classification of helpers is the definition of the access policy of a helper. The access policy determines the mechanism that should be implemented in order to establish a trustworthy relationship among nodes in the oppnet.

For instance, if the candidate helper node is a server containing maps which are open to the public, and this service could be leveraged onto the Oppnet, then, a reputation-based method will be implemented. If the candidate helper node is a secure governmental network, then policy-driven negotiations [43] will be considered in order to access and retrieve information. If the service requested by the seed node or helper is looking into accessing confidential information from the police database (for instance, criminal records about an individual) then an identity-based access method will be used to access the required data given that a trusted relationship among the nodes is pre-existent.

3.1.2 Design and Functionality of the SCOW-Q Model

The design of the SCOW-Q model is inspired from that of a trust management system introduced by Ruohomaa et al. in [10]. The authors recommended that three tasks be fulfilled. The first task is the *initialization of a trust relationship* responsible for determining a suitable initial trust among entities. This task performs an initial assessment to determine if the entity with which a new relationship is to be established could be trustworthy or if this entity poses a risk of performing a malicious behavior. The second task is the *observation*. In this activity, the actual behavior of a trusted entity is monitored. Observation could be performed in a collaborative way providing that a feedback be active and constant. It could also be performed by an outsider, who acts as a silent third trusted party. Finally, the third task is the *evolution of reputation and trust*, where updating trust measurements is mandatory, specially to maintain in a good state the reputation scores of an entity. In [10], the authors also proposed some mechanisms for ensuring QoS during the semantic service discovery process, for each type of node in the network.

More precisely, the SCOW-Q model is based on the semantic service discovery process using ontologies. This process is applied to Seed Oppnet nodes to perform the *capability discovery* task in Oppnets. Common concepts utilized in this model are *helper advertisement* and *helper registry*, both based on the concepts known from the semantic service discovery process. *Helper advertisement* describes the capabilities (including services) provided by an entity. An *entity* is identified as a node, a seed node, or a helper node. These entities could vary in nature, from a group of PCs on a network to a PDA, or from a satellite connection to an intelligent appliance or a sensor. These participating devices must have a mechanism to advertise their capabilities. The *helper registry* is a list of all identified services that have some degree of functional and non-functional (including QoS) characteristics that can be matched based on the requestor's requirements.

The first entity considered in the SCOW-Q model (Fig. 3.1) is the Control Center (CC) node. This node has a great deal of interaction with other seed nodes. CC is responsible for having agents that constantly monitor the environment, searching for potential capabilities, maintaining and updating changes in ontologies, and keeping a repository in which a helper registry is maintained. When an Oppnet needs new capabilities, candidate helper nodes are located, and some of them are invited to join Oppnet as its helpers, providing their capabilities. Helper nodes can have assistants with limited functionalities but (in addition to providing their own simple services) able to locate other nodes and identify their advertised capabilities. These assistants are called *lites* (short for *lightweight helpers*).

The establishment and functionality of the Control Center, as well as its collaboration with seed nodes, are demonstrated within the innermost oval. The remaining two ovals, surrounding the innermost oval, connect different types of helpers that can exist during the lifetime of an Oppnet (i.e. trusted known helpers, public unknown helpers, private unknown helpers and Oppnet reservist as previously defined), with the outermost oval connecting only lites.

It should be emphasized that the capability discovery approach presented in this model could be implemented in open environments such as the Web, which already has an underlying layer that binds up services using protocols such as HTTP. Moreover, this model is adequate for environments using ad hoc connectivity, such as Mobile Ad Hoc Networks (MANETs) or peer-to-peer (P2P) systems.

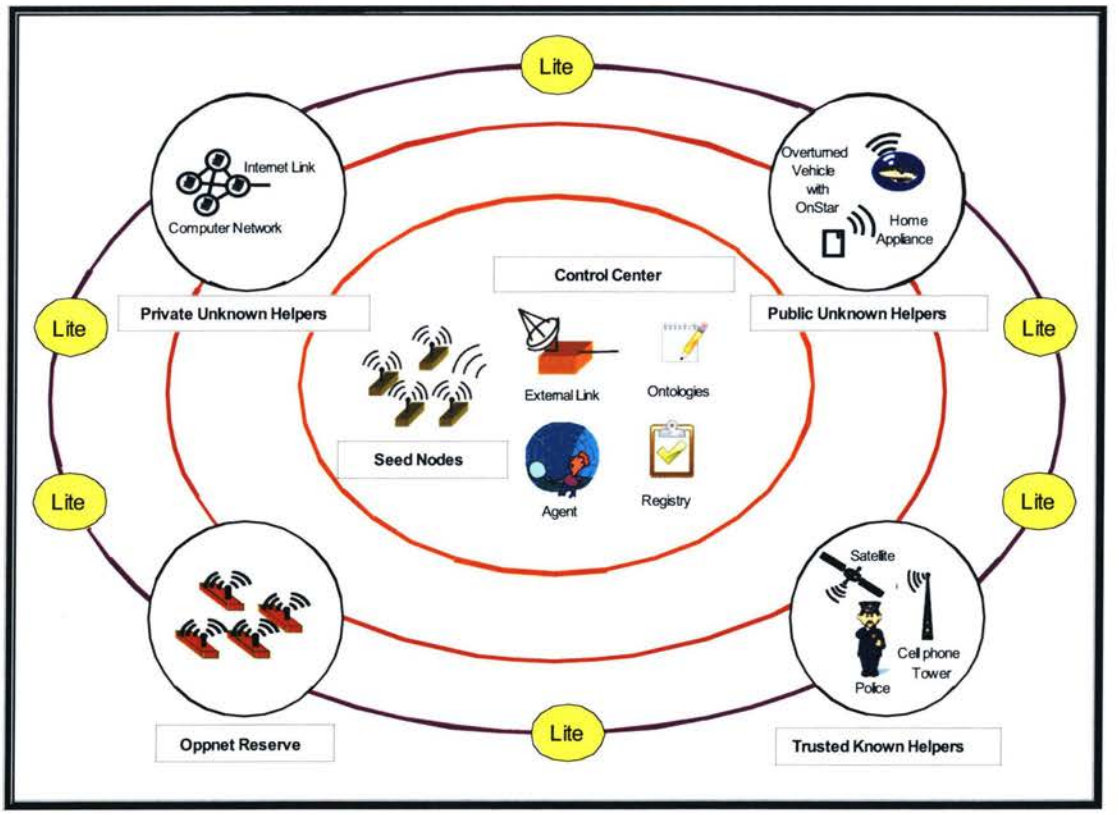


Figure 3-1: Oppnet SCOW-Q Model for Semantic Capability Discovery with QoS Parameters

3.1.2.1 Oppnet SCOW-Q for Seed Nodes and the Control Center

Seed nodes are a group of nodes simultaneously deployed to form an initial Oppnet. A distributed *control center* (CC) is a subset of the seed nodes that have more management power than other Oppnet nodes. This subsection describes the proposed interaction between Seed nodes and the Control Center in the Oppnet SCOW-Q model.

To simplify Oppnet architecture, we assume that only agents within CC nodes can discover needed helper nodes and identify their capabilities through helper (or service) advertisements. Given that the CC nodes are able to accept or reject candidate helpers, the agent management and the helper registry based on ontologies can reside in them. In an Expanded Oppnet, CC continues to carry on its tasks of maintaining up-to-date ontologies, and updating the registry of advertised capabilities whenever new advertisements are received (or discovered). When a new helper joins the Oppnet, a copy of the most current ontology and helper registry is downloaded to the helper from CC. Also, a matchmaking agent is extracted from CC and deployed on the helper to assist in the autonomous helper-matching process of the helper.

3.1.2.2 Establishing Trust with QoS specifications between Seed Nodes and the Control Center Using the SCOW-Q Model

Seed nodes are a group of nodes simultaneously activated at a initial Oppnet deployment. A Distributed Control Center (CC) is a subset of seed nodes that have the ability to accept or reject more nodes. However, given that the latter is a subset of seed nodes, the focus of this section is to describe the elements required to establish an environment of trust for seed nodes, including additional functionality expected to be performed by the CC in order to semantically enable an environment for Oppnets.

a) Initializing a Trust Relationship: The SCOW-Q model suggests that once the seed node is deployed, an identity-based access control mechanism between seed nodes and the Control Center nodes (CC) should be implemented. This assumes that seed nodes and the CC nodes are known amongst them and trustworthy. Given that CC nodes are able to accept or reject nodes, the authorization and authentication control could reside in the CC. Once the Oppnet continues to grow the CC nodes will continue to carry on their tasks of issuing commands to other nodes. Besides, the CC node can be responsible for managing the ontologies that semantically enable the communication among nodes. The interaction between these nodes via our SCOW-Q model confirms that it is possible to implement identity-based access control mechanisms in a Semantic Web environment and it is possible that two or more nodes could have a pre-established trusted relationship.

b) Observation Task: In the SCOW-Q model, the seed nodes actively participate in monitoring the helpers, keeping track of their behaviors. Meanwhile, the CC node acts as a silent observer of the helpers' behaviors joining the Oppnet. The CC node can assist in building a reputation-based model where ratings are received and scores are produced while the Oppnet is configured. It is worth mentioning that, in this case, it is recommended to implement a Distributed Reputation System (DRS) such as the one proposed in [22]. Although the DRS architecture does not require a centralized location for consolidating the reputation scores, it utilizes "score stores" to hold and distribute ratings to other parties. Hence, once the Oppnet is deployed, the CC node can perform the task of a "score store" to consolidate and make available reputation ratings to the seed

and insiders who intend to establish a transactional relationship with outsiders, i.e. potential candidate nodes.

c) Evolution of Reputation and Trust: In the SCOW-Q model, the seed nodes exchange the information with the CC node regarding helpers. The CC node then maintains a record of entities that can be trusted, and which can later be used as an additional layer of security when reputation-based access control methods are implemented. Before each node is released from the Oppnet, the CC node uploads an updated ontology containing ratings and feedbacks on the performance of that node during the lifetime of the aforementioned interaction.

d) Enabling QoS Specifications: In the SCOW-Q model, a CC node is responsible for having agents who will constantly monitor the environment, looking for potential services, maintaining and updating changes to the service registry and ontologies. Agents will discover Web services by accessing the Service Profile. In the Service Profile, the agent will have the opportunity to look at the QoS specifications advertised by the service and validate them if there is a match with the QoS specifications required by the requester of the service or if they comply with the default QoS requirements (i.e. the minimum expected QoS parameters which are yet to be determined). Given that a CC node is able to accept or reject nodes, the Management Agent and the Service Registry based on ontologies should reside in here. Once the Oppnet is expanded, the CC nodes will continue to maintain ontologies up to date and update the registry of advertised services as new advertisements are added. A snapshot of agents that realize this functionality in the CC node is depicted in Fig. 3.2.

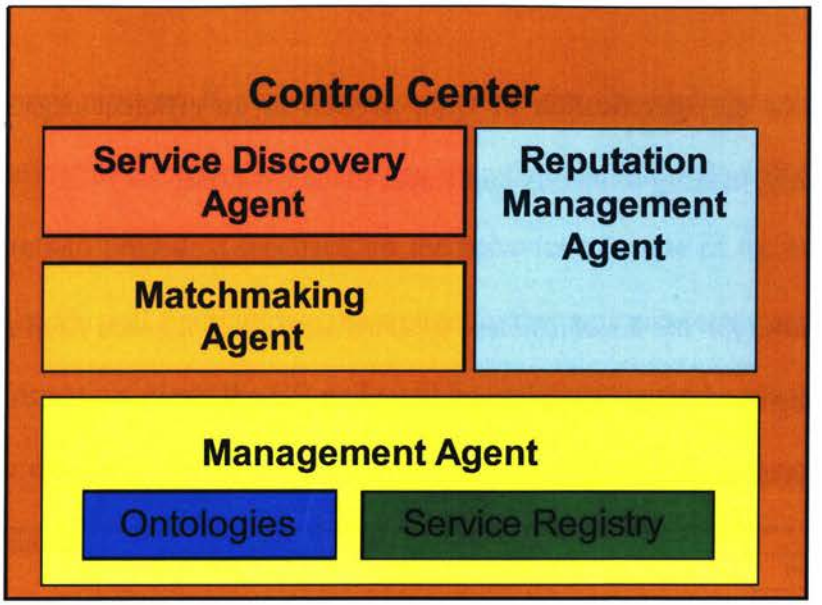


Figure 3-2: Agents' Architecture in the Control Center.

In Fig. 3.2, a number of agents perform some key tasks in order to assist in the configuration and maintenance of the Oppnet. The Service Discovery Agent looks up for services advertised by other entities, interprets their access control policies to establish a trustworthy relationship, then it validates the QoS specifications. The Matchmaking Agent or matching process is responsible for comparing the services provided with those that were requested, determining if the services are useful or not depending on the context in which the Oppnet has been deployed. The Reputation Management Agent is responsible for holding reputation ratings, and in some cases, computing the reputation scores. However, its most important features is to make these scores available to other nodes in the Oppnet. The Management Agent has as the core task to maintain the ontologies and the service registry up-to-date, i.e. to maintain a list of all the identified services, including their functional and non-functional QoS specifications. Here, the EASY framework [65] could be implemented to efficiently match the advertised service capabilities and properties with those requested by the seed nodes. EASY provides and efficient mechanism for service discovery which is independent of underlying layers,

thus, allows for the specification of QoS, as well as an efficient organization of the Service Registry, making it directly applicable to the CC node.

e) Process Flows for the Establishment of Trust with QoS between Seed Nodes and the CC Node Using the SCOW-Q Model:

This process is depicted in Fig. 3.3.

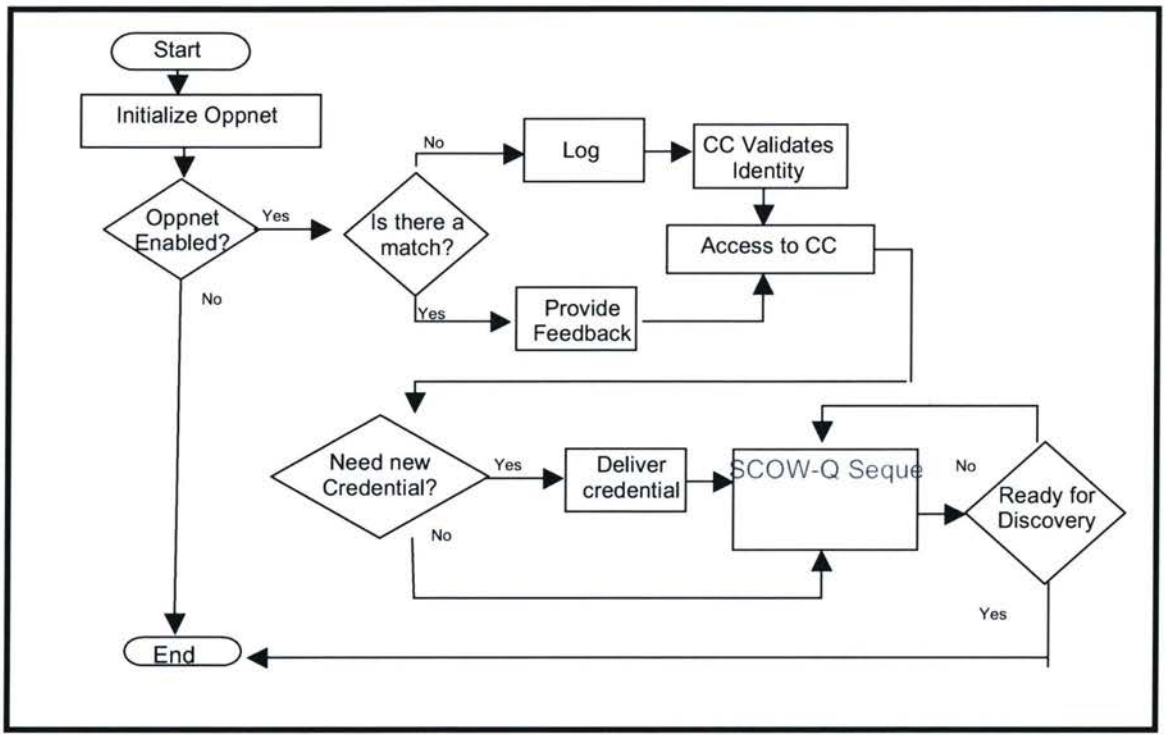


Figure 3-3: Process Flows for the establishment of Trust with QoS between Seed Nodes and the CC Node

In Fig. 3.3, the first step is to identify if a node is Oppnet enabled and if so, it becomes a seed node. If the answer is positive, the next step is to determine the identity of that seed node. If this is the first contact, the seed node will form the Oppnet and the CC node will issue a new credential that will be valid for the duration of the configuration of this new Seed Oppnet. A new download containing and updated ontology and service registry will be delivered to the seed node in order to start the

service discovery process. If a communication with the CC node has previously been established during the Seed Oppnet's configuration then the CC node will assume that the seed node wants to provide a feedback on the behavior of some of the entities. To this effect, the CC node will first validate the credential that was previously issued, then, will accept the information. Next, the CC node will be updating the seed node with any up-to-date ontology or service registry. The last step indicates that the seed node is ready to start or continue the service discovery task.

3.1.2.3 Process Flows in the SCOW-Q Model

A critical task for a seed node is to locate and discover potential helpers. This section discusses the process of establishing trust with QoS when dealing with the seed nodes and potential helpers, according to the aforementioned helpers' classification.

a) Initializing a Trust Relationship:

- *Case of Trusted Known Helpers (TKHs):* The SCOW-Q model assumes that the helpers in this category are known to each other. In this case, an identity-based trust model can be used to ensure trust amongst the helpers and allow them to exchange their credentials or signatures, thus to share their services. Typically, when a seed node or a helper node wishes to access one or more of the services provided by a TKH, it looks for a particular service in the service registry of that TKH. To access the capabilities of this TKH, it is necessary for the seed node to have a credential to interact with that entity. In this case, the request could be placed directly through the seed node – which has already gotten a credential issued by the CC – or could be placed to the CC who will behave as a broker between the seed node and the TKH. Afterwards, a decision will be made on the

trustworthiness of the node (TKH) and a recommendation will be issued by the CC based on prior knowledge of that node and its rated level of reputation. Once the trustworthiness of each TKH is validated, one can use a Web of trust such as the one proposed in [25], [75], or a mechanism referrals such as the one described in [23] to manage the network of current available trusted entities.

- *Case of Public Unknown Helpers (PUHs):* The SCOW-Q model assumes that the helpers in this category are public, i.e. their policy requirements are opened and disclosed. In this case, a policy-based trust model can be implemented to manage trust amongst PUHs. To this effect, the REI [46] or the KAoS [44] policy languages can be used.
- *Case of Private Unknown Helpers (PriUHs):* Seed nodes could discover helper nodes in which their access policies are private and do not allow the exchange of information unless a policy negotiation takes place and an agreement is reached. In this case, a policy-driven negotiation access control model for trust (such as the one introduced in [43]) is recommended, which ensures the protection of sensitive resources and information. Its implementation could be achieved by developing policies through the use of languages such as PeerTrust [48] or Protune [49].
- *Case of Oppnet Reservists (Reservists):* In the SCOW-Q model, when a seed node calls for a reservist to join the network, it implements an identity-based access control mechanisms to issue a formal request to the reservist. In this case, the seed node should automatically know the list of reservists from a list of trusted volunteered nodes, their capabilities and their level of trustworthiness.

b) Observation Tasks Applicable to All Types of Helpers: In the SCOW-Q model, seed nodes should be able to monitor directly the behavior of helpers. Additionally, any anomaly in the activities performed by helpers should be recorded and captured (via an intrusion detection mechanism implemented at the helper's level) to construct a list of references with respect to the performance of a node. This list could be used by the Oppnet to build the helper's reputation. On the other hand, helpers will be able to provide feedback on other helpers and recommend the inclusion of them in the current Oppnet setting. Helpers with limited capability (referred to as *lites*) should be able to perform a trust assessment when they are requested to produce their own information. A lower level security should be implemented at these nodes to ensure that the infrastructure they own does not cause a threat to the Oppnet. The above requiremnts can be implemented using the framework such as the one introduced in [68]. This model, originally developed for MANETs environment, advocates the use of a gossip protocol for the ontology dissemination and a walk mechanism to pinpoint potential "secure" service providers.

c) Capability Discovery Process in the SCOW-Q Model: In an Oppnet, candidate nodes could decide to join or not the Oppnet, if a candidate node decides not to join the Oppnet, it could make some of its services available to be used by Oppnets, and this will be considered as a non-functional specification. The SCOW-Q model suggests developing a set of default QoS parameters. These QoS parameters should be application-dependent, which makes the QoS issue even more difficult. For instance, in an Oppnet for military applications, real-time response (with hard deadlines) is imperative. In Oppnet applications that require meeting soft or hard deadlines, users must be allowed to define QoS requirements (including throughput and delay). In general, an Oppnet realizing

users' jobs needs to invoke its own and helpers' capabilities and utilize them in such a way that the QoS demands are met.

A process flow for the semantic helper (capability) discovery process in the Oppnet SCOW-Q model is depicted in Fig. 3.4. Here, the model presented by Nedos *et al.* [68] — originally developed for MANETs — is followed in Oppnets. The reason is that the dynamic characteristics of Oppnets are satisfied by the distributed ontology matching, using a gossip protocol ontology dissemination, and a walk mechanism to pinpoint potential capability providers. Also, the EASY Language and EASY Matching solutions for pervasive computing [65] can be used to more efficiently match the advertised capabilities and the capability requests issued by Oppnet nodes. The efficient mechanism for service discovery provided by EASY is independent of underlying layers, and allows for QoS specifications.

Oppnet nodes actively monitor each other and the candidate helper nodes, keeping track of their behavior. Any anomaly is noticed and recorded. Aggregated feedback is submitted to CC for integration at the helper registry level, building reputation database for Oppnet nodes and candidate helpers. Feedback and reputation data might be disseminated via the gossip and walk mechanisms [65].

Another component from EASY applied in Oppnets, in particular in CC, is the technique to efficiently organize the helper registry of the Oppnet SCOW-Q model (corresponding to the service repository in EASY). The EASY approach to helper registry allows for an efficient insertion of service advertisements, minimizing the impact to already registered services. It also helps reducing the number of matchings considered for a given request. It

should be noticed that in EASY the semantic reasoning employed to build the classified ontology hierarchies occur offline.

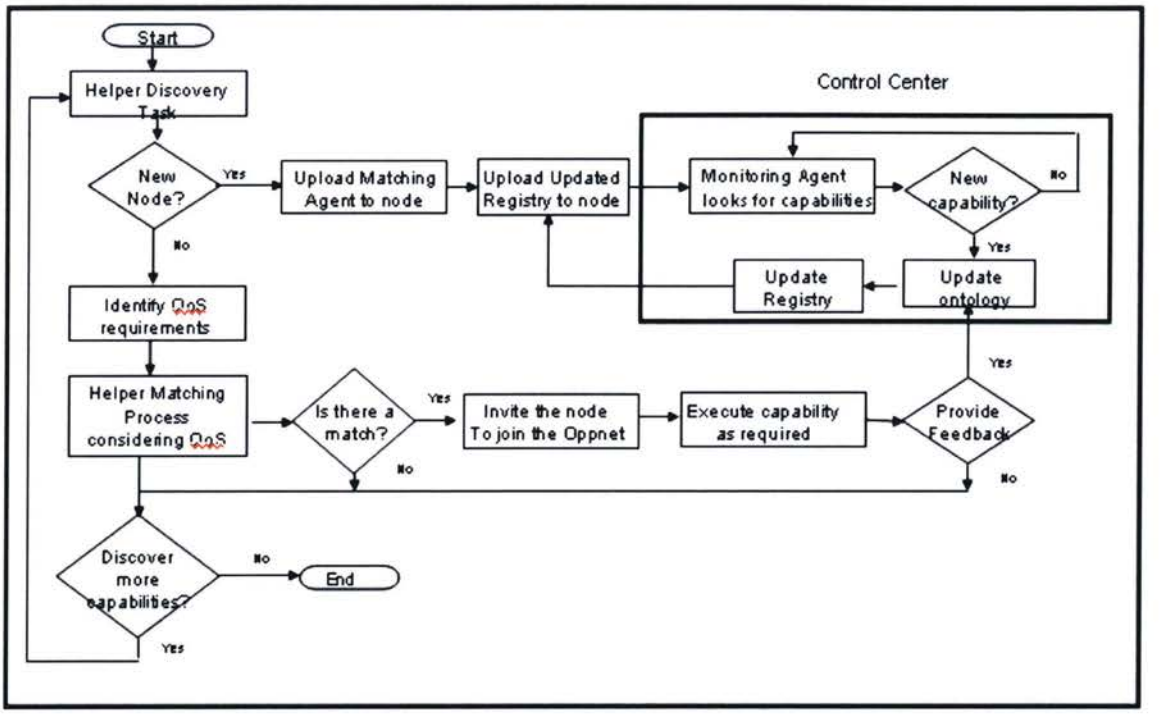


Figure 3-4: Capability Discovery Process in the SCOW-Q Model [3].

In Fig. 3.4, once a helper node has been identified by the current Oppnet and a trust management protocol has been invoked according to the aforementioned classification of helper types, the capability discovery process starts. If that helper node is considered as new, i.e. there is no previous reference of being a formal helper node of the Oppnet, and no reference of identified requirements exists, a matching agent is uploaded to that potential helper node along with an updated registry of capabilities. The CC node follows an iterative approach. In it, a monitoring agent is constantly looking for capabilities and takes responsibility for updating the registry and ontology as appropriate. It is important to mention that feedbacks with respect the QoS provided by the node, are

then captured and transferred to the CC node to be retrofitted in to the ontology. Once the matching information has been loaded into the node, it follows the helper's discovery process, but this time identified as an existing node. If the matching process finds that there is a match between the requirements and the capabilities, then the helper node is formally invited to join the Oppnet. As the helper node executes its capabilities as required, a reputation-based approach of trust is used to capture feedbacks on the performance of the service and evaluate the level of QoS provided by this helper node. Such information is then feed into the CC node.

The capability discovery process flow in the SCOW-Q model is illustrated in Chapter 4 using a series of Use Cases and sequence diagrams, following the conventions and specifications of the Unified Modeling Language (UML) [76]. These Use Cases are meant to demonstrate the process of establishing trust among nodes in an Oppnet environment. The scenarios presented through these Use Cases describe the sequence of events from the time a seed node is initialized to the time a candidate helper node has become a formal helper by accepting to participate to the Oppnet task, in the context of establishing trust during the service discovery and matching processes.

Chapter 4 Illustrating the SCOW-Q Model

This chapter illustrates the SCOW-Q model. First, the model is presented from a system's perspective using two types of diagrams: (1) the system use case diagram accompanied by a detail use case description, and (2) the sequence diagram depicting the sequences of events required for the establishment of trust in Oppnet. The diagrams mentioned above follow the standard Unified Modeling Language (UML) notation to assist in visualizing the architectural blueprint of the proposed model. Second, a representation of the working SCOW-Q model is illustrated through a typical use case scenario.

4.1 System Use Case Diagram

The System Use Case diagram depicted in Fig. 4.1 shows the number of actors that could exist at a given time in an Oppnet set up. It also identifies the events (so-called uses cases) required to describe the interactions that occur when there is an attempt to establish a trustworthy relationship using the SCOW-Q model.

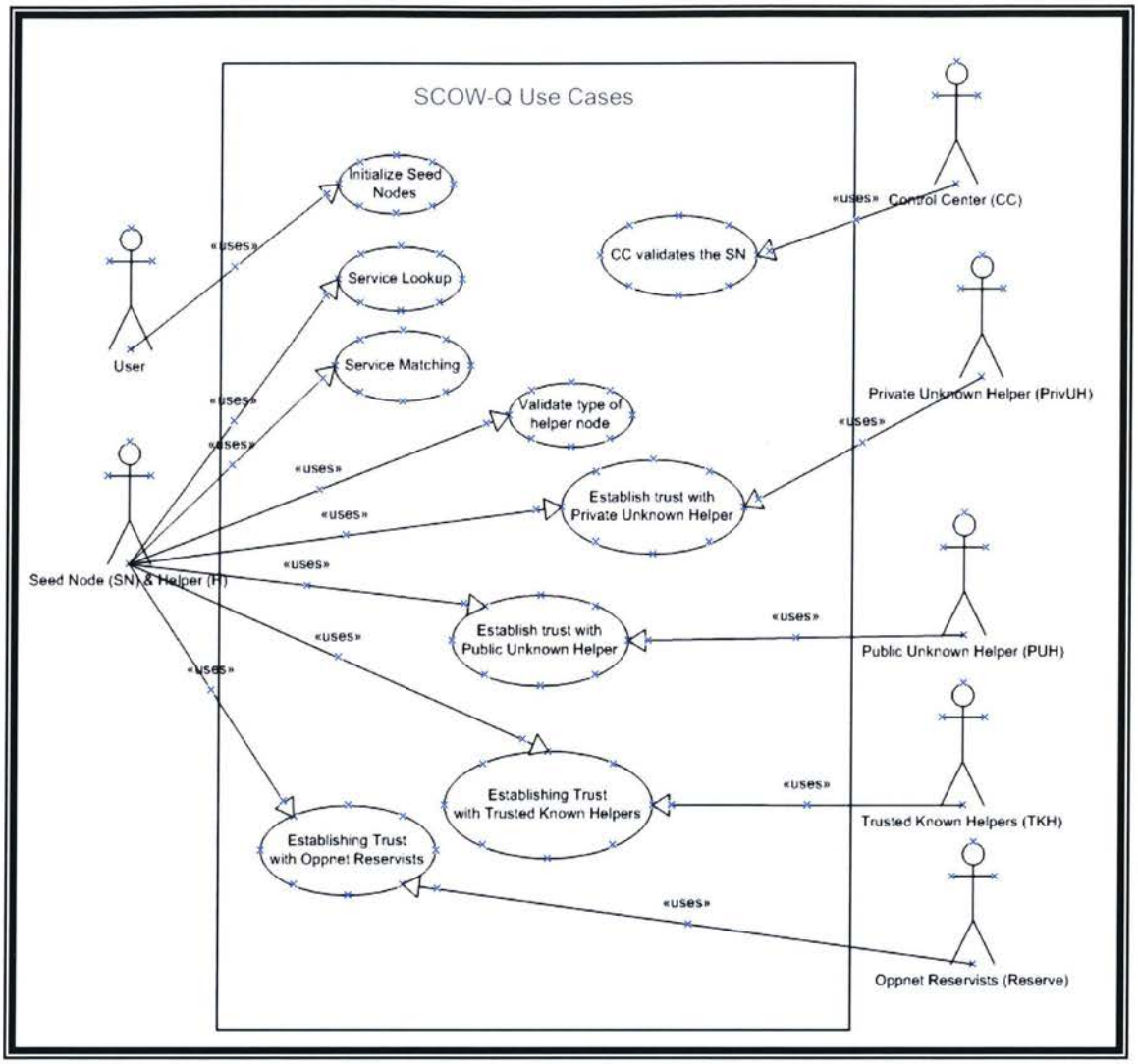


Figure 4-1: System Use Case Diagram of the SCOW-Q Model.

4.1.1 Use Case 1: Initializing the Seed Nodes

Goal Seed Nodes initializes the Oppnet and establishes a communication with the CC node
Actors 1. Seed Node (SN); 2. Control Center (CC).
Basic Flow 1. User executes request to initialize an Oppnet as a SN; 2. SN locates CC; 3. SN exchanges credentials to establish a trusted relationship with the CC node using an identity-based trust method; 4. CC validates the SN identity and issues a ticket which could be used for the duration of this particular Oppnet lifetime; 5. CC sends ticket to SN; 6. CC uploads ontology and service registry to SN; 7. SN is enabled to perform the service lookup.
Alternative Flow 1. If the SN credentials are not valid, the CC will not interact with the SN and no upload of ontologies or service directory will occur; 2. SN will use its own resources or previous ontologies to perform the service lookup and become itself a CC.
Special Requirements 1. No especial requirements.
Preconditions 1. User must have a device that is considered "Oppnet enabled" meaning that a program is installed in the device which will allow it to initialize the deployment of an Oppnet; 2. User has a previously established access key to communicate with the CC node; 3. The CC node will have previous knowledge of the existence of the SN.
Postconditions 1. The SN will use the credentials (ticket) issued by the CC node to access it for the life duration of the Oppnet.
Extension Points 1. No extension point.

4.1.2 Use Case 2: Service Lookup

Goal

Identify what services are available to the Oppnet and identify the provider of those services as potential candidate nodes.

Actors

1. Seed Node (SN);
2. Potential Candidate Node (CN).

Basic Flow

1. SN requires a service or a capability;
2. SN accesses the service registry provided by the CC node and determines what entities have provided the requested service;
3. SN scans the Oppnet environment for a specific entity or capability;
4. SN reads the service profile for those entities available on the Oppnet environment, looking for the service description;
5. SN identifies potential candidate nodes.

Alternative Flow

1. If an entity listed in the service registry is found, the SN confirms (using the service profile) that the service really exists;
2. SN identifies the type of access policy.

Special Requirements

1. No special requirements.

Preconditions

1. The service profile has been defined using an ontology language;
2. The service registry contains a list of entities (helpers) that have had a previous trusted transaction experience in an Oppnet environment and that posses a desirable reputation rating.

Postconditions

1. No postconditions.

Extension Points

1. No extension point.

4.1.3 Use Case 3: Service Matching

Goal

Identify access policies, functional and non-functional QoS properties, in order to find a match with those of the user's request. Determine which candidate helpers could become a helper of the Oppnet.

Actors

1. Seed Node (SN);
2. Potential Candidate Node (CN).

Basic Flow

1. SN identifies the type of access policy to the service;
2. SN looks for the QoS properties of the candidate node;
3. SN compares the QoS specifications of the requester of the service with those of the service;
4. SN determines whether a CN could become a helper or not for the Oppnet.

Alternative Flow

1. If no QoS specification was provided by the requester of the service, the SN could use the default QoS specifications developed for the Oppnet;
2. If the CN does not match the requirements of the service request, the SN abandons that service profile and performs the *Service Lookup* use case.

Special Requirements

1. Default QoS specifications must be determined;
2. Default QoS specifications must be expressed in an Ontology language.

Preconditions

1. The SN is accessing the service profile which has been defined using an ontology language.

Postconditions

1. No postcondition.

Extension Points

1. No extension point.

4.1.4 Use Case 4: Validating the type of helper node to determine the trust implementation method

Goal

Based on the classification of helper nodes, this use case identifies to which category the found helper belongs.

Actors

1. Seed Node (SN) / Existing helpers (H);
2. Potential Helper (PH);
3. Public Unknown Helper (PUH);
4. Private Unknown Helper (PriUH);
5. Oppnet Reservist (Reservist);
6. Trusted Known Helper (TKH).

Basic Flow

1. SN looks into the access policies advertised by the PH;
2. Based on the access policies, SN determines what type of helper node could PH be;
3. SN executes the corresponding alternative flow.

Alternative Flow

1. If the SN/H had full access to the advertised access policies, then, the PH will be considered a PUH. Hence, the SN/H will be directed to establish a relationship according to the *Public Unknown Helper* use case using a policy-based access control method.
2. If the SN/H had partial access to the advertised access policies, then, the PH will be considered as a PriUH. Hence, the SN/H will be directed to establish a relationship with the PH according to the *Private Unknown Helper* use case using a negotiation-based policy access control method.
3. If the SN/H had partial access to the advertised access policies and has discovered that a private key is required to access the service, then the PH will be considered as an Oppnet Reservist. Hence, a trusted relationship will be assumed. The SN/H will then follow the *Oppnet Reservist* use case using an identity-based access control method.
4. If the SN/H had partial access to the advertised access policies, and discovered that a private key is required to access the service, then the SN/H will identify whether the PH belongs to a Web of Trust or is only a TKH. In the case the PH belongs to a Web of Trust, the SN/H will use a reputation-based access control method. Otherwise it will use an identity-based access control method.

Special Requirements

1. No special requirement.

Preconditions
1. The services provided by the PH and its QoS properties have matched the user's request. Hence the PH could be considered as a helper (if it agrees to join the Oppnet). 2. Policies are able to support semantic representation and are written in an ontology language.
Postconditions
1. Confirm that a potential helper could become a helper as long as a trusted relationship could be established and the helper agrees to join the Oppnet.
Extension Points
1. No extension point.

4.1.5 Use Case 5: Establishing a trusted relationship with a Public Unknown Helper

Goal Establishes a trusted relationship between a Seed node / helpers and potential helper nodes. A Public Unknown Helper has been identified for having their access policies openly exposed. A policy-based access control method is invoked to exploit the requested services.
Actors 1. Seed Node (SN) / existing Helpers (H) 2. Potential Helper (PH)
Basic Flow 1. The SN/H looks for the rules specified in the access policies 2. Actions such as encryption mode are described 3. Specification of the required service is provided 4. Public Keys are exchanged 5. The SN/H invites the potential helper to join the Oppnet. 6. Potential helper confirms if it will join or not the Oppnet
Alternative Flow 1. If the potential helper rejects the invitation, then the SN/H will have to go back and perform the service lookup again.
Special Requirements 1. No special requirement.
Preconditions 1. No pre-establish relationship exists among SN/H and Potential helpers 2. Policies are able to support the semantic representation and are written in an ontology language.
Postconditions 1. Once the potential helper confirms that it will join the Oppnet, it will become a helper node. 2. The CC node will upload the required ontology and service registry to the helper. 3. Information about this helper will be sent to the CC for monitoring purposes. 4. The CC node through its Management Agent will keep the service registry and ontology up to date and will track the behavior of the new node with the help of the Reputation Management Agent.
Extension Points 1. Upload ontology and service registry to a new node 2. Monitor the helper's behavior

4.1.6 Use Case 6: Establishing a trusted relationship with a Private Unknown Helper

Goal

Establishing a trusted relationship between a Seed node/Existing helpers and potential helper nodes. A policy-driven negotiation-based control method is used.

Actors

1. Seed Node (SN) / existing Helpers (H)
2. Private Unknown Helper (PriUH)

Basic Flow

1. The SN/H looks for the rules specified in the access policies and identifies that a negotiation protocol has to take place in addition exchanging the private keys;
2. The SN/H reaches the PriUH and sends its identity information;
3. The PriUH authenticates the identity of the SN/H and submits a request for further information;
4. In order for the SN/H to disclose further credentials, the SN/H requests to the PriUH some further information via a certificate;
5. The certificate is sent by the PriUH to the SN/H;
6. The SN/H validates the certificate and discloses further details;
7. The PriUH agrees with the credentials it has received from the SN/H and it is willing to establish a relationship;
8. The SN/H invites the PriUH to join the Oppnet;
9. The PriUH confirms that if it will join or not the Oppnet.

Alternative Flow

1. If the PriUH rejects the invitation, then the SN/H will have to go back and perform the service lookup use case again.

Special Requirements

1. No special requirement.

Preconditions

1. No pre-established relationship exists among the SN/H and the PriUH;
2. Policies support the semantic representation and are written in an ontology language.

Postconditions

1. Once the PriUH confirms that it will join the Oppnet, it becomes a Helper node;
2. The CC node will upload the required ontology and service registry to the helper;
3. Information about this helper will be sent to the CC node for monitoring purpose;
4. Through its Management Agent, the CC node will keep the service registry and ontology up to date and will track the behavior of the new node with the help of the Reputation Management Agent.

Extension Points

1. Upload ontology and service registry to new node
2. Monitor the helper's behavior

4.1.7 Use Case 7: Establishing a trust relationship with Oppnet Reservists

Goal

Establishes a trusted relationship with nodes that have volunteered their services to the Oppnet, hence, are known. Use an identity-based access method to exploit the requested services.

Actors

1. Seed Node (SN)/Helper (H)
2. Oppnet Reservists (Reserve)

Basic Flow

1. The SN/H looks for the rules specified in the access policies;
2. The SH/H identifies that an identity-based access method is required in addition to exchanging the private keys;
3. SN/H reaches the Reservist by sending the required identity information;
4. The Reservist authenticates the identity of the SN/H;
5. The Reservist sends a certificate to the SN/H;
6. The Reservist agrees with the credentials it has received from of SN and is willing to establish a relationship;
7. The SN/H invites the Reservist to join the Oppnet;
8. The Reservist confirms that if it will or will not join the Oppnet.

Alternative Flow

1. If an emergency occured, then Step 7 of the above basic flow will be by-passed and the SN/H will declare that the Reservist has to join the Oppnet.
2. If an emergency has not occured and the Reservist does not want to join the Oppnet, then the SN/H will have to go back and perform the service lookup again.

Special Requirements

1. No special requirement.

Preconditions

1. Reservists are also TKHs and must have their private keys.
2. Policies are able to support the semantic representation and are written in an ontology language.

Postconditions

1. Confirm that a potential helper could become a helper as long as a trusted relationship could be established and the helper agrees to join the Oppnet.

Extension Points

1. No extension point.

4.1.8 Use Case 8: Establishing a trusted relationship with Trusted Known Helpers

Goal

Establishes trusted relationships between nodes that are known and considered as trusted entities. The trusted known helper nodes could also be members of a Web of Trust. To access these nodes, a reputation-based access control method will be invoked when the node is a member of the Web of trust. Otherwise, an identity-based access control method will be invoked.

Actors

1. Seed Node (SN)/Helper (H)
2. Trusted Known Helpers (TKH)
3. Ticket Granting Server (TGS)

Basic Flow

1. The SN/H looks at the access policies.
2. The SN/H identifies that a reputation-based access method is specified, followed by an identity-based mechanism, for the purpose of accessing the service
3. The SN/H reaches the TKH, by sending the required identity information
4. The TKH authenticates the identity of the SN/H via a third trusted party TGS.
5. The TGS issues a credential which is sent to the SN/H via the TKH
6. The TKH establishes a relationship with the SN/H
7. The SN/H invites the TKH to join the Oppnet.
8. The TKH confirms that if it will join or not the Oppnet

Alternative Flow

1. The SN/H could request to be added to the Web of Trust.

Special Requirements

1. No special requirement.

Preconditions

1. The TKH are known helpers and must have their private keys.
2. Policies are able to support the semantic representation and are written in an ontology language.

Postconditions

1. Confirm that a potential helper could become a helper as long as a trusted relationship could be established and the helper agrees to join the Oppnet.

Extension Points

1. Request to add a trusted helper into the Web of Trust.

4.2 Sequence Diagram

The sequence diagram of the SCOW-Q model is depicted in Fig. 4.2. Therein, a sequential representation of the steps required for the establishment of trust in an Oppnet based on the aforementioned categorization of helpers, is presented

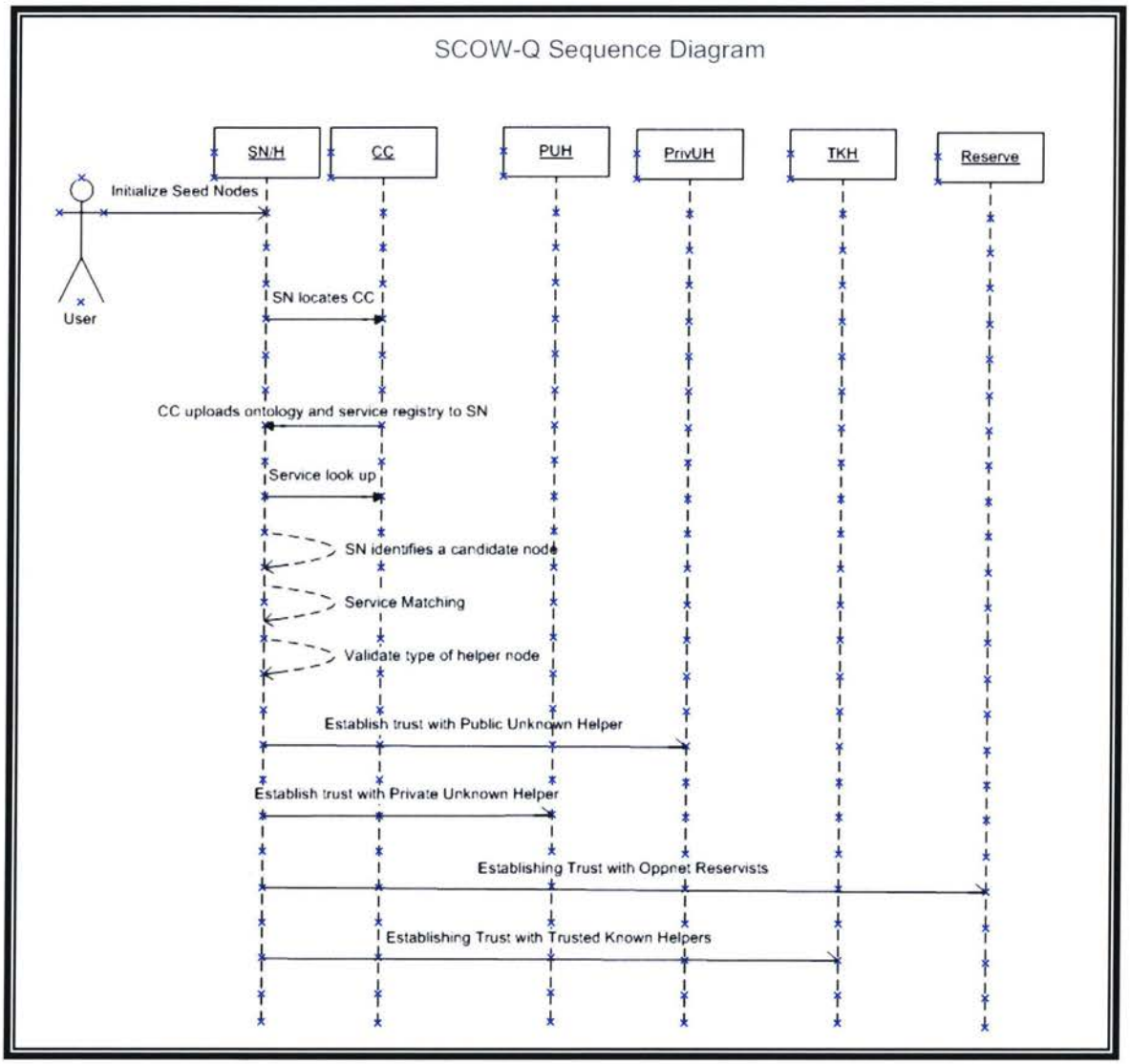


Figure 4-2: Sequence Diagram for the SCOW-Q Model.

4.3 Use Case Scenario for Implementing Trust in Oppnets Using the SCOW-Q Model

In this section, we are interested in showing how technologies such as the Semantic Web Services (SWSs) can be used to promote the realization of trust in Oppnets. To this effect, the above mentioned *use case 8 entitled "Establishing a trusted relationship with Trusted Known Helpers"* has been chosen and its achievement illustrated by means of a scenario (referred to as "Use Case Scenario". In this representation, it is possible to visualize the roles of the SWS and identity-based access control methods in the execution of capabilities requested by a seed node or a helper in an Oppnet setup. The next section defines the artifacts required to define, execute and realize the aforementioned scenario.

4.3.1 Underlying Technologies

Based on the principles of Service Oriented Architecture (SOA), Web services are geared towards achieving a seamless integration of distributed software components, particularly when services are semantically modeled considering their functional and non-functional features. The latter are known as SWS which are equipped with an ontology-based semantic reasoning that assists in the discovery and matching of required service capabilities. The use of ontologies enables the exchange of information between agents (programs) and machines without human intervention [64]. Various semantic service description languages have been developed, such as WSDL-S, WSMO, FLOWS, OWL-S, to name a few.

In this thesis, the OWL-S (a Web Ontology Language) is the framework chosen because it supports the specification of a service profile required in the SCOW-Q model to identify the capabilities of advertised services. It also allows the specification of

service conversations for the dynamic realization of user tasks from available services in an Oppnet environment. We also select *Amigo (Ambient Intelligence for the Networked Home Environment)* [77] as the framework that enables the implementation of the SWS. This framework has in its core a middleware containing the implementation of a working prototype called *COCOA: Conversational-base service Composition in pervasive computing environments with QoS support*, which assist in the discovery, matching, and composition of SWS, and which plays a key role in the provision of semantic reasoning. A summary of the COCOA architecture [78] is provided below.

The Amigo project provides a middleware that dynamically integrates heterogeneous systems to achieve interoperability between services and devices, which could be extended across different locations [6]. Figure 4-3 illustrates its architecture [77], which is composed of: (1) the *Applications & Services component* – In this layer, native or foreign applications to Amigo are developed, (2) the *Middleware component* – which comprises three main sub-layers: (i) *Intelligent User Service layer* – This sub-layer provides the functionality required to facilitate an ambient in-house network, (ii) *Base Middleware Layer*- This sub-layer deals with the implementation of COCOA. It provides the functionality needed to establish the network environment. In this sub-layer, the semantics represented by the use of ontologies are employed to communicate, discover services and devices on the network, and to define functional and non-functional properties. Finally, this layer enables the interoperability and ad hoc composition of heterogeneous service platforms and resources; hence, it is crucial to enable trust management in an Oppnet environment, (iii) *Programming and Deployment framework* – This layer provides the tools (.NET and Java) to develop the Amigo-aware services.

Finally, (3) the *Platform component* provides the support for multiple protocols which enable interoperability between different service technologies.

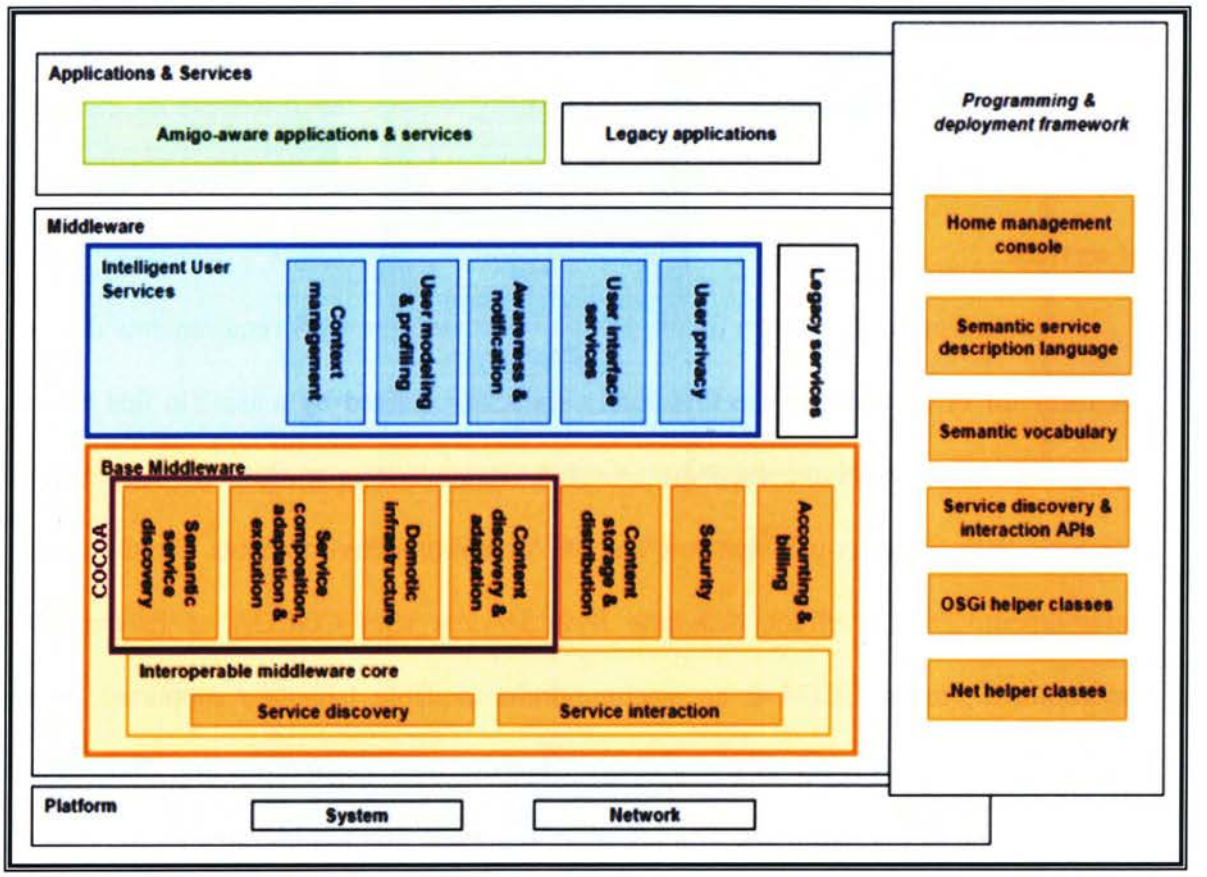


Figure 4-3: Amigo-Framework Architecture [77].

The COCOA architecture (Fig. 4.4) is composed of the following components [78]: (1) *User's device* – where the application that exploits the functionality provided by COCOA is hosted, (2) *Remote devices* – where the services that the application will use are hosted, (3) *the Amigo home server* - where the COCOA Semantic Service Repository and Execution Engine are hosted. These two subcomponents could be hosted on independent machines, however for simplicity in the context of the use case scenario described in this section; both components are considered to be allocated on the same machine. The integration of the above three components allow for the registration of

services, the dynamic composition of available services and their execution. These components exploit the following three elements found in the core of COCOA System: (1) COCOA-L, a OWL-S based language for semantic specification of services in pervasive environments, (2) COCOA-SD - which realizes the discovery of candidate services and matching, and (3) COCOA-CI – used to perform the dynamic composition of services.

According to the design of our SCOW-Q model, the QoS requirements describe the behavior of a capability (in this case a service) expected by a user (in this case an agent). Therefore, we have used the COCOA framework as a mean for realizing the Semantic Web Services in a dynamic and pen computing environments like the Oppnet environment. To this effect, *Semantic Web Services* uses COCOA as its execution environment[1], and COCOA-L as machine-understandable language supported by the OWL-S ontology, in order to semantically advertise their capabilities and descriptions, while assisting in the discovery, composition, matching, and binding of services (via the COCOA-SD). The goal of using the SCOW-Q model coupled with the COCOA framework as its underlying SWS framework, is to succeed in describing the objectives of any service requester in terms of QoS, while achieving trust in an Oppnet, thereby, dynamically assist in the realization of the user's tasks according to the specifics of the Oppnet environment in terms of available services. In the next section, we assess the above objectives by describing an instance of the capability discovery process of the SCOW-Q model in the form of our so-called "Use Case Scenario".

[1] The execution environment refers to the SWS execution environment, in which the coupling of SWS and Oppnet technology are integrated and operate.

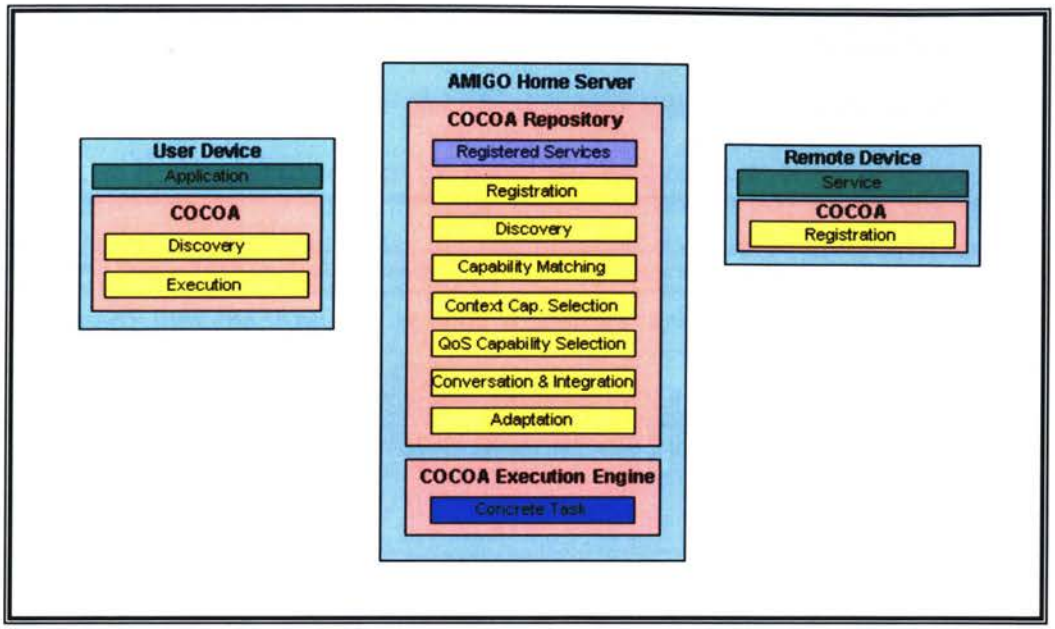


Figure 4-4: COCOA Architecture [78].

4.3.2 Use Case Scenario

The proposed use case scenario is depicted in Fig 4.5

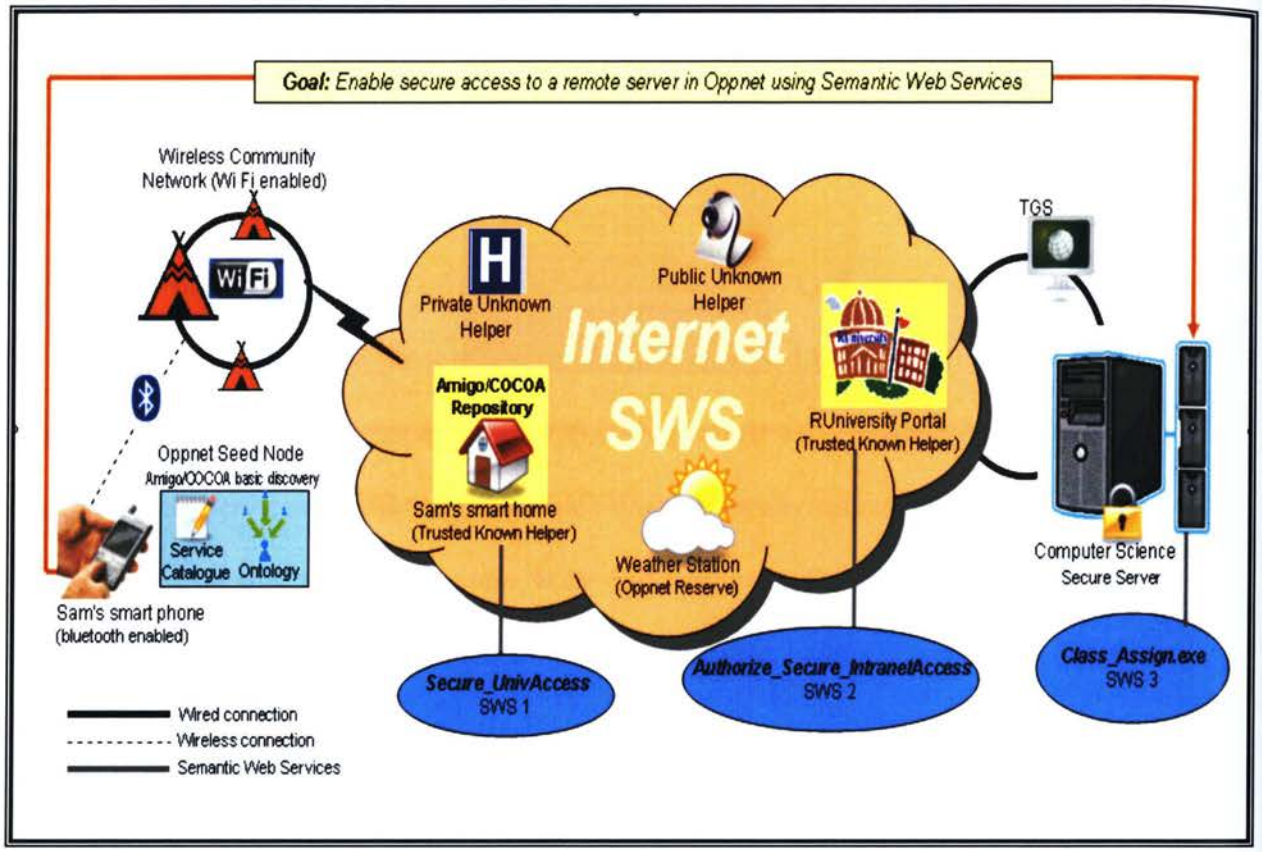


Figure 4-5: Use Case Scenario.

4.3.2.1 Use Case: Description

Sam is a student at Ryerson University (RUniversity) who has developed an application for assigning classrooms after all professors' schedules in the Computer Science Department have been completed. Amongst the list of system requirements that Sam has developed, he had to provide a SWS that allows the execution of the classroom assignment (called *Class_Assign*). However, since the application handles private information about professors, it was decided to locate the application in a secure (physically and logically) server within the confines of the Department of Computer Science. Moreover, the RUniversity's Web portal has advertised a SWS called

Authorize_Secure_IntranetAccess that has the ability to provide secure access to students through the University's Intranet. Sam not only is a bright student but also a technology savvy, so he decided to convert his house into an intelligent home (equipped with a Home Area Network) which is under the command of the Amigo Home Server. He has also advertised into the Amigo's service repository a SWS called *Secure_UnivAcces* web service which will allow him to access the secure server at RUniversity using a Virtual Private Network (VPN) connection.

Sam is taking a vacation after working long hours to finish the project. He decides to go camping near to a First Nations Reserve area (where a wireless community network exist with WiFi connection enabled). Obviously, he did not take his computer with him. Rather, he only took his Smart phone (which is Bluetooth enabled), hoping that in case he needs it, he might get access to a resource which could help him to get the Internet access. Before he goes and as a precaution, he downloads some applications that might be useful, for instance, a Java runtime, an Amigo/COCOA basic discovery software and the Oppnet deployed seed node application[2]). While out in the woods, Sam suddenly remember that a process needs to be run in order to collect all the professors' schedules and perform the desired classroom assignment. If he does not execute the process, all the students in the Department of Computer Science will not know where to go on the first day of classes. He immediately remembers that his Smart phone has the ability to become a seed node and deployed Oppnet that will help him to find the necessary capabilities to reach the RUniversity's Intranet and get access to the server in the Computer Science Department, to be able to run the classroom assignment process. He also would like to

[2] The Oppnet deployed seed node application is assumed to have the ability to deploy an Oppnet by becoming a seed node and also is assumed to have an updated ontology and service catalogue.

find the fastest mechanism to access the file. The application layer control protocol used for the creation, modification, and termination of session is the Kerberos protocol. This protocol is also used for authentication purpose.

4.3.2.2 Use Case: Goal

In this use case scenario, we infer that Sam's goal is to access and execute the file located in the secure server at RUniversity, in the minimum possible time and using the capabilities available through the Oppnet.

In order to accomplish Sam's desire, the following events have to occur. Given Sam's request (i.e. access to an executable file in a secure server at RUniverstity) including his preference (as fast as possible), (1) he will have to initiate a seed Oppnet, (2) and this seed Oppnet will have to look for capabilities that could help him to discover, compose and invoke the Semantic Web Services that will authorize a secure access to the Computer Science server and execute the class assignment process. Since the RUniversity Web portal has been identified as a Trusted Known Helper (TKH) – in the Oppnet context, the access policies (in this case, identity-based access control methods) advertised in the service profile, will be considered to authenticate Sam as user and fulfill his goal. The authentication will follow the Kerberos protocol, in which a third trusted party server located at the RUniversity network identified as the Ticket Granting Server (TGS) will issue the appropriate credentials that will later be used by the Computer Science server to grant access to the executable file.

In order to fulfill the above Sam's request (i.e. to access an executable file in a secure server at RUniverstity in the fastest possible way), three Semantic Web services have been identified, and shall be discovered, selected, and invoked. In the sequel, we

describe each of them, along with an example of a simplified COCOA-L based semantic description.

4.3.2.3 Use Case: Identified Semantic Web Services

1) Authorize_Secure_IntranetAccess SWS

The purpose of the *Auhtorize_Secure_IntranetAccess* SWS is to receive the request from a user, authenticate its identity to authorize access to the capabilities offered by the resource that the Web service is representing. An assumption has been made that a previous relationship between the requester of the service and the provider has already been established. Hence, the TGS in the RUniversity network will have already known the requester. Otherwise, access to the service will be denied. In order to access the secure server, Sam must be authenticated via the main student's portal, provide the appropriate identification and receive the credentials (ticket) to access the Computer Science server and invoke the *Class_Assign* SWS. The latter will then execute the desired process. The *Class_Assign* SWS is located at the Computer Science Department's server and is considered as a subclass of the *Auhtorize_Secure_IntranetAccess.* This allows to leverage the authentication capabilities. Moreover, the *Auhtorize_Secure_IntranetAccess* SWS also advertises in its service profile that one of its functional properties (QoS) is a speedy response time which could be determined as 5 milliseconds.

- Requested inputs: *person.name=U*
- Requested outputs: *Authorize_SecureAccess (U, remote.server)*
- Provided input values: *person.name="Sam"*
  remote.server="ComSci"
  ticket.granting="TGS"
  Ticket="credential"
- Requested effect: *Authorize(U,ticket.granting.server)*
- Provided effect: *Credential(U,ticket)*

2) Secure_Univ_Access SWS

The purpose of the *Secure_UnivAccess* SWS is to act as a gateway using an existing VPN connection already set up on a local network (in this case Sam's Home Area Network) to access the RUniversity secure server. The drawback of this SWS is that in order to allow access to a secondary server, it has first to authenticate and authorize access to the local home network and then negotiate the access to the RUniversity server, resulting in a time expensive activity. If we recall, one of the Sam's preference is to reach the secure server in a timely manner. This service has described in the service profile that one of its functional properties (QoS) is a response time of 10 milliseconds.

- Requested inputs: *person.name="U"*
- Requested outputs: *Authorize_SecureAccess (U,home.server)*
  Authorize_SecureAccess (U, remote.server)
- Provided input values: *person.name="Sam"*
  remote.server="ComSci"
  home.server="Sam's smarthome"
  ticket.granting.server="TGS"
  ticket="credential"
- Requested effect: *Authorize(U,ticket.granting.server)*
- Provided effect: *Credential(U,ticket)*

3) Class_Assign SWS

The purpose of the *Class_Assign* SWS is to validate that Sam is an authenticated user of the network resource and allow him to run the process. Other capabilities of this service are to stop the process and to schedule its execution.

- Requested inputs: *Credential(U,ticket)*
- Provided input values: *ticket="credential"*
  Scheduletime="hhmmss"
- Provided output values: *Status ="OK"*
  Status = "failed"
- Requested effect: *Class_Assign(status)*

A simplified SWS definition for the services is depicted in Fig.4.6. In it, the Namespace is defined as well as the concepts and capabilities offered by the service specified in WSDL notation. Moreover, an example of the capability definition is provided as well. Note that some trivial definitions have been omitted for simplicity purposes:

```
<!—Semantic Web Service Definition for TrustedKnownHelper. -->
-<wsdl:definitions
targetNamespace="http://localhost:9080/axis/service/TrustedKnownHelper">
 <wsdl:message name="IdentityBasePolicy"></wsdl:message>
 <wsdl:message name="AuthenticateUser"></wsdl:message>
 <wsdl:message name="IssueCredential"></wsdl:message>
 <wsdl:message name="SendCredential"></wsdl:message>

<!—Ontology specification. -->
<owl:Class rdf:ID="TrustedknownHelper">
<rdfs:subClassOf
rdf:resource="http://amigo.gforge.inria.fr/owl/HelperTypes.owl#HelperTypes" />
</owl:Class>

<!--Capability semantics for TrustedKnownHelper. -->

<owl:Class rdf:ID=" IdentityBasePolicy ">
<rdfs:subClassOf rdf:resource=
"http://amigo.gforge.inria.fr/owl/Capabilities.owl#ServiceCapability"/>
</owl:Class>

<owl:Class rdf:ID=" AuthenticateUser ">
<rdfs:subClassOf rdf:resource=
"http://amigo.gforge.inria.fr/owl/Capabilities.owl#ServiceCapability"/>
</owl:Class>

<owl:Class rdf:ID=" IssueCredential ">
<rdfs:subClassOf rdf:resource=
"http://amigo.gforge.inria.fr/owl/Capabilities.owl#ServiceCapability"/>
</owl:Class>

<owl:Class rdf:ID=" SendCredential ">
<rdfs:subClassOf rdf:resource=
"http://amigo.gforge.inria.fr/owl/Capabilities.owl#ServiceCapability"/>
</owl:Class>

<service:presents>
<capabilities:ServiceProfile rdf:ID="AuthorizeSecureIntranetAccessProfile">

  <!--Provided capability IdentityBasePolicy -->
  <lang:hasProvidedCapability>
     <capabilities: IdentityBasePolicy rdf:ID=
"AuthorizeSecureIntranetAccessServiceIdentityBaseCapability">
        <lang:hasConversation
```

```
rdf:resource="#AuthorizeSecureIntranetAccessServiceIdentityBaseConversation"/>
        <lang:hasOutput rdf:resource=
"#AuthorizeSecureIntranetAccessServiceIdentityBaseOutput"/>
</capabilities: AuthorizeSecureIntranetAccessIdentityBase>
</lang:hasProvidedCapability>
```

Figure 4-6: Simplified SWS Definition.

4.3.2.4 Establishing a Trusted Relationship with the Trusted Known Helper

This section describes the process of establishing a trusted relationship between the seed node and the trusted known helper according to the use case scenario depicted in Fig. 4.5. According to the sequence diagram depicted in Fig. 4.7, this process can be divided into the following phases: (1) Registration, (2) Initialize Oppnet, (3) Service Lookup, (4) Achieving Goal, and (5) Achieving Desire.

1) Registration

The purpose of this phase is to register into the COCOA Repository (CR) those services offered by remote devices that are available to be exploited by service requestors. In the context of Oppnets, certain types of helper nodes (such as Oppnet Reservist and Trusted Known Helpers) will have the ability to "pre-register" their services. Moreover, after a Private or Public Unknown Helper has established a trusted relationship with other Oppnet nodes and has received a high reputation scores, it could be converted into Trusted Known Helpers and formally register its services and capabilities in the repository. The Registration process can be described as follows: (1) The remote device who whishes to register or publish a service executes the COCOA registration process that allows him to locate the CR, (2) The remote device performs a self-service registration by passing to the CR the service profile in the format of a WSDL

(Web Service Description Language) file. This file contains the concepts and capabilities delivered by the service. An example of a WSDL file is as follows:

```
<!–Semantic Web Service Definition for TrustedKnownHelper. -->
-<wsdl:definitions
targetNamespace="http://localhost:9080/axis/service/TrustedKnownHelper">
 <wsdl:message name="IdentityBasePolicy"></wsdl:message>
 <wsdl:message name="AuthenticateUser"></wsdl:message>
 <wsdl:message name="IssueCredential"></wsdl:message>
 <wsdl:message name="SendCredential"></wsdl:message>
```

The service description provides a collection of atomic (or individual) capabilities that allows the service to be discovered. Among those capabilities is the QoS specification parameter. Finally, (3) The registration is completed by including the WSDL file into the Registered Services Catalogue. The list of available services composed in this catalogue will be later used for the service discovery activities.

For our use case scenario, the registration process is assumed to have been executed prior to the initial deployment of the Oppnet.

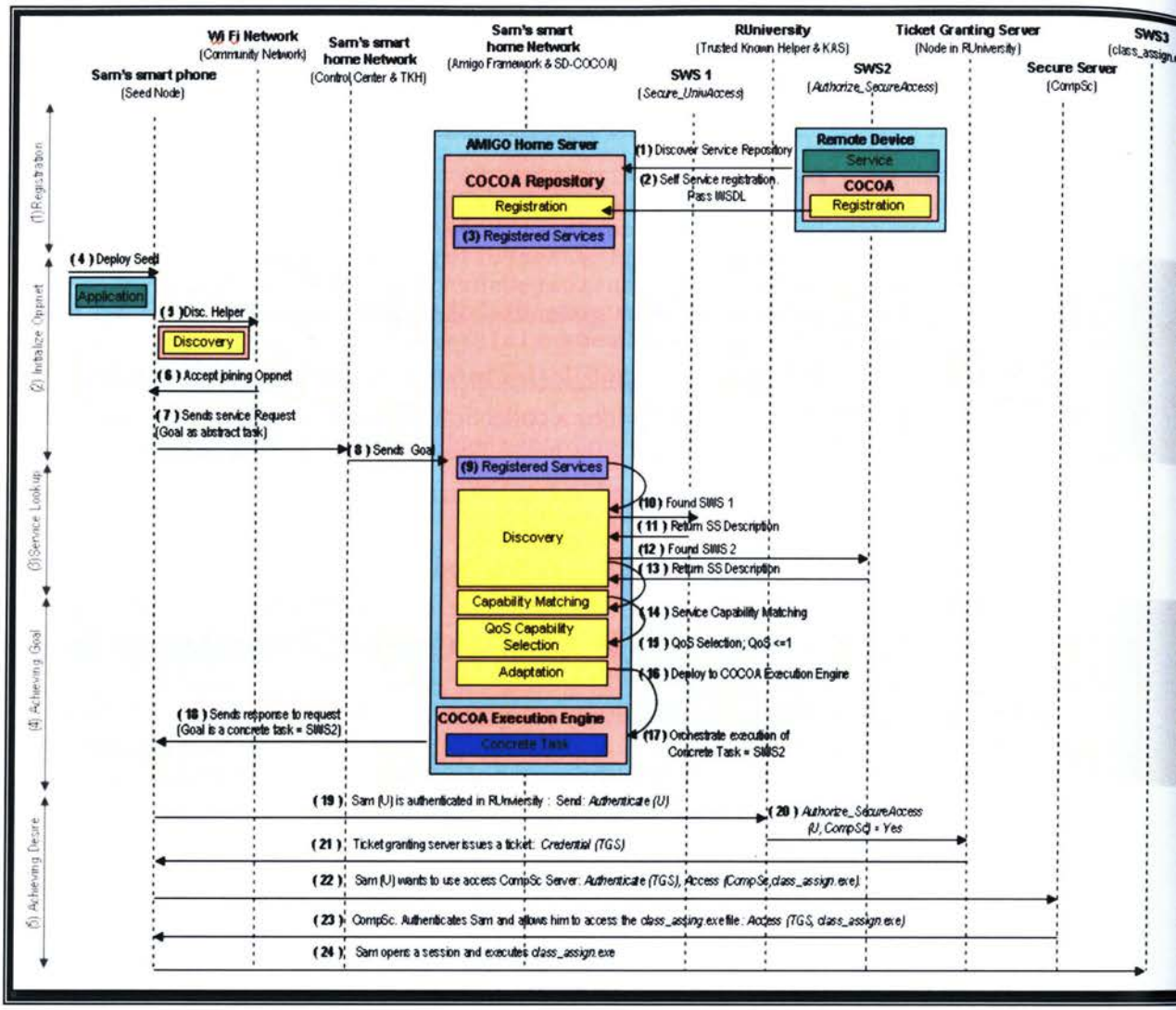


Figure 4-7: Sequence Diagram for establishing a trusted relationship between the Seed Node and a Trusted Known Helper.

2) Initializing the Oppnet

The purpose of this phase is to initialize the Oppnet and establish the initial contact with the Amigo Home Server and COCOA Repository (CR). The steps required in this phase are a continuation of the above mentioned steps. They are described as follows. According to the use case scenario description (Section 4.3.2.1), Sam has downloaded a Java runtime, a Amigo/COCOA basic discovery software (including a light version of the service catalogue and ontology) and the Oppnet deployed seed node

application into his Smartphone. (4) Using his Smartphone, (Fig. 4.5) Sam executes that application to deploy an Oppnet. The Smartphone becomes the seed node, (5) This seed node employs the COCOA basic discovery mechanism (embedded in it) to locate a nearby candidate node that could help having a connection with the Internet. Using its Bluetooth capability, the Smartphone locates a Public Unknown Helper (PUH), which is a Wireless Community Network (WCN) that uses a WiFi as a backbone connection media to access the Internet. The seed node in the Smartphone looks at the service profile and identifies that the type of node is a PUH, then identifies the security access method and establishes a trusted relationship by inviting the node to join the Oppnet (It should be noticed that the description of how to establish a trusted relationship with an PUH could be infer from the system's use case 5 in Section 4.1.5. Our focus here is on showing how trust management can be achieved with a TKH), (6) The WCN accepts to join the Oppnet and becomes a helper who will provide Internet capabilities to the seed node, (7) The WCN knows the goal of the seed node and translates it into an abstract task, meaning that no service has been match to it. In addition, the WCN provides the possibility to the Smartphone to reach the smart home network, which is a TKH. Later on, this TKH becomes a Control Center (CC node) for the Oppnet, (8) Sam's smart home network, now hosting the CC node, sends the abstract task to the Amigo Home Server to perform the semantic service lookup.

3) Service Lookup

The purpose of this phase is to perform service discovery and identify those services that are candidates to become suitable to fulfill the user's request. The Service Lookup process steps are a continuation of the above steps. Their descriptions are as follows. (9)

The abstract task sent by the Amigo Home Server is received by the CR. In the list of registered services, the CR identifies two SWS as candidates to fulfill Sam's goal. (10) The first SWS is the *Secure_UnivAccess (SWS1)*, located by the Service Discovery functionality within the CR. This SWS has been published by a device in Sam's home network. It allows Sam to access the secure server at RUniversity using a VPN connection from his home (Note that the security aspects of VPN are not considered in this example. This only illustrates the additional step in getting to the RUniversity server and the additional time implications that this will have when considering QoS parameters). The ResponseTime QoS parameter value for the SWS is set to 0.10 milliseconds. The updated semantic service description for the SWS is then sent back to the CR. (11) The second SWS is the Authorize_*Secure_IntranetAccess (SWS2)*, which is also located by the Service Discovery functionality. This SWS has the ability to provide a secure access to the University's Intranet to students. It also includes the *Class_Assign* SWS that allows for the execution of the classroom assignment process. (12) The ResponseTime QoS parameter value for the Authorize_*Secure_IntranetAccess* SWS is set to 0.05 milliseconds, and the updated semantic service description for this SWS is sent back to the CR. (13) Afterwards, the service matching process is activated.

A simplified view of the ontology that defines the ResponseTime QoS parameter is presented in Fig. 4.8 based on the context required for the Amigo framework.

```
?xml version="1.0"?>
<rdf:RDF
    xmlns:Amigo="http://www.owl-ontologies.com/Amigo/Amigo.owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
    xmlns="http://www.owl-ontologies.com/Amigo/QoSVocabulary.owl#"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:j.0="http://www.owl-
ontologies.com/Amigo/ContextVocabulary.owl#"
  xml:base="http://www.owl-ontologies.com/Amigo/QoSVocabulary.owl">

  <owl:Ontology rdf:about="">
    <owl:imports rdf:resource="http://www.owl-
ontologies.com/Amigo/Amigo.owl"/>

  </owl:Ontology>
  <owl:Class rdf:ID="Performance">
    <rdfs:subClassOf rdf:resource="http://www.owl-
ontologies.com/Amigo/Amigo.owl#QoSConcept"/>
  </owl:Class>

  <owl:Class rdf:ID="ResponseTime">
    <rdfs:subClassOf rdf:resource="#Performance"/>
  </owl:Class>

  <owl:ObjectProperty rdf:ID="hasDeviceCapabilities">
    <rdfs:subPropertyOf rdf:resource="http://www.owl-
ontologies.com/Amigo/Amigo.owl#hasDeviceContext"/>
    <rdfs:domain rdf:resource="http://www.owl-
ontologies.com/Amigo/Amigo.owl#Device"/>
  </owl:ObjectProperty>

<!-- Qos Parameter for SecureUnivAccess -->
  <ResponseTime rdf:ID="CD_QoSP_ResponseTime">
      <Amigo:QC_Value
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >0.10</Amigo:QC_Value>
      <Amigo:QC_Metric
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >millisec</Amigo:QC_Metric>
  </ResponseTime>

  <!-- Qos Parameter for Authorize_Secure_IntranetAccess -->
    <ResponseTime rdf:ID="CD_QoSP_ResponseTime">
        <Amigo:QC_Value
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >0.05</Amigo:QC_Value>
        <Amigo:QC_Metric
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
```

```
        >millisec</Amigo:QC_Metric>
  </ResponseTime>
</rdf:RDF>
```

Figure 4-8: Simplified view of the ontology defining the Response Time QoS parameter using the Amigo Framework.

4) Achieving Goal

The purpose of this phase is to select the most suitable service that will fulfill the user's request including the QoS specifications provided in the goal. In this phase, the service are composed and deployed into the COCOA Execution Engine. The steps involved are as follows: (14) The Service Matching process in the CR attempts to select a suitable service based on the matching capabilities of the two services found; (15) Among the available services (in this case, only SWS1 and SWS2), the one that comply with the specified QoS provided by the requester of the service is chosen. In this case, the QoS properties for SWS1 and SWS2 are compared and the one with the the shortest ResponseTime is selected; (16) The chosen SWS is then composed, i.e., its status is now changed from "abstract talk containing only the goal of the user" to "an instantiated concrete task with real service capabilities ready to be deployed onto the home server COCOA Execution Engine (CEE). The resulting composed service is generated as an executable *ActiveBPEL* bundle and then deployed into the CEE; (17) The CEE executes each of the composite service capabilities, and sends back a response to the original requestor indicating that the stated goal is a concrete task that could exploit the chosen SWS.

5) Achieving Desire

After the goal has been put into a concrete task that is ready to exploit the chosen SWS, it needs to be realized by the SWS execution environment. This phase is to

accomplish that task, the goal being to access a secure server at RUniversity and invoke the relevant process that executes the file located in the secure server. It should be noticed that in this phase, the Kerberos protocol is invoked as the trust management mechanism to authenticate the user who claimed to be the requester of the service (i.e. access to a file in the secure server). This authentication process is described in Section 4.3.2.5. Typically, a TGS is used to authenticate the identity of the user. It assigns a credential which provides the access to the target server to execute the *Class_Assing* SWS, which in turns runs the classroom assignment process. The steps of the "Achieving Desire" are as follows: (18) Once the seed node initiated by the user receives a notification that there is a SWS (in this case SWS2) that could be used to access the secure server and that could provide the shortest time to access such service based on the QoS specifications, Sam sends the required key to the RUniversity server to be authenticated; (19) The RUniversity server forwards the information to the TGS who validates the user and confirms that the user is indeed the registered user. The TGS sends a confirmation to Sam in the form of a credential (known as a ticket); (22) Sam sends back a request to the TGS, stating that he would like to access the secure server in the Computer Science Department. To this effect, the request is sent along with the credential issued by the TGS. This will assure the Secure Server in the Computer Science Department that the user (Sam) has been identified and has the adequate permission to access the requested process; (23) The secure server in the Computer Science Department then allows Sam to access the *Class_Assign* service in order to execute the process; (24) Sam launches a Telnet session and executes the *Class_Assign* process to fulfill his desired goal.

4.3.2.5 Establishing an Identity-Based Trust Mechanism in Oppnet Between a Seed Node and a Trusted Helper

In the realization of the above Use Case Scenario, there is a need for establishing an identity-based trust in Oppnet. The Kerberos protocol is chosen as the application layer control protocol for the creation, modification, and termination of sessions, as well as the authentication protocol. This section is meant to describe its settings.

The proposed SCOW-Q model advocates that to establish a trust relationship between the seed node and a trusted known helper, an identity-based access control mechanism between the seed nodes and the CC node should be implemented.

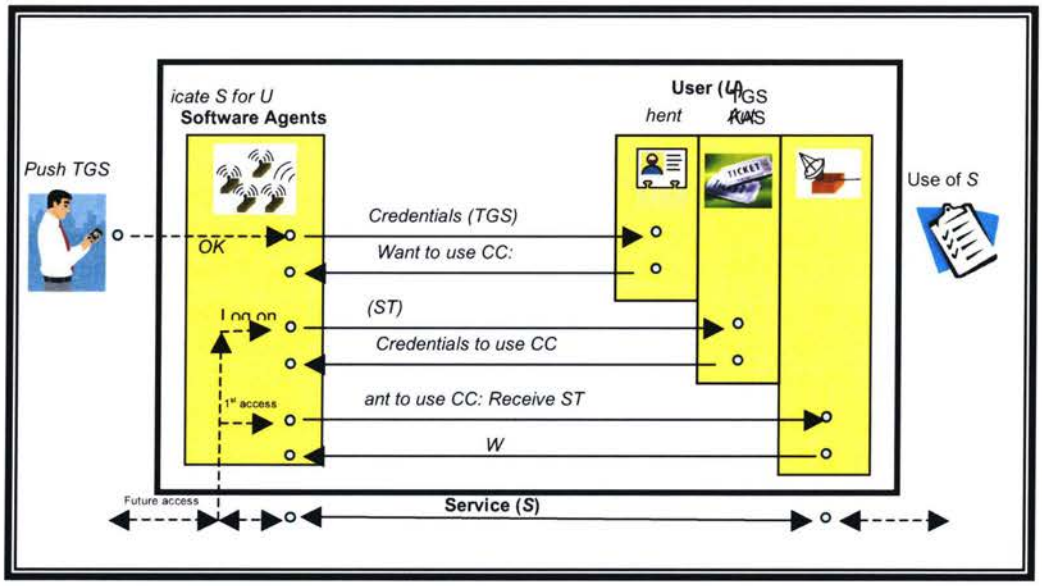


Figure 4-9: A Process Flow of Kerberos Protocol in an Oppnet Environment.

The Kerberos protocol consists of software agents that handle different tasks in the authentication process. It implements a *client* process responsible for authenticating the user's password with the objective to process the requests to reach a networked *service* process. Two software agents are required to assist in the authentication process: (1) the *Kerberos Authentication Server (KAS)* - its function is to authenticate the user and

supplies the credentials that allow the use of the network for a given period of time, (2) the *Ticket Granting Server (TGS)* - which is responsible for authenticating the client's requests based on the provided credentials. Fig. 4.9 depicts an implementation flow of the Kerberos protocol in an Oppnet environment. The steps required establishing the first contact between the seed nodes and the CC node are as follows.

1. The user (U) initiates a communication with a client seed node (*SN*).

2. While the User (*U*) logs on, the KAS process authenticates the SN client and provides credentials. These credentials are provided in the form of a called *Ticket Granting Ticket (TGT)*:

Send: *Authenticate SN for U,* Return: *Credentials (TGT)*

3. On behalf of the user, the SN client requests access to the CC node to lookup the catalogue of entities and services. This will help the user to locate a media server to upload a video clip containing a sequence from a surveillance camera.

4. The SN client presents the *TGT* to the *TGS*, then, the TGS provides a Service Ticket (*ST*) which is equivalent to the above received credentials:

Send: *SN wants to use service CC => Present TGT,* Return: *Service Ticket (ST)*

5. Once the SN client has received the ST, it is free to use it as many times as desired without involving the *TGS*:

Send: *SN wants to use service CC again => Present ST,* Return: *Confirmation (OK)*

6. On behalf of the user, the SN client uses the service to access the catalogue of entities and services in order to locate the media server.

It should be noted that for establishing a trust relationship between the seed node and the other categories of helpers (i.e. public unknown helpers and private unknown helpers), other types of trust mechanisms must be implemented within the COCOA-L language. In case of public unknown helpers (PUHs), the SCOW-Q model assumes that each candidate PUH should possess its public key and should have its services/functionality publicly advertised, and there shouldn't be any previous relationship established amongst the seed node and the PUH. Based on these requirements, the KAoS policy-based language [44] can be extended to fit the application requirements of the SWS execution environment due to its customized user interface referred to as KPAT. In the case of private unknown helpers (PriUHs), one can employ a policy-driven negotiation mechanism based on the PeerTrust language [43], [79].

Chapter 5 CONCLUSIONS

In this thesis, we have analyzed various methods for handling trust and trust management in a Semantic Web environment. Our findings revealed that some of these approaches could also be applied to the Oppnets domain, providing that a suitable Semantic Web Service framework be defined, which involve the settings of appropriate access control methods

The contributions of this thesis are twofold: (1) we have introduced for the first time a novel composite model of trust (called SCOW-Q (Semantic Capability discOvery With QoS), which provides an architectural basis for representing trust and trust management in Oppnet. This model can be viewed as a hybrid approach that aims at defining the steps for trust management in each of types of nodes found in an Oppnet, (2) we have illustrated the model through a use case scenario, using a well-known defined Semantic Web Services framework.

In the future, it is an interesting work to take advantage of the completed and available COCOA framework to develop and evaluate a complete working prototype implementation of the SCOW-Q model. When proposing a formal evaluation of this forthcoming prototype one of the suggested primary targets to be considered will be a response time comparison against the time spent for the XML parsing of service and task descriptions, which is inherent to the use of Web services and the Semantic Web technologies.

BIBLIOGRAPHY

[1] Haggle "Project, Deliverable D4.1, Preliminary design of trust and security mechanisms", [Last Accessed on August 1, 2009] *http://www.haggleproject.org/images /d/d3/ D4_1.eurecom.pdf*, July 2007.

[2] Lilien, L., Kamal, Z. H., Bhuse, V., and Gupta A. "*Opportunistic Networks: The Concept and Research Challenges in Privacy and Security*", in: Proc. International Workshop on Research Challenges in Security and Privacy for Mobile and Wireless Networks (WSPWN '06), pp. 134-147. Miami, FL, Mar. 2006.

[3] Lilien, L., Zill-E-Huma, K., Gupta, A., Woungang, I., and Tamez, E. B. *"Quality of Service in an Opportunistic Capability Utilization Network"*, Chapter in Book entitled "Mobile Opportunistic Networks: Architectures, Protocols and Applications". Auerbach Publications (Acceptance letter available). In press, to appear, 2009.

[4] Berners-Lee, T., Hendler, J., Lassila, O. "*The Semantic Web*", Source: Scientific American; Vol. 284 Issue 5, p34, 10p, 2 diagrams, May2001.

[5] OASIS (Organization for the Advancement of Structured Information Standards). "*Service Oriented Architecture Committee*", [Last Accessed on August 1, 2009] http://www.oasis-open.org/committees/tc_cat.php?cat=soa

[6] Ben Mokhtar, S., Georgantas, N., and Issarny, V."*COCOA: Conversation-based service composition in pervasive computing environments with QoS support*", In J. of Systems and Software, Vol. 80 (12 Spec. Iss.), pp. 1941-1955. 2007.

[7] Lilien, L., Gupta, A., and Yang, Z. "Opportunistic Networks for Emergency Applications and Their Standard Implementation Framework", In Proc. The First Intl. Workshop on Next Generation Networks for First Responders and Critical Infrastructure (NetCri07), New Orleans, Louisiana, Apr. 2007.

[8] Jøsang, A. "*The right type of trust for computer networks*", In Proceedings of the ACM New Security Paradigms Workshop. ACM, 1996

[9] Olmedilla, D., Rana, O., Matthews, B., Nejdl, W."*Security and trust issues in semantic grids*", In Proceedings of the Dagsthul Seminar, Semantic Grid: The Convergence of Technologies, vol. 05271, 2005

[10] Ruohomaa, S., Kutvonen, L.: "*Trust management survey*", In Proceedings of iTrust 2005, Lecture Notes in Computer Science Springer pp. 77-92. 2005.

[11] Nguyen, C.T., Camp, O., and Loiseau, S. "A Bayesian Network Based Trust Model for Improving Collaboration in Mobile Ad Hoc Networks", In Proc. of the IEEE Intern. Conference on Research, Innovation & Vision for the Future, pp. 144-151, 2007.

[12] Kohl. J., Neuman, BC. "*The Kerberos network authentication service*", IETF RFC 1510, 1993

[13] OASIS. "*Web Service Security - Kerberos Token Profile 1.1-standard specification*" [Last Accessed on August 1, 2009] http://www.oasis-pen.org/committees/download.php/16788/wss-v1.1-spec-os-KerberosTokenProfile.pdf

[14] Kauster, U., Kaonig-Ries, B., Petrie, C., and Klusch, M. "*On the evaluation of A Policy based approach to security for the Semantic Web service Frameworks*", In International Journal on Semantic Web and Information Systems 4(4), Dec. 2008.

[15] Brands, S. "Rethinking Public Key Infrastructures and Digital Certificates" Building in Privacy. The MIT Press, 2000

[16] International Telecommunication Union. "*Rec. X.509*" - Information Technology - Open Systems Interconnection - The Directory: Authentication Framework, Aug. 1997

[17] Kagal, L., Finin, T., and Joshi, A. "*A Policy based approach to security for the Semantic Web*", In Proc. of the 2nd Intl. Semantic Web Conference (ISWC03), 2003.

[18] Krukow, K., Nielsen, M.: "*Trust structures: Denotational and operational semantics*", In International Journal of Information Security 6 (2-3), pp. 153-181

[19] Bonatti, P., Duma, C., Olmedilla, D., Shahmehri, N. "*An integration of reputation-based and policy-based trust management*", In Proceedings of the Semantic Web Policy Workshop, 2005.

[20] Mui, L., Mohtashemi, M., Halberstadt, A. "*A computational model of trust and reputation*", In 35th Annual Hawaii International Conference on System Sciences (HICSS'02), Vol. 7., IEEE Computer Society. 2002.

[21] Huynh, T.D., Jennings, R.D., Shadbolt, N.R. "*An integrated trust and reputation model for open multi-agent systems*", In Autonomous Agents and Multi-Agent Systems 13 (2), pp. 119-154. 2006.

[22] Jøsang, A., Ismail, R., and Boyd, C. "*A survey of trust and reputation systems for online service provision*", In Decision Support Systems Volume: 43 Issue: 2 pp. 618-644. March 2007.

[23] Beth, T., Borcherding, M., Klein, B. "*Valuation of trust in open networks*", In Proceedings of the 3rd European Symposium on Research in Computer Security, pp. 3–18. 1994.

[24] Yu, B., Singh, M.P. "*Distributed Reputation Management for Electronic Commerce*", In Proceedings of the Computational Intelligence, Volume 18, Number 4, 2002.

[25] Agrawal, R., Domingos, P., Richardson, M. "*Trust management for the Semantic Web*", In Proceedings of the 2nd Intern. Semantic Web Conference, ISWC2003. 2003.

[26] Zhang, Y., Chen, H., Wu, Z. "*A Social Network-Based Trust Model for the Semantic Web*", In Lecture Notes in Computer Science (including sub-series Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 4158 LNCS, pp. 183-192

[27] Golbeck, J., Parsia, B., Hendler, J. "*Trust Networks on the Semantic Web*", In Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science) 2782, pp. 238-249. 2003.

[28] Zhang, J., and Cohen, R. "Demonstrating the effectiveness of a trust model for sharing ratings of information providers on the Semantic Web", Computational Intelligence journal, 2007.

[29] Pennock, D.M., Nielsen, F.A., Giles, C. L. "*Extracting collective probabilistic forecasts from Web games*", In Proc. of the 7th ACM SIGKDD Intern. Conference on Knowledge Discovery and Data Mining, pp. 174-183, San Francisco, CA: ACM Press, 2001.

[30] Kamvar, S., Schlosser, M., and Garcia Molina, H. "*The Eigen trust algorithm for reputation management in P2P networks*", In Proc. of the 12th Intl. World Wide Web Conference, 2003.

[31] Gil, Y., Artz, D. "*Towards content trust of web resources*", In Web Semantics 5 (4), pp. 227-239. 2007.

[32] Artz, D., Gil, Y. "*A survey of trust in computer science and the semantic web*", In Journal of Web Semantics 5 (2). pp. 58–71. 2007.

[33] Chklovski, T., Gil, Y., Ratnakar, V., Lee, J. "*Trellis: Supporting decision making via argumentation in the semantic web*", In Proceedings of the 2nd International Semantic Web Conference, 2003.

[34] Y. Gil and V. Ratnakar, "*Trusting information sources one citizen at a time*," In Proc. of the Intl. Semantic Web Conference, pp. 162-176. Sardinia, Italy, 2002.

[35] Blaze, M. Feigenbaum J, and Lacy, J. "Decentralized trust management", In Proceedings of the IEEE Symposium on Security and Privacy. IEEE, May 1996.

[36] Marsh, S. "*Formalising Trust as a Computational Concept*". PhD thesis, Department of Mathematics and Computer Science, University of Stirling, 1994.

[37] Blaze, M. J., Feigenbaum, J., and Keromytis, A. D. "*KeyNote: Trust management for public key infrastructures*"(position paper), In Proceedings of the 6th International Workshop on Security Protocols, volume LNCS 1550/1998, pages 59 – 63. Springer-Verlag, April 1998.

[38] Chu, Y.-H., Feigenbaum, J., LaMacchia, B., Resnick, P., and Strauss, M. "*REFEREE: Trust management for Web applications*", In Computer Networks and ISDN Systems, 29(8–13): pp. 953–964, 1997.

[39] Viljanen, L. "*Towards an Ontology of Trust*" Lecture Notes in Computer Science: Towards an Ontology of Trust, pp. 175-184

[40] Kaneiwa, K., Mizoguchi, R. "*Distributed reasoning with ontologies and rules in order-sorted logic programming*", In Web Semantics2009.

[41] Lloyd, J. W.: "*Foundations of Logic Programming*", Springer, 2nd edition, 1987.

[42] OASIS (Organization for the Advancement of Structured Information Standards). [Last accessed on August 1, 2009] Available at: http://www.oasis-open.org

[43] Olmedilla, D. "*Security and privacy on the semantic web*", In Security, Privacy and Trust in Modern Data Management Petkovic M., and Jonker W. (Eds), Spring 2006.

[44] Uszok, A., Bradshaw, J.M., Jeffers, R., Suri, N., Hayes, P.J., Breedy, M.R., Bunch, L., Johnson, M., Kulkarni, S.and Lott. J. "*KAoS policy and domain services: Toward a description-logic approach to policy representation, deconfliction, and enforcement*" In POLICY, 93 pages, 2003.

[45] IHMC Ontology and Policy Management. "*Core Ontology for KAoS (KAoSOntology)*" [Last accessed on August 1, 2009]

http://ontology.ihmc.us/KAoSOntologies.owl

[46] Kagal, L. "*REI_: A Policy Language for the Me-Centric Project*" HP Labs, Palo Alto September 6, 2002

[47] Tonti, G., Bradshaw, J.M., Jeffers, Montanar, R., Suri, N., and Uszok, A. "*Semantic Web Languages for Policy Representation and Reasoning: A Comparison of KAoS, Rei, and Ponder*", In Lecture Notes in Computer Science (including sub-series Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 2870, pp. 419-437. 2005.

[48] Gavriloaie, R., Nejdl, W., Olmedilla, D., Seamons, K.E., and Winslett, M. "*No registration needed: How to use declarative policies and negotiation to access sensitive resources on the semantic web*", In 1st European Semantic Web Symposium (ESWS 2004), Vol. 3053 of LNCS, pp. 342–356. Heraklion, Crete, Greece, May 2004. Springer

[49] Bonatti, P., Olmedilla, D. "*Driving and monitoring provisional trust negotiation with metapolicies*", In Proceedings - Sixth IEEE International Workshop on Policies for Distributed Systems and Networks, POLICY 2005, vol. 2005, pp. 14-23. 2005

[50] eBay Site. (http://www.ebay.com) World Wide Web

[51] Amazon Site. (http://www.amazon.com) World Wide Web

[52] Kauster, U., Kaonig-Ries, B., Petrie, C., and Klusch, M. "*On the Evaluation of a Policy-based Approach to Security for the Semantic Web Service Frameworks*", In International Journal on Semantic Web and Information Systems 4(4), Dec. 2008.

[53] Eastlake, D. Reagle, J. and Solo, D. "*Xml-signature syntax and processing*", W3C Recommendation, Feb. 2002.

[54] Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., and Patel-Schneider, P.F. "*The Description Logic Handbook: Theory, Implementation, and Applications*", Cambridge University Press, 2003.

[55] The World Wide Web Consortium. [Last Accessed on August 1, 2009] http://www.w3.org

[56] Kiss, C. "*Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 2.0*", W3C Working Draft, April 30, 2007. Online, [last accessed on Aug. 15, 2008] http://www.w3.org/TR/2007/WD-CCPP-struct-vocab2-20070430

[57] The DAML Services Coalition, 2004

[58] The W3C Organization. "*Web Services Description Language (WSDL) Version 2.0. Part 1: Core Language*" 2007. [Last accessed on Aug. 9, 2009] http://www.w3.org/TR/wsdl20/

[59] The DARPA Agent Markup Language (DAML) Program, Semantic Web Services, 2004. [Last accessed on Aug. 9, 2009] *http://www.daml.org/services/*

[60] The ESSI WSMO working group. [Last accessed on Aug. 9, 2009] http://www.wsmo.org/

[61] The OWL Service Coalition – OWL-S Semantic Mark-up for Web Services, [Last accessed on Aug 9,2009] http://www.daml.org/services/owl-s/1.0/

[62] Liu, Y., A.H.H.N., Zeng. L. "*QoS Computation and Policing in Dynamic Web Service Selection*". In Thirteenth International World Wide Web Conference Proceedings, WWW 2004, pp. 798-805. New York, USA. 2004. ACM Press

[63] Gaber. J. "*Spontaneous Emergence Model for Pervasive Environments*" In GLOBECOM - IEEE Global Telecom. Conference, art. No. 4437807, Nov. 2007.

[64] Amigo Consortium. "*Deliverable D2.1: Specification of the Amigo Abstract Middleware Architecture*" April, 2005. [Last accessed on July 30, 2009]

http://www.hitechprojects.com/euprojects/amigo/deliverables/Amigo_WP2_D2.1_v10%20final.pdf

[65] Ben Mokhtar, S., Preuveneers, D., Georgantas, N., Issarny, V., and Berbers, Y. "*EASY: Efficient Semantic Service Discovery in pervasive computing environments with QoS and context support*", In J. of Systems and Software, Vol. 81 (5), pp. 785-808. 2007.

[66] Chakraborty, D., Joshi, A., Yesha, Y. and Finin, T. "*Toward distributed service discovery in pervasive computing environments*", In IEEE Transactions on Mobile Computing, Vol. 5 (2). pp. 97-112. 2006.

[67] Gagnes, T., Plagemann, T., Munthe-Kaas, E. "*A Conceptual Service Discovery Architecture for Semantic Web Services in Dynamic Environments*" In Proc. of 22nd International Conference on Data Engineering Workshops, pp. 74. 2006

[68] Nedos, A., Singh, K., and Clarke, S. "*Mobile Ad Hoc Services: Semantic Service Discovery in Mobile Ad Hoc Networks*", In Proc. Intl. Conf. on Service Oriented Computing (ICSOC 2006), Lecture Notes in Computer Science 4294, pp. 90–103. 2006.

[69] Le-Hung, V. , Hauswirth, M., and Aberer, K. "*Towards P2P-Based Semantic Web Service Discovery with QoS Support*", Lecture Notes in Computer Science 3812, Springer-Verlag, pp. 18–31. 2006.

[70] Despotovic, Z., and Aberer, K. "*Possibilities for managing trust in P2P networks*", Technical Report IC200484, Swiss Federal Institute of Technology at Lausanne (EPFL), Switzerland, Nov. 2004.

[71] Verma, K., Sivashanmugam, K., Sheth, A., Patil, A., Oundhakar, S., and Miller, J. "*METEOR-S WSDI: A scalable P2P infrastructure of registries for semantic publication*

and discovery of web services" In Information Technology and Management, Vol.6 (1), pp. 17-39. 2005,

[72] Ren, K., Chen, J., Chen, T., Song, J., and Xiao, N. "*Grid-based Semantic Web Service Discovery Model with QoS Constraints*", In Proc. IEEE 3rd Intl. Conf. on Semantics, Knowledge and Grid, pp 479–482. Oct. 2007.

[73] Kopena, J., Sultanik, E., Naik, G., Howley, I. "Service-Based Computing on MANETs: Enabling Dynamic Interoperability of First Responders", In IEEE Intelligent Systems, Vol. 20 (5), pp. 17-25. Sep.-Oct. 2005

[74] Srinivasan, N., Paolucci, M., and Sycara, K. "*An Efficient Algorithm for OWL-S based Semantic Search in UDDI*" In LNCS in Semantic Web Service and Web Process composition 3387. pp. 96-100. 2005.

[75] Golbeck, J., Parsia, B., and Hendler, J. "*Trust Networks on the Semantic Web*", In Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science), Vol. 2782, pp. 238-249. 2003.

[76] Object Management Group. "*Unified Modeling language*" [Last accessed on August 1, 2009] http://www.uml.org/

[77] Amigo Consortium. "*Deliverable D4.7: Intelligent User Services, 1-Introduction*", January 2008. [Last accessed on July 30, 2009]
http://www.hitech-projects.com/euprojects/amigo/deliverables/amigo_1_d4.7_final.pdf

[78] Thomson G., Bianco S. "*iCOCOA: Software Developer's Guide*", July 2008 [Last Accessed on July 30, 2009] http://www-rocq.inria.fr/arles/download/iCOCOA/iCOCOA_Developer_Guide.pdf

[79] Bonilla Tamez, E., Woungang, I., Lilien, L., and Denko, M. K. "*Trust Management in Opportunistic Networks: A Semantic Web Approach*", In Proc. of the 7th Annual Conference on Privacy, Security and Trust (PST 2009), August 25-27, Saint Johns, New Brunswick, Canada, 2009.

Candidate Helper	An entity which offered services have a potential match to those specified by the service requestor
Capability	Refers to services, programs or resources available in a device to be exploited by other entities
Control Center	A node in an Oppnet that has the ability to invite helpers to join the network keeps track of their reputation, upload ontologies and service registry.
Helper	A node that has accepted to join an Oppnet
Oppnet	Opportunistic Network Class 2
Oppnet Reserve	Entities recognized as volunteers who that have already registered the services they could provide in an Oppnet context.
Potential Helper	An entity that has shown compatibility with the services or capabilities requested by other entities
Private Unknown Helper	An entity which exposes limited information about the services it provides.
Public Unknown Helper	An entity which publicly exposes its services and that has not had a previous relationship with a given Oppnet set up.
Seed Node	Initial node responsible for deploying (or initializing) an Oppnet
Semantic Web Services	Services that offer their service description in an ontology based language that facilitates the semantic interpretation of the functionality provided by the service.
Trusted Known Helper	An entity known to the Oppnet